

EXTRACTING HIDDEN STRUCTURES IN SOCIAL AND INFORMATION NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Bruno Dias Abrahao

August 2014

© 2014 Bruno Dias Abrahao
ALL RIGHTS RESERVED

EXTRACTING HIDDEN STRUCTURES IN SOCIAL AND INFORMATION NETWORKS

Bruno Dias Abrahao, Ph.D.

Cornell University 2014

The accelerated evolution of online social interactions and information systems has brought a marked growth in their complexity. The dynamics of these systems are not entirely regulated or engineered, but are instead governed by hidden structures that form as a result of organic growth, but are neither directly observable nor predictable by design. Understanding these structures is crucial for harnessing the benefits of social and information networks while supporting their health and growth.

Leveraging the vast amounts of data generated by the Web and the sciences, Network Science has achieved remarkable progress at identifying and modeling certain hidden structures. However, as datasets become larger, they come with higher levels of noise and represent increasingly complex, multifaceted, and multidimensional structures. Therefore, approaching these problems in a rigorous way is crucial to enable further discoveries. This thesis advances the field by providing insights gained from applying novel, principled approaches to three existing modeling and learning tasks: community detection, network inference, and Internet modeling. It concludes with an illustration of the application of sociological theories to guide the empirical analysis of online social network data, revealing hidden social structures that enable a deeper understanding of our own behavior.

BIOGRAPHICAL SKETCH

Bruno Abrahao was born on November 16, 1978 in Belo Horizonte, MG, Brazil where he obtained BSc and MSc degrees in Computer Science from the Universidade Federal de Minas Gerais in 2005. He expects to receive a Ph.D. in Computer Science from Cornell University in May 2014.

For Thomas Abrahao.

ACKNOWLEDGEMENTS

Owing the authorship of this dissertation does not make justice to all the collaboration, encouragement, and support that I received from many mentors, colleagues, and friends whom I had the privilege to learn from and work with. I hope this can be redeemed by my gratitude, which I express here.

At Cornell, I had the honor to have Robert Kleinberg as my thesis advisor. I was extremely fortunate to have the opportunity to benefit from his extensive experience and expertise in Mathematics and Computer Science as well as his contagious zest for scientific discoveries. I thank him for encouraging me to follow my own instincts and research interests, and for giving me the opportunity to take the lead on our projects, which gave me an enormous opportunity to develop independence as a researcher. I am enormously grateful to him for standing by me all along the way and for providing me, without hesitation, with all the resources and personal support I needed during my Ph.D. program.

This thesis is a fruit of the work with exceptional researchers with whom I had the honor to collaborate, Flavio Chierichetti, Karen Cook, John Hopcroft, Alessandro Panconesi, Sucheta Soundarajan, and Bogdan State. Each had a unique and profound impact on the way I view research.

I thank the members of my thesis special committee, Jon Kleinberg, Dexter Kozen, and Robbert Van Renesse for agreeing to be part of it and for taking the time to advise and guide me during the development of this thesis.

Renato Paes Leme, Eduardo Morato, Guilherme Pinto, Yogi Sharma, Eduardo Valle, and Marcos Vaz-Salles, and Yisong Yue are colleagues and friends who directly influenced the directions of my research through technical discussions, advice, inspiration, and by being role models to me.

The support systems I found at Cornell have been a key ingredient to the accomplish-

ment of this thesis. I am deeply indebted to my host family Dominic and Ines Versage for supporting me in ways that I could not possibly have dreamed of. I also specially thank Sharon Mier who, since my day one at Cornell, has been lovingly providing me with guidance to navigate the system.

Every Ph.D. owes great part of their success to the diligent work of several administrative assistants. In particular, I thank Maria Witlox, Becky Stewart, and Stephanie Meik for their attention and dedication and for taking care of basically everything outside my research.

This thesis would never been written were it not for the great mentors and teachers I had early on as an undergraduate student at UFMG. In particular, Wagner Meira Jr. and Virgilio Almeida have engaged and advised me in computer science research and provided me with invaluable professional opportunities. I also thank Alex Zhang (HP Labs), Daniel Menasce (George Mason University), and Mark Crovella (Boston University) for supporting me all along my academic trajectory.

Finally, I profoundly thank my parents Walter and Terezinha Abrahao whose unconditional love and wholehearted support are the reasons why I am able to write this thesis.

This research was funded by the AFOSR grant FA9550-09-1-0100 and Cornell University. I thank the Santa Fe Institute for granting me a scholarship to attend the Complex Systems Summer School, which dramatically influenced the direction of my research.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vii
List of Tables	ix
List of Figures	xi
1 Introduction	1
1.1 Roadmap	5
1.1.1 Preliminaries	6
1.1.2 The Link Structure Level	7
1.1.3 The Infra-Structure Level	9
1.1.4 The Social Structure Level	10
1.1.5 Bibliographical Notes	11
2 A Separability Framework for Characterizing Community Structure	12
2.1 Related Work	19
2.2 Framework Overview	21
2.3 Building Structural Classes	24
2.3.1 Datasets	25
2.3.2 Structural Classes and Feature Space	29
2.4 Framework and Application	37
2.4.1 Class Separability Measures	37
2.4.2 Class Selection Method	43
2.4.3 Class Consistency Across Networks	47
2.5 Structural Tendencies of Communities	49
2.6 Network Consistency	53
2.7 Computational Performance	58
2.8 Discussion	59
3 Trace Complexity of Network Inference	64
3.1 Related Work	67
3.2 Cascade Model	70
3.3 The Head of a Trace	71
3.3.1 The First-Edge Algorithm	71
3.3.2 Lower Bounds	72
3.3.3 Reconstructing the Degree Distribution	76
3.4 The Tail of the Trace	77
3.4.1 Reconstructing Trees	79
3.4.2 Bounded-Degree Graphs	84
3.5 Experimental Analysis	93
3.6 Discussion	98
3.7 Appendix: Proofs from Section 3.3.2	99

4	On the Internet Delay Space Dimensionality	110
4.1	Related Work	114
4.2	Methodology	118
4.3	Dimensionality Measures	122
4.3.1	Embedding dimension	123
4.3.2	Correlation Dimension	126
4.3.3	Other Dimensionality Measures	131
4.4	On the Delay Space Structure	135
4.4.1	The Geographic Component	135
4.4.2	Dimensionality Reducing Decomposition	136
4.5	Discussion	143
5	Status and Power in Online Social Exchange	145
5.1	Materials and Methods	148
5.1.1	The Couchsurfing Dataset	148
5.1.2	Linear Mixed Effects	149
5.2	Study 1: Status Giving From Surfer to Host	150
5.2.1	Between-dyads Analysis	155
5.3	Study 2: Relative Valuation of Hospitality	156
5.4	Discussion	160
	Bibliography	163

LIST OF TABLES

2.1	List of features corresponding to measures of the subgraphs that communities induce.	29
2.2	Average number of communities from each class that each node belongs to, after sampling.	35
2.3	Percentage of the probability mass of classification of elements in the test set into the correct class, using SVM, for all networks. The last row presents global separability ration between the scatter matrices J3 scores measuring the separability of classes to a baseline consisting of the J3 scores of the same data with shuffled class labels. If a value is in parentheses, this indicates that a plurality of the probability mass from that class was assigned to some other class.	43
2.4	Pairwise separability for classes in network Amazon, calculated using scatter matrices. Ratios are measured relative to the baseline value obtained by shuffling class labels.	44
2.5	Percentage of the probability mass of classification of elements in the test set into the correct class, using SVM, for all networks, after merging classes.	47
2.6	The percentage of probability mass from each class that was correctly classified. The column titles indicate the networks on which the classifiers were trained. The values in row M , column N indicate the average percentage of probability mass from each class over all networks except N that was correctly classified.	48
2.7	Summary of the feature selection results. Features are ranked in order of their frequency in the selection list over the networks. (* reporting how many quartiles of the property were selected. ** feature number according to Table 2.1.	50
2.8	k -Nearest-Neighbors classification performance using both the full set of features and the subset of the most discriminative features selected by CFS.	51
2.9	Summary of the feature selection results when classifying implicitly and explicitly defined communities. Features are ranked in order of their frequency in the selection list over the networks.(* reporting how many quartiles of the property were selected. ** feature number according to Table 2.1.)	52
2.10	The percentage of probability mass from each network that was correctly classified as belonging to that network. The row titles indicate the class that the classifier was trained on.	55
2.11	The percentage of probability mass from elements in each network that was classified as each network. Row $N1$, Column $N2$ contains the fraction of probability mass of elements from network $N1$ in the test set that was classified as network $N2$	56

2.12	The percentage of probability mass from each network that was correctly classified as belonging to that network, where the training set contained elements from the specified class.	57
2.13	Runtime results for networks DBLP, Amazon, LJ1, and LJ2. Each row represents a different class sample size. Cells contain average classification accuracy across all classes and running time (in seconds) for that sample size.	60
4.1	List of the major Tier-1 AS together with the number of IPs in their downstream networks represented in the Meridian dataset.	120
4.2	Dimensionality measures: Correlation Dimension (D_2), Hausdorff Dimension (D_0) and PCA found for the different AS networks.	140
5.1	Means and standard deviations for continuous variables, and frequency and counts for discrete variables (datasets 1 and 2) Study 1: N=80,194 post hosting interactions from 34,755 verified surfers to 32,093 verified hosts. Study 2: N = 36,156 requests from 19,100 verified surfers to 19,192 verified hosts. Hosts in Study 2 were living in 4,533 cities having at least two other CouchSurfing hosts. On average 7.97 Study 2 hosts lived in any one city. Study 1 data collected between January 2003 and November 2011. Study 2 data collected between July 2010 and November 2011.	151
5.2	Host and surfer's reports of friendship strength. $\chi^2 = 29136.6$, $df = 25$. T-stat(H_a : Surfer > Host) = 10.885, $df=75078$	152
5.3	Host and surfer's reports of trust $\chi^2 = 2190.05$, $df = 16$. T-stat(H_a : Surfer > Host) = 44.131, $df=71811$	152
5.4	Linear mixed-effects regressions comparing first ratings given in each dyad. <i>Source</i> : CouchSurfing dataset No. 1. Only one-directional ratings included. Sample sizes: 75,902; 74,311; 74,263. Scaled deviances: 168,841; 159,782; 156,170. Log-likelihoods: -84,429; -79,900; -78,107. AIC: 168,869; 159,810; 156,233. * $p < .10$, ** $p < .05$, *** $p < .01$. Two-tailed tests.	154
5.5	Linear mixed-effects regression. Response: friendship rating from surfer to host. <i>Source</i> : CouchSurfing dataset No.2. Sample size: 20,917 Scaled deviance: 39,322.29. Log-likelihood: -19,705.3. AIC: 39,446.6. 1 was added to all log-transformed quantities before taking the natural logarithm. * $p < .10$, ** $p < .05$, *** $p < .01$. Two-tailed tests. ‡Measured as the mean number of requests received and accepted by other hosts in the host's city, during the 90-day interval prior to when the surfer made the initial request for the host's hospitality.	157

LIST OF FIGURES

2.1	Six different communities of 100 nodes each, identified on the LiveJournal network through different methods, namely Metis, Annotated Community, Random Walk, Infomap, Newman-Modularity, and Louvain. The communities comprise different node sets of the network, which were displayed by applying the same network layout algorithm. The visual diversity of the collection provides a rough and ready illustration of the structural variability that can be produced by the different methods. To aid the identification of structural nuances, the lightness of the red node colors reflect node degree, from fully illuminated (low degree) to dark (high degree).	15
2.2	The process of extracting examples of communities. We apply a given algorithm to a network to extract examples of typical structures that it produces, i.e, the communities extracted as its output. The set of examples extracted is further annotated with the name of the algorithm that produced them.	30
2.3	Probabilistic multi-class supervised classification of an element in the test set in the cross-validation phase of the method to assess the separability of structural classes of communities.	40
2.4	Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the 11 classes.	41
2.5	Distribution of probability mass resulting from the SVM, classification of annotated communities.	42
2.6	Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the 9 classes after merging redundant classes.	45
2.7	Distribution of probability mass resulting from the SVM, classification of annotated communities on the 9 classes after merging redundant classes.	46
2.8	Tendency of algorithms with respect to various features. Scores are calculated by three times the number of networks on which the feature had a high score on that class plus two times the number of networks on which the feature had a medium score on that class plus the number of networks on which the feature had a low score on that class.	54
3.1	Complementary cumulative density function (CCDF) of degree reconstruction using $\Omega(n)$ traces for (a) a synthetic network with 1,024 nodes generated using the Barabasi-Albert algorithm, and two real social networks: two subsets of the Facebook network comprising 503 graduate students (a) and 1220 undergraduate students (c), respectively, from Rice University.	93
3.2	F1 score of the First-Edge, First-Edge+, and NETINF algorithms applied to different real and synthetic networks against a varying number of traces. (best viewed in color)	95

4.1	Geographic location of nodes in the Meridian dataset.	122
4.2	Percentiles of relative errors produced by Vivaldi using different values of d	123
4.3	Correlation Fractal Dimension, D_2 , of a) a set of 2500 random points in a unit square, and the Internet delay space as represented by b) the Meridian matrix and c) the MIT King matrix.	125
4.4	Pair-count plot of the Meridian dataset embedded into the Euclidean 7-space via Vivaldi.	128
4.5	Measures of dimensionality applied to the Meridian matrix: a) the Hausdorff dimension computed by the greedy set cover plot, and b) Principal Component Analysis (PCA) applied to the Meridian matrix.	129
4.6	Some fractals and fractal measures. (a) A 1.5-dimensional fractal curve. (b) Template for recursively constructing the fractal curve. (c) A fractal measure on the square. Gray level indicates density.	130
4.7	Dimensionality of the distance space based on great circle distances via a) the correlation Dimension and b) The Hausdorff Dimension. c) The correlation dimension of one of the latency-based clusters.	132
4.8	Network decomposition into intersecting pieces, each corresponding to a Tier-1 AS together with its downstream network.	137
4.9	The scree plot outputted by Isomap.	142

CHAPTER 1

INTRODUCTION

Many natural and physical phenomena can be naturally modeled as networks. Networks capture entities and their relationships within a given context. Among many examples, networks can model social structures, where entities correspond to individuals, and their connections reflect meaningful interactions, e.g., friendship, co-authorship, business partnerships, mating, and so on. In the knowledge domain, we can structure information as a web connecting related pieces of information via hyperlinks. Networks can also capture natural phenomena, such as in the biological domain in which we can model interactions between proteins through gene networks or model consumer-resource ecosystems through food webs. Investigating systems through network abstractions represents a recent divergent philosophical paradigm in the sciences. As opposed to breaking systems down into parts that we can study in isolation, researchers have been increasingly recognizing that this traditional practice oftentimes results in incomplete understanding, and that taking into account the relationships among the different parts is fundamental to capture the nature of a phenomenon.

As nature and human factors present us with outcomes that are exquisitely sensitive to the structure of connections, understanding network structure is crucial for allowing us to interact with these complex systems in a way that ensures their evolution while supporting their health and preventing disruptions that may cause them to unravel. Indeed, a lack of systematic understanding of the connected nature of our physical, natural, and engineered systems puts our environment, our economical and political well-being, and our technology at risk from collapse. Accordingly, inadvertent interventions in public policies, in the genetic code, or in computer systems, just to mention a few examples, even if localized, could lead to undesirable effects as events propagate through the net-

work and influence the other parts, sometimes resulting in irrevocable damages to these systems.

This is an exciting moment in time for the computational sciences to study networks. Not only has the network abstraction recently proved extremely powerful in the sciences, but this is also one of the first opportunities in history for us to closely investigate complex systems at large scales. Indeed, we are witnessing an unprecedented production of vast amounts of data by the Web and the sciences. This puts us in the midst of a revolution in which we can collect, analyze, and compute on fine-grained data representing phenomena emerging from the interactions among entities in ways that were not possible in the past. This revolution is further fueled by the current high interest of the private sector in leveraging data to improve their business practices and to maximize profits, and the multi-million investments from the U.S. government and from private foundations to improve tools and techniques used to access, organize and glean discoveries from huge volumes of digital data [3,4].

When we, the data scientists, look at network data, we are often filled with the same sense of wonder and fascination as we experience by observing the complexity of nature or by staring at works of art. Surely, this is in part due to the networks' overwhelming complexity and beauty when visualized. Nevertheless, in my opinion, there is a more profound reason why networks are awe inspiring: *they are mysterious objects*. Accordingly, whatever the domain from which a network originates, the bewildering complexity of the underlying phenomena is usually more than the network abstraction by itself can readily offer to an observer. In addition, the dynamics that take place in networks are not centrally regulated or engineered, but are instead governed by *hidden structures* that form as a result of organic evolution, but are neither directly observable nor predictable by design.

In most cases, we do not understand even the structure of the networks that we design. For example, the design of computer networks, which includes the Internet, accounts only for a partial explanation of their actual behavior. That is because while we develop and control the infra structure to support communication, the communication protocols, and the interface that allows for interaction, when we use these systems, we are not simply operating a device; we effectively become part of it. The complex patterns of interaction among the different parts, which are driven by decentralized and independent decisions, give rise to networks at planetary scales that evolve in an organic fashion, the global structure of which we barely understand. Consequently, developing an understanding of the structural properties of these networks through measurements is crucial for harnessing their benefits and supporting their health and growth.

Reasoning about networks and understanding the dynamics of processes taking place on them is a challenging endeavor, requiring advanced techniques to model and extract the hidden patterns that drive their structure. Examples of questions that require the discovery of hidden patterns are whether we can decompose a network into sets of entities whose members have a distinctive relationship with one another. This question, in turn, prompts us to precisely identify the properties of interaction within a group of related entities or whether different entities have different roles in shaping the network structure. As another example, we may try to extract hidden structures to model networks as geometric objects so that we can employ powerful mathematical methods to model their behavior. In some cases, we may observe processes that take place in networks, such as the identity of entities that receive messages from their connections and the time in which these events occur, but the network is unknown, i.e., the network itself is the hidden structure.

The field of Network Science, which is remarkably interdisciplinary, employing methods and aiding the investigation of fundamental questions in a number of different fields, such as Computer Science, Economics, Physics, Biology, and the Social Sciences, has emerged with the goal of seeking an orderly view of networks. The field has achieved remarkable progress at identifying and modeling certain hidden structures, examples of which are small world networks [138], navigation properties [77], centrality [76, 110], and a number of other structural properties. Despite its success, we have just begun to uncover properties of networks. In fact, Network Science is a young, vibrant field whose foundations are still to be established. This thesis advances the field by illuminating core concepts, exposing the limitations of network models, developing models and algorithms to extract structure, and demonstrating how we can use insights from fields, such as Sociology, whose long history and undisputed merit has produced both beautiful theoretical results and important practical applications, to guide the empirical analysis of social networks.

Hidden structures in networks emerge at different levels of abstraction, and this thesis aims to address the identification of these structures in social and information networks at three different levels. First, at the *link structure* level, we reason about hidden structures that form as a result of the way groups of related network entities interact and of the way they influence one another under processes taking place on the network. Second, at the *infra structure* level we study the hidden patterns of interconnection and communication of computer networks, which in turn support other systems that can be studied at the link structure level, such as online social networks. Last, at higher levels of abstraction, such as at the *social structure* level, we investigate hidden structures that form as a result of social tension induced by relationship with power imbalances. In online systems supported by the Internet, social structure is actually prevalent at virtually all levels of abstraction as the structures that form are usually driven by human

factors, such as social, political, financial, and geographical. Accordingly, understanding network structure represents an incredible opportunity for us to uncover patterns of our own behavior, through the technology we developed.

Aiming at extracting hidden structures, we leverage large datasets available in this era of “Big Data”. By “big” we do not simply mean that the datasets represent large stockpiles of readily accessible information. On the contrary, they represent noisy, multifaceted, and multidimensional descriptions of increasingly complex structures and processes, most of which we only partially understand. Therefore, approaching these problems in a principled way is crucial to enable further discoveries. As the networks become more and more complex, a natural approach is to try to master complexity. In this thesis, however, we ask a different question. We are interested in using the insights of CS Theory, Machine Learning, and the Social Sciences in a principled way to *extract simplicity*. Chapters 2 through 5 are illustrations of this approach.

1.1 Roadmap

We start with our investigations at the link structure level in Chapters 2 and 3. In Chapter 2 we are presented with network data and we are interested in mining the group structure of the network, capturing distinctive relationships among related entities, which is also known as *community structure*. As the literature contains dozens of different definitions of community structure and dozens of algorithms to extract it from networks, we aim to capture the essence of a community, by inspecting examples of real communities annotated by experts in data. This study has the side-effect of categorizing community detection algorithms by their tendencies and of challenging core definitions in the field.

Sometimes, the network is not available in the data, which contains only information

of some process taking place on the network, such as epidemics. This gives us the quintessential example of hidden structure, i.e., the network itself is hidden. In this case, we need to be able to turn data into networks. We study this question in Chapter 3 by investigating the limits on what we can learn about network structure from the network models using data.

Next, in Chapter 4 we study the infra structure level by investigating whether computer networks, such as the Internet, possess hidden structures that allow us to model them as geometric objects so that we can reason about their behavior through powerful mathematical models.

Last, as the reader will notice, in Chapter 3 we deal with influence and diffusion while we assume that every entity has the same power to influence others in the network. In Chapter 5, we employ theories from social psychology to show that entities in networks may have different roles, induced by heterogeneous resource endowments.

In the next subsections, we present some preliminaries as well as a more detailed overview of each stage of our study.

1.1.1 Preliminaries

Different fields use different terminology to describe networks. In the preceding discussion, we used the term “entities” to describe the set of objects whose interactions we want to investigate. In Computer Science and Physics we usually refer to networks as “graphs” and we use the term “nodes” or “vertices” for the entities, while in the Social Sciences we term them “actors”, as in this domain entities usually correspond to people. As for the interactions, they are usually referred to as “links”, “edges”, or “rela-

tionships”. In this thesis we are going to use the different terms that describe the same concept interchangeably, employing the most appropriate term in each context.

1.1.2 The Link Structure Level

Chapter 2 is concerned with the identification of hidden patterns of interaction among network entities, reflecting meaningful subgroups, or communities, whose nodes have a distinctive relationship with one another. This problem, known as the *community detection problem*, has gained considerable attention in the past few years.

Four major factors govern the intricacies of community detection in networks: (1) the literature offers a multitude of disparate community detection algorithms whose output exhibits high structural variability across the collection, (2) communities identified by algorithms may differ structurally from real communities that arise in practice, (3) there is no consensus characterizing how to discriminate communities from non-communities, and (4) the application domain includes a wide variety of networks of fundamentally different natures. We present a class separability framework to tackle these challenges through a comprehensive analysis of community properties. Our approach enables the assessment of the structural dissimilarity among the output of multiple community detection algorithms and between the output of algorithms and communities that arise in practice. In addition, our method provides us with a way to organize the vast collection of community detection algorithms by grouping those that behave similarly. Finally, we identify the most discriminative graph-theoretical properties of community signature and the small subset of properties that account for most of the biases of the different community detection algorithms. We illustrate our approach with an experimental analysis, which reveals nuances of the structure of real and extracted communities. In

our experiments, we furnish our framework with the output of ten different community detection procedures, representative of categories of popular algorithms available in the literature, applied to a diverse collection of large-scale real network datasets whose domains span biology, on-line shopping, and social systems. We also analyze communities identified by annotations that accompany the data, which reflect exemplar communities in various domains. We characterize these communities using a broad spectrum of community properties to produce the different structural classes. As our experiments show that community structure is not a universal concept, our framework enables an informed choice of the most suitable community detection method for identifying communities of a specific type in a given network and allows for a comparison of existing community detection algorithms while guiding the design of new ones.

Next, in Chapter 3 we address the *network inference* problem, which consists of reconstructing the edge set of a network given traces representing the chronology of infection times as epidemics spread through the network. This problem is a paradigmatic representative of prediction tasks in machine learning that require deducing a latent structure from observed patterns of activity in a network, which often require an unrealistically large number of resources (e.g., amount of available data, or computational time). A fundamental question is to understand which properties we can predict with a reasonable degree of accuracy with the available resources, and which we cannot. We define the *trace complexity* as the number of distinct traces required to achieve high fidelity in reconstructing the topology of the unobserved network or, more generally, some of its properties. We give algorithms that are competitive with, while being simpler and more efficient than, existing network inference approaches. Moreover, we prove that our algorithms are nearly optimal, by proving an information-theoretic lower bound on the number of traces that an optimal inference algorithm requires for performing this task in the general case. Given these strong lower bounds, we turn our attention to special

cases, such as trees and bounded-degree graphs, and to property recovery tasks, such as reconstructing the degree distribution without inferring the network. We show that these problems require a much smaller (and more realistic) number of traces, making them potentially solvable in practice.

1.1.3 The Infra-Structure Level

In Chapter 4, we investigate the dimensionality properties of the Internet delay space, i.e., the matrix of measured round-trip latencies between Internet hosts. Previous work on network coordinates has indicated that this matrix can be embedded, with reasonably low distortion, into a 4 to 7-dimensional Euclidean space. The application of Principal Component Analysis (PCA) reveals the same dimensionality values. Our work addresses the question: to what extent is the dimensionality an *intrinsic* property of the delay space, defined without reference to a host metric such as Euclidean space? Is the intrinsic dimensionality of the Internet delay space approximately equal to the dimension determined using embedding techniques or PCA? If not, what explains the discrepancy? What properties of the network contribute to its overall dimensionality? Using datasets obtained via the King [64] method, we study different measures of dimensionality to establish the following conclusions. First, based on its power-law behavior, the structure of the delay space can be better characterized by *fractal* measures. Second, the intrinsic dimension is significantly *smaller* than the value predicted by the previous studies; in fact by our measures it is less than 2. Third, the AS topology is reflected in the delay space; subnetworks composed of hosts which share an upstream Tier-1 autonomous system in common possess lower dimensionality the combined delay space. Finally, we observe that fractal measures, due to their sensitivity to non-linear structures, display higher precision for measuring the influence of subtle features of the delay space

geometry which are not captured by other dimensionality measures.

1.1.4 The Social Structure Level

Last, in Chapter 5, we guide the empirical analysis of social networks by employing sociological theories dating back to the 1960s, which predict that heterogeneity in resource endowments among actors often leads to power inequality. Those lacking desired resources are dependent on those who own the resources they need or want. According to Power-Dependence Theory, the power imbalance will tend toward balance, an outcome that can be reached in several ways. Among the possible power-balancing mechanisms, *status giving* is one way in which low-power actors may lessen dependence on their more powerful partners. While this mechanism may be at work in many contexts, its effects have proved challenging to investigate, especially in large-scale, real-world situations. We provide the first test of this power-balancing mechanism outside the laboratory by analyzing data from CouchSurfing.org, an international online hospitality exchange network. We consider “hosting” to be a social exchange in which hosts offer a resource (i.e., hospitality) to others (“surfers”) who need a place to stay. We then test the predictions concerning status giving and its consequences on a scale not addressed in previous work. Using mutual user-reported ratings to quantify status, we show the tendency of CouchSurfers to give higher status to their hosts, especially under conditions of resource scarcity. Status acquisition is observable at the dyadic and community levels. We explore implications of this mechanism for the evolution of CouchSurfing. We demonstrate that having status in the organization results in an increased likelihood of users achieving their goals and serves as an incentive for users to provide the resources critical to the survival of the organization.

1.1.5 Bibliographical Notes

Most of the work in this thesis has appeared previously in research papers [8–10, 12, 13, 128]. The results in Chapter 2 appeared in [12] and [10], Chapter 3 is based on the results in [8], and Chapter 4 on [9] and [13]. Last, Chapter 5 is an extended version of the contents of [128].

CHAPTER 2

A SEPARABILITY FRAMEWORK FOR CHARACTERIZING COMMUNITY STRUCTURE

Community structure captures the tendency of entities in a network to group together in meaningful subsets whose members have a distinctive relationship to one another. The identification of these subsets allows for the analysis of networks at different levels of detail, which is instrumental in illuminating the structure underlying large-scale systems [35, 54, 56, 103, 105].

Despite playing a fundamental role in the structure and function of networks, community structure has proved to be frustratingly difficult to define, quantify, and extract. In addition to challenges related to computational tractability, four major factors account for the intricacies of community extraction.

First, the literature offers a multitude of disparate community detection algorithms. Due to differences in concept and design, the output of these procedures exhibits high structural variability across the collection. Given the diverse nature of networks, the notion of meaningful communities is necessarily context dependent, involving interpretations and expectations of domain experts. Therefore, many attempts to define communities are grounded on the notion of mathematical optimization. Starting with an a priori expectation about what a community should look like, researchers specify an objective function for a search problem whose solution provides the desired communities. This process has given rise to a large collection of community detection algorithms, each aiming at optimizing a particular objective function through a particular heuristic. As an illustration, Figure 2.1 shows six communities of size one hundred extracted from the same network (the LiveJournal network, see Section 2.3.1), by different methods, namely Metis, Random Walk, Infomap, Newman-Modularity, and Louvain (described

in Section 2.3.2) and one community of that network identified by data annotation. By visually inspecting these examples, we can see that, even with this limited number of examples and despite the varying network layouts (which were a product of the application of the same network layout algorithm to the subnetworks) the structural variability produced by the different methods is readily apparent. For example, some of the communities exhibit a dense, compact core, such as those extracted by the Infomap and the Louvain methods, whereas others spread along “tendrils” consisting of long paths of low degree nodes that connect small clusters, such as the communities labeled as Metis, Random Walk, and Newman modularity. The annotated community, on the other hand, seems to possess a sparse core, including a larger number of low degree nodes that spread on its periphery.

Second, communities in real networks often emerge as a result of multiple driving forces that make up the underlying complex system. Therefore, an attempt to capture community structure by maximizing a given objective function may represent an unrealistic expectation. As a consequence, communities identified by methods that reflect mathematical constructs may differ structurally from real communities that arise in practice.

Third, there is no established consensus on the question of what properties distinguish subgraphs that are communities from those that are not communities. While we can examine examples of community structure, e.g., by asking experts to identify communities in a given domain, we can only characterize a population in the presence of negative examples. However, finding negative examples of community structure is a challenging task. Any other subset of nodes in the network that is not explicitly identified as a community is a potential negative example; however, in large networks, exhaustively enumerating all forms of negative examples is computationally intractable.

Moreover, even if we could enumerate every other set in the network, we are still faced with the possibility that these seemingly negative examples could also be valid communities that were simply not identified by the expert.

Last, the application domain includes a wide variety of networks of fundamentally different natures. Each of these networks contains meaningful communities that may possess their own distinctive structural profiles.

In this chapter, we present a framework to tackle these challenges through a comprehensive characterization of community properties. By using different notions of communities as references, our methodology enables the analysis of community structure without requiring the identification of negative examples. Our method presents a scalable framework that enables researchers to assess the structural dissimilarity among the output of new and existing community detection algorithms, and between the output of algorithms and communities that arise in practice. The analysis may guide the design of novel community detection procedures. Given the significant structural variability among the output of different algorithms (which we established in our experimental analysis, introduced at the end of this section and discussed in Section 2.4), our framework serves as a tool for practitioners to decide on the most suitable algorithm for identifying communities of a specific structure in a given network. The intended structure can be specified by producing examples of communities that are either representative of real communities in the network or possess the particular structural features we wish to find. Another dividend of our method is a way to organize the menagerie of community structures that the collection of algorithms in the literature exhibit. The framework exposes the tendencies produced by the different algorithms, which allows us to group those that behave similarly. By complementing the approach with a feature selection analysis, we are able to determine what graph-theoretical properties of a subgraph are

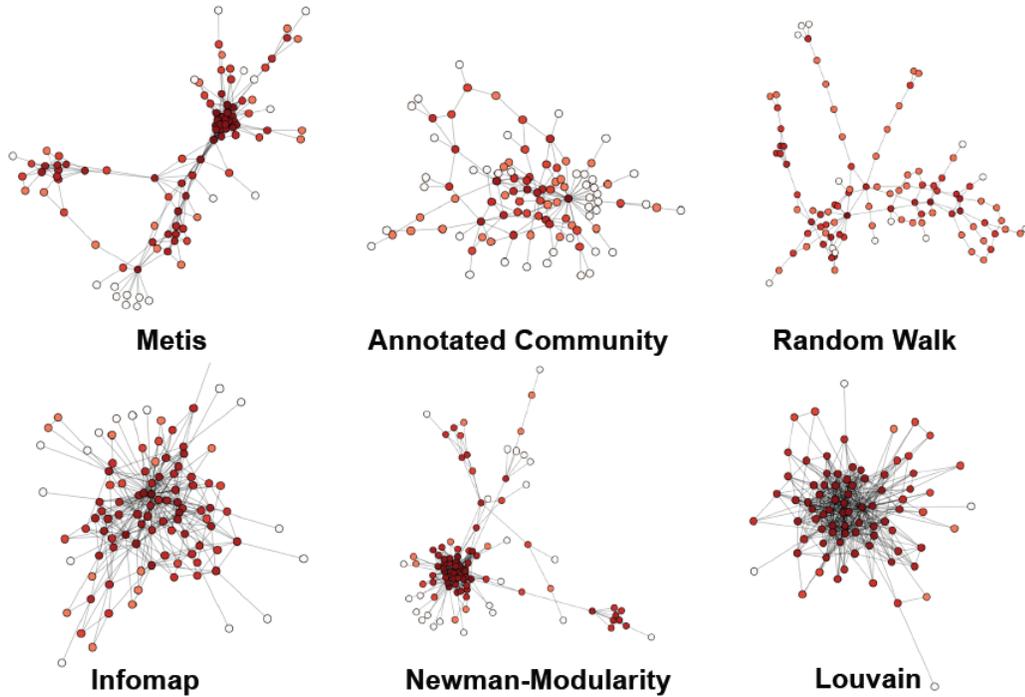


Figure 2.1: Six different communities of 100 nodes each, identified on the LiveJournal network through different methods, namely Metis, Annotated Community, Random Walk, Infomap, Newman-Modularity, and Louvain. The communities comprise different node sets of the network, which were displayed by applying the same network layout algorithm. The visual diversity of the collection provides a rough and ready illustration of the structural variability that can be produced by the different methods. To aid the identification of structural nuances, the lightness of the red node colors reflect node degree, from fully illuminated (low degree) to dark (high degree).

the most discriminative of community signature and identify those properties that account for most of the biases of the different community detection algorithms. Finally, our approach can be used to study the consistency of the output of algorithms across different networks.

We frame our approach as a class separability problem, which is able to simultaneously handle a large number of classes of communities and a diverse set of structural

properties. To this end, we specify a learning problem in which we map the distinct communities into a feature space, where the dimensions represent measures that characterize a community’s link structure. The separability of classes provides information on the extent to which different communities come from the same (or fundamentally different) distributions of feature values.

The heart of our framework is the assessment of class separability. To this end, we use traditional methods in machine learning, such as Scatter Matrices [135]. In addition, to produce more fine-grained separability information, we propose quantifying class separability through the cross-validation performance of existing multi-class supervised classifiers, both parametric, namely Support Vector Machines [136], and non-parametric, namely k -Nearest Neighbors [15]. To study the most relevant properties to analyze communities, we employ a feature selection analysis using a correlation-based method [66].

In this work we consider communities defined in two different ways and extract different classes of communities that can be grouped into two main categories: intrinsically-defined and extrinsically-defined communities.

We define the first set of communities by properties intrinsic to their link structure. For our purposes, these are the sets that are output by community detection algorithms. Each class of intrinsically defined communities comprises a set of examples that a specific algorithm extracts. The separability of these classes demonstrates the extent to which different algorithms output structurally distinguishable subgraphs. A feature selection analysis can then be employed to highlight the properties that exhibit the highest degree of inter-class variability, thereby making explicit the structural bias produced by different algorithms.

We also define communities by the context, the dynamics, or the function associated with the networks, but extrinsic to the link structure. We identify these communities through meaningful annotations provided with the datasets, such as explicit declaration of group membership, product categories, grouping by protein function, and so on. In this fashion, for each network, we form a class of extrinsically-defined communities, henceforth called *annotated communities*. These communities enable a large-scale rigorous analysis of community detection methods. The separability of the class comprising annotated communities from the classes of intrinsically-defined communities determines the extent to which community detection algorithms succeed in extracting subgraphs that are structurally comparable to the communities formed by nodes sharing extrinsic properties in common.

An important question that arises in our framework is how to select a collection of community classes that are independent enough to populate the feature space with enough diversity. A space with these properties provides a strong reference to analyze the structure of communities based on structural diversity that other notions of community exhibit. For this purpose, we propose the application of a pairwise Scatter Matrix measurement [135] for analyzing class similarity and determine which classes from a collection are redundant or independent. Depending on the application, redundant classes can be merged or be represented in a reduced collection by one representative.

We illustrate our approach with an experimental analysis, which reveals nuances of the structure of real and extracted communities. In our experiments, we furnish our framework with the output of ten different community detection procedures, representative of categories of popular algorithms available in the literature, as well as with annotated data, which reflect exemplar communities in various domains. We comprehensively characterize these examples, which we extract from large-scale real network

data spanning diverse domains, such as biology, on-line shopping, and social systems, using a broad spectrum of community properties.

In our experimental analysis, we reach the following conclusions about the communities in question. First, for all networks, the strong cross validation performance indicates that the different community detection algorithms produce fundamentally different structures that are separable on the feature space defined. Second, we observe that in nearly all cases, the annotated communities are structurally distinguishable from the output of all community detection algorithms. Nevertheless, a surprising outcome reveals that the structure of annotated communities bears closest structural resemblance to the output of simple procedures that encode little structure and were originally included in our framework as baseline procedures, such as random walk and breadth-first search. In addition, in spite of the diversity of the domains from which our networks are drawn, this observation applies to all of the networks, except to two of them for which we have a small population size. Third, we show that the different community detection algorithms produce structures that are not only consistent within a network, but even across networks. Finally, a small subset of the features is consistently observed as the most discriminative. This observation allows for a dimensionality reduction by a factor as large as 4, preserving an equivalent 10-fold cross validation performance. The most discriminative features identify the graph theoretical properties that account for most of the biases of the different algorithms. In addition to considering class structure and separability, we show that even though communities generated through the same method applied to different networks resemble one another in important ways, they also have significant differences. We demonstrate that when considering only one community detection method at a time, communities from different networks are highly separable. Even more interestingly, we show that when a classifier is trained on examples produced from only one specific community detection method, it can still identify which network

other communities came from, even when those communities were produced through other methods.

This chapter is organized as follows. Section 2.1 discusses background information and related work. Section 2.2 presents an overview of our framework. Section 2.3 introduces the datasets we use, the algorithms we consider, and the measures we apply to construct the feature space. Section 2.4 describes the heart of our framework and presents an experimental analysis thereof. Next, Section 2.5 presents the structural tendencies of communities through a feature selection analysis. Section 2.6 contains results of our network separability experiments. Finally, Section 2.8 offers our concluding remarks¹.

2.1 Related Work

The work by [56] sparked a recent wave of interest in the notion of *community structure* as a decomposition of a network that reflects meaningful properties of the underlying system [54]. Nevertheless, this area has its roots in the related problem of graph partitioning whose initial contributions date back to the 1970s [74].

As mentioned in the preceding section, the multitude of community structure definitions is a source of high variability between the output of different community detection algorithms. Among the objective functions introduced in previous work, the notion of *modularity* [56] has become an influential one. Modularity assigns high scores to communities whose internal edges outnumber the ones established in expectation by a random-network model that preserves the degree distribution of the original network. Another notion, inspired by electrical networks, is that of *conductance* [35]. The con-

¹Preliminary results appear in an earlier conference publication [11].

ductance of a set S with complement S^C is the ratio of the number of edges connecting nodes in S to nodes in S^C by the total number of edges incident to S or to S^C (whichever number is smaller). The common theme underlying the preceding notions is the search for node sets that are internally cohesive and yet sparsely connected to the rest of the network. Therefore, these measures tend to penalize sets having a large number of edges crossing the set relative to the count of internal edges.

Communities in general, however, display features that modularity and conductance may not capture, such as a preponderance of links to the outside over internal links and an arbitrary degree of overlap. This fact is substantiated by an investigation of real networks revealing that they do not split well into low-conductance communities [88] as most networks are expander-like [68]. These considerations lead to the development of alternative definitions, such as (α, β) -community [96], and algorithms, such as *Link Communities* [16] and *Clique Percolation* [111].

Despite the vast literature on community detection, the work by [16], [145], as well as ours, are among the few that attempt to analyze the structural resemblance between communities extracted by algorithms and annotated communities, which represent examples of meaningful communities in various domains.

Even though network analysts expect the output of the different algorithms to display dissimilar structural profiles due their conceptual diversity, the structural variability does not hinge simply on the choice of optimization problem. In most cases of interest, the search for a collection of node sets that maximize a given objective function is computationally intractable [54]. Therefore, in an attempt to handle the massive scale of today's networks, popular methods of community detection rely on efficient heuristics. As a consequence, previous works have quantified a significant output variability among different approximation algorithms that aim at maximizing the exact same func-

tion [81, 89].

[37] provide an excellent survey of modern community detection algorithms, which proposes a useful categorization of the different methods based on their definition. However, the authors do not include an experimental analysis to estimate the output variability that algorithms in the same category exhibit, or an assessment of whether the structures produced by these algorithms are faithful to the communities they aim to extract. Our work provides framework that allows for a comparison between algorithms empirically, based on their behavior when applied to networks, without the need to understand their definition.

In the spirit of studying the structural variability exhibited by different algorithms, closest to ours is the work by [88], which discusses properties of communities produced by multiple algorithms that aim at maximizing conductance. They consider the values of a handful of features, e.g., set compactness and internal conductance, produced by different algorithms. In contrast, here we present the first study that is simultaneously comprehensive with respect to the diversity of structural properties, of domains, of algorithms, and of scale. To illustrate this point, we demonstrate the applicability of our approach through the analysis of a collection of different communities. We take account of a set of 36 features, measured from the output produced by 10 different community detection processes. We derive our results from a diverse collection of datasets from small- and large-scale networks arising from multiple domains.

2.2 Framework Overview

The purpose of our framework is to assist researchers and practitioners in understanding the behavior of different community detection algorithms. Given the large collection of

community detection algorithms in the literature, we expect that different methods might produce different outputs when given the same input. However, little is understood about the dissimilarities among the output of different algorithms, and as these methods optimize for different criteria and use different heuristics, they may indeed search for fundamentally different types of communities. How can we understand existing algorithms, and which methods should we use as comparisons against new algorithms?

To answer this question, we need to understand what communities are and what properties they possess. However, it is not clear that the community detection problem is well-defined, and different people may have very different notions of communities. Moreover, the concept of community may not only vary from individual to individual, but also may be context and domain specific. For example, there is no reason to expect that the communities in a social network would resemble the structure of those in biological networks.

To address these issues, we can analyze real communities from different domains as identified by domain experts, and by looking at these examples, may attempt to determine what properties they have. To identify these properties, one might exhaustively enumerate all possible forms of non-communities and compare these sets against known communities. However, finding negative examples of community structure is a challenging task. Any other subset of nodes in the network that is not explicitly identified as a community is a potential negative example; therefore, in large networks, exhaustively enumerating all forms of negative examples is computationally intractable. Moreover, even if we could enumerate every other set in the network, we are still faced with the possibility that these seemingly negative examples could also be valid communities that were simply not identified by the expert.

The traditional statistical characterization task demands a training set where we dis-

tinguish communities from non-communities. Given such data, we could then train a binary classifier to distinguish between the two structures, and use the output of community detection algorithms as the test set. The classifier would then tell us the extent to which the output of some community detection algorithm resembles communities or non-communities. From this model, we would be able to extract the exclusive features that communities possess and would better understand how well the algorithms capture these properties, which can be used as a performance measure. However, as discussed in the preceding paragraph, setting up this experiment is non-trivial due to the absence of negative examples.

Our contribution in this work is to provide a way to overcome these challenging obstacles in characterizing community structure by applying machine learning techniques in unorthodox ways. The key idea in our approach is to eliminate the requirement of presenting the classifier with negative examples by learning the structural distinction among known community notions. We build a feature space that allows us to model the problem as a separability framework [53, 135] by extracting a comprehensive set of features, i.e. community properties, using graph-theoretical concepts, from each example of community we have extracted. These examples of communities, which are labeled with the name of the method used to identify them in networks, are then projected onto a feature space. This experiment allows for any outcome between two extreme scenarios. The different notions of communities we want to study may encode and capture similar enough structures so that the different classes would be hard to separate in a feature space, or at the other extreme, we might observe a clear separation of the classes in the feature space. The extent to which the classes are separable can be used as a measure of dissimilarity among the different community notions considered. Given this model, we can additionally assess the structural similarities of real communities with the communities produced by some algorithm. Here we use the diversity of community structures

exhibited by different processes as references to understand other community structures. This powerful framework allows us to assess the structural differences among different notions of communities as well as to structurally assess the quality of the output of algorithms with respect to real communities. Furthermore, we are able to concurrently consider a broad spectrum of structural features in a scalable way.

In Section 2.3, we discuss how we build structural classes to use in the training and test sets of the separability framework. In Section 2.4, we discuss how to set up the separability experiments to answer the questions we are interested in, such as measuring the extent to which different structural classes, each containing labeled examples of communities extracted by different algorithms (one structural class per algorithm) are separable. Furthermore, the distance, in terms of interclass separability and intra-class dispersion, between the classes corresponding to the output of community detection algorithms and the examples of real communities, tells us which algorithms produce communities that most closely resemble real communities. We also discuss existing measures of class separability and propose the cross-validation performance of classifiers as a measure. Finally, Section 2.5 concludes the presentation of our framework with a method for reducing the dimensionality of the data to reveal the most discriminative features on which the different algorithms load their biases. From this analysis, we can compare the behavior of different algorithms in terms of a few structural features.

2.3 Building Structural Classes

Our main goal is to capture the structural signature exhibited by different communities. We do not know a priori the extent and the source of structural variability among seemingly different communities. In addition, to answer the research questions we pose

in the preceding section, we want to relate specific structural properties with the methods that tend to produce them. To address this question, we divide the space of known community examples into classes, each corresponding to the method that generated or identified the instances therein contained. As we expect that the structural tendencies of a given community identification method are going to be reflected by the examples in its corresponding class, we refer to these classes as *structural classes*. This denomination does not imply that the classes have distinctive structures. In fact, some classes may be compact, exhibiting a clear signature, whereas others may exhibit high variance that could be difficult to characterize. Moreover, multiple classes may overlap, indicating that they contain somewhat similar structures. Accordingly, the purpose of the class separability measures we propose here is to assess the structural dissimilarity among the structural classes we consider.

Before describing our framework and delving into our analysis, in this section we present the datasets we use, as well as our methodology for building structural classes of communities from the network data. We also describe the process of projecting the communities we extract onto a feature space that allows us to treat the question of class dissimilarity as a learning problem.

2.3.1 Datasets

We analyze eight large-scale datasets, namely DBLP, LiveJournal, two portions of the Facebook network (denoted by Facebook — Rice University Undergraduate and Graduate), Amazon, and three biological networks denoted by HS, SC, and Fly. The collection encompasses different forms of entities and relationships originating from diverse domains.

The LiveJournal dataset consists of a snapshot of a large network of bloggers, previously explored by [17]. The snapshot includes 4,847,571 bloggers who explicitly declare their friendship links. Due to the massive size of this dataset, we consider two portions of it, which we obtain by starting at a random node and performing a breadth-first search from that node. The datasets, henceforth named LJ1 and LJ2, contain 500,000 nodes each. LJ1 and LJ2 contain 10,736,588 and 10,640,429 edges, respectively.

DBLP data is publicly collectible and our dataset consists of a snapshot taken in May 2009 of the on-line publications database site DBLP. The data include a collection of editions of publication venues (i.e., conferences and journals) in computer science. A pair of the 744,386 authors present in the dataset are linked if they have co-authored at least one paper in any of the venues.

Facebook — Rice University Undergraduate (Ugrad) and Graduate (Grad) are an anonymized portion of the Facebook network which include Rice University students, collected by crawling public friends lists on Facebook on May 17, 2008. They consist of two disjoint sets of 1220 undergraduate students and 503 graduate students, respectively. [97] present a detailed description of these datasets.

The Amazon dataset [87] is a product co-purchasing network from the on-line retailer Amazon.com. Each node represents a book, and an edge exists between two nodes if one was frequently purchased with the other. The network contains 270,347 nodes and 741,142 edges. For each book, Amazon.com reports up to 5 other items that were frequently purchased with the book.

Biological networks HS, SC, and Fly describe protein-protein interactions for *H. Sapiens* (human), *S. Cerevisiae* (a type of yeast), and *Drosophila* (a fruit fly species) [113], respectively. In these networks, a node represents a protein, and two

nodes are connected if scientific evidence of their interaction exists. HS contains 10,298 nodes and 54,655 edges, SC contains 5523 nodes and 82,656 edges, and Fly contains 15,326 nodes and 486,970 edges.

Annotated Communities

The networks we analyze contain annotations reflecting examples of communities that arise in these domains². Some of these sets are user-defined, i.e., users explicitly declare their participation in the community, while others reflect contextual information of the underlying process or organization, e.g., university department, protein function, product category, etc. Below we describe how we identify and clean the annotated communities for each dataset.

For the social networks, in LiveJournal, users explicitly declare their membership in zero or more communities created and administered by users. In DBLP, conferences where authors publish their research work reflect the community memberships. Finally, for Facebook — Rice University Undergraduate and Graduate, users who possess common academic attributes, such as department, major, or dormitory, form the communities. These attributes were obtained by matching Facebook names with student records from the university’s directory [97].

For each item in Amazon.com, the on-line store provides several product categories, such as “Photo Essays” or “Landscape Architecture Textbooks”. We identify a set of nodes possessing a common categorical label as a community.

For HS, SC, and Fly, a number of proteins (though not all) have annotations regarding one or more gene ontology IDs describing the known functions that the protein

²These communities, however, may not represent an unbiased sample of communities in these networks as other communities that are not annotated might also exist.

serves (e.g., metabolic regulation). We use these gene ontology values to identify the communities.

Because we define annotated communities by identifying sets of nodes with common labels, we sometimes encounter annotated communities containing multiple components. Considering disconnected communities would produce a less informative feature space, because some metrics achieve extreme values, such as infinity, for disconnected subgraphs, i.e., shortest paths, diameter, and so on. In these cases, rather than discarding the communities, we simply consider each component to be a separate community. Therefore, to capture the community information implicit in the annotations, we consider each connected component of the graph induced by a node set possessing a common label as an annotated community by itself.

Earlier experiments suggest that these structural features are not well represented in small communities, as the extracted features of these communities exhibit high correlation, i.e., the intrinsic dimensionality of their feature space is too low to represent the structural variability we want to observe. Therefore, it was our early decision to establish a cut-off point for the community size, below which the objects are too small to be representative. In the other extreme are the communities that are too large and sparse that make the features distorted in meaningless ways. Therefore, we filtered out small communities with fewer than 10 members and large communities with more than 1000 members.

Overall, we identified 29,955 annotated communities for LJ1, 39,598 for LJ2, 10,595 for DBLP, 24 for RICE-grad, 41 for RICE-ugrad, 9439 for Amazon, 64 for HS, 76 for SC, and 54 for Fly.

#	Feature	Description
1	n	Number of nodes
2	m	Number of edges
3	Diameter	Greatest distance between two nodes by traversing shortest paths
4	Edge Density	Ratio of m to the maximum possible number of edges
5	Conductance	Ratio of m to the sum of the total degrees of the n nodes, including edges to rest of the network
6	Transitivity	Ratio of the number of 3-node cycles (triangles) to the number of 2-hop paths (open triangles)
7	Triangle Density	Ratio of the number of 3-node cycles (triangles) to the number of possible node triples
8-11	Shortest Path	All pairs shortest paths The features are the three quartiles and the maximum.
12-15	Edge Betweenness	For each edge, fraction of all-pairs shortest paths that include that edge. The features are the three quartiles and the maximum.
16-20	Node Betweenness	For each node, fraction of all-pairs shortest paths that include that node The features are the minimum, the three quartiles, and the maximum.
21-25	α	For each non-member on the fringe of the community, number of members that this node is connected to. The features are the minimum, the three quartiles, and the maximum.
26-30	β	For each member, number of other members this node is connected to The features are the minimum, the three quartiles, and the maximum.
31	Treesum	Total number of spanning trees of the community graph, divided by the total number of spanning trees of a K_n -clique (computed using Kirchoff's matrix tree theorem [93])
32-36	Information Centrality	For each node, its Stephenson and Zelen's information centrality index [129] The features are the minimum, the three quartiles, and the maximum.

Table 2.1: List of features corresponding to measures of the subgraphs that communities induce.

2.3.2 Structural Classes and Feature Space

In this section we describe how to produce examples that constitute the structural classes and how to build the feature space for our learning framework. The process consists of two steps. First, we produce the examples by applying community detection algorithms, one for each class, to the network data. Second, we extract features by measuring a broad spectrum of properties of the subgraphs induced by communities. This latter step uses a set of examples consisting of the output produced in the previous step along with the

set of annotated communities.

Producing the Examples

To illustrate the study of classes representing intrinsically defined communities, we selected a collection of 10 community detection procedures. We applied these procedures to each of the nine network datasets to extract examples of subgraphs produced by these methods. We labeled examples with the identity of the community detection procedure that produced them. In total, for each network, we created 11 structural classes of communities: one class of extrinsically-defined communities, which comprises examples of annotated communities, and each of the other 10 classes corresponding to intrinsically-defined communities, which comprise examples extracted by each of the 10 community detection algorithms respectively. Figure 2.2 presents a graphical illustration of this process.

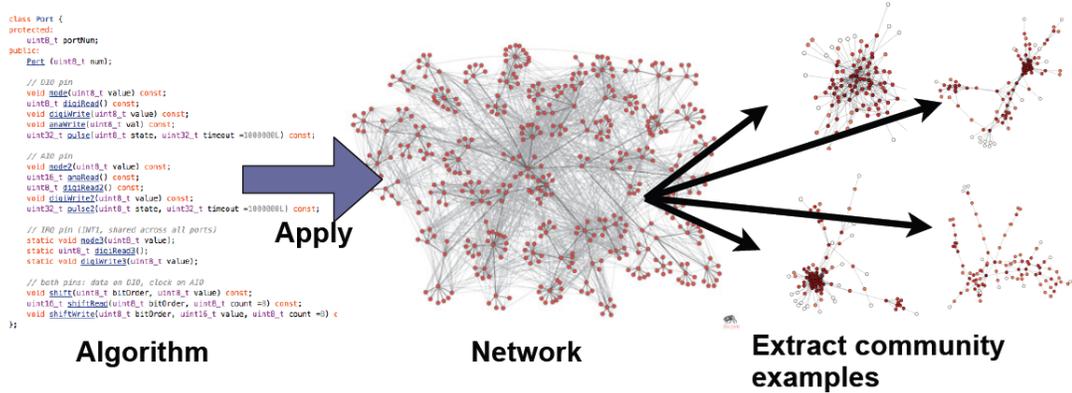


Figure 2.2: The process of extracting examples of communities. We apply a given algorithm to a network to extract examples of typical structures that it produces, i.e, the communities extracted as its output. The set of examples extracted is further annotated with the name of the algorithm that produced them.

Without any assumptions on the structures that the algorithms produce, we chose

our collection with the goal of including algorithms that are representative of strategies employed by a broad range of algorithms in the literature, purely based on their description. This description-based diversity is reflected in the categorization of [37]. Below we briefly describe the community detection procedures we consider, together with some examples of algorithms in the same description-based category according to [37].

1. **Breadth First Search (BFS):** To establish a baseline, we use breadth first search to extract sets that serve as examples of random connected communities. To create one BFS community of size k , we begin with a randomly selected node and perform a breadth first search from that node until we visit k elements.
2. **Random Walk 0 (RW0):** The central idea in many community detection algorithms is that random walks tend to concentrate within a community [117, 140]. To create communities of size k , we begin with a random node and perform a uniformly-random walk from that node until k different nodes are visited. This method represents a way to extract a connected community that encodes little structure and serves as another baseline procedure.
3. **Random Walk 0.15 (RW15):** This is similar to the preceding method with the twist that at each step we restart the walk from the starting node with 0.15 probability. RW15 concentrates the random walk distribution around a center, thereby forming more compact sets, whereas RW0 communities tend to spread out.
4. (α, β) **(AB):** An (α, β) -community, for $\alpha < \beta$, requires every member of the community to be connected to at least β other members while non-members have at most α links to the community [96]. This definition allows for overlapping communities whose out-links may outnumber the in-links. To produce an (α, β) -community of size k , we produce a BFS community of size k and then apply a limited number of sequential node swaps that aim at making the set an approx-

imate or exact (α, β) -community. In each step we remove the community node with the fewest member neighbors and add the fringe node with the most member neighbors. The $\alpha - \beta$ algorithm can be considered an example of the class of algorithms that search for sets of nodes that meet some specific structural criteria. Other algorithms that find communities meeting specific structural criteria are Clique Percolation [112], S-Plexes Enumeration [78], Bi-Clique [86], and EAGLE [126].

5. **Link Communities (LC):** In contrast with the majority of the available algorithms, Link Communities [16] aims at addressing the overlapping and hierarchical nature of community structure by treating communities as groups of links rather than nodes, defining a similarity function on edges based on their shared node neighborhoods, and then using hierarchical single-linkage clustering to identify communities of edges. We extract examples of this structure by applying a standard implementation of this algorithm to our networks. We include Link Communities as a representative of algorithms that cluster links, rather than nodes. Other algorithms in this category include the Link Modularity [50] and Link Maximum Likelihood [21] algorithms.
6. **Infomap (IM):** The Infomap algorithm [121] views the problem of finding communities as akin to the problem of a map-maker deciding on a level of granularity. The communities and the nodes therein have names. A random walk in the network is described by appending the community name followed by the name of nodes visited while in the community to a transcript. The goal is to find the community structure that minimizes the expected length of the description. Intuitively, such a structure would cause random walks to rarely escape communities. Infomap is a representative of algorithms that define communities as a group of nodes that are closer to each other than to nodes outside the community with

respect to the number of hops between two nodes. Other examples of these approaches are Walktrap [117] and DOCS [139].

7. **Louvain:** The Louvain method [30] is a popular method for greedy modularity optimization. The algorithm consists of iteratively aggregating nodes into communities whenever this move locally improves modularity. The process outputs communities when no further merge produces a significant gain in modularity. The Louvain method is a classic example of a community detection method that optimizes for internal community density. Other such algorithms include the MetaFac [90], Variational Bayes [67], $LA \rightarrow IS^2$ [24], and Local Density [122] algorithms.
8. **Newman-Clauset-Moore (Newman):** This method is another example of greedy modularity maximization [36]. Unlike the Louvain method, which considers merges that locally improve modularity, Newman-Clauset-Moore identifies a hierarchical community structure from which communities are extracted by cutting the dendrogram that reflects the hierarchy at the level that maximizes a global value of modularity. As with the Louvain method above, the Newman-Clauset-Moore method for modularity optimization is an example of an algorithm that attempts to find communities with high internal density.
9. **Markov Clustering Algorithm (MCL):** MCL [41] is a random-walk-based method. It consists of two alternating steps. It begins with the random-walk matrix of a graph (the normalized adjacency matrix). The first step, namely “expansion”, squares this matrix; this corresponds to computing the flow between clusters. The second step called “inflation”, squares each element of the matrix individually, and then re-normalizes; this step corresponds to increasing the strength of intra-community ties. This process converges to a stationary matrix with several connected components, which the algorithm outputs as the commu-

nities. MCL is a member of the class of community detection algorithms that also includes Infomap.

10. **Metis:** Metis [72] is a graph partitioning method which is a variation of the Kernighan-Lin algorithm [74]. Metis partitions a node-weighted network into a specified number of equal weight sets while minimizing the number of edges between the sets. Here we used a version of Metis we adapted for finding high-conductance sets. The original Metis algorithm can be considered as representative of the class of ‘bridge detection’ community detection algorithms, which identify communities by removing bridges between dense sections of the network. Other algorithms in this class include Edge Betweenness [57], CONGA [61], L-Shell [18], and Internal-External Degree [82]. Additionally, because we modify the algorithm to identify high-conductance sets, it might also be considered as part of the class of algorithms that optimize for internal community density.

As we are interested in the structural signature produced by the different methods, we run the parametrized algorithms multiple times for each network, randomly varying the parameter settings. Some of the procedures are non-deterministic and generate different communities at each run, even if the same parameters are used. To preserve uniformity with the set of annotated communities, in the process of generating examples, we discard communities of size less than 10 or greater than 1000, as we are interested in the structure of reasonably sized communities. (See Subsection 2.3.1 for the details.) We also filtered out communities that contain multiple components, which are rarely extracted by the methods we used. The number of examples extracted varies among the procedures.

However, we inherit sensitivity to class imbalance from the methods we use for class separability. The effects of class imbalance on the performance of machine learning

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
BFS	9.9	5.3	0.9	1.0	0.2	0.2	0.2	0.2	0.1
RW0	9.9	5.3	0.9	1.0	0.2	0.2	0.1	0.1	0.1
RW15	9.9	5.3	0.9	1.0	0.2	0.2	0.2	0.1	0.1
AB	9.9	5.3	0.9	1.0	0.2	0.03	0.1	0.1	0.1
IM	2.7	7.3	0.1	0.4	1.1	0.04	0.06	0.3	0.1
LC	4.1	0.5	0.2	0.4	0.2	0.004	0.01	0.3	0.3
Louv.	9.1	12.5	4.8	10.0	9.0	2.1	1.2	20.4	16.4
Newm.	16.8	24.8	5.3	19.8	9.0	0.8	0.9	1.0	1.0
MCL	1.7	4.1	0.03	0.06	0.9	0.01	0.02	0.01	0.01
Metis	4.3	10.4	0.5	1.7	0.7	0.1	0.2	0.5	0.3
Annot.	10.6	6.2	1.4	1.3	3.2	0.1	0.2	0.2	0.1

Table 2.2: Average number of communities from each class that each node belongs to, after sampling.

methods is the subject of an extensive literature in machine learning. For our experiments, we have applied standard methods in the literature for minimizing the problem. For an overview, we refer the reader to the work of Chawla [34]. More specifically, we under-sample the large classes and to a lesser extent over-sample small classes to reduce this source of bias. Naturally, under-sampling can result in some portions of a network (particularly a large network) not being represented in our final set of communities. Table 2.2 contains the average number of communities from each class that each node belongs to, after under- or over-sampling. For the small networks (Grad, Ugrad, HS, SC, and Fly), we sample 100 communities from each class, and for the larger networks, we sample 1000 communities from each class. If a community detection method tends to produce a large number of small communities, and we then under-sample this set, we will see small average values, whereas if a method produces a small number of large communities that we over-sample, we will see higher average values.

Our algorithm selection has the purpose of illustrating the applicability of our framework. The approach, however, is not limited to the list we consider. Our method scales to a large number of classes, and a collection of classes should include enough informa-

tion to reflect the analysis intended. As discussed in the next section, a pair of classes may be highly correlated to each other (e.g., RW0 and RW15). As a result, they may split the predictions in such a way as to obfuscate the interpretation of the outcome. To avoid this pitfall, in Section 2.4.2 we consider an inter-class correlation analysis to assess the independence of an algorithm collection.

Feature Extraction

In the feature extraction phase, we measure the subgraphs induced by the communities produced in the previous step and those induced by annotated communities. We use a large spectrum of measurements that cover many properties of both the internal link structure and the external interaction of the community with the rest of the network. Each measurement corresponds to a dimension of our feature space. Table 2.1 lists the features and describes their corresponding measures.

Most of the features can be understood from their table description. The feature Information Centrality, however, deserves further explanation. This measure captures a node’s degree of centrality as a function of how fast its information can sequentially reach every other node in the network. For a given node, the information centrality computes a harmonic mean of the amount of “signal” that a node receives from other nodes. A signal between two nodes corresponds to a path between them, which varies according to the “noise”, instantiated here as the path length [129].

By measuring the structural properties described in Table 2.1 for each example of a community derived in the previous phase, we obtain 11 classes of labeled examples in feature space, which constitute the input in our framework.

2.4 Framework and Application

In this section we describe our class separability framework and illustrate its applicability with an experimental application using the data we processed through the steps described in the previous section.

2.4.1 Class Separability Measures

Methods for measuring class separability are popular in machine learning for guiding feature selection analysis. Accordingly, effective feature sets for classification tasks are the ones that simultaneously lead to high inter-class and low intra-class variability [135]. Methods of class separability allow for a rigorous analysis of independence among classes. Unfortunately, many of these methods are computationally demanding or dependent on assumptions which are often mismatched with applications [53].

In this work, we frame the research question of discriminating the structure of different communities as a class separability problem. The separability of structural classes of communities provides information on whether different communities come from the same (or fundamentally different) distributions of feature values. This analysis is informative of the extent to which different algorithms produce structural differences and the extent to which community detection algorithms succeed in producing sets that resemble annotated communities.

To measure class separability, we first use the J3 metric [135], which is based on within-class and between-class scatter matrices. The J3 criterion calculates two scatter matrices. The between-class scatter matrix S_m is simply the global covariance matrix, and the within-class scatter matrix S_w is the average of the covariance matrices for each

class, weighted by the size of that class. The J3 score is then the trace of $|S_w^{-1}S_m|$. A high J3 score indicates that within-class covariance is low, and global covariance is high. The last row of Table 2.3 contains the ratio of the global J3 value for each network to a baseline J3 value that is obtained by measuring the separability of the same data, but with the class labels shuffled. A value of 1 indicates separability close to a random re-labeling, and higher values indicate greater separability. These values demonstrate that the classes in each network are separable to some extent, even when evaluated through a fairly simple measure.

While the J3 criterion is useful for gaining a coarse overview of class separability, it is a global measure, which tells us little about the separability behavior of each class. Aiming at achieving more fine-grained separability information while including all the classes simultaneously and preserving computational scalability, we propose the use of the cross-validation performance of existing probabilistic multi-class supervised classifiers as a measure of class separability. To our definition, classes are separable to the extent that a classifier can correctly distinguish their structure by exhibiting an accurate classification. More specifically, we employ two techniques, one parametric, namely Support Vector Machines (SVM) [136], and one nonparametric, namely k -Nearest-Neighbors (kNN) [15], to confirm each other’s outcomes while ruling out variability due to the specifics of each algorithm. We select hyperparameters in both cases via grid search using the performance of 10-fold cross-validation as the objective function.

The primary goal is to gain insight into whether the classes are separable in the feature space defined. This will tell us the extent to which the community detection algorithms produce structural profiles that are specific to each algorithm. Using a slight variation of the same approach, we can determine which algorithms produce outputs that most closely resemble the annotated communities.

Our approach to measure the separability of the structural classes of algorithms using probabilistic multi-class supervised classifiers is as follows. We measure class separability using the performance of a 3-fold cross-validation. For each network, we train a multi-class classifier on a set containing two-thirds of the examples, which are selected at random, and then evaluate the performance of the model on the remaining third, which constitute the test set. We perform 3 rounds of this process and average the outcomes. For each element in a test set, the probabilistic SVM or kNN model outputs a probability mass vector indicating the probability that each data point belongs to each class. Figure 2.3 illustrates the cross-validation phase as applied to one element in the test set.

Using the structural classes computed via the steps described in the preceding section, we perform this cross-validation on a dataset containing all eleven classes: ten classes corresponding to the structure that the algorithms produce, and one class corresponding to the structure of annotated communities. We first observe that the experiments suffer little variability between the two classifiers. Figure 2.4 presents the analysis of the outcome produced by the SVM-based method applied to the DBLP network. In the picture, we show a bar graph of the distribution of probability mass for each class derived from the network DBLP. This graph visually demonstrates that the bulk of the probability mass from each class was correctly classified.

Table 2.3 contains a summary of results for all networks. Each entry in the table represents the fraction of probability mass from that class that was correctly assigned. When a value appears in parentheses, this indicates that more of the probability mass was assigned to some other class. As this table shows, only 17 out of 99 network-class pairs failed to have a plurality of the probability mass correctly classified. Newman Modularity is frequently misclassified; however, it is a small class in all networks, especially in the smaller ones (e.g., on network SC, Newman Modularity found only 3

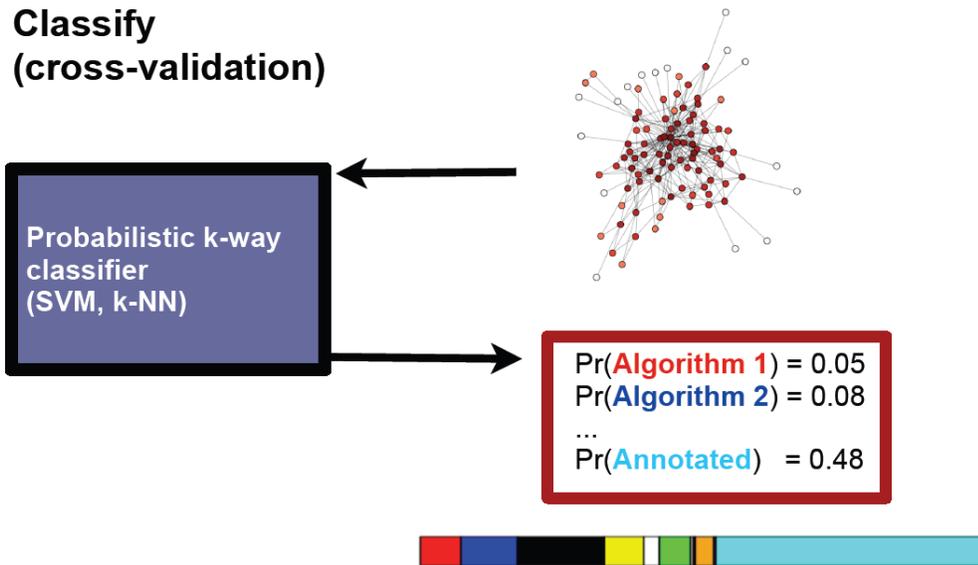


Figure 2.3: Probabilistic multi-class supervised classification of an element in the test set in the cross-validation phase of the method to assess the separability of structural classes of communities.

communities of size between 10 and 1000). In the case of annotated communities a plurality of their corresponding classes tend to be correctly classified, with the exception of network Fly. Figure 2.4 serves as a visual reference of network DBLP, whose classes have a global separability score of 1.58. The other networks exhibit similar distributions, with the exception of network Fly, whose classes are less well separated.

The previous experiment shows that annotated communities tend to form their own, separable class that is significantly distinct from all other classes. However, a question of interest to the design and application of community detection procedures is which algorithms output communities bearing the closest structural resemblance to the annotated communities. To answer this question we perform a slight variation of the classification task previously described. We train a classifier on the ten classes corresponding to the community detection algorithms and leave the class of annotated communities out of

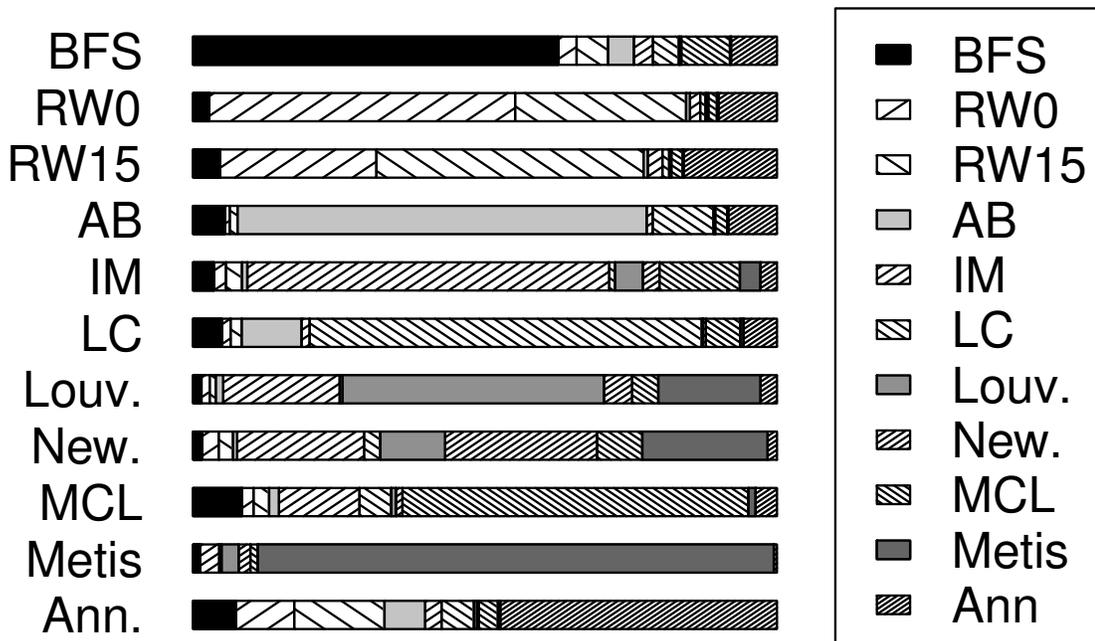


Figure 2.4: Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the 11 classes.

the training set. The goal of this experiment is to evaluate to which class of intrinsically defined communities the annotated examples of the test set are classified.

Figure 2.5 shows the distribution of probability mass of the annotated communities classified into the different classes corresponding to community detection algorithms. The structure that the random-walk and BFS procedures produce is clearly the most similar to that of the annotated communities. For 7 of the 9 networks, a plurality of the probability mass from the annotated communities was assigned to either RW15 or RW0, followed by BFS. Due to the high similarity between the two random-walk classes, the classifier confuses these two as shown in Figure 2.4. The exceptions to this trend are networks Grad and Fly. For Grad, the annotated communities' probability is spread

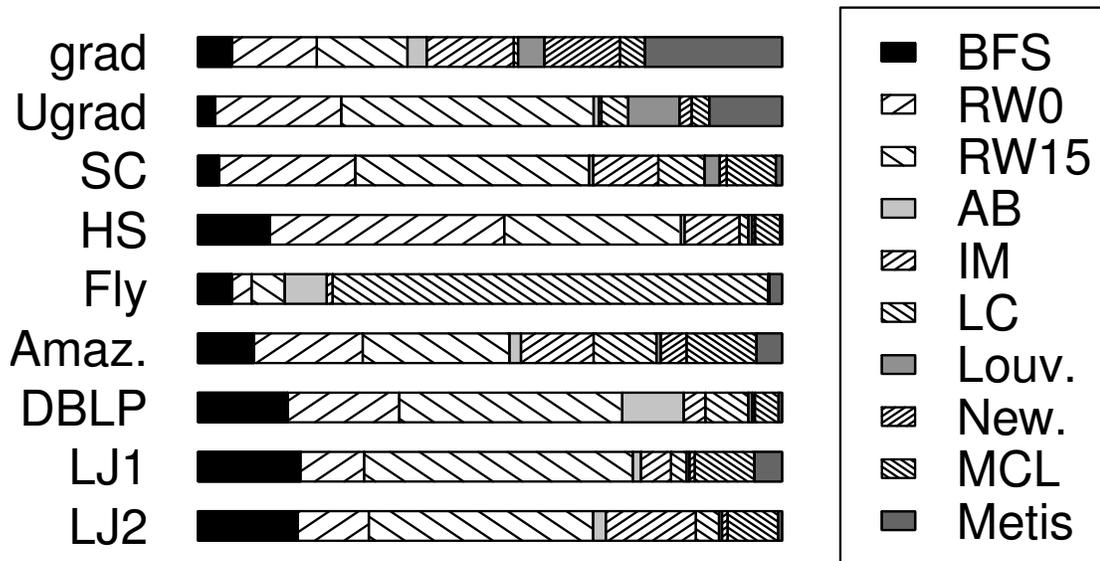


Figure 2.5: Distribution of probability mass resulting from the SVM, classification of annotated communities.

across many classes, with Metis receiving the plurality of the mass. In the Fly network, the greatest share of the mass of annotated communities is assigned to LC. These exceptions are associated with small network datasets, therefore the variability could be due to small population sample size.

Given the diverse nature of these networks, it is perhaps surprising that in virtually all domains the random-walk and BFS communities bear the closest structural resemblance to the annotated communities. Even more astonishing is the fact that the structure of annotated communities is closer to that of the baseline procedures than to that of more structured approaches.

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
BFS	60%	88%	73%	70%	(40%)	63%	55%	86%	81%
RW0	44%	55%	43%	(39%)	(27%)	52%	43%	61%	63%
RW15	40%	(29%)	44%	42%	34%	46%	39%	57%	57%
AB	83%	91%	90%	71%	60%	70%	74%	90%	89%
IM	27%	(23%)	72%	73%	(2%)	62%	51%	82%	66%
LC	68%	96%	83%	85%	83%	67%	56%	90%	89%
Louv.	24%	(3%)	49%	(1%)	(0%)	45%	58%	38%	49%
Newm.	(14%)	(25%)	(15%)	(0%)	90%	26%	39%	45%	56%
MCL	19%	(22%)	57%	28%	(34%)	59%	46%	80%	74%
Metis	61%	73%	81%	90%	(42%)	88%	66%	92%	86%
Annot.	37%	33%	50%	46%	(8%)	47%	40%	72%	71%
Global	1.44	2.11	1.7	1.81	1.35	1.58	1.38	1.68	1.76

Table 2.3: Percentage of the probability mass of classification of elements in the test set into the correct class, using SVM, for all networks. The last row presents global separability ration between the scatter matrices J3 scores measuring the separability of classes to a baseline consisting of the J3 scores of the same data with shuffled class labels. If a value is in parentheses, this indicates that a plurality of the probability mass from that class was assigned to some other class.

2.4.2 Class Selection Method

The feature space generated by the data forms a reference system for analyzing community structure from the viewpoint of the diversity of structures that come from different notions of communities. When we employ class separability measures, it is particularly important to ensure that the classes considered are independent, and not redundant. For example, when identifying which type of community detection method produces structure that most resembles that of the annotated communities, we saw that one of the two random walk classes tended to be assigned the bulk of the probability mass of the annotated communities on most networks. However, these two classes are remarkably similar. When we present a classifier with two almost indistinguishable classes, the classifier splits the examples' probability masses equally between the classes.

	BFS	RW0	RW15	AB	IM	LC	Louv.	Newm.	MCL	Metis	Ann.
BFS	1.00	1.14	1.14	1.16	1.22	1.19	1.40	1.22	1.15	1.57	1.13
RW0	1.14	1.00	1.04	1.33	1.40	1.33	1.52	1.31	1.28	2.01	1.14
RW15	1.14	1.04	1.00	1.27	1.33	1.36	1.54	1.35	1.28	1.94	1.12
AB	1.16	1.33	1.27	1.00	1.15	1.08	1.22	1.18	1.15	1.25	1.16
IM	1.22	1.40	1.33	1.15	1.00	1.36	1.39	1.15	1.13	1.14	1.12
LC	1.19	1.33	1.36	1.08	1.36	1.00	1.65	1.26	1.09	1.27	1.09
Louv.	1.40	1.52	1.54	1.22	1.39	1.65	1.00	1.06	1.63	1.19	1.26
Newm.	1.22	1.31	1.35	1.18	1.15	1.26	1.06	1.00	1.24	1.13	1.12
MCL	1.15	1.28	1.28	1.15	1.13	1.09	1.63	1.24	1.00	1.22	1.06
Metis	1.57	2.01	1.94	1.25	1.14	1.27	1.19	1.13	1.22	1.00	1.27
Ann.	1.13	1.14	1.12	1.16	1.12	1.09	1.26	1.12	1.06	1.27	1.00

Table 2.4: Pairwise separability for classes in network Amazon, calculated using scatter matrices. Ratios are measured relative to the baseline value obtained by shuffling class labels.

To provide a rigorous method of class selection, we propose the application of a pairwise Scatter Matrix measurement [135] for analyzing class similarity and determine which classes from a collection are redundant or independent. We again use the J3 criterion that we originally used to measure overall class separability within a network, but, this time, we measure pairwise separability by considering each pair of classes separately. These values give us insight into whether certain pairs of classes are redundant, correlated, or overlap significantly (e.g., the two methods for maximizing modularity). Values for network Amazon are shown in Table 2.4. In these tables, values are presented as a ratio of the J3 score of a pair of classes to the J3 score of a baseline separability value, which is obtained by shuffling the examples’ class labels. As we see in Table 2.4, no two classes are completely identical, though the two random walk classes are quite similar, as are the two modularity-based classes. We see this pattern repeated consistently in other networks, though occasionally other pairs of classes are similar (e.g., in network Amazon, the MCL and Annotated classes have a fairly low separability score, but this pattern does not occur in other networks).

We now repeat our earlier experiments after applying the results obtained by our class selection method. We learned from Table 2.4 that some pairs of classes, namely the two random walk and the two modularity classes, are somewhat redundant. We thus merge each of these pairs of classes into one single class, and again perform the above experiments³.

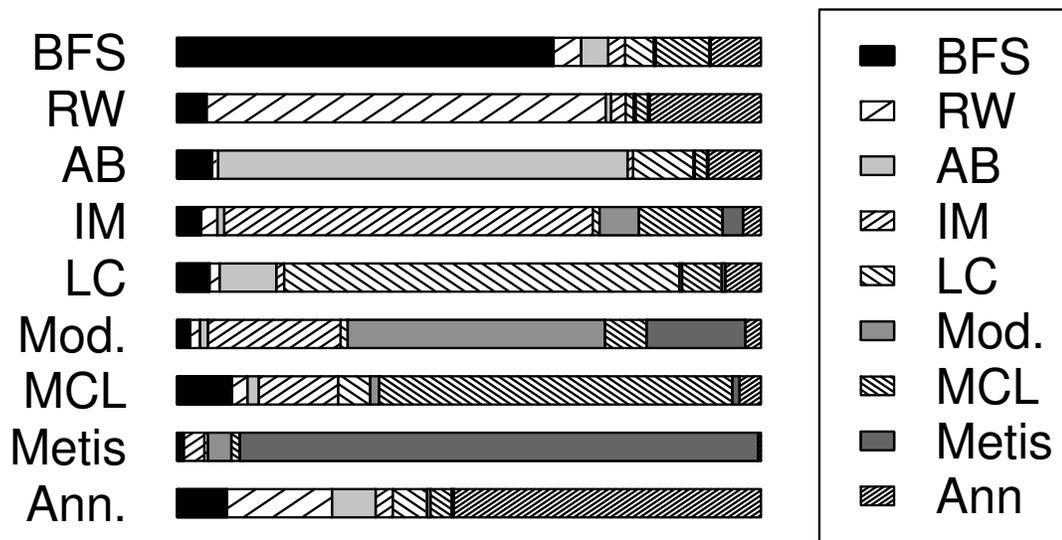


Figure 2.6: Distribution of probability mass resulting from the SVM on network DBLP, cross validation on the 9 classes after merging redundant classes.

Figure 2.6 contains the distribution of probability mass obtained by applying the SVM to all 9 classes on network DBLP, and Table 2.4.2 contains the percentage of probability mass from each class in each network that was correctly classified. We saw in our first experiment that the classifiers tended to confuse the two random walk classes with one another; e.g., elements from RW15 were frequently misclassified as belonging

³Here we could have preserved one representative of each group of similar structural classes instead of merging the similar classes. Whether to use one approach or the other depends on the intended application.

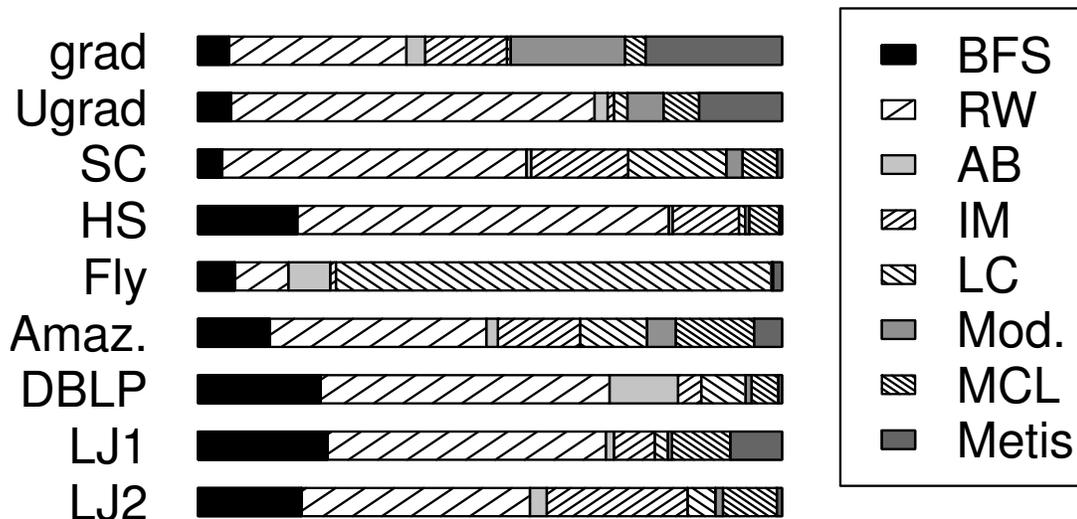


Figure 2.7: Distribution of probability mass resulting from the SVM, classification of annotated communities on the 9 classes after merging redundant classes.

to class RW0. As expected, we typically see that the single random walk class and single modularity class are substantially more consistent than either of their sub-classes alone. This effect is particularly pronounced on certain networks, such as Undergrad, where over 90% of the probability mass from the large random walk class is correctly classified (as opposed to values of 55.1% and 61.4% for RW0 and RW15 separately).

Figure 2.7 contains the distribution of probability mass obtained when classifying the annotated communities into one of the 8 algorithm classes. As predicted by our class selection method, the merged classes receive all the mass assigned to the corresponding separate classes. Moreover, previously, for network Grad, we saw that a plurality of the mass of the annotated communities was previously assigned to the Metis class; however, now that we combine the two random walk classes, we see that the single random walk

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
BFS	70%	90%	74%	69%	(39%)	65%	60%	85%	81%
RW	74%	91%	83%	83%	61%	68%	65%	91%	88%
AB	79%	93%	92%	77%	54%	70%	76%	91%	89%
IM	4%	(15%)	76%	72%	(2%)	63%	53%	83%	67%
LC	62%	95%	87%	81%	84%	68%	58%	91%	88%
Modul.	35%	(25%)	39%	33%	(34%)	44%	52%	53%	62%
MCL	25%	(21%)	59%	33%	(25%)	61%	47%	81%	74%
Metis	67%	74%	73%	83%	(41%)	89%	70%	93%	85%
Annot.	27%	40%	52%	47%	(7%)	53%	45%	74%	74%

Table 2.5: Percentage of the probability mass of classification of elements in the test set into the correct class, using SVM, for all networks, after merging classes.

class receives more probability mass than either of the two random walk classes alone, and thus more than the Metis class.

2.4.3 Class Consistency Across Networks

We now turn our attention to the problem of analyzing class consistency across networks. In our previous experiments, we analyzed each network separately and saw that the 11 classes of communities all tended to be fairly consistent (e.g., elements from class BFS in network Grad shared important structural features with one another). In the next experiment, rather than studying each network in isolation, we simultaneously examine classes across all networks.

We perform 9 experiments, each corresponding to a network N . In each experiment, we create a training set containing elements from all 11 community classes in network N , labeled with their community type (e.g., BFS, Louvain, etc.). We create a test set containing elements from the 11 community classes of the other 8 networks. In this test set, each element is labeled with its community type, but we do not identify the network

Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	60.3	34.7	42.6	36.4	25.2	46.2	63.4	37.1	48.5
RW0	44.5	26.2	30.6	21.0	20.8	38.8	30.8	22.8	20.9
RW0.15	41.0	23.6	24.8	16.0	20.1	48.4	26.4	19.8	24.9
AB	58.4	41.7	63.6	44.5	9.9	20.0	55.1	64.3	62.8
InfoMap	4.7	5.5	22.0	42.5	23.8	6.0	25.1	34.0	33.0
LinkCom	6.6	14.6	29.4	33.8	15.6	14.8	27.7	23.2	28.4
Louvain	2.1	43.5	8.0	25.0	41.0	1.1	7.8	20.9	9.1
Newman	6.3	4.9	8.4	23.2	3.6	9.9	5.4	34.2	53.6
MCL	7.3	13.9	23.7	20.3	27.1	13.6	18.9	40.4	7.3
Metis	20.2	11.9	31.0	12.1	34.8	3.5	5.9	14.8	14.0
Ann.	5.8	19.7	23.8	10.8	20.6	31.0	17.3	36.1	34.2

Table 2.6: The percentage of probability mass from each class that was correctly classified. The column titles indicate the networks on which the classifiers were trained. The values in row M , column N indicate the average percentage of probability mass from each class over all networks except N that was correctly classified.

from which it came.

The purpose of this experiment is to determine whether communities generated by a specific method tend to share structural features, even across networks. For example, we wish to learn whether an SVM trained only on representatives from network Grad can correctly classify communities from other networks: does a BFS community from network DBLP resemble a BFS community from network Grad?

Table 2.6 contains the results of this experiment. The element in row N , column C contains the average percentage of probability mass from class C that was correctly classified when the classifier was trained on elements from network N . For example, when the SVM was trained on elements from network Grad, an average of 60.3% of the probability mass from class BFS in the other networks was correctly classified as BFS.

We see that some classes, particularly BFS, RW0, RW0.15, and AB tend to have a great deal of consistency across networks. The other classes tend to have a less distinct signature, often performing worse than random.

2.5 Structural Tendencies of Communities

As we have seen in the preceding experiment, each community detection algorithm extracts a distinct structure, which our method is able to separate when projected onto the feature space we define. In this section, we are concerned with identifying the ways in which the algorithms produce these distinguishable biases by finding which properties exhibit the highest degree of between-class variability. We are also concerned with identifying the properties that are the most discriminative to distinguish between annotated communities and the synthetic data produced by the algorithms. Finally, a dividend of our approach is the ability to organize a collection of algorithms by grouping those that exhibit similar behaviors.

To address this question we use the Correlation-based Feature Selection (CFS) algorithm [66] to identify subsets of the most discriminative features for each network. CFS is intended to identify a set of features that are well correlated with the class label and poorly correlated with each other. For a given subset S containing k features, CFS defines a merit function M_S .

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \quad (2.1)$$

In this equation, $\overline{r_{cf}}$ represents the mean correlation between each individual feature f in S with the class label c , and $\overline{r_{ff}}$ represents the mean correlation between pairs of features in K . Intuitively, this function gives a high score to sets of features that are highly predictive of the class and are not redundant with one another. Using this function, one can rank all subsets of features, but this would be inefficient in most cases. CFS begins with no elements in the feature set, and it then employs a hill-climbing algorithm to search the space of feature subsets. The algorithm includes the ability to backtrack up to 5 times per iteration to search for a subset S with a greater value of M_S .

Network	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
Num. of Features Selected	6	7	10	5	6	10	8	12	11
Rank	Feature								
1	Conductance	1	1	1	1		1	1	1
1	Diameter		1	1	1	1	1	1	1
3	Info Centrality*	2	2	3	1	1	2	1	2
4	Node Betweenness*			2	2		2	1	5
5	Shortest Path*			1		3	2	1	1
6	β^*	1	1	1			2	1	1
7	α^*	1		1		1		2	1
Other features**		#6	#4, #7						

Table 2.7: Summary of the feature selection results. Features are ranked in order of their frequency in the selection list over the networks. (* reporting how many quartiles of the property were selected. ** feature number according to Table 2.1.

Table 2.7 lists the features selected by CFS for each network ranked in order of the frequency with which they appear in the selection over the networks. The table lists the most frequent features, or, for those properties calculated with quartiles, sets of features. The entries for row “Features” and column “Network” that contain the value 1 indicate the presence of that feature in the feature selection process applied to the data from that particular network, whereas empty cells indicate the absence thereof. Integers larger than 1 can be found in some of the entries and indicate the number of quartiles from that feature that were selected by CFS. In nearly every network, conductance, diameter, information centrality, and node betweenness were the most discriminative features.

Surprisingly, in several cases, multiple quartiles of a feature appear: for example, Fly has 3 path length quartiles, and LJ1 and LJ2 each contain all 5 node betweenness features. We had expected that different quartiles of the same feature would be highly correlated to each other, and therefore they would be unlikely to co-occur among the features selected by CFS. Instead, these results suggest that varying the choice of community detection algorithm results in fine-grained variation in the distribution of such

	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2
All	62.9%	86%	82.2%	80.9%	93.6%	81.3%	65.3%	89.1%	88.5%
Selection	61.5%	84.7%	85.1%	81%	90.6%	79.4%	63.0%	78.8%	76%

Table 2.8: k -Nearest-Neighbors classification performance using both the full set of features and the subset of the most discriminative features selected by CFS.

features.

To assess the effectiveness of the features CFS found, Table 2.8 presents for all networks the classification performance of the k NN cross validation using both the full set of features and the subset of features found by CFS. We see that in most cases, there is very little loss in accuracy. We observe a similar qualitative outcome for the SVM cross validation. In the table, the largest drops happened for LJ1 and LJ2 and reduced the accuracy by less than 15%. Being nearly as discriminative as the full set, a reduced set containing a handful of features retains the relevant information needed to analyze the bias produced by different algorithms.

To identify the properties that are the most discriminative to distinguish between annotated communities and the synthetic data produced by the algorithms, we next perform an experiment similar to the preceding. However, instead of considering separate classes of communities that the algorithms produce, we merge all implicitly defined classes into a single class. We consider the set of annotated communities to be one class, and the set of communities extracted by all algorithms to be another single class. In this experiment, we wish to identify those features that are most useful for distinguishing between explicitly and implicitly defined communities. As before, we use the Correlation-based Feature Selection Method to identify useful features. Table 2.9 contains the results of this experiment. Again, we see that conductance, node betweenness, and information centrality are particularly important.

Network	Grad	Ugrad	HS	SC	Fly	DBLP	Amaz	LJ1	LJ2	
Num. of Features Selected	6	5	3	6	5	7	6	4	5	
Rank	Feature									
1	Node Betweenness*	2	3	1	2	2		2	2	3
2	Conductance			1	1	1	1	1	1	1
3	Info Centrality*	1	2	1	3	2				
4	α^*						5	1	1	
5	Edge Betweenness*	2						1		
Other features**		#7					#9	#4		#3

Table 2.9: Summary of the feature selection results when classifying implicitly and explicitly defined communities. Features are ranked in order of their frequency in the selection list over the networks. (* reporting how many quartiles of the property were selected. ** feature number according to Table 2.1.)

Finally, to organize a collection of algorithms by grouping those that exhibit similar behaviors, we use the sets of the most discriminative features found in Table 2.7 to study which tendencies in feature values are associated with which algorithms. To this end, we conducted a range analysis which distinguishes the different algorithms according to the value of their features. In the interest of space, we summarize the qualitative outcome of this experiment in Figure 2.8. The entries correspond to the bias produced by each of the algorithms, considering all networks. Features take on a varying range of values across different networks. Thus, to label the magnitude of features, we compute the mean value of each class and compute a global median of these averages over all classes. The averages occurring between the 33rd and 67th percentile constitute the *medium* denomination; whereas those below the 33rd and above the 67th constitute *low* and *high*, respectively. Finally, we count how many times each feature produced each of the denominations across all the networks. From this count, we calculate three times the number of networks on which the feature had a high score on that class plus two times the number of networks on which the feature had a medium score on that class plus the number of networks on which the feature had a low score on that class, and present these

values in Figure 2.8.

Using this analysis, we are able to group algorithms with similar behavior. For example, the random-walk procedures produce the same structural bias. The same holds for Louvain and Newman; and AB and LC. Our method identified these similarities without any prior knowledge about the similar goals shared by these algorithms. Metis and IM differ only in behavior of the node betweenness feature. The profile of annotated communities is close to that of random-walk procedures, with a few nuances. Annotated communities exhibit medium conductance whereas RW0 and RW15 extract low conductance sets. In addition, the diameter of annotated communities was measured as high for four of the networks, medium for one of them, and low for the remaining four. This contrasts with RW0 and RW15, which produce sets with high diameter. Nevertheless, the similarity due to other features explains the ways in which annotated communities resemble the output of random-walk-based algorithms.

2.6 Network Consistency

Our previous experiments have considered various problems of distinguishing between communities generated through different methods. We saw that many classes of communities tended to have a strong ‘signature,’ both within and even across networks; for example, communities generated by a BFS algorithm were fundamentally different from communities generated by the Louvain method. In the next three experiments, instead of analyzing whether communities generated by different methods have different structures, we ask whether communities from different networks are distinguishable from one another.

In the first of these experiments, we consider each of the 11 methods of defining

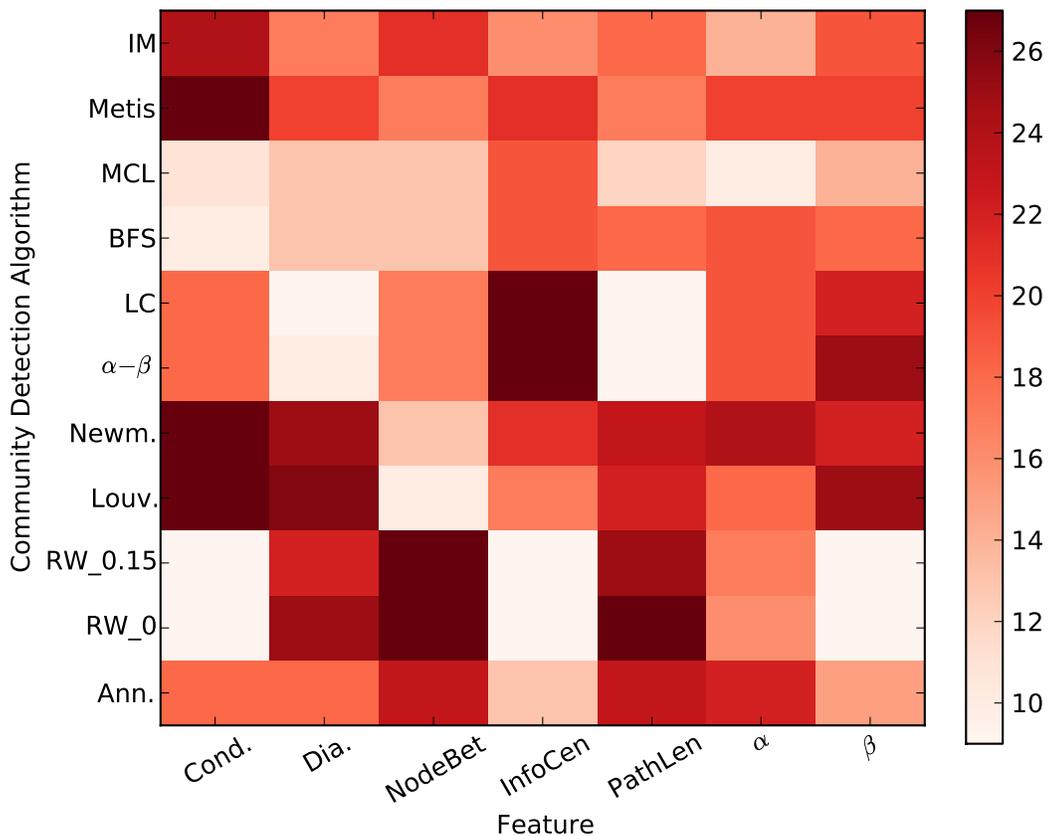


Figure 2.8: Tendency of algorithms with respect to various features. Scores are calculated by three times the number of networks on which the feature had a high score on that class plus two times the number of networks on which the feature had a medium score on that class plus the number of networks on which the feature had a low score on that class.

communities separately (e.g., we perform a different experiment for each of BFS, Louvain, etc.). In each experiment, we train an SVM classifier on a set containing communities from one specific class from all 9 networks, where each community is labeled with the network from which it came. We then evaluate this classifier on other communities from the same class, again from all 9 networks, also labeled with their networks of origin. For example, in our first experiment, both the training and test sets contain elements from Grad BFS, Undergrad BFS, DBLP BFS, and so on, labeled as Grad, Undergrad, or

Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	68.9%	67.4%	36.6%	25.0%	61.3%	78.6%	71.0%	47.3%	43.3%
RW0	67.5%	74.2%	45.1%	49.5%	68.2%	82.8%	74.1%	55.4%	51.0%
RW0.15	69.3%	76.2%	47.6%	50.1%	77.2%	83.7%	76.0%	55.5%	49.2%
AB	69.8%	55.1%	30.5%	31.2%	72.0%	82.1%	70.9%	43.4%	40.4%
InfoMap	33.2%	87.8%	47.6%	51.9%	54.7%	82.1%	78.3%	62.3%	57.1%
LinkCom	38.8%	94.2%	72.9%	66.7%	74.6%	81.1%	78.1%	54.1%	53.5%
Louvain	50.3%	71.9%	72.4%	46.4%	2.6%	90.1%	88.3%	39.9%	48.5%
Newman	0.2%	0.1%	0.3%	0.1%	0.1%	94.8%	21.5%	15.2%	60.8%
MCL	20.0%	48.1%	49.9%	22.3%	22.4%	80.6%	72.5%	53.0%	55.8%
Metis	92.3%	98.1%	88.5%	83.9%	81.5%	98.2%	98.0%	92.0%	92.1%
Ann.	60.4%	52.8%	22.4%	27.3%	38.3%	91.8%	85.7%	44.5%	45.8%

Table 2.10: The percentage of probability mass from each network that was correctly classified as belonging to that network. The row titles indicate the class that the classifier was trained on.

DBLP, respectively.

Table 2.10 contains the results of this experiment. The element in row C , column N contains the percentage of the probability mass from network N that was correctly classified when the SVM was trained on representatives of class C from all networks. We see that all of the networks have a very distinctive signature. (As before, the Newman classes are often very small and so training on this class produces unreliable results.)

We saw earlier that elements from the same class and different networks had structural similarities (for certain classes). For example, a Grad BFS community had some resemblance to a DBLP BFS community. In this experiment, we see that elements from the same class and different networks generally also have important differences; that is, although a Grad BFS community resembles a DBLP BFS community in important ways, it is still possible to distinguish between the two. However, there are some notable exceptions to this statement. While for most classes, the classifier does a very good job at distinguishing between networks, the performance of the classifier trained on elements from the Newman (and, for network Fly, Louvain) class is particularly weak.

	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
Grad	62.5%	0.5%	6.0%	1.8%	0.6%	6.9%	8.3%	4.5%	8.8%
Ugrad	3.7%	74.0%	2.0%	9.0%	1.4%	0.4%	0.9%	5.0%	7.0%
HS	2.3%	1.6%	44.2%	7.1%	2.0%	13.0%	5.9%	11.4%	12.6%
SC	1.6%	4.9%	6.3%	42.8%	4.3%	2.1%	4.7%	17.7%	15.5%
Fly	0.5%	2.2%	4.1%	6.5%	60.7%	1.2%	3.1%	11.7%	9.9%
Amazon	1.8%	0.3%	4.6%	0.8%	0.4%	71.6%	11.3%	3.4%	5.3%
DBLP	2.4%	0.5%	3.0%	2.1%	0.9%	12.5%	64.1%	6.1%	48.5%
LJ1	2.0%	1.9%	6.4%	7.7%	4.3%	4.8%	6.8%	35.8%	30.3%
LJ2	2.8%	2.2%	6.1%	7.4%	3.9%	5.1%	8.1%	27.3%	37.3%

Table 2.11: The percentage of probability mass from elements in each network that was classified as each network. Row $N1$, Column $N2$ contains the fraction of probability mass of elements from network $N1$ in the test set that was classified as network $N2$.

We see also that some networks tend to have a stronger signature than others. For instance, the SVMs consistently classify elements from Amazon and DBLP correctly; in contrast, they are less accurate for LJ1 and LJ2 (a closer examination of the data shows that, unsurprisingly, elements from LJ1 and LJ2 tend to be confused with one another).

Our next experiment is structured similarly, except instead of considering each method of community definition separately, we merge all 11 community detection methods together into a single experiment. In this experiment, the training and test sets again contains elements from all 9 networks; however, each network is represented by elements from all 11 community identification methods. For example, the training and test sets contain elements from both Grad BFS and Grad Louvain, all of which are labeled simply as ‘Grad,’ and elements from both Undergrad BFS and Undergrad Louvain, all of which are labeled as ‘Undergrad.’

Table 2.11 contains the results of this experiment, with values along the diagonal representing the percentage of probability mass that was correctly classified. We see again from this experiment that all of these networks have a distinct signature, although some are stronger than others. As before, LJ1 and LJ2 are often confused with one

Class	Grad	Ugrad	HS	SC	Fly	Amazon	DBLP	LJ1	LJ2
BFS	43.5%	11.5%	11.0%	9.7%	79.2%	73.9%	45.4%	31.2%	5.2%
RW0	40.0%	13.1%	11.0%	11.5%	68.5%	76.0%	44.1%	22.7%	7.3%
RW0.15	35.9%	14.7%	10.9%	15.2%	48.8%	76.6%	47.7%	19.5%	9.2%
AB	30.5%	12.0%	8.7%	12.6%	44.3%	73.4%	46.8%	18.2%	11.2%
InfoMap	25.4%	12.7%	9.0%	19.2%	49.6%	66.0%	44.8%	17.5%	14.2%
LinkCom	21.8%	11.7%	14.4%	21.8%	49.1%	60.4%	44.2%	15.1%	12.2%
Louvain	19.2%	10.6%	13.4%	19.8%	43.2%	52.8%	43.9%	15.8%	11.3%
Newman	17.3%	10.3%	12.0%	17.1%	37.1%	50.3%	40.3%	18.1%	16.8%
MCL	16.4%	11.1%	14.2%	18.2%	32.7%	49.0%	39.9%	19.8%	16.4%
Metis	16.8%	11.4%	14.6%	16.7%	39.2%	45.9%	37.0%	20.3%	16.8%
Ann.	28.6%	20.5%	10.9%	12.8%	60.4%	60.3%	41.7%	22.3%	16.8%

Table 2.12: The percentage of probability mass from each network that was correctly classified as belonging to that network, where the training set contained elements from the specified class.

another. Undergrad and Amazon, in particular, have communities that are highly distinguishable from those of other networks. Again, we see that most networks have a strong signature (and even LJ1 and LJ2 can be somewhat distinguished from one another).

In our final experiment, we again perform a separate experiment for each of the 11 methods of defining communities. For each method M (e.g., BFS), we train the classifier on a set containing representatives of method M from all 9 networks, where each element is labeled with its network of origin. The test set contains representatives of every method *except* M , again from all 9 networks, also labeled by the network. In this experiment, we determine whether a classifier can identify which network a community came from, even when that community was defined through a different method than the examples on which the classifier was trained.

Table 2.6 contains the results of this experiment. Naturally, the classifier’s performance in this experiment is much worse than in the preceding two experiments, because the elements in the training set are fundamentally different from the elements in the test set. Even so, we see again that some networks have very strong signatures and

are easily differentiated from the others, whereas for other networks, the accuracy is approximately what one would expect if the classifier made decisions at random. In particular, communities in Amazon tend to have structural similarities, regardless of how that community was defined; that is, a classifier trained to identify BFS sets in Amazon can also identify other types of communities in Amazon. We see similar behavior in networks Fly and DBLP. Interestingly, we also see that the algorithm class used in the training set is also important. For example, SVMs trained on BFS communities tend to do much better at classifying other communities than do SVMs trained on Metis. This suggests that BFS communities are, in some sense, more representative of the entire set of communities from a particular network.

2.7 Computational Performance

Here we present a note about the computational performance of our framework. The machine learning methods we used can easily handle large numbers of features and community detection methods, but the performance of the framework is limited by the performance of these methods.

What makes our approach practical is the fact that virtually every step of our method is highly parallelizable and can take advantage of simple task distributions over large clusters and multi-cores. Once we obtain output from various community detection methods (where each algorithm can run on an independent processor), we can easily calculate the community properties by spreading the computation thereof amongst an arbitrary number of computers. Calculations of the features that we considered in this chapter were fairly fast. In addition, most of the communities that arise in practice are relatively small graphs. In the classification portion of our framework, classifiers may

have difficulty with large datasets, but one can simply sample feature vectors from each class; indeed, this may even be necessary to ensure class balance.

2.8 Discussion

This chapter presents a methodology to address the complexity of analyzing community structure in light of the different notions of communities. This approach contrasts with traditional approaches where the characterization of objects requires the exhaustive enumeration of negative examples. Here we use the diversity of community structures exhibited by different processes as references to understand other community structures. Our approach simultaneously considers a large number of algorithms, multiple domains of application, and a broad spectrum of metrics to characterize community structure.

The machine learning methods we used can easily handle large numbers of features and community detection methods, but the performance of the framework is limited by the performance of these methods.

Almost every step of our method is highly parallelizable and can take advantage of simple task distributions over large clusters and multi-cores. Once we obtain output from various community detection methods (where each algorithm can run on an independent processor), we can easily calculate the community properties by spreading the computation thereof amongst an arbitrary number of computers. Calculations of the features that we considered in this chapter were fairly fast. In addition, most of the communities that arise in practice are relatively small graphs. In the classification portion of our framework, classifiers may have difficulty with large datasets, but one can simply sample feature vectors from each class; indeed, this may even be necessary to ensure class balance.

	DBLP		Amazon		LJ1		LJ2	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
100	42%	174	38%	187	56%	146	53%	149
250	44%	912	40%	1069	58%	751	56%	773
500	47%	3368	42%	4051	59%	2724	58%	2736
750	48%	7725	43%	9253	61%	5841	59%	6059
1000	49%	13718	44%	16678	61%	10219	60%	10723

Table 2.13: Runtime results for networks DBLP, Amazon, LJ1, and LJ2. Each row represents a different class sample size. Cells contain average classification accuracy across all classes and running time (in seconds) for that sample size.

Table 2.13 contains run-time and accuracy values for each of our four larger networks at different sample sizes. In this experiment, which is similar to the experiment described in Section 2.4 and presented in Table 2.3, we trained an SVM on balanced classes of various different sizes, and evaluated the accuracy of the resulting classifier on withheld elements on those classes. For each sample size, we present both the average accuracy of the classifier over all classes as well as the time required to create and evaluate the model.

We see that smaller sample sizes result in remarkably faster model creation and evaluation, with only a slight drop in accuracy. A practitioner highly concerned with efficiency can thus obtain a fairly accurate model very quickly. These results illustrate the scalability of our framework, and so demonstrate its feasibility on large datasets.

It is important to emphasize that our work focuses on structural similarity, which is a weaker requirement than accuracy. In other words, communities with similar properties to real communities may not correspond exactly to the communities we may expect to find. Nevertheless, we firmly believe that mastering structure is a fundamental stepping stone in the development of algorithms to accurately find the communities of interest.

Our experimental analysis includes 10 community detection algorithms, represen-

tative of popular algorithms in the literature, a collection of 9 different networks from diverse domains, and 36 structural properties. The results reveal, first, a high variability among the output of community detection methods, which is demonstrated by cross-validation performance of the classifiers, showing that each of these 10 classes of communities was remarkably structurally consistent. Second, annotated communities have a distinct structure from what we expect; a classifier can distinguish most of the mass of annotated communities from that of communities produced by algorithms. In fact, their structure is closer to the output of baseline procedures, such as random walks and BFS, than to that of more structured popular algorithms. Third, a small set of features explain the biases produced by different algorithms and expose the structural signature of real communities, which were represented in this study via meaningful annotations present in our datasets. Fourth, we can organize the menagerie of available community detection algorithms by grouping them with respect to structural similarities in behavior. This can be done via two mechanisms: (1) by analyzing the behavior of algorithms with respect to the most relevant features found in the preceding step, and (2) by applying a class selection method we defined based on the scatter matrices to determine whether two classes are independent, i.e., each class bringing a new viewpoint to our analysis, or redundant, i.e., where multiple classes contribute the same information to the collection. The latter approach is also applied in our frameworks as a pre-processing step to improve the interpretability of the separability measures. We saw that the two random walk classes and the two modularity-based classes were quite similar, and so repeated our earlier experiments after merging these pairs of classes. This naturally resulted in greater class consistency, and further cemented the dominance of the random walk methods when classifying the annotated communities. Last, in contrast to our earlier experiments, which analyzed the consistency of classes defined by community detection methods, we analyzed whether communities from the same network tended to

resemble each other. We first restricted our analysis to one type of community detection method at a time, and showed that communities that were produced by the same method but from different networks tended to be different in important ways. Interestingly, we showed that even when a classifier was trained to differentiate between networks by using communities produced by one particular community detection method, that same classifier was successful even when applied to a test set containing communities produced by the other community detection methods.

Our approach differs fundamentally from previous work in the area due to its supervised nature. The main message that resulted from our experimental analysis is that community structure is not clearly defined, and moreover, not universal. In fact, there is a broad range of structures that are generated by multiple definitions, heuristics, and expectations. Accordingly, our supervised approach may be used by a practitioner to make an informed decision about the most suitable algorithm for a given network in the following way. First, we produce a test set comprising examples of the communities we are interested in finding, which could be either real or synthetic. Second, we choose a set of algorithms we want to evaluate. Finally, we apply our approach using the target network and present the classifier with the test set. The classifier assigns the probability mass of the test set to the class of algorithm that bears closest resemblance to the examples. The algorithm that receives the bulk of the mass is the algorithm that is most likely to succeed in extracting communities that structurally resemble the ones we are interested in. Researchers may also benefit from our methodology when designing new community detection algorithms as a way to compare the behavior of new methods with existing ones. Finally, our framework suggests a change in the way we approach the problem of community detection. Typical community detection methods treat the task of extracting communities as an unsupervised problem in which a particular structure is extracted. However, this approach presents little sensitivity to different purposes, dif-

ferent structures of interest and different domains of application. In contrast, we could start thinking about a supervised approach that allows the user to specify the particular community structure they intend to find through examples. Then, a hypothetical algorithm would learn structure from these examples and retrieve similar structures from the network.

CHAPTER 3

TRACE COMPLEXITY OF NETWORK INFERENCE

Many technological, social, and biological phenomena are naturally modeled as the propagation of a contagion through a network. For instance, in the blogosphere, “memes” spread through an underlying social network of bloggers [14], and, in biology, a virus spreads over a population through a network of contacts [19]. In many such cases, an observer may not directly probe the underlying network structure, but may have access to the sequence of times at which the nodes are infected. Given one or more such records, or *traces*, and a probabilistic model of the epidemic process, we can hope to deduce the underlying graph structure or at least estimate some of its properties. This is the *network inference* problem, which researchers have studied extensively in recent years [14, 42, 59, 60, 102].

In this chapter we focus on the number of traces that network inference tasks require, which we define as the *trace complexity* of the problem. Our work provides inference algorithms with rigorous upper bounds on their trace complexity, along with information-theoretic lower bounds. We consider network inference tasks under a diffusion model presented in [60], whose suitability for representing real-world cascading phenomena in networks is supported by empirical evidence. In short, the model consists of a random cascade process that starts at a single node of a network, and each edge $\{u, v\}$ independently propagates the epidemic, once u is infected, with probability p after a random *incubation time*.

Overview of results. In the first part of this chapter, we focus on determining the number of traces that are necessary and/or sufficient to perfectly recover the edge set of the whole graph with high probability. We present algorithms and (almost) matching lower bounds for exact inference by showing that in the worst case, $\Omega\left(\frac{n\Delta}{\log^2 \Delta}\right)$ traces

are necessary and $O(n\Delta \log n)$ traces are sufficient, where n is the number of nodes in the network and Δ is its maximum degree. In the second part, we consider a natural line of investigation, given the preceding strong lower bounds, where we ask whether exact inference is possible using a smaller number of traces for special classes of networks that frequently arise in the analysis of social and information networks. Accordingly, we present improved algorithms and trace complexity bounds for two such cases. We give a very simple and natural algorithm for exact inferences of trees that uses only $O(\log n)$ traces.¹ To further pursue this point, we give an algorithm that exactly reconstructs graphs of degree bounded by Δ using only $O(\text{poly}(\Delta) \log n)$ traces, under the assumption that epidemics always spread throughout the whole graph. Finally, given that recovering the topology of a hidden network in the worst case requires an impractical number of traces, a natural question is whether some non-trivial property of the network can be accurately determined using a moderate number of traces. Accordingly, we present a highly efficient algorithm that, using vastly fewer traces than are necessary for reconstructing the entire edge set, reconstructs the degree distribution of the network with high fidelity by using $O(n)$ traces.

The information contained in a trace. Our asymptotic results also provide some insight into the usefulness of information contained in a trace. Notice that the first two nodes of a trace unambiguously reveal one edge — the one that connects them. As we keep scanning a trace the signal becomes more and more blurred: the third node could be a neighbor of the first or of the second node, or both. The fourth node could be the neighbor of any nonempty subset of the first three nodes, and so on. The main technical challenge in our context is whether we can extract any useful information from the *tail* of a trace, i.e., the suffix consisting of all nodes from the second to the last. As it turns out, our lower bounds show that, for perfect inference on general connected graphs, the

¹All inference results in this chapter hold with high probability.

answer is “no”: we show that the *First-Edge algorithm*, which just returns the edges corresponding to the first two nodes in each trace and ignores the rest, is essentially optimal. This limitation precludes optimal algorithms with practical trace complexity². This result motivates further exploration of trace complexity for special-case graphs. Accordingly, for trees and bounded degree graphs, we illustrate how the tail of traces can be extremely useful for network inference tasks.

Our aforementioned algorithms for special-case graphs make use of maximum likelihood estimation (MLE) but in different ways. Previous approaches, with which we compare our results, have also employed MLE for network inference. For instance, NETINF [60] is an algorithm that attempts to reconstruct the network from a set of independent traces by exploring a submodular property of its MLE formulation. Another example, and closest to ours, is the work by Netrapalli and Sangahvi [102], whose results include qualitatively similar bounds on trace complexity in a quite different epidemic model.

Turning our attention back to our algorithms, our tree reconstruction algorithm performs global likelihood maximization over the entire graph, like the NETINF algorithm [60], whereas our bounded-degree reconstruction algorithm, like the algorithm in [102], performs MLE at each individual vertex. Our algorithms and analysis techniques, however, differ markedly from those of [60] and [102], and may be of independent interest.

In the literature on this rapidly expanding topic, researchers have validated their findings using small or stylized graphs and a relatively large number of traces. In this work, we aim to provide, in the same spirit as [102], a formal and rigorous understanding

²On the other hand, the use of short traces may not be only a theoretical limitation, given the real world traces that we observe in modern social networks. For example, Bakshy et al. [20] report that most cascades in Twitter (`twitter.com`) are short, involving one or two hops.

of the potentialities and limitations of algorithms that aim to solve the network inference problem.

This chapter is organized as follows. Section 3.1 presents an overview of previous approaches to network learning. Section 3.2 presents the cascade model we consider throughout the chapter. Section 3.3 deals with the *head of the trace*: it presents the First-Edge algorithm for network inference, shows that it is essentially optimal in the worst case, and shows how the first edges' timestamps can be used to guess the degree distribution of the network. Section 3.4, instead, deals with the *tail of the trace*: it presents efficient algorithms for perfect reconstruction of the topology of trees and of bounded degree networks. Section 3.5 presents an experimental analysis that compares ours and existing results through the lens of trace complexity. Section 3.6 offers our conclusions. The proofs missing from the main body of the chapter can be found in Appendix 3.7.

3.1 Related Work

Network inference has been a highly active area of investigation in data mining and machine learning [14, 42, 59, 60, 102]. It is usually assumed that an event initially activates one or more nodes in a network, triggering a cascading process, e.g., bloggers acquire a piece of information that interests other bloggers [63], a group of people are the first infected by a contagious virus [19], or a small group of consumers are the early adopters of a new piece of technology that subsequently becomes popular [120]. In general, the process spreads like an epidemic over a network (i.e., the network formed by blog readers, the friendship network, the coworkers network). Researchers derive observations from each cascade in the form of *traces* — the identities of the people that are activated in the process and the timestamps of their activation. However, while we do see traces,

we do not directly observe the network over which the cascade spreads. The network inference problem consists of recovering the underlying network using the epidemic data.

In this chapter we study the cascade model that Gomez-Rodrigues et al. [60] introduced, which consists of a variation of the independent cascade model [73]. Gomez-Rodrigues et al. propose NETINF, a maximum likelihood algorithm, for network reconstruction. Their method is evaluated under the exponential and power-law distributed incubation times. In our work, we restrict our analysis to the case where the incubation times are exponentially distributed as this makes for a rich arena of study.

Gomez-Rodrigues et al. have further generalized the model to include different transmission rates for different edges and a broader collection of waiting time distributions [59, 101]. Later on, Du et al. [42] proposed a kernel-based method that is able to recover the network without prior assumptions on the waiting time distributions. These methods have significantly higher computational costs than NETINF, and, therefore, than ours. Nevertheless, experiments on real and synthetic data show a marked improvement in accuracy, in addition to gains in flexibility. Using a more combinatorial approach, Gripon and Rabbat [62] consider the problem of reconstructing a graph from traces defined as sets of unordered nodes, in which the nodes that appear in the same trace are connected by a path containing exactly the nodes in the trace. In this work, traces of size three are considered, and the authors identify necessary and sufficient conditions to reconstruct graphs in this setting.

The performance of network inference algorithms is dependent on the amount of information available for the reconstruction, i.e., the number and length of traces. The dependency on the number of traces have been illustrated in [42], [59], and [60] by plotting the performance of the algorithms against the number of available traces. Nev-

ertheless, we find little research on a rigorous analysis of this dependency, with the exception of one paper [102] that we now discuss.

Similarly to our work, Netrapalli and Sangahvi [102] present quantitative bounds on trace complexity in a quite different epidemic model. The model studied in [102] is another variation of the independent cascade model. It differs from the model we study in a number of key aspects, which make that model a simplification of the model we consider here. For instance, (i) [102] assumes a cascading process over discrete time steps, while we assume continuous time (which has been shown to be a realistic model of several real-world processes [60]), (ii) the complexity analyzed in [102] applies to a model where nodes are active for a single time step — once a node is infected, it has a single time step to infect its neighbors, after which it becomes permanently inactive. The model we consider does not bound the time that a node can wait before infecting a neighbor. Finally, (iii) [102] rely crucially on the “correlation decay” assumption, which implies – for instance — that each node can be infected during the course of the epidemics by *less* than 1 neighbor in expectation. The simplifications in the model presented by [102] make it less realistic — and, also, make the inference task significantly easier than the one we consider here.

We believe that our analysis introduces a rigorous foundation to assess the performance of existing and new algorithms for network inference. In addition, to the best of our knowledge, our work is the first to study how different parts of the trace can be useful for different network inference tasks. Also, it is the first to study the trace complexity of special case graphs, such as bounded degree graphs, and for reconstructing non-trivial properties of the network (without reconstructing the network itself), such as the node degree distribution.

3.2 Cascade Model

The cascade model we consider is defined as follows. It starts with one activated node, henceforth called the *source* of the epidemic, which is considered to be activated, without loss of generality, at time $t = 0$.

As soon as a node u gets activated, for each neighbor v_i , u flips an independent coin: with probability p it will start a countdown on the edge $\{u, v_i\}$. The length of the countdown will be a random variable distributed according to $\text{Exp}(\lambda)$ (exponential³ with parameter λ). When the countdown reaches 0, that edge is *traversed* — that is, that epidemic reaches v_i via u .

The “trace” produced by the model will be a sequence of tuples (node v , $t(v)$) where $t(v)$ is the first time at which the epidemics reaches v .

In [60], the source of the epidemics is chosen uniformly at random from the nodes of the network. In general, though, the source can be chosen arbitrarily⁴.

The cascade process considered here admits a number of equivalent descriptions. The following happens to be quite handy: independently for each edge of G , remove the edge with probability $1 - p$ and otherwise assign a random edge length sampled from $\text{Exp}(\lambda)$. Run Dijkstra’s single-source shortest path algorithm on the subgraph formed by the edges that remain, using source s and the sampled edge lengths. Output vertices in the order they are discovered, accompanied by a timestamp representing the shortest path length.

³ [42, 59, 60] consider other random timer distributions; we will mainly be interested in exponential variables as this setting is already rich enough to make for an interesting and extensive analysis.

⁴Choosing sources in a realistic way is an open problem — the data that could offer a solution to this problem seems to be extremely scarce at this time.

3.3 The Head of a Trace

In this section we will deal with the head of a trace — that is, with the edge connecting the first and the second nodes of a trace. We show that, for general graphs, that edge is the only useful information that can be extracted from traces. Moreover, and perhaps surprisingly, this information is enough to achieve close-to-optimal trace complexity, i.e., no network inference algorithm can achieve better performance than a simple algorithm that only extracts the head of the trace and ignores the rest. We analyze this algorithm in the next section.

3.3.1 The First-Edge Algorithm

The First-Edge algorithm is simple to state. For each trace in the input, it extracts the edge connecting the first two nodes, and adds this edge to the guessed edge set, ignoring the rest of the trace. This procedure is not only optimal in trace complexity, but, as it turns out, it is also computationally efficient.

We start by showing that First-Edge is able to reconstruct the full graph with maximum degree Δ using $\Theta(n\Delta \log n)$ traces, under the cascade model we consider.

Theorem 3.3.1. *Suppose that the source $s \in V$ is chosen uniformly at random. Let $G = (V, E)$ be a graph with maximum degree $\Delta \leq n - 1$. With $\Theta\left(\frac{n\Delta}{p} \log n\right)$ traces, First-Edge correctly returns the graph G with probability at least $1 - \frac{1}{\text{poly}(n)}$.*

Proof. Let $e = \{u, v\}$ be any edge in E . The probability that a trace starts with u , and continues with v can be lower bounded by $\frac{p}{n\Delta}$, that is, by the product of the probabilities that u is selected as the source, that the edge $\{u, v\}$ is not removed from the graph, and

that v is the first neighbor of u that gets infected. Therefore, if we run $c \frac{n\Delta}{p} \ln n$ traces, the probability that none of them starts with the ordered couple of neighboring nodes u, v is at most:

$$\left(1 - \frac{p}{n\Delta}\right)^{\frac{n\Delta}{p} c \ln n} \leq \exp(-c \ln n) = n^{-c}.$$

Therefore, the assertion is proved for any constant $c > 2$. □

We notice that a more careful analysis leads to a proof that

$$\Theta((\Delta + p^{-1}) n \log n)$$

traces are enough to reconstruct the whole graph with high probability. To prove this stronger assertion, it is sufficient to show the probability that a specific edge will be the first one to be traversed is at least $\frac{2}{n} \cdot (1 - e^{-1}) \cdot \min(\Delta^{-1}, p)$. In fact one can even show that, for each $d \leq \Delta$, if the First-Edge algorithm has access to $O((d + p^{-1}) n \log n)$ traces, then it will recover all the edges having at least one endpoint of degree less than or equal d . As we will see in our experimental section, this allows us to reconstruct a large fraction of the edges using a number of traces that is significantly smaller than the maximum degree times the number of nodes.

Finally, we note that the above proof also entails that First-Edge performs as stated for any waiting time distribution (that is, not just for the exponential one). In fact, the only property that we need for the above bounds to hold, is that the first node, and the first neighbor of the first node, are chosen independently and uniformly at random by the process.

3.3.2 Lower Bounds

In this section we discuss a number of lower bounds for network inference.

We start by observing that if the source node is chosen adversarially — and, say if the graph is disconnected — no algorithm can reconstruct the graph (traces are trapped in one connected component and, therefore, do not contain any information about the rest of the graph.) Moreover, even if the graph is forced to be connected, by choosing $p = \frac{1}{2}$ (that is, edges are traversed with probability $\frac{1}{2}$) an algorithm will require at least $2^{\Omega(n)}$ traces even if the graph is known to be a path. Indeed, if we select one endpoint as the source, it will take $2^{\Omega(n)}$ trials for a trace to reach the other end of the path, since at each node, the trace flips an unbiased coin and dies out with probability $\frac{1}{2}$.

This is the reason why we need the assumption that the epidemic selects $s \in V$ uniformly at random — we recall that this is also an assumption in [60]. Whenever possible, we will consider more realistic assumptions, and determine how this changes the trace complexity of the reconstruction problem.

We now turn our attention to our main lower bound result. Namely, even if traces never die (that is, if $p = 1$), and assuming that the source is chosen uniformly at random, we need $\tilde{\Omega}(n\Delta)$ traces to reconstruct the graph.

First, let G_0 be the clique on the node set $V = \{1, \dots, n\}$, and let G_1 be the clique on V minus the edge $\{1, 2\}$.

Suppose that Nature selects the unknown graph uniformly at random in the set $\{G_0, G_1\}$. We will show that with $o\left(\frac{n^2}{\log^2 n}\right)$ traces, the probability that we are able to guess the unknown graph is at most $\frac{1}{2} + o(1)$ — that is, flipping a coin is close to being the best one can do for guessing the existence of the edge $\{1, 2\}$.

Before embarking on this task, though, we show that this result directly entails that $o\left(n \cdot \frac{\Delta}{\log^2 \Delta}\right)$ traces are not enough for reconstruction even if the graph has maximum degree Δ , for each $1 \leq \Delta \leq n - 1$. Indeed, let the graph G'_0 be composed of a clique on

$\Delta + 1$ nodes, and of $n - \Delta - 1$ disconnected nodes. Let G'_1 be composed of a clique on $\Delta + 1$ nodes, minus an edge, and of $n - \Delta - 1$ disconnected nodes. Then, due to our yet-unproven lower bound, we need at least $\Omega\left(\frac{\Delta^2}{\log^2 \Delta}\right)$ traces to start in the large connected component for the reconstruction to succeed. The probability that a trace starts in the large connected component is $O\left(\frac{\Delta}{n}\right)$. Hence, we need at least $\Omega\left(n \cdot \frac{\Delta}{\log^2 \Delta}\right)$ traces.

We now highlight the main ideas that we used to prove the main lower bound, by stating the intermediate lemmas that lead to it. The proofs of these Lemmas can be found in Appendix 3.7.

The first lemma states that the random ordering of nodes produced by a trace in G_0 is uniform at random, and that the random ordering produced by a trace in G_1 is “very close” to being uniform at random. Intuitively, this entails that one needs many traces to be able to infer the unknown graph by using the orderings given by the traces.

Lemma 3.3.2. *Let π be the random ordering of nodes produced by the random process on G_0 , and π' be the random ordering of nodes produced by the random process on G_1 . Then,*

1. π is a uniform at random permutation over $[n]$;
2. for each $1 \leq a < b \leq n$, the permutation π' conditioned on the vertices 1, 2 appearing (in an arbitrary order) in the positions a, b , is uniform at random in that set;
3. moreover, the probability $p_{a,b}$ that π' has the vertices 1, 2 appearing (in an arbitrary order) in the positions $a < b$ is equal to $p_{a,b} = \frac{1+d(a,b)}{\binom{n}{2}}$, with
 - $d(a, b) = -1$ if $a = 1, b = 2$; otherwise $d(a, b) > -1$;
 - moreover $d_{a,b} = O\left(\frac{\ln n}{n}\right) - O\left(\frac{1}{b}\right)$.
4. Finally, $\sum_{a=1}^{n-1} \sum_{b=a+1}^n d(a, b) = 0$.

The preceding Lemma can be used to prove Lemma 3.3.3: if one is forced not to use timestamps, $o\left(\frac{n^2}{\log^2 n}\right)$ traces are not enough to guess the unknown graph with probability more than $\frac{1}{2} + o(1)$.

Lemma 3.3.3. *Let \mathcal{P} the sequence of the ℓ orderings of nodes given by ℓ traces, with $\ell = o\left(\frac{n^2}{\ln^2 n}\right)$.*

The probability that the likelihood of \mathcal{P} is higher in the graph G_0 is equal to $\frac{1}{2} \pm o(1)$, regardless of the unknown graph.

The next Lemma, which also needs Lemma 3.3.2, takes care of the waiting times in the timestamps. Specifically, it shows that – under a conditioning having high probability – the probability that the sequence of waiting times of the traces has higher likelihood in G_0 than in G_1 is $\frac{1}{2} \pm o(1)$, regardless of the unknown graph.

Lemma 3.3.4. *Let α satisfy $\alpha = o(1)$, and $\alpha = \omega\left(\frac{\log n}{n}\right)$. Also, let ℓ_i be the number of traces that have exactly one of the nodes in $\{1, 2\}$ the first i informed nodes.*

Let \mathcal{W} be the random waiting times of the traces. Then, if we condition on $\ell_i = \Theta(\alpha \cdot i \cdot (n - i))$ for each $i = 1, \dots, n$ (independently of the actual node permutations), the probability that the likelihood of \mathcal{W} is higher in the graph G_0 is equal to $\frac{1}{2} \pm o(1)$, regardless of the unknown graph.

Finally, the following corollary follows directly from Lemma 3.3.3 and Lemma 3.3.4, and by a trivial application of the Chernoff Bound.

Corollary 3.3.5. *If Nature chooses between G_0 and G_1 uniformly at random, and one has access to $o\left(\frac{n^2}{\log^2 n}\right)$ traces, then no algorithm can correctly guess the graph with probability more than $\frac{1}{2} + o(1)$.*

As already noted, the lower bound of Corollary 3.3.5 can be easily transformed in a $\Omega\left(n \cdot \frac{\Delta}{\log^2 \Delta}\right)$ lower bound, for any $\Delta \leq n - 1$.

3.3.3 Reconstructing the Degree Distribution

In this section we study the problem of recovering the degree distribution of a hidden network and show that this can be done with $\Omega(n)$ traces while achieving high accuracy, using, again, only the first edge of a trace.

The degree distribution of a network is a characteristic structural property of networks, which influences their dynamics, function, and evolution [104]. Accordingly, many networks, including the Internet and the world wide web exhibit distinct degree distributions [51]. Thus, recovering this property allows us to make inferences about the behavior of processes that take place in these networks, without knowledge of their actual link structure.

Let ℓ traces starting from the same node v be given. For trace i , let t_i be the differences between the time of exposure of v , and the the time of exposure of the second node in the trace.

Recall that in the cascade model, the waiting times are distributed according to an exponential random variable with a known parameter λ . If we have ℓ traces starting at a node v , we aim to estimate the degree of v the time gaps t_1, \dots, t_ℓ between the first node and the second node of each trace.

If v has degree d in the graph, then t_i ($1 \leq i \leq \ell$) will be distributed as an exponential random variable with parameter $d\lambda$ [43]. Furthermore, the sum T of the t_i 's, $T = \sum_{i=1}^{\ell} t_i$, is distributed as an Erlang random variable with parameters $(\ell, d\lambda)$ [43].

In general, if X is an Erlang variable with parameters (n, λ) , and Y is a Poisson variable with parameter $z \cdot \lambda$, we have that $\Pr[X < z] = \Pr[Y \geq n]$. Then, by using the tail bound for the Poisson distribution [31, 79], we have that the probability that T is at most $(1 + \epsilon) \cdot \frac{\ell}{d\lambda}$ is

$$\Pr[\text{Pois}((1 + \epsilon) \cdot \ell) \geq \ell] \geq 1 - e^{-\Theta(\epsilon^2 \ell)}.$$

Similarly, the probability that T is at least $(1 - \epsilon) \cdot \frac{\ell}{d\lambda}$ is

$$1 - \Pr[\text{Pois}((1 - \epsilon) \cdot \ell) \geq \ell] \geq 1 - e^{-\Theta(\epsilon^2 \ell)}.$$

We then have:

$$\Pr\left[\left|T - \frac{\ell}{d\lambda}\right| \leq \epsilon \cdot \frac{\ell}{d\lambda}\right] \geq 1 - e^{-\Theta(\epsilon^2 \ell)}.$$

Let our degree inference algorithm return $\hat{d} = \frac{\ell}{T\lambda}$ as the degree of v . Also, let d be the actual degree of v . We have:

$$\Pr\left[\left|\hat{d} - d\right| \leq \epsilon d\right] \geq 1 - e^{-\Theta(\epsilon^2 \ell)}.$$

We have then proved the following theorem:

Theorem 3.3.6. *Provided that $\Omega\left(\frac{\ln \delta^{-1}}{\epsilon^2}\right)$ traces start from v , the degree algorithm returns a $1 \pm \epsilon$ multiplicative approximation to the degree of v with probability at least $1 - \delta$.*

3.4 The Tail of the Trace

A naïve interpretation of the lower bound for perfect reconstruction, Corollary 3.3.5, would conclude that the information in the “tail” of the trace — the list of nodes infected

after the first two nodes, and their timestamps — is of negligible use in achieving the task of perfect reconstruction. In this section we will see that the opposite conclusion holds for important classes of graphs. We specialize to two such classes, trees and bounded-degree graphs, in both cases designing algorithms that rely heavily on information in the tails of traces to achieve perfect reconstruction with trace complexity $O(\log n)$, an exponential improvement from the worst-case lower bound in Corollary 3.3.5. The algorithms are quite different: for trees we essentially perform maximum likelihood estimation (MLE) of the entire edge set all at once, while for bounded-degree graphs we run MLE separately for each vertex to attempt to find its set of neighbors, then we combine those sets while resolving inconsistencies.

In Section 3.5 we provide one more example of an algorithm, which we denote by First-Edge+, that makes use of information in the tail of the trace. Unlike the algorithms in this section, we do not know of a theoretical performance guarantee for First-Edge+ so we have instead analyzed it experimentally.

It is natural to compare the algorithms in this section with the NETINF algorithm [60], since both are based on MLE. While NETINF is a general-purpose algorithm, and the algorithms developed here are limited to special classes of graphs, we believe our approach offers several advantages. First, and most importantly, we offer provable trace complexity guarantees: $\Omega(\log n)$ complete traces suffice for perfect reconstruction of a tree with high probability, and $\Omega(\text{poly}(\Delta) \log n)$ traces suffice for perfect reconstruction of a graph with maximum degree Δ . Previous work has not provided rigorous guarantees on the number of traces required to ensure that algorithms achieve specified reconstruction tasks. Second, our tree reconstruction algorithm is simple (an easy pre-processing step followed by computing a minimum spanning tree) and has worst-case running time $O(n^2\ell)$, where n is the number of nodes and $\ell = \Omega(\log n)$ is the number

of traces, which compares favorably with the running time of NETINF.

3.4.1 Reconstructing Trees

In this section we consider the special case in which the underlying graph G is a tree, and we provide a simple algorithm that requires $\Omega(\log n)$ complete traces and succeeds in perfect reconstruction with high probability. Intuitively, reconstructing trees is much simpler than reconstructing general graphs for the following reason. As noted in [60], the probability that an arbitrary graph G generates trace T is a sum, over all spanning trees F of G , of the probability that T was generated by an epidemic propagating along the edges of F . When G itself is a tree, this sum degenerates to a single term and this greatly simplifies the process of doing maximum likelihood estimation. In practical applications of the network inference problem, it is unlikely that the latent network will be a tree; nevertheless we believe the results in this section are of theoretical interest and that they may provide a roadmap for analyzing the trace complexity of other algorithms based on maximum likelihood estimation.

Input: A collection T_1, \dots, T_ℓ of complete traces generated by repeatedly running the infection process with $p = 1$ on a fixed tree.

Let $t_i(v)$ denote the infection time of node v in trace T_i .

Output: An estimate, \hat{G} , of the tree.

- 1: **for all** pairs of nodes u, v **do**
- 2: Let $c(u, v)$ be the median of the set $\{|t_i(u) - t_i(v)|\}_{i=1}^\ell$.
- 3: **if** \exists a node p and a pair of traces T_i, T_j such that $t_i(p) < t_i(u) < t_i(v)$ and $t_j(p) < t_j(v) < t_j(u)$ **then**
- 4: Set $c(u, v) = \infty$.
- 5: **Output** $\hat{G} =$ minimum spanning tree with respect to cost matrix $c(u, v)$.

Algorithm 1: The tree reconstruction algorithm.

The tree reconstruction algorithm is very simple. It defines a cost for each edge $\{u, v\}$ as shown in Figure 1, and then it outputs the minimum spanning tree with respect

to those edge costs. The most time-consuming step is the test in step 3, which checks whether there is a node p whose infection time precedes the infection times of both u and v in two distinct traces T_i, T_j such that the infection times of u and v are oppositely ordered in T_i and T_j . (If so, then G contains a path from p to u that does not include v , and a path from p to v that does not include u , and consequently $\{u, v\}$ cannot be an edge of the tree G . This justifies setting $c(u, v) = \infty$ in step 4.) To save time, one can use lazy evaluation to avoid performing this test for every pair u, v . The lazy version of the algorithm computes edge costs $c(u, v)$ as in step 3 and then proceeds straight to the minimum spanning tree computation, using Kruskal's algorithm. Any time Kruskal's algorithm decides to insert an edge $\{u, v\}$ into the tree, we instead perform the test in step 3 and delete edge $\{u, v\}$ from the graph if it violates the test.

The analysis of the algorithm is based on the following outline: first, we show that if $\{u, v\}$ is any edge of G , then $c(u, v) < \lambda^{-1}$ with high probability (Lemma 3.4.1). Second, we show that if $\{u, v\}$ is any edge not in G , then $c(u, v) > \lambda^{-1}$ with high probability (Lemma 3.4.2). The edge pruning in steps 3 and 4 of the algorithm is vital for attaining the latter high-probability guarantee. When both of these high-probability events occur, it is trivial to see that the minimum spanning tree coincides with G .

Lemma 3.4.1. *If $\{u, v\}$ is an edge of the tree G , then Algorithm 1 sets $c(u, v) < \lambda^{-1}$ with probability at least $1 - c_1^\lambda$, for some absolute constant $c_1 < 1$.*

Proof. First, note that the algorithm never sets $c(u, v) = \infty$. This is because if one were to delete edge $\{u, v\}$ from G , it would disconnect the graph into two connected components G_u, G_v , containing u and v , respectively. The infection process cannot spread from G_u to G_v or vice-versa without traversing edge $\{u, v\}$. Consequently, for every node $p \in G_u$, the infection time $t_i(u)$ occurs strictly between $t_i(p)$ and $t_i(v)$ in all traces. Similarly, if $p \in G_v$ then the infection time $t_i(v)$ occurs strictly between $t_i(p)$

and $t_i(u)$ in all traces.

Therefore, the value of $c(u, v)$ is equal to the median of $|t_i(u) - t_i(v)|$ over all the traces T_1, \dots, T_ℓ . In any execution of the infection process, if the first endpoint of edge $\{u, v\}$ becomes infected at time t , then the opposite endpoint receives a timestamp $t + X$ where $X \sim \text{Exp}(\lambda)$. Consequently the random variable $|t_i(u) - t_i(v)|$ is an independent sample from $\text{Exp}(\lambda)$ in each trace. The probability that any one of these samples is greater than λ^{-1} is $1/e$, so the probability that their median exceeds λ^{-1} is equal to the probability of observing at least $\ell/2$ heads in ℓ tosses of a coin with bias $1/e$. By Chernoff's bound [99], this is less than $(\sqrt{2}e^{1/e})^{-\ell}$. \square

The remaining step in analyzing the tree reconstruction algorithm is to prove that $c(u, v) > \lambda^{-1}$ with high probability when $\{u, v\}$ is not an edge of the tree G .

Lemma 3.4.2. *If $\{u, v\}$ is not an edge of G , then Algorithm 1 sets $c(u, v) > \lambda^{-1}$ with probability at least $1 - c_2 \cdot c_3^\ell$ for some absolute constants $c_2 < \infty$ and $c_3 < 1$.*

Proof. G is a tree, so for any two nodes u, v , there is a unique path $P(u, v)$ in G that starts at u and ends at v . Furthermore, for every $s \in G$, there is a unique node $z(s) \in P(u, v)$ such that the paths $P(s, u)$ and $P(s, v)$ are identical up until they reach $z(s)$, and they are vertex-disjoint afterward. When the infection process starts at s and spreads throughout G , it always holds that $t(z(s)) \leq \min\{t(u), t(v)\}$. Conditional on the value of $t(z(s))$, the infection times of vertices on the paths $P(z(s), u)$ and $P(z(s), v)$ constitute two independent Poisson processes each with rate λ . Let $n_u(s)$ and $n_v(s)$ denote the number of edges in the paths $P(z(s), u)$ and $P(z(s), v)$, respectively. The infection times $t(u), t(v)$ occur at the $n_u(s)^{\text{th}}$ and $n_v(s)^{\text{th}}$ arrival times, respectively, in the two independent Poisson processes.

Let s_1, \dots, s_ℓ denote the sources of traces T_1, \dots, T_ℓ . We distinguish two cases.

First, suppose at least $\frac{\ell}{10}$ of the traces satisfy $n_u(s_i) = n_v(s_i)$. In any of these traces, the events $t_i(u) < t_i(v)$ and $t_i(v) < t_i(u)$ both have probability $1/2$, by symmetry. Hence, with probability at least $1 - 2 \cdot 2^{-\ell/10}$, there exist traces T_i, T_j such that $z(s_i), z(s_j)$ are both equal to the midpoint of the path $P(u, v)$, but $t_i(u) < t_i(v)$ whereas $t_j(v) < t_j(u)$. If this high-probability event happens, the condition in step 3 of the algorithm will be satisfied with $p = z(s_i) = z(s_j)$ and the cost $c(u, v)$ will be set to ∞ .

The remaining case is that at least $\frac{9\ell}{10}$ of the traces satisfy $n_u(s_i) \neq n_v(s_i)$. In this case, we reason about the distribution of $|t_i(u) - t_i(v)|$ as follows. Let q denote the number of uninfected nodes on path P at the time t when an element of $\{u, v\}$ is first infected. Conditional on the value of t , the remaining infection times of the nodes on path P are the arrival times in a Poisson process of rate λ . The conditional probability that $|t_i(u) - t_i(v)| > \lambda^{-1}$, given q , is therefore equal to the probability that a $\text{Pois}(1)$ random variable is less than q . This conditional probability is equal to $1/e$ when $q = 1$ and is at least $2/e$ when $q > 1$. (The value of q is always at least 1, because at time t exactly one element of $\{u, v\}$ is infected and the other is not yet infected.)

When $n_u(s_i) \neq n_v(s_i)$, we claim that $\Pr(q > 1)$ is at least $1/2$. To see why, assume without loss of generality that $n_u(s_i) < n_v(s_i)$ and let x be the node on path $P(u, v)$ such that $x \neq u$ but u and x are equidistant from $z(s_i)$. (In other words, the paths $P(z(s_i), x)$ and $P(z(s_i), u)$ have the same number of edges.) By symmetry, the events $t_i(u) < t_i(x)$ and $t_i(x) < t_i(u)$ both have probability $1/2$. Conditional on the event $t_i(u) < t_i(x)$, we have $q > 1$ because x, v are two distinct nodes that are uninfected at time $t_i(u)$. Consequently, $\Pr(q > 1) \geq 1/2$ as claimed.

Now let us combine the conclusions of the preceding two paragraphs. For notational convenience, we use t_i^{uv} as shorthand for $|t_i(u) - t_i(v)|$. When $n_u(s_i) \neq n_v(s_i)$ we have

derived:

$$\begin{aligned} \Pr(t_i^{uv} > \lambda^{-1}) &= \Pr(t_i^{uv} > \lambda^{-1} \mid q = 1) \Pr(q = 1) + \Pr(t_i^{uv} > \lambda^{-1} \mid q > 1) \Pr(q > 1) \\ &\geq \frac{1}{2} \left(\frac{1}{e}\right) + \frac{1}{2} \left(\frac{2}{e}\right) = \frac{1.5}{e}. \end{aligned}$$

When $n_u(s_i) = n_v(s_i)$ we have derived:

$$\Pr(t_i^{uv} > \lambda^{-1}) \geq \Pr(t_i^{uv} > \lambda^{-1} \mid q = 1) = \frac{1}{e}.$$

Recall that $c(u, v)$ is the median of t_i^{uv} for $i = 1, \dots, \ell$. The probability that this median is less than λ^{-1} is bounded above by the probability of observing fewer than $\ell/2$ heads when tossing $\ell/10$ coins with bias $\frac{1}{e}$ and $9\ell/10$ coins with bias $\frac{1.5}{e}$. The expected number of heads in such an experiment is $\frac{0.1+(0.9)(1.5)}{e} = \frac{1.45}{e} > \frac{8}{15}$. Once again applying Chernoff's bound (to the random variable that counts the number of *tails*) the probability that at least $\ell/2$ tails are observed is bounded above by $\left(\frac{14}{15}e^{1/15}\right)^{\ell/2} < (0.999)^\ell$. \square

Combining Lemmas 3.4.1 and 3.4.2, and using the union bound, we find that with probability at least $1 - (n-1)c_1^\ell - \binom{n-1}{2}c_2c_3^\ell$, the set of pairs (u, v) such that $c(u, v) < \lambda^{-1}$ coincides with the set of edges of the tree G . Whenever the $n - 1$ cheapest edges in a graph form a spanning tree, it is always the minimum spanning tree of the graph. Thus, we have proven the following theorem.

Theorem 3.4.3. *If G is a tree, then Algorithm 1 perfectly reconstructs G with probability at least $1 - (n - 1)c_1^\ell - \binom{n-1}{2}c_2c_3^\ell$, for some absolute constants $c_1, c_3 < 1$ and $c_2 < \infty$. This probability can be made greater than $1 - 1/n^c$, for any specified $c > 0$, by using $\ell \geq c_4 \cdot c \cdot \log n$ traces, where $c_4 < \infty$ is an absolute constant.*

3.4.2 Bounded-Degree Graphs

In this section, we show that $O(\text{poly}(\Delta) \log n)$ complete traces suffice for perfect reconstruction (with high probability) when the graph G has maximum degree Δ . In fact, our proof shows a somewhat stronger result: it shows that for any pair of nodes u, v , there is an algorithm that predicts whether $\{u, v\}$ is an edge of G with failure probability at most $1 - 1/n^c$, for any specified constant $c > 0$, and the algorithm requires only $\Omega(\text{poly}(\Delta) \log n)$ independent partial traces in which u and v are both infected. However, for simplicity we will assume complete traces throughout this section.

Input: An infection rate parameter, λ .

A set of vertices, V .

An upper bound, Δ , on the degrees of vertices.

A collection T_1, \dots, T_ℓ of complete traces generated by repeatedly running the infection process on a fixed graph G with vertex set V and maximum degree Δ .

Let $t_i(v)$ denote the infection time of node v in trace T_i .

Output: An estimate, \hat{G} , of G .

- 1: **for all** nodes u **do**
- 2: **for all** sets $S \subseteq V \setminus \{u\}$ of at most Δ vertices **do**
- 3: **for all** traces T_i **do**
- 4: Let $S_i^u = \{v \in S \mid t_i(v) < t_i(u)\}$.
- 5: **if** $S_i^u = \emptyset$ **then**
- 6: Let $\text{score}_i(S, u) = 0$ if u is the source of T_i , otherwise $\text{score}_i(S, u) = -\infty$.
- 7: **else**
- 8: $\text{score}_i(S, u) = \log |S_i^u| - \lambda \sum_{v \in S_i^u} [t_i(u) - t_i(v)]$.
- 9: Let $\text{score}(S, u) = \ell^{-1} \cdot \sum_i \text{score}_i(S, u)$.
- 10: Let $R(u) = \text{argmax}\{\text{score}(S, u)\}$.
- 11: **for all** ordered pairs of vertices u, v **do**
- 12: **if** $t_i(v) < t_i(u)$ in at least $\ell/3$ traces and $v \in R(u)$ **then**
- 13: Insert edge $\{u, v\}$ into \hat{G} .
- 14: Output \hat{G} .

Algorithm 2: Bounded-degree reconstruction algorithm.

The basic intuition behind our algorithm can be summarized as follows. To determine if $\{u, v\}$ is an edge of G , we try to reconstruct the entire set of neighbors of u and

then test if v belongs to this set. We use the following insight to test whether a candidate set S is equal to the set $N(u)$ of all neighbors of u . Any such set defines a “forecasting model” that specifies a probability distribution for the infection time $t(u)$. To test the validity of the forecast we use a strictly proper scoring rule [58], specifically the logarithmic scoring rule, which is defined formally in the paragraph following Equation (3.1). Let us say that a set S differs significantly from the set of neighbors of u (henceforth denoted $N(u)$) if the symmetric difference $S \oplus N(u)$ contains a vertex that is infected before u with constant probability. We prove that the expected score assigned to $N(u)$ by the logarithmic scoring rule is at least $\Omega(\Delta^{-4})$ greater than the score assigned to any set that differs significantly from $N(u)$. Averaging over $\Omega(\Delta^4 \log \Delta \log n)$ trials is then sufficient to ensure that all sets differing significantly from $N(u)$ receive strictly smaller average scores.

The scoring rule algorithm thus succeeds (with high probability) in reconstructing a set $R(u)$ whose difference from $N(u)$ is insignificant, meaning that the elements of $R(u) \oplus N(u)$ are usually infected after u . To test if edge $\{u, v\}$ belongs to G , we can now use the following procedure: if the event $t(v) < t(u)$ occurs in a constant fraction of the traces containing both u and v , then we predict that edge $\{u, v\}$ is present if $v \in R(u)$; this prediction must be correct with high probability, as otherwise the element $v \in R(u) \oplus N(u)$ would constitute a significant difference. Symmetrically, if $t(u) < t(v)$ occurs in a constant fraction of the traces containing both u and v , then we predict that edge $\{u, v\}$ is present if $u \in R(v)$.

KL-divergence. For distributions p, q on \mathbb{R} having density functions f and g , respectively, their KL-divergence is defined by

$$D(p \parallel q) = \int f(x) \log \left(\frac{f(x)}{g(x)} \right) dx. \quad (3.1)$$

One interpretation of the KL-divergence is that it is the expected difference between

$\log(f(x))$ and $\log(g(x))$ when x is randomly sampled using distribution p . If one thinks of p and q as two forecasts of the distribution of x , and one samples x using p and applies the *logarithmic scoring rule*, which outputs a score equal to the log-density of the forecast distribution at the sampled point, then $D(p \parallel q)$ is the difference in the expected scores of the correct and the incorrect forecast. A useful lower bound on this difference is supplied by Pinsker’s Inequality:

$$D(p \parallel q) \geq 2 \|p - q\|_{\text{TV}}^2, \quad (3.2)$$

where $\|\cdot\|_{\text{TV}}$ denotes the total variation distance. In particular, the fact that $D(p \parallel q) > 0$ when $p \neq q$ means that the true distribution, p , is the unique distribution that attains the maximum expected score, a property that is summarized by stating that the logarithmic scoring rule is *strictly proper*.

Quasi-timestamps and conditional distributions From now on in this section, we assume $\lambda = 1$. This assumption is without loss of generality, since the algorithm’s behavior is unchanged if we modify its input by setting $\lambda = 1$ and multiplying the timestamps in all traces by λ ; after modifying the input in this way, the input distribution is the same as if the traces had originally been sampled using the infection process with parameter $\lambda = 1$.

Our analysis of Algorithm 2 hinges on understanding the conditional distribution of the infection time $t(u)$, given the infection times of its neighbors. Directly analyzing this conditional distribution is surprisingly tricky, however. The reason is that u itself may infect some of its neighbors, so conditioning on the event that a neighbor of u was infected at time t_0 influences the probability density of $t(u)$ in a straightforward way at times $t > t_0$ but in a much less straightforward way at times $t < t_0$. We can avoid this “backward conditioning” by applying the following artifice.

Recall the description of the infection process in terms of Dijkstra's algorithm in Section 3.2: edges sample i.i.d. edge lengths and the timestamps $t(v)$ are equal to the distance labels assigned by Dijkstra's algorithm when computing single-source shortest paths from source s . Now consider the sample space defined by the tuple of independent random edge lengths $y(v, w)$. For any vertices $u \neq v$, define a random variable $\mathring{t}(v)$ to be the distance label assigned to v when we *delete* u and its incident edges from G to obtain a subgraph $G - u$, and then we run Dijkstra's algorithm on this subgraph. One can think of $\mathring{t}(v)$ as the time when v would have been infected if u did not exist. We will call $\mathring{t}(v)$ the *quasi-timestamp of v* (with respect to u). If $N(u) = \{v_1, \dots, v_k\}$ is the set of neighbors of u , and if we sample a trace originating at a source $s \neq u$, then the executions of Dijkstra's algorithm in G and $G - u$ will coincide until the step in which u is discovered and is assigned the distance label $t(u) = \min_j \{\mathring{t}(v_j) + y(v_j, u)\}$. From this equation, it is easy to deduce a formula for the conditional distribution of $t(u)$ given the k -tuple of quasi-timestamps $\mathring{\mathbf{t}} = (\mathring{t}(v_j))_{j=1}^k$. Using the standard notation z^+ to denote $\max\{z, 0\}$ for any real number z , we have

$$\Pr(t(u) > t \mid \mathring{\mathbf{t}}) = \exp\left(-\sum_{j=1}^k (t - \mathring{t}(v_j))^+\right). \quad (3.3)$$

The conditional probability density is easy to calculate by differentiating the right side of (3.3) with respect to t . For any vertex set S not containing u , let $S\langle t \rangle$ denote the set of vertices $v \in S$ such that $\mathring{t}(v) < t$, and let $\rho(t, S) = |S\langle t \rangle|$. Then the conditional probability density function of $t(u)$ satisfies

$$f(t) = \rho(t, N(u)) \exp\left(-\sum_{j=1}^k (t - \mathring{t}(v_j))^+\right) \quad (3.4)$$

$$\log f(t) = \log(\rho(t, N(u))) - \sum_{v \in N(u)} (t - \mathring{t}(v))^+. \quad (3.5)$$

It is worth pausing here to note an important and subtle point. The information contained in a trace T is insufficient to determine the vector of quasi-timestamps $\mathring{\mathbf{t}}$, since quasi-timestamps are defined by running the infection process in the graph $G - u$, whereas

the trace represents the outcome of running the same process in G . Consequently, our algorithm does not have sufficient information to evaluate $\log f(t)$ at arbitrary values of t . Luckily, the equation

$$(t(u) - t(v))^+ = (t(u) - \mathring{t}(v))^+$$

holds for all $v \neq u$, since $\mathring{t}(v)$ differs from $t(v)$ only when both quantities are greater than $t(u)$. Thus, our algorithm has sufficient information to evaluate $\log f(t(u))$, and in fact the value $\text{score}_i(S, u)$ defined in Algorithm 2, coincides with the formula for $\log f(t(u))$ on the right side of (3.5), when $S = N(u)$ and $\lambda = 1$.

Analysis of the reconstruction algorithm. The foregoing discussion prompts the following definitions. Fix a vector of quasi-timestamps $\mathring{\mathbf{t}} = (\mathring{t}(v))_{v \neq u}$, and for any set of vertices S not containing u , let p^S be the probability distribution on \mathbb{R} with density function

$$f^S(t) = \rho(t, S) \exp \left(- \sum_{v \in S} (t - \mathring{t}(v))^+ \right). \quad (3.6)$$

One can think of p^S as the distribution of the infection time $t(u)$ that would be predicted by a forecaster who knows the values $\mathring{t}(v)$ for $v \in S$ and who believes that S is the set of neighbors of u . Letting $N = N(u)$, each timestamp $t_i(u)$ is a random sample from the distribution p^N , and $\text{score}_i(S, u)$ is the result of applying the logarithmic scoring rule to the distribution p^S and the random sample $t(u)$. Therefore

$$\mathbb{E}[\text{score}_i(N, u) - \text{score}_i(S, u)] = D(p^N \parallel p^S) \geq 2 \|p^N - p^S\|_{\text{TV}}^2. \quad (3.7)$$

The key to analyzing Algorithm 2 lies in proving a lower bound on the expected total variation distance between p^N and p^S . The following lemma supplies the lower bound.

Lemma 3.4.4. *Fix a vertex u , let $N = N(u)$ be its neighbor set, and fix some $S \subseteq V \setminus \{u\}$ distinct from N . Letting $\pi(S \oplus N, u)$ denote the probability that at least one*

element of the set $S \oplus N$ is infected before u , we have

$$\mathbb{E} (\|p^N - p^S\|_{\text{TV}}) \geq \frac{1}{10} \Delta^{-2} \pi(S \oplus N, u). \quad (3.8)$$

Proof. For a fixed vector of quasi-timestamps $(\mathring{t}(v))_{v \neq u}$ we can bound $\|p^N - p^S\|_{\text{TV}}$ from below by the following method. Let v_0 denote the vertex in $S \oplus N$ whose quasi-timestamp t_0 is earliest. Let b be the largest number in the range $0 \leq b \leq \frac{1}{\Delta}$ such that the open interval $I = (t_0, t_0 + b)$ does not contain the quasi-timestamps of any element of $S \cup N$. The value $|p^N(I) - p^S(I)|$ is a lower bound on $\|p^N - p^S\|_{\text{TV}}$.

One may verify by inspection that the density function $f^S(t)$ defined in equation (3.6) satisfies the differential equation $f^S(t) = \frac{d}{dt} (f^S(t)/\rho(t, S))$ for almost all t . By integrating both sides of the equation we find that for all t ,

$$1 - F^S(t) = \frac{f^S(t)}{\rho(t, S)} = \exp \left(- \sum_{v \in S} (t - \mathring{t}(v))^+ \right), \quad (3.9)$$

where F^S denotes the cumulative distribution function of p^S . A similar formula holds for F^N . Let $G = 1 - F^S(t_0) = 1 - F^N(t_0)$, where the latter equation holds because of our choice of t_0 . We have

$$\begin{aligned} p^S(I) &= G - (1 - F^N(t_0 + b)) \\ &= G \cdot (1 - e^{-\rho(t_0+b, S)b}) \\ p^N(I) &= G - (1 - F^S(t_0 + b)) \\ &= G \cdot (1 - e^{-\rho(t_0+b, N)b}), \end{aligned}$$

where the second and fourth lines follow from the formula (3.9), using the fact that none of the quasi-timestamps $\mathring{t}(v)$ for $v \in S \cup N$ occur in the interval $(t_0, t_0 + b)$.

Let $\rho = \rho(t_0, S) = \rho(t_0, N)$. We have

$$\begin{aligned} |p^N(I) - p^S(I)| &= G \cdot |e^{-\rho(t_0+b, N)b} - e^{-\rho(t_0+b, S)b}| \\ &= G e^{-\rho b} (1 - e^{-b}), \end{aligned} \quad (3.10)$$

using the fact that exactly one of $\rho(t_0 + b, S), \rho(t_0 + b, N)$ is equal to ρ and the other is equal to $\rho + 1$. To bound the right side of (3.10), we reason as follows. First, $\rho = |N\langle t_0 \rangle| \leq |N| \leq \Delta$. Second, $b \leq \Delta^{-1}$ by construction. Hence $e^{-\rho b} \geq e^{-1}$. Also, the inequality $1 - e^{-x} \geq (1 - e^{-1})x$ holds for all $x \in [0, 1]$, since the left side is a concave function, the right side is a linear function, and the two sides agree at $x = 0$ and $x = 1$. Thus, $1 - e^{-b} \geq (1 - e^{-1})b$, and we have derived

$$|p^N(I) - p^S(I)| \geq (e^{-1} - e^{-2})Gb.$$

To complete the proof of the lemma we need to derive a lower bound on the expectation of the product Gb . First note that $G = 1 - F^N(t_0)$ is the probability $t(u) > t_0$ when $t(u)$ is sampled from the distribution p^N . Since p^N is the conditional distribution of $t(u)$ given \mathfrak{t} , we can now take the expectation of both sides of the equation $G = 1 - F^N(t_0)$ and conclude that $\mathbb{E}[G] = \pi(S \oplus N, u)$. Finally, to place a lower bound on $\mathbb{E}[b | G]$, we reason as follows. In the infection process on $G - u$, let R denote the set of vertices in $S \cup N$ whose quasi-timestamps are strictly greater than t_0 . The number of edges joining R to the rest of $V \setminus \{u\}$ is at most $\Delta|R| < 2\Delta$, so the waiting time from t_0 until the next quasi-timestamp of an element of R stochastically dominates the minimum of $2\Delta^2$ i.i.d. $\text{Exp}(1)$ random variables. Thus the conditional distribution of b given G stochastically dominates the minimum of $2\Delta^2$ i.i.d. $\text{Exp}(1)$ random variables and the constant $1/\Delta$, so

$$\mathbb{E}[b|G] \geq \int_0^{1/\Delta} e^{-2\Delta^2 t} dt = \frac{1}{2}\Delta^{-2} [1 - e^{-2\Delta}] \geq \frac{1}{2}\Delta^{-2} [1 - e^{-2}].$$

Putting all of these bounds together, we obtain

$$\mathbb{E}(\|p^N - p^S\|_{\text{TV}}) \geq \frac{1}{2}\Delta^{-2}(e^{-1} - e^{-2})(1 - e^{-2})\pi(S \oplus N, u),$$

and the inequality (3.8) follows by direct calculation. \square

Combining Pinsker's Inequality with Lemma 3.4.4 we immediately obtain the following corollary.

Corollary 3.4.5. *If $N = N(u)$ and S is any set such that $\pi(S \oplus N, u) > 1/4$, then for each trace T_i the expected value of $\text{score}_i(N) - \text{score}_i(S)$ is $\Omega(\Delta^{-4})$.*

Using this corollary, we are ready to prove our main theorem.

Theorem 3.4.6. *For any constant $c > 0$, the probability that Algorithm 2 fails to perfectly reconstruct G , when given*

$$\ell = \Omega(\Delta^9 \log^2 \Delta \log n)$$

complete traces, is at most $1/n^c$.

Proof. Let us say that a set S differs significantly from $N(u)$ if $\pi(S \oplus N(u), u) > 1/4$. When ℓ is as specified in the theorem statement, with probability at least $1 - 1/n^{c+1}$, there is no vertex u such that the algorithm's estimate of u 's neighbor set, $R(u)$, differs significantly from $N(u)$. Indeed, when S, N satisfy the hypotheses of Corollary 3.4.5, the random variables $\text{score}_i(N) - \text{score}_i(S)$ are i.i.d. samples from a distribution that has expectation $\Omega(\Delta^{-4})$, is bounded above by $O(\log \Delta)$ with probability $1 - 1/\text{poly}(\Delta)$, and has an exponential tail. Exponential concentration inequalities for such distributions imply that for all $\delta > 0$, the average of $\ell = \Omega(\Delta^8 \log^2(\Delta) \log(1/\delta))$ i.i.d. samples will be non-negative with probability at least $1 - \delta$. Setting $\delta = n^{-\Delta-c-2}$ and taking the union bound over all vertex sets S of cardinality Δ or smaller, we conclude that when $\ell = \Omega(\Delta^9 \log^2(\Delta) \log n)$, the algorithm has less than n^{-c-2} probability of selecting a set $R(u)$ that differs significantly from $N(u)$. Taking the union bound over all vertices u we obtain a proof of the claim stated earlier in this paragraph: with probability $1 - 1/n^{c+1}$, there is no u such that $R(u)$ differs significantly from $N(u)$.

Let us say that an ordered pair of vertices (u, v) violates the *empirical frequency property* if the empirical frequency of the event $t_i(v) < t_i(u)$ among the traces T_1, \dots, T_ℓ differs by more than $\frac{1}{12}$ from the probability that $t(v) < t(u)$ in a random trace. The probability of any given pair (u, v) violating this property is exponentially small in ℓ , hence we can assume it is less than $1/n^{c+3}$ by taking the constant inside the $\Omega(\cdot)$ to be sufficiently large. Summing over pairs (u, v) , the probability that there exists a pair violating the empirical frequency property is less than $1/n^{c+1}$ and we henceforth assume that no such pair exists.

Assuming that no set $R(u)$ differs significantly from $N(u)$ and that no pair (u, v) violates the empirical frequency property, we now prove that the algorithm's output, \hat{G} , is equal to G . If $\{u, v\}$ is an edge of G , assume without loss of generality that the event $t(v) < t(u)$ has probability at least $1/2$. By the empirical frequency property, at least $\ell/3$ traces satisfy $t_i(v) < t_i(u)$. Furthermore, v must belong to $R(u)$, since if it belonged to $R(u) \oplus N(u)$ it would imply that $\pi(R(u) \oplus N(u), u) \geq \Pr(t(v) < t(u)) \geq 1/2$, violating our assumption that $R(u)$ doesn't differ significantly from $N(u)$. Therefore $v \in R(u)$ and the algorithm adds $\{u, v\}$ to \hat{G} . Now suppose $\{u, v\}$ is an edge of \hat{G} , and assume without loss of generality that this edge was inserted when processing the ordered pair (u, v) . Thus, at least $\ell/3$ traces satisfy $t_i(v) < t_i(u)$, and $v \in R(u)$. By the empirical frequency property, we know that a random trace satisfies $t(v) < t(u)$ with probability at least $1/4$. As before, if v belonged to $R(u) \oplus N(u)$ this would violate our assumption that $R(u)$ does not differ significantly from $N(u)$. Hence $v \in N(u)$, which means that $\{u, v\}$ is an edge of G as well. \square

3.5 Experimental Analysis

In the preceding sections we have established trace complexity results for various network inference tasks. In this section, our goal is to assess our predictions on real and synthetic social and information networks whose type, number of nodes, and maximum degree (Δ) we now describe.

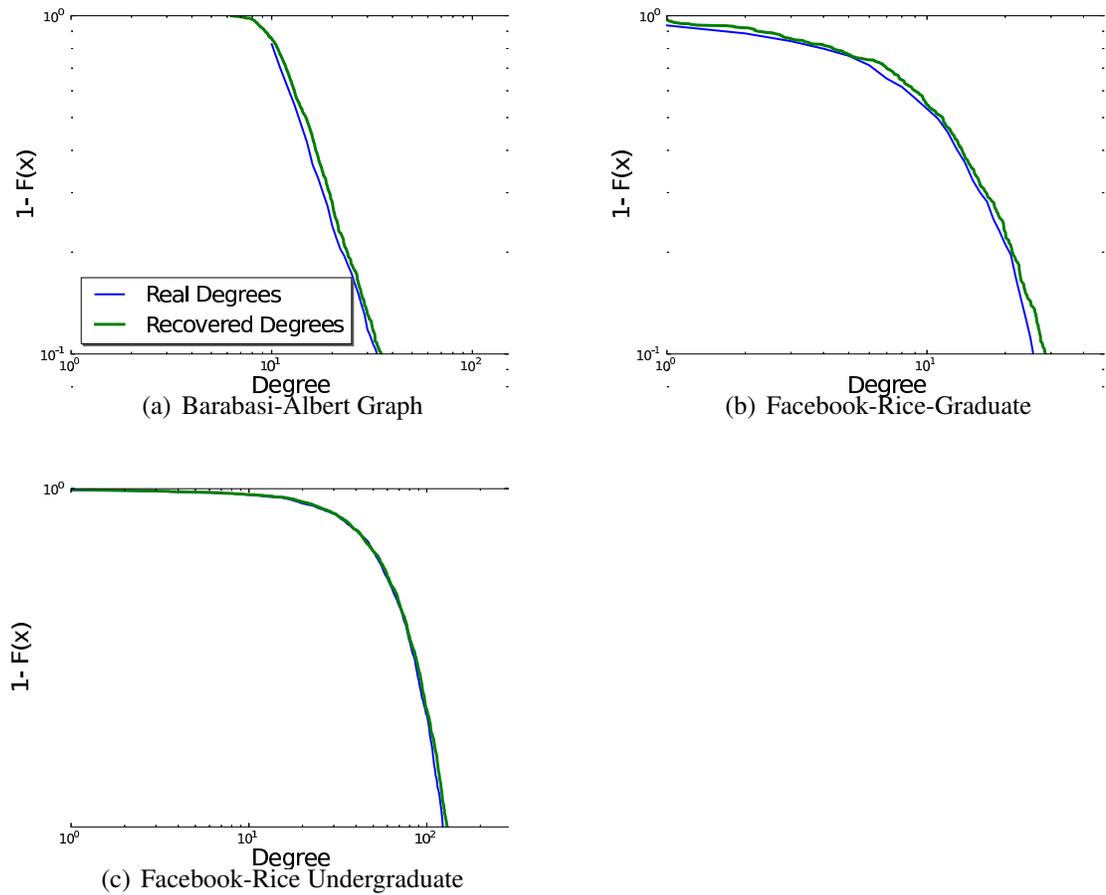


Figure 3.1: Complementary cumulative density function (CCDF) of degree reconstruction using $\Omega(n)$ traces for (a) a synthetic network with 1,024 nodes generated using the Barabasi-Albert algorithm, and two real social networks: two subsets of the Facebook network comprising 503 graduate students (a) and 1220 undergraduate students (c), respectively, from Rice University.

We use two real social networks, namely two Facebook subnetworks comprising 503 ($\Delta = 48$) graduate and 1220 ($\Delta = 287$) undergraduate students, respectively [98]. We also generate three synthetic networks, each possessing 1024 vertices, whose generative models frequently arise in practice in the analysis of networks. We generated a *Barabasi-Albert Network* [22] ($\Delta = 174$), which is a preferential attachment model, a *$G_{(n,p)}$ Network* [48] ($\Delta = 253$) with $p = 0.2$, and a *Power-Law Tree*, whose node degree distribution follows a power-law distribution with exponent 3 ($\Delta = 94$).

First, we evaluate the performance of the algorithm to reconstruct the degree distribution of networks without inferring the network itself (Section 3.3.3). Figure 3.1 shows the reconstruction of the degree distribution using $\Omega(n)$ traces of the Barabasi-Albert Network and the two Facebook subnetworks. We used $10n$ traces, and the plots show that the CCDF curves for the real degrees and for the reconstructed distribution have almost perfect overlap.

Turning our attention back to network inference, the $\Omega(n\Delta^{1-\epsilon})$ lower-bound established in Section 3.2 tells us that the First-Edge algorithm is nearly optimal for perfect network inference in the general case. Thus, we assess the performance of our algorithms against this limit. The performance of First-Edge is notoriously predictable: if we use ℓ traces where ℓ is less than the total number of edges in the network, then it returns nearly ℓ edges which are all true positives, and it never returns false positives.

If we allow false positives, we can use heuristics to improve the First-Edge’s recall. To this end, we propose the following heuristic that uses the degree distribution reconstruction algorithm (Section 3.3.3) in a pre-processing phase, and places an edge in the inferred network provided the edge has probability at least p of being in the graph. We call this heuristic *First-Edge+*.

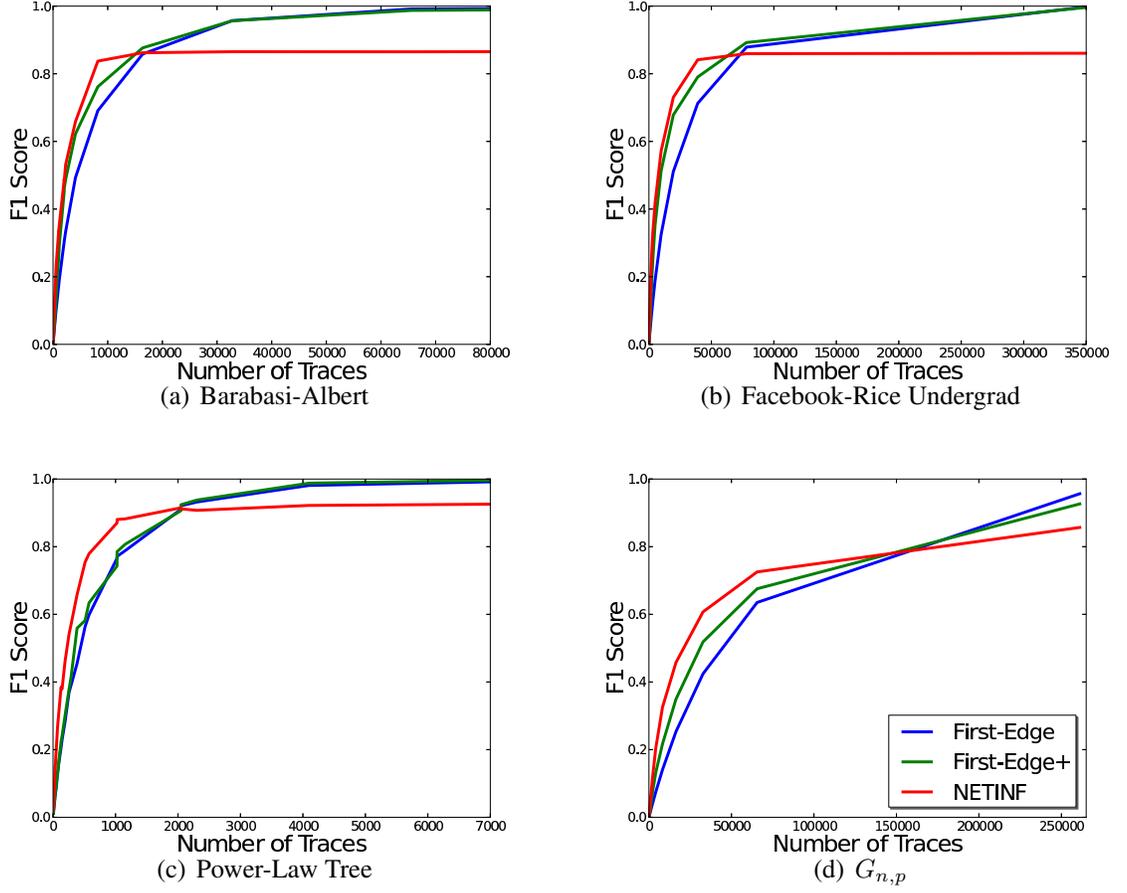


Figure 3.2: F1 score of the First-Edge, First-Edge+, and NETINF algorithms applied to different real and synthetic networks against a varying number of traces. (best viewed in color)

In First-Edge+, we use the memoryless property of the exponential distribution to establish the probability p of an edge pertaining to a network G . The algorithm works as follows. Consider a node u that appears as the root of a trace at time $t_0 = 0$. When u spreads the epidemic, some node v is going to be the next infected at time t_1 , which was sampled from an exponential distribution with parameter λ . At time t_1 , notice that there are exactly $d_u - 1$ nodes waiting to be infected by u , and exactly $d_v - 1$ waiting to be infected by v , where d_u and d_v are the degrees of u and v respectively. At time t_1 any of these nodes is equally likely to be infected, due to the memoryless property.

Moreover, the next node w that appears in a trace after time t_1 is going to be infected by u with probability $p_{(u,w)} = \frac{d_u-1}{d_u+d_v-2}$ and by v with probability $p_{(v,w)} = \frac{d_v-1}{d_u+d_v-2}$. We can approximate⁵ this reasoning for larger prefixes of the trace: given a sequence u_1, \dots, u_k of infected nodes starting at the source of the epidemic, the probability that u_{k+1} is a neighbor of u_i is roughly $p_{(u_i, u_{k+1})} \simeq \frac{d_{u_i}}{\sum_j d_{u_j}}$. Therefore, for every segment of a trace that starts at the source, we infer an edge (u, v) if $p_{(u,v)} > p$, computed using the reconstructed degrees, where p is a tunable parameter. In our experiments we arbitrarily chose $p = 0.5$.

Note that First-Edge+ may not terminate as soon as we have inferred enough edges, even in the event that all true positives have been found, an effect that degrades its precision performance. To prevent this, we keep a variable T , which can be thought of as the *temperature* of the inference process. Let M be a counter of the edges inferred at any given time during the inference process, and \hat{E} be an estimate of the total number of edges, computed using the degree reconstruction algorithm in the pre-processing phase. We define $T = \frac{M}{\hat{E}}$ and run the algorithm as long as $T < 1.0$. In addition, whenever we infer a new edge, we flip a coin and remove, with probability T , a previously inferred edge with the lowest estimated probability of existence. Thus, while the network is “cold”, i.e., many undiscovered edges, edges are rapidly added and a few are removed, which boosts the recall. When the network is “warm”, i.e., the number of inferred edges approaches $|E|$, we carefully select edges by exchanging previously inferred ones with better choices, thereby contributing to the precision.

Figure 3.2 contrasts the performance of First-Edge, First-Edge+ and an existing network algorithm, NETINF [60], with respect to the F1 measure. NETINF requires the number of edges in the network as input, and thus we give it an advantage, by setting

⁵The exact probability depends on the number of edges between each of the nodes u_1, \dots, u_k and the rest of the graph.

the number of edges to the true cardinality of edges for each network.

In Figures 3.2(a) and 3.2(b), we observe that, as First-Edge+ and NETINF are less conservative, their F1 performances have an advantage over First-Edge for small numbers of traces, with First-Edge+ approaching the performance to NETINF. Interestingly, in Figure 3.2(c), we see that First-Edge and First-Edge+ achieve perfect tree inference with roughly 5,000 traces, which reflects a trace complexity in $\Omega(n)$ rather than in $O(\log n)$, which is the trace complexity of Algorithm 1.⁶ This result illustrates the relevance of the algorithms for special cases we developed in Section 3.4. Last, we observe that $G_{n,p}$ random graphs seem to have very large trace complexity. This is shown in Figure 3.2(d), where neither our algorithms nor NETINF can achieve high inference performance, even for large numbers of traces.

In accordance with our discussion in Section 3.3.1, we confirm that, in practice, we need significantly fewer than $n * \Delta$ traces for inferring most of the edges. It is perhaps surprising that First-Edge+, which is extremely simple, achieves comparable performance to the more elaborate counterpart, NETINF. In addition, while NETINF reaches a plateau that limits its performance, First-Edge+ approaches perfect inference as the number of traces goes to $\Omega(n\Delta)$. In the cases in which NETINF achieves higher performance than First-Edge+, the latter is never much worse than the former. This presents a practitioner with a trade-off between the two algorithms. For large networks, while First-Edge+ is extremely easy to implement and makes network inferences (in a preemptive fashion) in a matter of seconds, NETINF takes a couple of hours to run to completion and requires the implementation of an elaborate algorithm.

⁶In our experiments Algorithm 1 consistently returned the true edge set without false positives with $O(\log n)$ traces for various networks of various sizes. Therefore, in the interest of space we omit the data from these experiments.

3.6 Discussion

Our goal is to provide the building blocks for a rigorous foundation to the rapidly-expanding network inference topic. Previous works have validated claims through experiments on relatively small graphs as compared to the large number of traces utilized, whereas the relation that binds these two quantities remains insufficiently understood. Accordingly, we believe that a solid foundation for the network inference problem remains a fundamental open question, and that works like [102], as well as ours, provide the initial contributions toward that goal.

Our results have direct applicability in the design of network inference algorithms. More specifically, we rigorously study how much useful information can be extracted from a trace for network inference, or more generally, the inference of network properties without reconstructing the network, such as the node degree distribution. We first show that, to perfectly reconstruct general graphs, nothing better than looking at the first pair of infected nodes in a trace can really be done. We additionally show that the remainder of a trace contains rich information that can reduce the trace complexity of the task for special case graphs. Finally, we build on the previous results to develop extremely simple and efficient reconstruction algorithms that exhibit competitive inference performance with the more elaborate and computationally costly ones.

Some open technical questions stemming from our work are immediately apparent. For instance, what is the true lower bound for perfect reconstruction? Is it $O(n^2)$, $O(n\Delta)$ or some other bound which, in the case of the clique, reduces to what we have shown? And, are there other meaningful statistics apart from the degree distribution that can be efficiently recovered? For graphs with maximum degree Δ , our perfect reconstruction algorithm has running time exponential in Δ : is this exponential dependence necessary?

And while the algorithm's trace complexity is polynomial in Δ , the upper bound of $\tilde{O}(\Delta^9)$ proven here is far from matching the lower bound $\Omega(\Delta^{2-\epsilon})$; what is the correct dependence of trace complexity on Δ ? The bounded-degree restriction, while natural, is unlikely to be satisfied by real-world networks; is perfect reconstruction possible for the types of graphs that are likely to occur in real-world situations?

Perhaps the most relevant avenue for future research in our context is to go beyond the notion of perfect reconstruction. This notion, while quite reasonable as a first step, is not flexible enough to be deployed in practical situations. One would need to take into account the possibility of accepting some noise, i.e. some false positives as well as false negatives. So the main issue is to look for algorithms and lower bounds that are expressed as a function of the precision and recall one is willing to accept in an approximate reconstruction algorithm.

Finally, it would be very interesting to develop similar results, or perhaps even a theory, of trace complexity for other types of information spreading dynamics.

3.7 Appendix: Proofs from Section 3.3.2

We start by proving Lemma 3.3.2, which is the combinatorial heart of our lower bound.

Proof of Lemma 3.3.2. Points 1, 2, 4 are either obvious or simple consequences of point 3, which we now consider. For $a \geq 2$, let $P_{a,b}$ be the probability that 1, 2 appear (in any order) in the positions a, b conditioned on the starting node being different from 1, 2. Moreover, let $P_{1,b}$ the probability that 1, 2 appear (in any order) in the positions 1, b conditioned on the starting node being one of 1, 2. Then $p_{a,b} = \frac{n-2}{n}P_{a,b}$, for $a \geq 2$, and $p_{1,b} = \frac{2}{n}P_{1,b}$.

We now compute $P_{a,b}$. First, we assume $a = 1$. Then

$$P_{1,b} = \prod_{i=2}^{b-1} \frac{i \cdot (n-i-1)}{(i-1) \cdot (n-i) + (n-i-1)} \cdot \frac{b-1}{(b-1)(n-b) + (n-b-1)}.$$

Now assume that $a \geq 2$. We have:

$$\begin{aligned} P_{a,b} &= \prod_{i=1}^{a-1} \frac{i(n-i-2)}{i(n-i)} \cdot \frac{a2}{a(n-a)} \\ &\cdot \prod_{i=a+1}^{b-1} \frac{i \cdot (n-i-1)}{(i-1)(n-i) + (n-i-1)} \cdot \frac{b-1}{(b-1)(n-b) + (n-b-1)}. \end{aligned}$$

By telescoping, specifically by $\prod_{i=s}^t \frac{n-i-2}{n-i} = \frac{(n-t-1)(n-t-2)}{(n-s)(n-s-1)}$, we can simplify the expression to:

$$P_{a,b} = 2 \cdot \frac{n-a-1}{(n-1)(n-2)} \cdot \prod_{i=a+1}^{b-1} \frac{i \cdot (n-i-1)}{(i-1)(n-i) + (n-i-1)} \cdot \frac{b-1}{(b-1)(n-b) + (n-b-1)}.$$

Moreover, by trying to simplify the product term, and by collecting the binomial, we get:

$$P_{a,b} = \frac{n-a-1}{\binom{n-1}{2}} \cdot \prod_{i=a+1}^{b-1} \frac{1}{1 + \frac{i-1}{i(n-i-1)}} \cdot \frac{b-1}{(b-1)(n-b) + (n-b-1)} \quad a \geq 2$$

Then, observe that for each $a \geq 1$, $b > a$, (and, if $a = 1$, $b \geq 3$) we have:

$$p_{a,b} = \left(1 \pm O\left(\frac{1}{n}\right)\right) \cdot \frac{n-a-1}{\binom{n}{2}} \cdot \prod_{i=a+1}^{b-1} \frac{1}{1 + \frac{i-1}{i(n-i-1)}} \cdot \frac{b-1}{(b-1)(n-b) + (n-b-1)}$$

We highlight the product inside $p_{a,b}$'s expression:

$$\begin{aligned}
\pi_{a,b} &= \prod_{i=a+1}^{b-1} \frac{1}{1 + \frac{i-1}{i(n-i-1)}} = \prod_{i=a+1}^{b-1} \frac{1}{1 - \frac{1}{i(n-i)}} \cdot \prod_{i=a+1}^{b-1} \left(\frac{1}{1 + \frac{i-1}{i(n-i-1)}} \cdot \left(1 - \frac{1}{i(n-i)} \right) \right) \\
&= \prod_{i=a+1}^{b-1} \frac{1}{1 - \frac{1}{i(n-i)}} \cdot \prod_{i=a+1}^{b-1} \left(\frac{i(n-i-1)}{i(n-i)-1} \cdot \frac{i(n-i)-1}{i(n-i)} \right) \\
&= \prod_{i=a+1}^{b-1} \frac{1}{1 - \frac{1}{i(n-i)}} \cdot \prod_{i=a+1}^{b-1} \frac{n-i-1}{n-i} = \prod_{i=a+1}^{b-1} \frac{1}{1 - \frac{1}{i(n-i)}} \cdot \prod_{i=a+1}^{b-1} \frac{1}{1 + \frac{1}{n-i-1}} \\
&= \frac{n-b}{n-a-1} \cdot \prod_{i=a+1}^{b-1} \frac{1}{1 - \frac{1}{i(n-i)}}.
\end{aligned}$$

We take the product of the denominators of the ratios, obtaining:

$$\prod_{i=a+1}^{b-1} \left(1 - \frac{1}{i(n-i)} \right) \geq \prod_{i=1}^{n-1} \left(1 - \frac{1}{i(n-i)} \right) \geq 1 - O\left(\frac{\ln n}{n}\right).$$

Therefore, we have

$$\pi_{a,b} = \left(1 + O\left(\frac{\ln n}{n}\right) \right) \frac{n-b}{n-a-1}.$$

We now turn back to $p_{a,b}$ expressions. Plugging in our approximation of $\pi_{a,b}$, we get:

$$p_{a,b} = \frac{1 + O\left(\frac{\ln n}{n}\right)}{\binom{n}{2}} \cdot \frac{b-1}{b-1 + \frac{n-b-1}{n-b}}.$$

The term $\frac{b-1}{b-1 + \frac{n-b-1}{n-b}}$ is bounded within $1 - \frac{1}{b}$ and 1. Therefore, if $a \geq 2, b \geq a+1$ (and if $a = 1, b \geq 3$), we have:

$$p_{a,b} = \frac{1 + O\left(\frac{\ln n}{n}\right) - O\left(\frac{1}{b}\right)}{\binom{n}{2}}.$$

□

Before moving on the two main Lemmas, we state (a corollary of) the Berry-Esseen Theorem [27, 49] which will be a crucial part of their proofs.

Theorem 3.7.1 (Berry-Esseen [27, 49]). *Let Z_1, \dots, Z_n be independent random variables, such that $E[Z_i] = 0$ for each $i = 1, \dots, n$, and such that $A = \sum_{i=1}^n E[|Z_i|^3]$ is finite. Then, if we let $B = \sqrt{\sum_{i=1}^n E[Z_i^2]}$ and $Z = \sum_{i=1}^n Z_i$, we have that*

$$\Pr[Z > \delta \cdot B] \geq \frac{1}{2} - \Theta\left(\delta + \frac{A}{B^3}\right),$$

and

$$\Pr[Z < -\delta \cdot B] \geq \frac{1}{2} - \Theta\left(\delta + \frac{A}{B^3}\right).$$

We will now use our Lemma 3.3.2, and the Berry-Esseen Theorem (Theorem 3.7.1), to prove Lemma 3.3.3.

Proof of Lemma 3.3.3. Let $\ell_{a,b}$ be the number of traces having one of the nodes in $\{1, 2\}$ in position a , and the other in position b . Then, $\sum_{b=2}^n \sum_{a=1}^{b-1} \ell_{a,b} = \ell$. We start by computing the likelihoods $\mathcal{L}_0, \mathcal{L}_1$ of \mathcal{P} assuming that the unknown graph is, respectively, G_0 or G_1 . We proved in Lemma 3.3.2, that the two likelihood of a trace only depends on the positions of 1 and 2. Therefore, if $p_{a,b}$ is the probability of obtaining a trace with 1, 2-positions equal to a, b in the graph G_1 , we have:

$$\frac{L_0(\mathcal{P})}{L_1(\mathcal{P})} = \frac{\binom{n}{2}^{-1}}{\prod_{a=1}^{n-1} \prod_{b=a+1}^n p_{a,b}^{\ell_{a,b}}} = \prod_{a=1}^{n-1} \prod_{b=a+1}^n (1 + d(a, b))^{-\ell_{a,b}}.$$

Regardless of the unknown graph, we have that the probability that there exists $b < \beta = \Theta(\sqrt{\log n})$ for which there exists at least one $a < b$ such that $\ell_{a,b} > 0$ is at most $O\left(\ell \cdot \frac{\beta^2}{n^2}\right) = o(\log^{-1} n)$. We condition on the opposite event. Then,

$$\begin{aligned} \mathcal{R} &= \ln \frac{L_0(\mathcal{P})}{L_1(\mathcal{P})} = \frac{\binom{n}{2}^{-1}}{\prod_{b=\beta}^n \prod_{a=1}^{b-1} p_{a,b}^{\ell_{a,b}}} = - \sum_{b=\beta}^n \sum_{a=1}^{b-1} (\ell_{a,b} \ln(1 + d(a, b))) \\ &= - \sum_{b=\beta}^n \sum_{a=1}^{b-1} (\ell_{a,b} (d(a, b) + O(d(a, b)^2))), \end{aligned}$$

where $d : \binom{[n]}{2} \rightarrow \mathbf{R}$ be the function defined in the statement of Lemma 3.3.2.

We aim to prove that the random variable \mathcal{R} is sufficiently anti-concentrated that, for any unknown graph G_i , the probability that $\Pr[\mathcal{R} > 0 | G_i] > \frac{1}{2} \pm o(1)$ and $\Pr[\mathcal{R} < 0 | G_i] > \frac{1}{2} \pm o(1)$. This will prove that one cannot guess what is the unknown graph with probability more than $\frac{1}{2} \pm o(1)$.

First, let X_0 and X_1 be two random variable having support $\binom{[n]}{2}$, the first uniform and the second distributed like the distribution p of Lemma 3.3.2.

We will compute a number of expectations so to finally apply Berry-Esseen theorem. First, recall that $\sum_{b=2}^n \sum_{a=1}^{b-1} d(a, b) = 0$. Therefore,

$$E[d(X_0)] = 0.$$

Moreover,

$$E[d(X_1)] = \sum_{b=2}^n \sum_{a=1}^{b-1} (p_{a,b} \cdot d(a, b)) = \sum_{b=2}^n \sum_{a=1}^{b-1} \left(\frac{1 + d(a, b)}{\binom{n}{2}} \cdot d(a, b) \right).$$

Recall that $\sum_{b=2}^n \sum_{a=1}^{b-1} d(a, b) = 0$. Then

$$E[d(X_1)] = \binom{n}{2}^{-1} \cdot \sum_{b=2}^n \sum_{a=1}^{b-1} d(a, b)^2.$$

Therefore, $E[d(X_1)] \geq 0$. Moreover,

$$E[d(X_1)] \leq O \left(\sum_{b=2}^{\Theta(n/\ln n)} \left(\frac{b}{n^2} \cdot \left(\frac{1}{b} \right)^2 \right) + \sum_{b=\Theta(n/\ln n)}^n \left(\frac{b}{n^2} \cdot \left(\frac{\ln n}{n} \right)^2 \right) \right) = O \left(\frac{\ln^2 n}{n^2} \right).$$

It follows that, for $i = 0, 1$, we have

$$0 \leq E[d(X_i)] \leq O \left(\frac{\ln^2 n}{n^2} \right).$$

We now move to the second moments. First observe that, for $i = 0, 1$,

$$E[d(X_i)^2] = \Theta \left(\binom{n}{2}^{-1} \sum_{b=2}^n \sum_{a=1}^{b-1} p_{a,b}^2 \right).$$

We lower bound both $E[d(X_0)^2]$ and $E[d(X_1)^2]$ with

$$E[d(X_i)^2] = \Omega \left(\sum_{b=\Theta(n/\ln n)}^n \left(\frac{b}{n^2} \cdot \left(\frac{\ln n}{n} \right)^2 \right) \right) = \Omega \left(\frac{\ln^2 n}{n^2} \right).$$

Analogously, we have that $E[d(X_0)^2]$ and $E[d(X_1)^2]$ can both be upper bounded by

$$O \left(\sum_{b=2}^{\Theta(n/\ln n)} \left(\frac{b}{n^2} \cdot \left(\frac{1}{b} \right)^2 \right) + \sum_{b=\Theta(n/\ln n)}^n \left(\frac{b}{n^2} \cdot \left(\frac{\ln n}{n} \right)^2 \right) \right) = O \left(\frac{\ln^2 n}{n^2} \right).$$

We then have, for $i = 0, 1$,

$$E[d(X_i)^2] = \Theta \left(\frac{\ln^2 n}{n^2} \right).$$

Moreover, the variance of $d(X_i)$, $i = 0, 1$, is equal to $S^2 = \Theta \left(\frac{\ln^2 n}{n^2} \right)$.

By linearity of expectation, regardless of the unknown graph, if we let $C = \Theta \left(\ell \cdot \frac{\ln^2 n}{n^2} \right)$, we have that

$$-C \leq E[\mathcal{R}] \leq C.$$

We upper bound both $E[|d(X_0)|^3]$ and $E[|d(X_1)|^3]$ with

$$\begin{aligned} O \left(\sum_{b=2}^{\Theta(n/\ln n)} \left(\frac{b}{n^2} \cdot \left(\frac{1}{b} \right)^3 \right) + \sum_{b=\Theta(n/\ln n)}^n \left(\frac{b}{n^2} \cdot \left(\frac{\ln n}{n} \right)^3 \right) \right) \\ \leq O \left(\frac{1}{n^2} + \frac{\ln^3 n}{n^3} \right) = O \left(\frac{1}{n^2} \right). \end{aligned}$$

It follows that

$$\begin{aligned} K &= E[|d(X_i) - E[d(X_i)]|^3] \leq E[\max(8|d(X_i)|^3, 8|E[d(X_i)]|^3)] \\ &\leq O \left(\max \left(E[|d(X_i)|^3], \frac{\ln^6 n}{n^6} \right) \right) \leq O \left(\frac{1}{n^2} \right). \end{aligned}$$

Now, we apply the Berry-Esseen bound with $A \leq \ell K = o \left(\frac{1}{\ln^2 n} \right)$ and $B = \Theta(\sqrt{\ell S^2}) = \Theta(1)$. We compute the error term of the Berry-Esseen theorem:

$$\frac{A}{B^3} \leq O \left(\frac{1}{\ln^2 n} \right) = o(1).$$

Therefore, \mathcal{R} will behave approximately like a gaussian in a radius of (at least) $\omega(1)$ standard deviations B around its mean. Observe that the standard deviation B satisfies $B = \Theta(\sqrt{C})$. Since $C = o(1)$ (by $\ell = o(\frac{n}{\ln^2 n})$), we have $B = \omega(C)$. Therefore, regardless of the unknown graph, the probability that \mathcal{R} will be positive is $\frac{1}{2} \pm o(1)$. \square

We finally prove Lemma 3.3.4, which deals with the likelihoods of the waiting times in the traces.

Proof of Lemma 3.3.4. Let $f_0(x) = te^{-tx}$ and $f_1(x) = (t-1)e^{-(t-1)x}$ be two exponential density functions with parameters t and $t-1$. Since we will be considering ratio of likelihoods, we compute the ratio of the two densities:

$$\frac{f_0(x)}{f_1(x)} = \left(1 + \frac{1}{t-1}\right) \cdot e^{-x}.$$

Observe that, if $q = o(t)$, it holds that

$$\left(\frac{f_0(\frac{1}{t})}{f_1(\frac{1}{t})}\right)^q = 1 + \frac{q}{2t^2} + o\left(\frac{q}{t^2}\right). \quad (3.11)$$

Let $\delta_{i,j} = 1$ if, in the j th trace, exactly one of the nodes in $\{1, 2\}$ was within the first i informed nodes; otherwise, let $\delta_{i,j} = 0$. Then $\ell_i = \sum_{j=1}^{\ell} \delta_{i,j}$.

For $i = 1, \dots, n-1$ and $j = 1, \dots, \ell_i$, let $t_{i,j}$ be the time waited (from the last time a node was informed) to inform the $(i+1)$ th node in the j th of the traces having exactly one of the two nodes 1, 2 in the first i positions. By the memoryless property of the exponential random variables, and by the fact that the minimum of n iid $\text{Exp}(\lambda)$ random variables is distributed like $\text{Exp}(n\lambda)$, we have that (once we condition on the ℓ_i 's) the $t_{i,j}$'s variables are independent, and that $t_{i,j}$ is distributed like $\text{Exp}(c\lambda)$ where c is the size of the cut induced by the first i nodes of the j th trace (of those having

one of the nodes 1, 2 within the first i nodes). Further, from the scaling property of the exponential random variables, we have that $\lambda t_{i,j}$ is distributed like $\text{Exp}(c)$.

Let $T = \lambda \cdot \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} t_{i,j} \delta_{i,j}$. Let T_0 be the random variable T conditioned on G_0 , and let T_1 be the random variable T conditioned on G_1 (observe that, since T is conditioned on $\ell_1, \dots, \ell_{n-1}$, both T_0 and T_1 will also be conditioned on $\ell_1, \dots, \ell_{n-1}$).

Then,

$$T_0 = \sum_{\substack{i,j \\ \delta_{i,j}=1}} (\lambda t_{i,j}) = \sum_{\substack{i,j \\ \delta_{i,j}=1}} \text{Exp}(i \cdot (n - i)) = \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} \text{Exp}(i \cdot (n - i)),$$

$$T_1 = \sum_{\substack{i,j \\ \delta_{i,j}=1}} (\lambda t_{i,j}) = \sum_{\substack{i,j \\ \delta_{i,j}=1}} \text{Exp}(i \cdot (n - i) - 1) = \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} \text{Exp}(i \cdot (n - i) - 1).$$

Now, let X be distributed like $\text{Exp}(x)$, for some $x > 0$. In general, we have $E[X^k] = \frac{k!}{x^k}$. If we let $Y = X - E[X]$, we obtain $E[Y] = 0$. Moreover, we have

$$E[Y^2] = E[X^2] - 2E[X]E[X] + E[X]^2 = E[X^2] - E[X]^2 = \frac{2}{x^2} - \frac{1}{x^2} = x^{-2}.$$

We also have,

$$\begin{aligned} E[|Y|^3] &= E[|X - E[X]|^3] \\ &\leq E[(X + E[X])^3] = E[X^3 + 3X^2E[X] + 3XE[X]^2 + E[X]^3] \\ &= E[X^3] + 3E[X^2]E[X] + 4E[X]^3 = 16x^{-3}, \end{aligned}$$

where the inequality follows from $X \geq 0$.

Let us consider T_k , $k = 0, 1$. They are the sum, over $i = 1, \dots, n - 1$, of ℓ_i independent exponential random variables with parameters $i \cdot (n - i) - k$. If we center each of those variables in 0 (that is, we remove the expected value of each exponential

variable), obtaining — say — variables $Y_{k,i,j} = \text{Exp}(i \cdot (n - i) - k) - \frac{1}{i \cdot (n-i) - k}$, we can write T_k as:

$$T_k = \sum_{i=1}^{n-1} \frac{\ell_i}{i \cdot (n - i) - k} + \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} Y_{k,i,j} = E[T_k] + \sum_{i=1}^{n-1} \sum_{j=1}^{\ell_i} Y_{k,i,j}.$$

Let us bound the absolute difference between the expected values of T_0 and T_1 , recalling that $\ell_i = \Theta(\alpha i(n - i))$, for each $i = 1, \dots, n - 1$:

$$E[T_1] - E[T_0] = \sum_{i=1}^{n-1} \left(\ell_i \cdot \left(\frac{1}{i(n - i) - 1} - \frac{1}{i(n - i)} \right) \right) \quad (3.12)$$

$$= \sum_{i=1}^{n-1} \frac{\ell_i}{i^2(n - i)^2 - i(n - i)} = \Theta \left(\alpha \cdot \frac{\log n}{n} \right) = D. \quad (3.13)$$

We now aim to show that $\Pr[T_k > E[T_0]] = \frac{1}{2} \pm o(1)$, for both $k = 0$ and $k = 1$. To do so, we use the Berry-Esseen theorem (Theorem 3.7.1). We apply it to our collection of variables $Y_{k,i,j}$ (which will play the Z_i variables' role in the Berry-Esseen theorem). We get:

$$A \leq O \left(\sum_{i=1}^{n-1} \frac{\ell_i}{i^3(n - i)^3} \right),$$

and

$$B = O \left(\sqrt{\sum_{i=1}^{n-1} \frac{\ell_i}{i^2(n - i)^2}} \right).$$

Therefore,

$$A \leq O \left(\alpha \cdot \sum_{i=1}^{n-1} \frac{1}{i^2(n - i)^2} \right) = O(\alpha \cdot n^{-2}),$$

and

$$B = \Theta \left(\sqrt{\alpha \cdot \sum_{i=1}^{n-1} \frac{1}{i(n - i)}} \right) = \Theta \left(\sqrt{\alpha \cdot \frac{\log n}{n}} \right).$$

Therefore, $\frac{A}{B^3} \leq O\left((\alpha n \log^3 n)^{-\frac{1}{2}}\right)$. We set $\delta = \frac{1}{\sqrt{\log n}}$. Then, for each $k, k' \in \{0, 1\}$, the probability that T_k exceeds $E[T_{k'}]$ by at least $\Omega\left(\sqrt{\frac{\alpha \log n}{n}}\right)$ is at least $\frac{1}{2} - \Theta\left(\frac{1}{\sqrt{\log n}}\right) = \frac{1}{2} - o(1)$.

Now consider, the likelihoods L_0, L_1 of the sequence of waiting times with each of the two graphs. We have:

$$L_0(\mathcal{W}) = \prod_{i=1}^{n-1} \prod_{j=1}^{\ell} (\lambda \cdot i \cdot (n-i) \cdot e^{-\lambda i(n-i)t_{i,j}})$$

$$L_1(\mathcal{W}) = \prod_{i=1}^{n-1} \prod_{j=1}^{\ell} (\lambda \cdot (i \cdot (n-i) - \delta_{i,j}) \cdot e^{-\lambda(i(n-i) - \delta_{i,j})t_{i,j}})$$

Then,

$$\begin{aligned} \frac{L_0(\mathcal{W})}{L_1(\mathcal{W})} &= \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} \left(\left(1 + \frac{1}{i \cdot (n-i) - 1}\right) \cdot e^{-t_{i,j}} \right) \\ &= \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} \left(\left(1 + \frac{1}{i \cdot (n-i) - 1}\right) \cdot e^{-\frac{1}{i(n-i)}} \right) \cdot \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} e^{-t_{i,j} + \frac{1}{i(n-i)}} \\ &= \prod_{i=1}^{n-1} \left(\left(1 + \frac{1}{i \cdot (n-i) - 1}\right)^{\ell_i} \cdot e^{-\frac{\ell_i}{i(n-i)}} \right) \cdot \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} e^{-t_{i,j} + \frac{1}{i(n-i)}} \end{aligned}$$

Since $\ell_i = o(i \cdot (n-i))$, we can apply Equation (3.11), and the former product equals

$$\begin{aligned} \frac{L_0(\mathcal{W})}{L_1(\mathcal{W})} &= \prod_{i=1}^{n-1} \left(1 + \Theta\left(\frac{\ell_i}{i^2 \cdot (n-i)^2}\right) \right) \cdot \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} e^{-t_{i,j} + \frac{1}{i(n-i)}} \\ &= \prod_{i=1}^{n-1} \left(1 + \Theta\left(\frac{\alpha}{i \cdot (n-i)}\right) \right) \cdot \prod_{i=1}^{n-1} \prod_{j=1}^{\ell_i} e^{-t_{i,j} + \frac{1}{i(n-i)}} \end{aligned}$$

The former product simplifies to $1 \pm \Theta\left(\alpha \cdot \frac{\log n}{n}\right) = 1 \pm o(1)$. Therefore,

$$\begin{aligned} \frac{L_0(\mathcal{W})}{L_1(\mathcal{W})} &= (1 \pm o(1)) \cdot e^{\sum_{i=1}^{n-1} \frac{\ell_i}{i(n-i)} - T} \\ &= (1 \pm o(1)) \cdot e^{E[T_0] - T}. \end{aligned}$$

Therefore, $L_0(\mathcal{W}) > L_1(\mathcal{W})$ if $T \leq E[T_0] - 1$, and $L_0(\mathcal{W}) < L_1(\mathcal{W})$ if $T \geq E[T_0] + 1$. We have that $|E[T_0 - T_1]| \leq D = \Theta\left(\alpha \cdot \frac{\log n}{n}\right)$; moreover, the probability that the difference between T and its expectation is at least $2D$, and the probability that it is at most $-2D$, are both $\frac{1}{2} - o(1)$, since the standard deviation B satisfies $B = \omega(D)$.

Therefore, the probability that the likelihood of graph G_0 is higher than the likelihood of graph G_1 is $\frac{1}{2} \pm o(1)$, regardless of whether the unknown graph was G_0 or G_1 . The proof is concluded. \square

CHAPTER 4

ON THE INTERNET DELAY SPACE DIMENSIONALITY

Network latency plays a central role in the design of a large class of Internet services as their performance is sensitive to the choice of the communicating participants, among a much larger set of alternatives. In light of this, coordinate-based network positioning systems have received considerable attention in the past few years [38, 40, 106, 116, 124, 131]. These approaches aim at providing a compact representation of the Internet delay space (i.e., the matrix of measured round-trip latencies between Internet hosts) by modeling the network as contained in a vector space. In this process, known as *network embedding*, each node is assigned a coordinate in a host metric space (e.g., Euclidean space) in such a way that the geometric distance between any two nodes estimates the real latency between them within a tolerable degree of error.

However, coordinate-based systems inherently suffer from embedding distortion, instability, slow convergence, and disappointing accuracy, as pointed out by [85] and [92]. Moreover, as discussed in [94] and [127], some aspects of the Internet graph make it difficult to model as a well-defined geometric object. As a result, these obstacles motivate positioning systems without coordinates [94, 95, 118, 144] as a more functionally viable alternative due to their improved accuracy, despite the fact that they are often measurement intensive and, in some cases, some types of queries are not available to network participants (e.g., the prediction of distances between any two other arbitrary nodes.).

As a way of understanding the potentialities and limitations of coordinate-based systems, this work investigates a critical aspect influencing the effectiveness of this approach, namely the dimensionality properties of the Internet delay space. The main component of coordinate-based systems consists of an embedding algorithm for which the number of dimensions of the host metric space (denoted hereafter by d) is a tunable

parameter.

Embeddings with different numbers of dimensions result in different degrees of accuracy, since distance matrices possess a minimum intrinsic dimensionality [40,106,131]. In addition, since embedding techniques are based on some variation of an optimization problem aimed at minimizing the prediction error, the algorithms suffer from the *curse of dimensionality*. That is to say that the algorithm's complexity increases with the number of dimensions to the point that it becomes unable to deal with the overwhelming number of degrees of freedom to explore. Often times, this phenomenon is due to the unnecessary inflation of the metric space [25]. Finally, the convergence time, which increases with d , affects the stability of coordinates and the adaptability to changes, thereby affecting the reliability of the predictions [40,85].

Previous studies indicated that embeddings of the Internet delay space can be created using 5 to 9 dimensions with reasonably low distortion in a Euclidean space [106,131]. Dabek *et al.* [40] demonstrated that augmenting the embedding with *height* vectors, which are thought of as distance penalties incurred by traversing the last-mile access links in the Internet topology, allows one to use a lower-dimensional Euclidean space while retaining a similar level of accuracy. The application of Principal Component Analysis (PCA) [130] reveals the same dimensionality values [131].

However, this work prompts many questions which are still elusive: Can the dimensionality of the network be determined by embedding it into a host metric such as Euclidean space, or is there a more robust way of defining dimensionality, independent of the choice of host metric? Is the observed number of dimensions that produces low distortion embeddings in [40,106,131] optimal? More importantly, what properties of the Internet contribute to its dimensionality?

The characterization of the Internet delay space dimensionality, apart from its implications to the performance of coordinate systems, is by itself a topic of practical interest as it uncovers properties and opens new questions on the nature and complexity of the network [115].

As illustrated by [132], certain features of the Internet can be deduced purely from its delay-space geometry. For example, the partition of Internet hosts according to continents can be approximately reconstructed by clustering unlabeled distance data. As another example, it is unknown how much empty space is left by embedding the Internet using the current algorithms. If the empty space is significantly large, then extra overhead and complexity are being unnecessarily incurred by positioning systems. In addition, previous studies — as well as ours — have focused on datasets with up to a few thousand data points. However, theoretical results assert that in the worst case, the number of dimensions and distortion of an embedding increase logarithmically with the cardinality of the point set [32]. Therefore, a practical characterization issue, also critical for synthetic delay space generation purposes [146], is whether or not the dimensionality behavior and embedding distortion observed in previous studies will remain invariant with scaling to millions of nodes. Finally, the performance of positioning systems without coordinates also benefits from the characterization of the delay space geometry. For instance, the scaling guarantees of Meridian [144] are based on the assumption of a doubling metric whose main parameter is its dimensionality.

Tang and Crovella [132], and Huffaker *et al.* [69] demonstrated that geographic location (henceforth, *geolocation*) is a strong component to the Internet delay space. However, due to routing inefficiencies, caused by sub-optimal behavior of protocols, wide-area routing policies, and triangle inequality violations [131, 137, 146], great circle distances are not able to fully explain the Internet delay space dimensionality behavior.

If this is the case, what other forces play a role in the dimensionality behavior?

This chapter presents measurements and analysis to shed insight on these questions. We study the Internet dimensionality, defined as an intrinsic property of the distance matrix. This constitutes a geometrical invariant which does not refer to an external host metric space, such as Euclidean space, and does not include any structural distortions and unnecessary dimensionality inflation incurred by the embedding algorithms. We also study tools for exploring how and to what extent network properties drive the delay space dimensionality behavior. Using datasets obtained via the King [64] method, we compare four intrinsically-defined measures of dimensionality with the dimension obtained using network embedding techniques (such as Vivaldi [40]). We present three main conclusions. First, based on its power-law behavior, the structure of the delay space is best described by fractal measures of dimension, which measure a dimensionality intrinsic to the dataset, rather than by integer-valued parameters, such as the embedding dimension or PCA. Second, the intrinsic dimension is *much smaller* than the dimension predicted by the latter methods (estimated to lie between 4 and 7): in fact, by some measures, the intrinsic dimension is less than 2. Third, we quantify to what extent geolocation drives the dimensionality behavior and we present observations that suggest how the AS topology is reflected in the delay space. More specifically, we show that subsets of the data which can reach each other without going through a transit link between two Tier-1 providers consistently exhibit lower fractal dimension than the combined delay space, and that no such dimensionality reduction is achieved when partitioning according to geolocation. Given the properties observed above, we finally show evidence that fractal measures of dimensionality, due to their sensitivity to non-linear structures, display higher precision for measuring the influence of subtle features of the delay space geometry which are not captured by other dimensionality measures.

The rest of this chapter is organized as follows. Section 4.1 presents the related work, Section 4.2 describes our experimental methodology, and Section 4.3 describes the notions of dimensionality we use in this work. Finally, Section 4.4 applies the proposed metrics to study delay space features, and Section 4.5 offers our conclusions.

4.1 Related Work

The first work to propose network embedding for the Internet was GNP (Global Network Positioning) [106]. In this work, Ng and Zhang tackle the complexity of the embedding by incrementally determining the coordinates of participants with respect to a set of a few previously chosen nodes (beacons or landmarks). These infrastructure nodes measure their inter-distance and determine their coordinates in some d -dimensional metric space. As a result they function as references in space so that subsequent incoming nodes can determine their own coordinates. This is done via a non-linear optimization problem that aims at minimizing the overall discrepancy between the geometric and measured distances. Hence, arriving participants compute the same minimization problem to determine their own absolute coordinates, by actively measuring their distance to the already-oriented beacons. Later on, a theoretical justification for the success of this approach was given by Kleinberg, Slivkins, and Wexler [75]; we discuss their work in greater detail below. Other examples of systems with similar scope are [38], [116], and [124].

On comparing the distributions of relative errors incurred by GNP using different number of dimensions, Ng and Zhang indicate that for the dataset studied, the best results were achieved using 7 to 9 dimensions.

Tang and Crovella [131] address the complexity and cost of network embedding by

proposing the Lipschitz embedding. This method relies on the assumption that, by the triangle inequality, two nearby points, say a and b , have similar distance to a third point x , that is $|d(a, x) - d(b, x)| \leq |d(a, b)|$ and is defined in terms of a set D of subsets of a point set X . The distance function d defines the distance from a point x to one of the sets $L_i \in D$ as the distance from x to the nearest point of L_i . Thus, the embedding is the mapping $\phi(x) = [d(x, L_1), d(x, L_2), \dots, d(x, L_{|D|})]$, where $|D|$ corresponds to the dimensionality of the embedding. When every L_i is a singleton, each element represents a beacon and, therefore, component j of vector \vec{x}_i is actually the measured distance from x to landmark j .

Since the number of dimensions required by the above embedding is high, the authors apply PCA, discussed in Section 4.3.3, to determine an r -dimensional space (where $r \ll n$) in which the data can be approximated with low loss of accuracy. As a result, each \vec{x}_i is transformed into a \vec{y}_i , where each component of the latter is a linear combination of the distances to landmarks. Thus, they can be seen as distances to virtual landmarks. The method is evaluated using several real Internet datasets, and the number of dimensions that capture most of the overall variation of the data is between 7 and 9, incurring mean relative errors of 8 to 25 percent, consistent across the different datasets.

Subsequently, Dabek *et al.* proposed Vivaldi [40], which is a coordinate-based system that uses a mass-spring relaxation problem to determine the coordinates of nodes. In spite of using beacons, like GNP, Vivaldi has the advantage of not requiring fixed nodes serving this role and provides a degree of accuracy that is competitive with that of GNP. More interestingly, the authors propose the idea of unidirectional height vectors to augment the geometric model. Intuitively, the core of the network is mapped into a vector space as before whereas the borders of the network are assigned heights that penalize the distances of access link traversals. As a result, nodes can be placed up or

down in order to accommodate conflicting distances in low dimensionality. It has been shown that this approach was able to embed the dataset considered into 2-space plus heights with competitive accuracy as compared to the embedding into 5-space.

The Vivaldi project also explores alternative geometric spaces, such as spherical and cylindrical, given that curved spaces resembles the surface of the globe around which the Internet is deployed. However, since most of the core links are centered in the U.S. and Europe, and due to the fact that there is no communication passing through the poles, the Internet does not wrap around the earth and, therefore, it has been shown that these approaches are no better than simply fitting the network into a plane space. Shavitt and Tankel explore embeddings in hyperbolic space [125] which accommodate distance conflicts in low dimensional spaces, achieving a similar level of success as the height vectors.

More recently, studies quantified the inaccuracy produced by current positioning systems. Ledlie, Gardner, and Seltzer demonstrated [85] that the relative errors increase with the cardinality of the set of hosts, and that the convergence to and maintenance of stable coordinates produced by the embedding algorithms are barriers to the effectiveness of such systems. Lua *et al.* [92] confirm that the degree of inaccuracy is beyond tolerable and propose metrics to quantify this fact, which is otherwise hidden by analyzing cumulative distributions of relative errors.

Zhang *et al.* characterized the delay space and proposed a synthetic data generator that improves upon existing topology generators [146]. Later on, the same group studied the impact of triangle inequality violations found in the delay space on overlay networks [137].

There is a rich body of theoretical work on questions regarding the existence of low-

distortion embeddings of finite point sets into Euclidean space and other host metrics, as well as the computational complexity of algorithms for computing such embeddings. The starting point for much of this research is Bourgain’s famous theorem [32] that every metric space of cardinality n may be embedded with distortion $O(\log n)$ in a Euclidean space of dimension $O(\log n)$. A randomized algorithm for computing such an embedding was supplied by Linial, London, and Rabinovich [91]; the algorithm is based on semidefinite programming combined with a random-projection method due to Johnson and Lindenstraus [71]. Although there has been progress on the problem of minimizing the *additive distortion*, i.e. the maximum additive error over all pairs of points, the corresponding problem of computing low *multiplicative distortion* embeddings into lower-dimensional spaces — e.g. Euclidean spaces of dimension $o(\log n)$ — is an algorithmic problem of daunting complexity. Indeed, it is NP-hard to approximate the minimum-distortion embedding of an n -point metric into a d -dimensional Euclidean space within an approximation factor less than $\Omega(n^{1/12})$ [70]. Finally, several recent papers [7, 75] have considered the problem of computing *embeddings with slack*, in which an ε fraction of all distances may be arbitrarily distorted and the rest must satisfy a low-distortion guarantee. This work, which uses beacon-based embedding techniques *a la* GNP, culminated in a theorem that every finite metric space admits an embedding in $O(\log^2 \frac{1}{\varepsilon})$ dimensions with $O(\log \frac{1}{\varepsilon})$ distortion and ε slack. Note that both the distortion and the dimensionality of the host metric in this theorem are still too high to be of practical value for network coordinate systems.

In Section 4.3 we study the power-law behavior of the delay space and its relationship with fractal dimensions. Many different power laws and self-similar phenomena have been documented in the literature on Internet measurement. Examples include power laws in the distribution of packet rates on an Ethernet link [143], inter-arrival times for FTP connections and TELNET packets [114], HTTP connections [39], differ-

ent aspects of the Internet topology [52], and round-trip measurements in a time series of pings between a single pair of hosts [6].

The measures of fractal dimensions used in this work were also used by Belussi and Faloutsos [26]. Their work demonstrated that when spatial datasets behave like fractals (defined in Section 4.3.2) over a wide range of distances, one can use measurements of their fractal dimensions to rapidly estimate the spatial selectivity in range queries.

To the best of our knowledge, our work is the first to study the underlying dimensionality of the Internet delay space as a separate issue from the embedding of this delay space in any particular metric space, the first to document power laws in the Internet delay space and to apply fractal geometry to its characterization, and the first to explore the impact of the Internet’s AS-level topology on its delay-space geometry. The dimensionality revealed by our techniques is significantly lower than the dimensionality of the embeddings used in the prior work reviewed above [40,106,131], although the existence of an algorithm that produces embeddings with this low dimensionality behavior is still an open question.

4.2 Methodology

Our main analysis is based on the Meridian dataset presented by Wong, Slivkins, and Surer [144], and some of our findings are also supported by observations made using the MIT King dataset [2,40].

The Meridian dataset was collected between May 5-13 2004 via the King [64] method, containing latency measurements between more than 5200 DNS servers. The list of sites to measure was determined by randomly picking website names from a set of

593160 entries obtained from the DMOZ and Yahoo directories. The raw data consists of a set of asymmetric measurements between pairs of DNS servers, that is, the RTT's in microseconds between two servers A and B , measured by recursively querying A for domains served by B , and vice-versa. The number of asymmetric measurements per pair varies between 1 and 20 entries with median 11. In order to create the matrix used in this work, we took the union of the asymmetric measurements for each pair, thereby making the dataset symmetric. We subsequently filtered out the pairs with less than 10 measurements, in order to minimize biases due to queuing delays at routers or DNS servers, and then computed the median of the symmetric measurements for the remaining pairs. Finally, we approximate the largest clique in the resulting incomplete matrix via a 2-approximation algorithm for the vertex cover problem (i.e., to eliminate the missing entries by removing the minimum number of nodes), resulting in a all-pairs matrix with 2385 hosts, annotated with their IP addresses, which we henceforth refer to as "IPs" for brevity.

The MIT King dataset was first used to study Vivaldi's behavior [40]. It was also collected using the King method and contains measurements among 1953 hosts, selected by finding the NS records of IP addresses of participants in a Gnutella network. Nevertheless, after applying the above data cleaning process, only 298 nodes remained in the dataset. Although the size of this dataset does not allow us to use it for analyzing all the aspects discussed in this work, it can still be used to support some of our findings.

We also merged the delay data in the Meridian dataset with the underlying AS topology by obtaining a snapshot, from the same period the delay dataset was collected, of the customer-provider AS graph from the CAIDA AS relationships dataset [5]. Using the combined data, we made a decomposition of the network into AS trees, each rooted at one major Tier-1 Autonomous Systems (AS). Accordingly, each piece com-

AS		Meridian
AS#	Name	Hosts
2914	NTT Comm.	1212
209	Qwest	1227
3561	SAVVIS	1389
3356	Level 3	1454
7018	AT&T	1487
3549	Global Crossing	1515
701	Verizon	1529
1239	Sprint Nextel	1604

Table 4.1: List of the major Tier-1 AS together with the number of IPs in their downstream networks represented in the Meridian dataset.

prises the Tier-1 network itself, together with its downstream network of AS customers. After decomposing the whole network in this fashion, we classified the IPs found in the Meridian dataset into each piece. The number of IPs found in each piece is summarized on Table 4.1¹.

Notice that the sum of the number of IPs in each network exceeds the total number of IPs in the dataset. This is because most of the IPs are located in multihomed networks (i.e., are served by multiple providers). In fact, according to our data, more than 60% of the customers have contracts with more than one provider, and the number of upstream providers per network can be as high as 13. Thus, our decomposition does not consist of a partition of the space, and, in fact, some of the subsets contain more than 50% of the nodes in the whole clique.

We emphasize that the King method is a convenient way to obtain vantage points

¹A recent result indicates that a single snapshot of the inferred AS topology map is believed to miss around 10% of the customer-provider links involving Tier-1 and Tier-2 networks [107]. While this does not lead to missclassification, it could possibly exclude the membership of some domains (and its downstream customers) in some of the pieces.

using DNS servers, which are generally well-connected hosts. Thus, although its hosts are geographically and topologically diverse, it can be argued that datasets collected via King could only give us an approximate picture of the delay space geometry as composed of core and edge networks. Furthermore, both datasets contains violations of the triangle inequality to a degree consistent with that found in the characterizations by [40], [131], [137], [144], and [146]. In addition, we analyze a static snapshot of the network, which does not capture the effects of temporal instability of distances (i.e., due to congestion, variable queuing time, path failures, etc).

The unavailability of Internet latency datasets is a major limitation to the study of the dimensionality of the delay space. We have investigated the other publicly available datasets of this kind but they are either limited in scale, i.e., do not contain a large enough all-pairs matrix so that our analysis can be applied, or they are not annotated with IP addresses, which makes the above decomposition impossible.

In order to analyze the geographic component of the delay space, we queried the *hostip.info* database [1] for the IPs contained in the Meridian dataset, obtaining their latitude and longitude at the time of writing. Since these IPs belong to DNS servers of large domains that are not as likely to have their IPs reassigned as are smaller domains, we resort to the assumption (not quantified) that a large fraction of the IPs contained in the Meridian dataset were not reassigned since 2004. Even though we believe that this is a reasonable approximation, it should be emphasized that, together with the fact that geolocation is currently a process with high degree of inaccuracy, this could combine several sources of error for the particular set of results in Section 4.4.1. Figure 4.4.1 presents a coarse-grained visualization of the geolocation of nodes in the Meridian dataset.

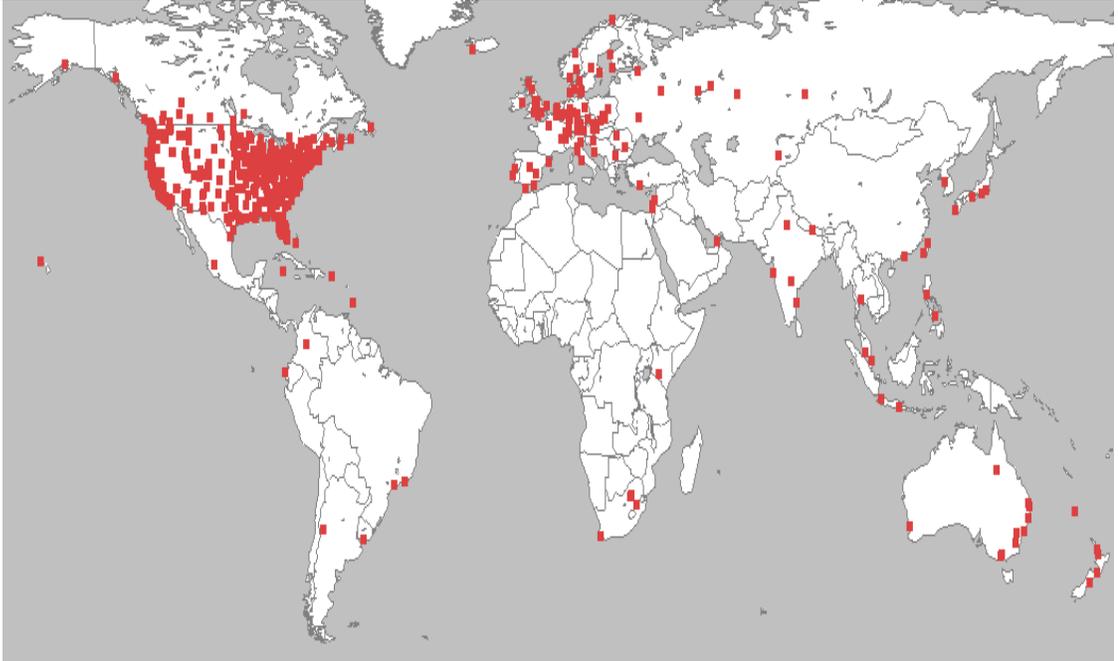


Figure 4.1: Geographic location of nodes in the Meridian dataset.

4.3 Dimensionality Measures

As a starting point for introducing the measures of dimensionality that we use in this work, let us consider the following problem. Suppose that a surveyor chooses a set X of 2500 random points in the plane and measures the distances between all pairs using a method that introduces 5% relative error due to measurement noise. Given the matrix of measurements, but not the coordinates of the actual points, how could one deduce that the data came from a point set in 2 dimensions, rather than 1 or 3? We consider this problem further in the following sections.

4.3.1 Embedding dimension

An obvious answer to the question question posed in the previous section is: for $d = 1, 2, 3, \dots$, try to embed the points in d dimensions using an embedding algorithm such as Vivaldi. Stop at the lowest dimension D which permits an embedding with small quartiles of relative errors and let D denote the *embedding dimension* of the dataset.

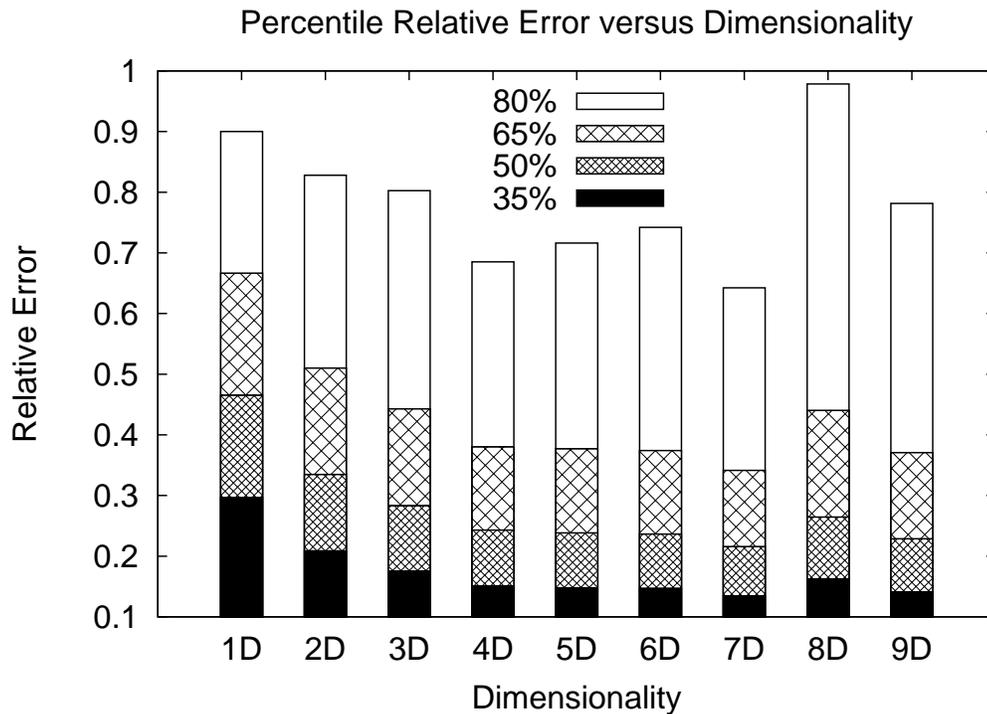


Figure 4.2: Percentiles of relative errors produced by Vivaldi using different values of d .

We applied this process to the Meridian dataset by embedding the network into Euclidean space using Vivaldi, available as part of the P2Psim package [2]. We varied the number of dimensions from 1D to 9D and Figure 4.2 presents the outcomes of this experiment by displaying the 35-th, 50-th, 65-th, and 80-th percentiles of relative errors incurred by embedding the whole delay space with different values of d (with the percentile values chosen in such a way that the discrepancy of the distributions could be

well captured). In this plot, we can observe that there is a fast improvement in accuracy up to $d = 4$ and a slow improvement up to $d = 7$. Surprisingly, after 7 dimensions, the accuracy of the algorithm gets worse, exposing the threshold beyond which the curse of dimensionality starts to affect the algorithm's performance.

A benefit of this approach is that, if successful, it actually recovers the coordinates of the original points (up to translations and rotations).

However, it also has many drawbacks: i) embedding algorithms are slow, even for relatively small values of d . ii) Finding an embedding that minimizes distortion is computationally intractable in the worst case [33]. iii) If the measured distances reflect a metric other than the Euclidean distance (e.g. the hyperbolic metric or the Manhattan metric), the algorithm may fail to find a low-distortion embedding in any dimension.

Moreover, by attempting to fit the distances precisely, the algorithm may produce a high-dimensional embedding with lots of empty space, obscuring the fact that the points of the embedding really lie in a lower dimensional subset of that space. For example, if a point set X were located on a hilly terrain instead of a flat plane, the algorithm would output an embedding using 3 dimensions despite the fact that all of the data lies along a 2-dimensional surface in 3-space.

Finally, the embedding may fail to reveal lower dimensional substructures which constitute important features of the distance matrix. For example, suppose that the entries of the distance matrix are estimates of the time required to walk between various locations in an office building with several floors. The geometry of the office building is most accurately modeled as a small number of 2-dimensional pieces (the floors) with a small number of "gateways" (the stairwells) connecting these pieces together. Embedding the distance matrix accurately in a Euclidean space would require at least 3

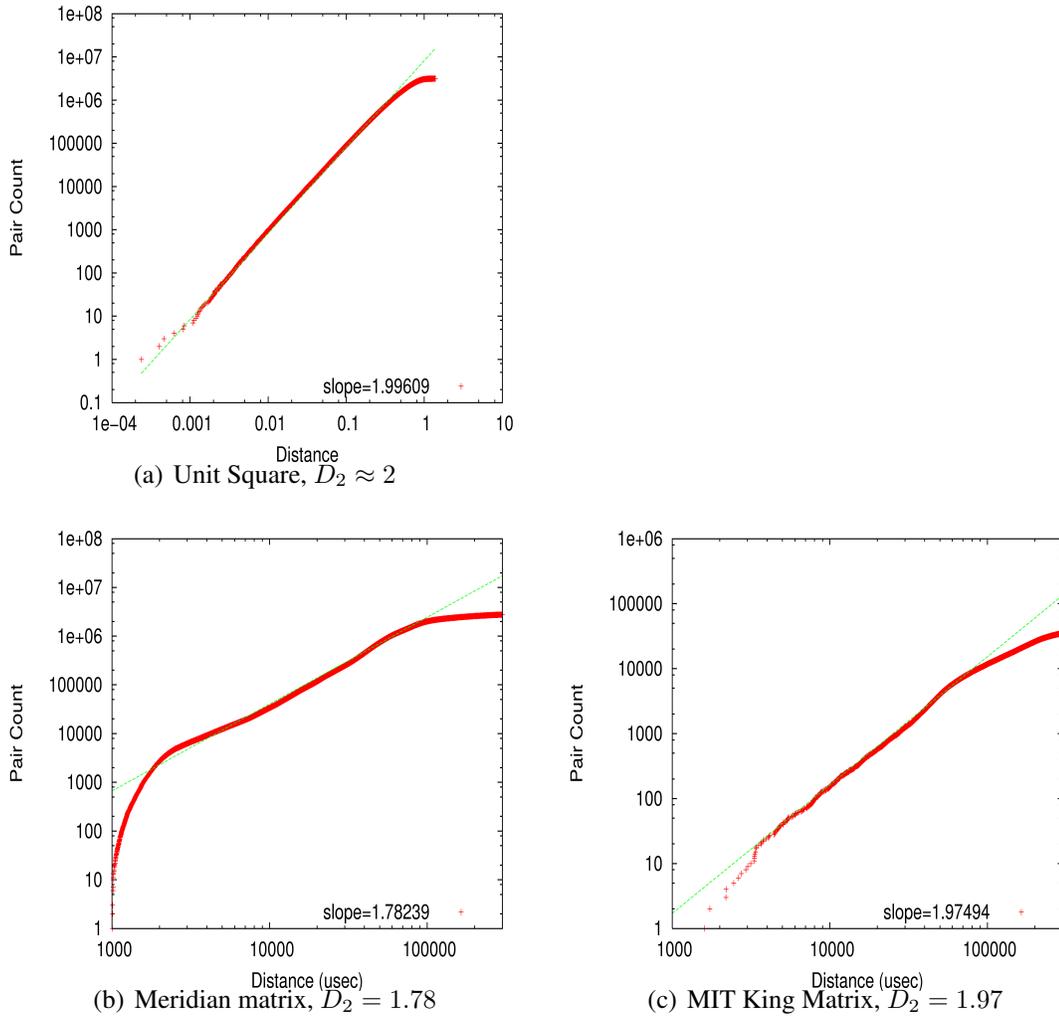


Figure 4.3: Correlation Fractal Dimension, D_2 , of a) a set of 2500 random points in a unit square, and the Internet delay space as represented by b) the Meridian matrix and c) the MIT King matrix.

dimensions — probably more, since shortest paths in the office building are very different from shortest paths in 3-space — obscuring the inherent 2-dimensionality of the office building’s floor plan. Like the office building, the Internet delay space is also composed of smaller pieces (autonomous systems) which meet only at prescribed gateways (customer-provider links and peering points). When representing the geometry of the Internet delay space, one should not choose a representation which obscures this

structure.

For purposes of estimating the dimensionality of a point set (rather than computing coordinates to represent its points) there are several other, more lightweight, ways of defining dimensionality using structural properties of the distance matrix itself, without making reference to an outside “host metric” such as Euclidean space. These methods also capture the effects of the intricate patterns mentioned above. We next introduce these definitions of dimensionality, explaining the applicability of each and their degree of accuracy for the solution to the problem introduced in the beginning of this section. Although the following notions of dimensionality were introduced in the theory of metric spaces (which assumes the triangle inequality), all of them have the desirable property that they are applicable even in datasets (like ours) which contain triangle inequality violations, often yielding meaningful results.

4.3.2 Correlation Dimension

Suppose we pick a random point $x \in X$ and a radius r , and we count the number of other points whose distance from x is at most r . If the points are random samples from a bounded region in the plane, then the expected number of such points in a region is proportional to its area. Hence the number of points within distance r of x should be proportional to r^2 . For the same reason, in d dimensions, the number of points would be proportional to r^d . Hence, upon plotting the radius r against the number of pairs whose distance is at most r in logscale (which we denote as the *pair-count* plot), for special point sets which produce a straight line over a given range of interest, i.e., exhibit power-law behavior, we can interpret the exponent of the power law, i.e., the slope of the line, d , as reflecting the underlying d -dimensionality of the space represented by the given

distance matrix. We refer to d as the *pair-count exponent* [26], which corresponds to the Correlation Fractal Dimension, D_2 [123], further discussed in Section 4.3.2.

Figure 4.3(a) illustrates the pair-count plot of a set of 2500 random points in a unit square surface. Notice the presence of a power law that persists over three decimal orders of magnitude, and observe that the exponent of this power law (i.e. the slope of the line) is almost exactly equal to the dimension of the space from which points were sampled, in accord with the theoretical prediction sketched above. In fact, for all “Euclidean objects”, i.e. distance matrices obtained from uniformly-random point clouds in Euclidean d -space, the fractal dimension matches the Euclidean dimension.

Figure 4.3 presents the pair-count plot of the Internet delay space as represented in the Meridian and MIT King datasets.

The first striking feature of these plots is a power-law that persists roughly over two orders of magnitude, i.e., from 3ms to 100ms (Note that this range of latencies includes almost every Internet route that is not trans-oceanic). As a result, the Internet delay space exhibits the desirable property that it can be measured by fractal metrics of dimensionality (see Section 4.3.2). The second unexpected observation is that the magnitude of its dimensionality is less than 2, represented by the pair-count exponents $D_2 = 1.782$ and $D_2 = 1.975$ in the Meridian and MIT King dataset respectively. The estimation of these values contains errors to a degree that would not affect the conclusions derived in this work. These dimensionality values are much smaller than the embedding dimension indicates (i.e., between 4 to 7 dimensions), suggesting a different geometric picture of the structure of the Internet delay space. Section 4.3.3 discusses the reasons for this discrepancy. Finally, the power-law behavior, including the dimensionality value, is consistent across random subsets of the data, as discussed in Section 4.4.2.

The fractal measures can help us understand the weaknesses of embedding algorithms by showing how they affect the properties of the original delay space. Accordingly, upon computing the pair-count plot of the embedded network produced by Vivaldi in 7 dimensions, we can observe how the resulting coordinate space does not preserve the geometric properties of the original delay space and suffers a major dimensionality inflation. Figure 4.4 shows the resulting pair-count plot of the delay matrix reconstructed from the 7-space coordinates. Notice that the curve has a concave shape, thereby deviating from the power-law behavior of the original space. Moreover, the best effort to measure its dimensionality, by finding the best straight line fit to the curve, results in a pair-count exponent of value 5.46.

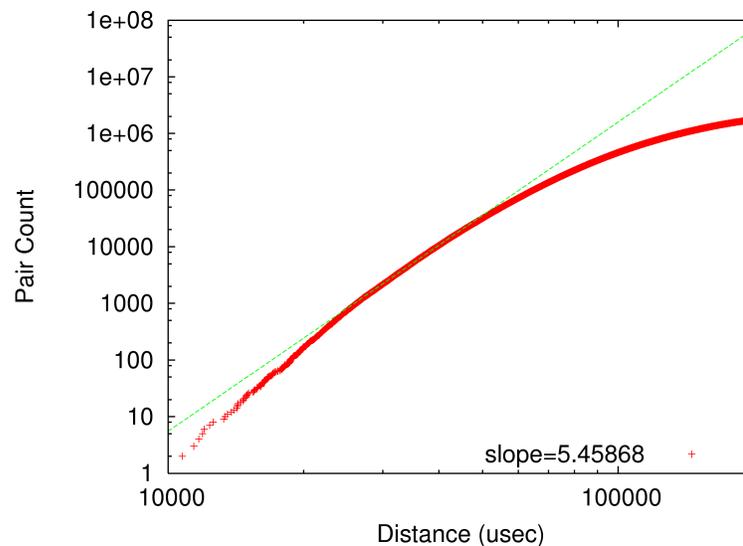


Figure 4.4: Pair-count plot of the Meridian dataset embedded into the Euclidean 7-space via Vivaldi.

Although the Internet hosts live in a sphere that can be described by two coordinates in spherical space, the values near 2 found for the delay space dimensionality are not a reflection of the 2-dimensional structure of a sphere's surface. This phenomenon would be further explored in Section 4.4.1.

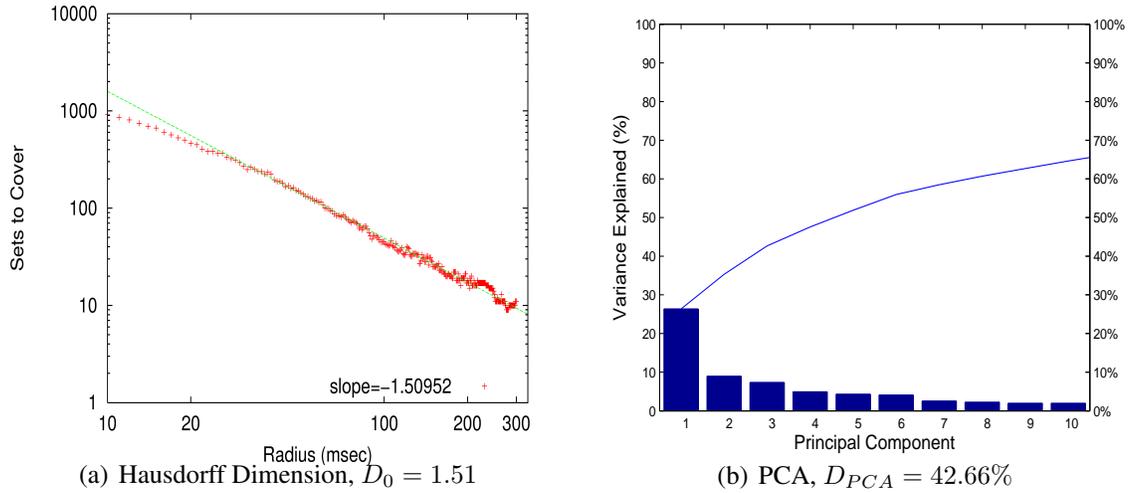


Figure 4.5: Measures of dimensionality applied to the Meridian matrix: a) the Hausdorff dimension computed by the greedy set cover plot, and b) Principal Component Analysis (PCA) applied to the Meridian matrix.

Fractal Dimensions

The previous section introduced a measure of dimensionality which was based on measuring the exponent of a power law arising in distance data. In general, this power law does not necessarily arise and, when it does, it need not have an integer exponent. Point sets whose pair-count plots display a power law are called *fractals*.

The correlation dimension is just an example from among an infinite family of fractal dimensions D_q , indexed by a non-negative number q . Formally, if μ is a fractal measure on a set Y and A_1, A_2, \dots, A_N is a partition of Y into pieces of diameter less than r , with $\mu(A_i) = p_i$ for $i = 1, 2, \dots, N$, then²

$$D_q(Y) = \frac{1}{q-1} \lim_{r \rightarrow 0} \frac{\log \left(\sum_{i=1}^N p_i^q \right)}{\log r}. \quad (4.1)$$

²The case $q = 1$ is exceptional. In equation (4.1) when $q = 1$, one uses the log of the entropy of the distribution $\{p_i\}$ in the numerator and drops the constant $1/(q-1)$ out front.

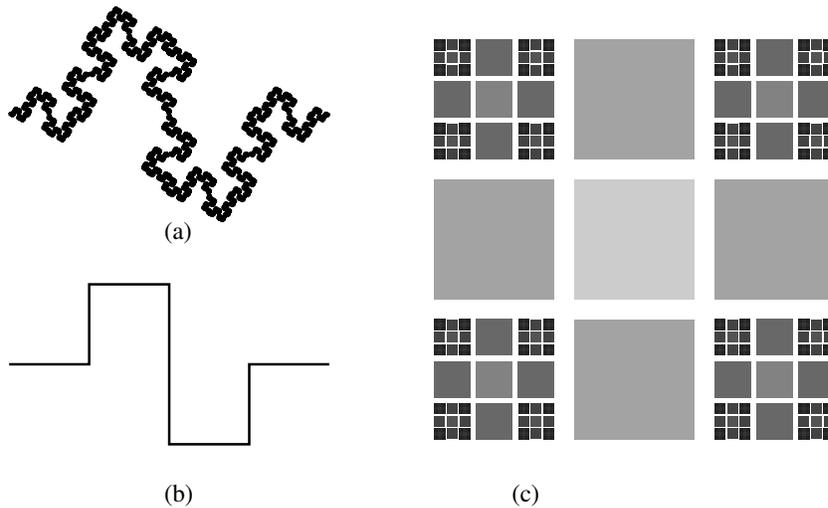


Figure 4.6: Some fractals and fractal measures. (a) A 1.5-dimensional fractal curve. (b) Template for recursively constructing the fractal curve. (c) A fractal measure on the square. Gray level indicates density.

The Correlation dimension corresponds to the fractal dimension D_2 . Among these dimensions, only the first three can be efficiently measured in practice.

Fractals may arise by applying recursive constructions in which a self-similar point set is composed of finitely many pieces, each of which is a scaled-down copy of the entire point set. If one samples a large number of points uniformly at random on such a perfect infinite fractal, and measures any of the fractal dimensions (e.g., for all q), they must coincide. For example, if a point set is made up of 2^p pieces, each of which is a copy of the entire set scaled down by 2^k , then all its fractal dimensions equal p/k . However, if the data is non-uniformly distributed inside the fractal, one gets a *fractal measure* or *multifractal* — a fractal together with a probability measure expressing the density of points at different locations.

Despite this recursive definition, the most surprising examples of fractal behavior

are non-recursive structures commonly found in nature. Some examples are snowflakes, coastlines and the surface of the human brain [123]. The question of exactly what features of the Internet delay space lead to its fractal behavior is still elusive. In the search for these properties, we discovered some hints that are discussed in Section 4.4.2. However, this question is currently a subject of our ongoing work.

4.3.3 Other Dimensionality Measures

In this section, we introduce another instance of fractal dimension, namely *Hausdorff dimension* (D_0) and two dimensionality reduction techniques, namely *Principal Component Analysis (PCA)* and *Isomap*, explaining the relevance of each of them to this work.

Hausdorff Dimension

Consider partitioning a point set X into low-diameter subsets. If X lies in a bounded region of the plane, then for every $r > 0$ it can be partitioned into $O(1/r^2)$ subsets of diameter less than $2r$, for example using grid cells of side length r . The analogous low-diameter covering in dimension d uses $O(1/r^d)$ subsets. Even if we are given only the distance matrix — so that it is infeasible to identify the partition into grid cells — a partition into low-diameter sets can still be constructed by considering the collection of all radius- r balls and selecting a sub-collection using the greedy set cover algorithm. For d -dimensional Euclidean objects the cardinality of this greedy covering will also be $O(1/r^d)$ with high probability, though it is less obvious than in the case of the grid-cell covering.

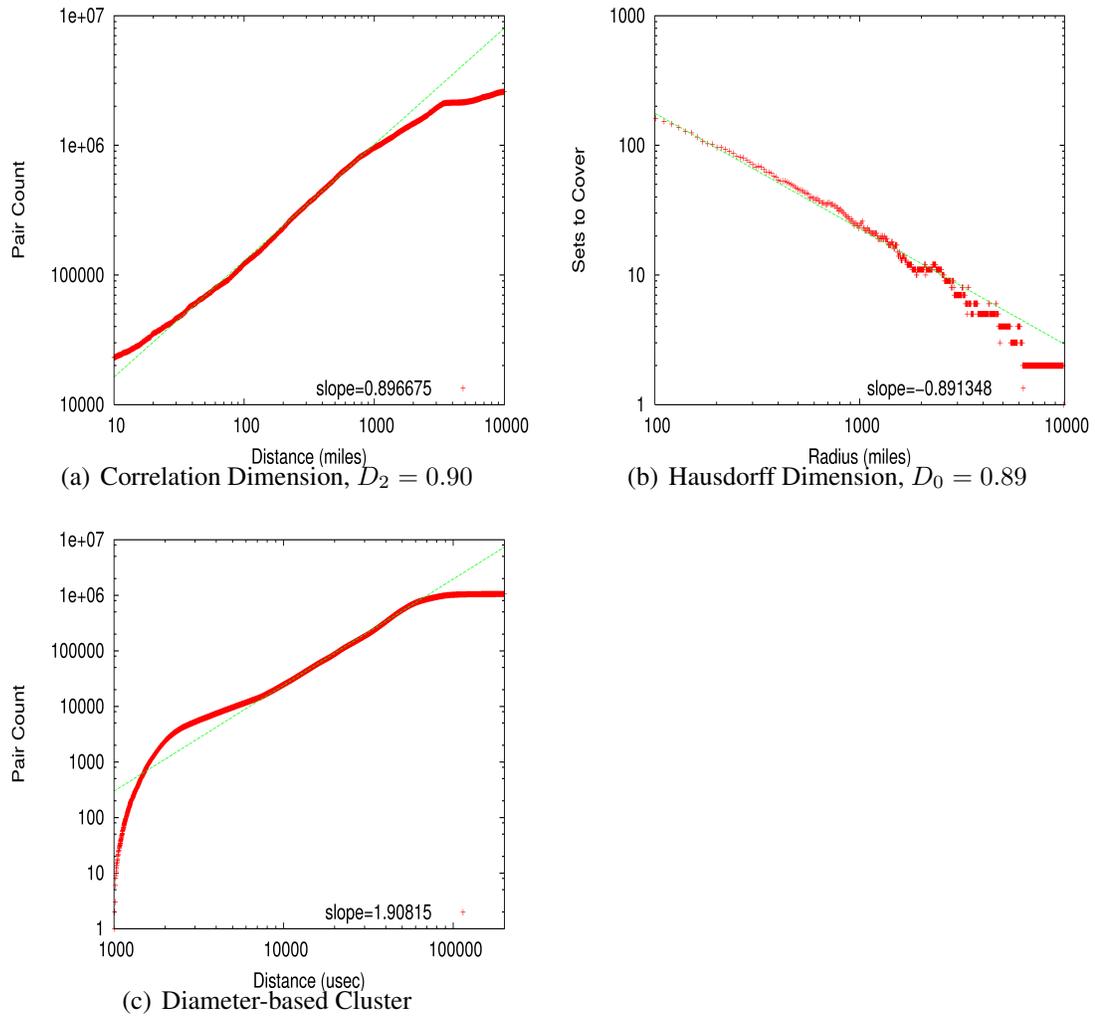


Figure 4.7: Dimensionality of the distance space based on great circle distances via a) the correlation Dimension and b) The Hausdorff Dimension. c) The correlation dimension of one of the latency-based clusters.

This suggests defining $N(r)$ to be the minimum size of a partition of X into pieces of diameter less than $2r$, and plotting r against $N(r)$ in logscale. For d -dimensional Euclidean objects we have seen that this will lead to a line of slope $-d$. For any distance matrix, if a power-law with exponent $-d$ is present over a given range of interest, we refer to d as the Hausdorff fractal dimension, D_0 [123], using the fractal definition presented in section 4.3.2, Equation 4.1.

Figure 4.5(a) presents the measure of D_0 for the Internet delay space as represented in the Meridian matrix. Notice the presence of a power-law with exponent -1.51 .

While D_2 encompasses the geometric structure of the delay space and also the spatial distribution of points, D_0 includes only the former notion [26, 123]. Nevertheless, we introduce D_0 for two main reasons. First, it exhibits the same power-law behavior displayed by D_2 when applied to the delay space as represented by the datasets considered, albeit with a different power-law exponent. Second, D_0 behaves similarly to D_2 when measuring networks that possess different intrinsic dimensionality values. Section 4.4.2 present this behavior in greater detail.

Finally, a number of algorithms are known to run efficiently on metric spaces that possess some form of bounded *doubling dimension* [65]. In general, that is to say that every ball of radius R in the metric can be covered by at most 2^d balls of radius $R/2$. Similarly to the Hausdorff dimension, a d -dimensional metric space has doubling dimension approximately d . While the doubling dimension has convenient algorithmic properties, it is difficult to precisely determine the doubling dimension of a dataset. (See [84] for an example of this process.) The difficulty arises because, unlike the correlation or Hausdorff dimension which are statistically robust against perturbing a few entries of the distance matrix, the doubling dimension is very sensitive to outliers as it involves taking the maximum covering number over all balls in the entire metric space.

Principal Component Analysis

PCA is the main technique adopted in previous work to characterize the Internet delay space dimensionality [40, 85, 131]. In our experiments we have applied PCA directly to the distance matrix. This is similar to the analysis applied in [131], where the method

was justified with the additional observation that the columns of the distance matrix itself represent the coordinates for a particular embedding of the delay matrix, namely the Lipschitz embedding in the L_∞ norm which was discussed earlier in Section 4.1.

Figure 4.5(b) presents the application of this approach to the delay space as represented by the Meridian matrix. The common practice suggests that the dimensionality of a dataset is determined by the point d in the x -axis in which the contribution of the corresponding component differs significantly from the previous one, and the percentage variance explained by components with labels greater than d becomes negligible. However, as in the case of this plot, it is not always clear where to establish this threshold. Since we use PCA for the purpose of comparison of the dimensionality values of different networks, we define our PCA measure as the percentage variance explained by the three most significant components. Accordingly, low dimensionality implies more variance being captured by these components whereas high dimensionality tends to spread the variance across a greater number of components. In the case of Figure 4.5(b), the PCA measure equals 42.66%.

As PCA is a method grounded in linear algebra, when applied directly to a distance matrix, it is oblivious to non-linear relationships between different dimensions³. For example, if the points are sampled uniformly at random from a circle in the plane, PCA applied to the distance matrix will strongly indicate a 2-dimensional dataset despite the fact that all of the points belong to a 1-dimensional curve. Here is where the fractal measures come into play: by capturing non-linear, as well as linear, relationships among the dimensions, the fractal measures are able to capture patterns in the data otherwise ignored by PCA. Therefore, they provide a more genuine characterization of the dimensionality of a metric space.

³PCA can in principle capture non-linear relationships when combined with kernels [100]

Isomap

Finally, we apply Isomap [133], a geometric dimensionality reduction technique, proposed in the Machine Learning community, which is sensitive to both linear and non-linear correlations between the dimensions. Like PCA, when applied to a dataset, Isomap outputs the fraction of the total variance explained by each of the dimensions and produces a d -dimensional non-linear embedding where the d is a tunable parameter. Our results indicate that, similarly to the fractal measures, Isomap displays higher sensitivity to Internet structural properties as compared to PCA and the Embedding dimension (see Section 4.4.2), thereby indicating that the delay space is rich in non-linearity. The Isomap results presented in this chapter were produced using the code that implements Isomap available on the authors' websites [133].

4.4 On the Delay Space Structure

In an attempt to understand the features of the delay space that contribute to its dimensionality behavior, we study how and to which extent network properties, such as the geographic structure and the AS topology, are reflected in its delay space dimensionality. The analysis in this section also demonstrates the applicability and effectiveness of each dimensionality measure in capturing these properties.

4.4.1 The Geographic Component

Our first experiment aims at quantifying the impact of the Internet's geographic structure on the delay space. For this purpose, we computed the great circle distances (in miles)

for every pair of nodes and generated a new distance matrix. The first observation is that the pair-count plot, presented in Figure 4.7(a), exhibits a power-law that also persists for approximately two decimal orders of magnitude. As a consequence, it can also be measured using the correlation dimension and has exponent $D_2 = 0.897$. The plot for Hausdorff dimension, shown in Figure 4.7(b), is less conclusive, although it can also be approximated by a straight line with slope $D_0 = 0.891$. As expected, PCA applied on the Lipschitz embedding for L_∞ norm (i.e., the distance matrix), results in the two first components explaining 100% of the variation in the data.

Note that the geographic dimensionality value is less than 1, while the surface of a sphere has dimensionality 2. This difference can be ascribed to the large empty spaces (i.e., oceans) and the non-uniform geolocation of nodes (i.e., dominant clusters in North America, Europe and Asia).

Since the dimensionality of the geographic space is significantly smaller than that of the delay space, the contribution of geolocation does not fully explain the delay space dimensionality, albeit it is indeed a strong component, in accordance with the analysis by [69] and [131]. However, the fractal measures of dimensionality allow us to quantify the extent to which geolocation contributes to the overall delay space structure. Furthermore, it shows that the fractal behavior of the delay space is in fact present in the underlying geodesic space.

4.4.2 Dimensionality Reducing Decomposition

On searching for a (possibly recursive) structural feature of the delay space that could explain its fractal behavior, we discovered a curious dimensionality shift that can be

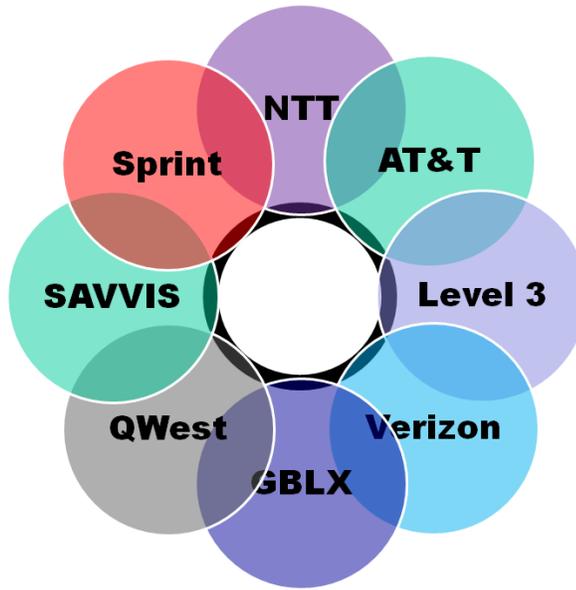


Figure 4.8: Network decomposition into intersecting pieces, each corresponding to a Tier-1 AS together with its downstream network.

ascribed to the structural configuration of the AS topology⁴

Our intuition is based on the observation that peering points, especially those corresponding to transit links between two Tier-1 networks, are contained on a significant fraction of the Internet routes joining nodes which are downstream from different Tier-1 providers. Even though there is an abundance of peering points between non-Tier-1 networks, and multihomed domains are the norm, the significance of paths traversing transit links between two Tier-1 networks could still have a major influence on the geometry of the Internet delay space. Thus, we ask the following question: what is the geometric effect of analyzing each Tier-1 AS downstream network in isolation, thus removing the distances that contain a contribution from the traversal of Tier-1 transit links?

We expected that this process would result in subnetworks whose delay space ge-

⁴In the interest of space, we present only a subset of the graphs from which we derived the conclusions in this section. However, the remaining graphs corresponding to all results presented here can be found in the companion website at <http://www.cs.cornell.edu/~abrahaoinetdim>

ometry is better behaved, as compared to their superposition, in which the subspaces corresponding to the different Tier-1 AS downstream networks would primarily connect to one another in the Tier-1 peering points, as illustrated in Figure 4.8 (again, ignoring other non-Tier-1 peering points and multihomed domains).

An example of this scenario was previously hinted in [106] where the geometry exhibited by a more homogeneous network, when observed in isolation, is better behaved, as compared to a more diverse network. In the case of this study, the embedding of the *Abilene* network, connecting hosts through a fast Internet2 backbone, resulted in gains in accuracy as high as 40% in relative errors, using the same number of dimensions as in the embedding of a global network. This improvement was ascribed to the overall short distances of the paths.

In order to quantify the extent to which transit links are reflected in the Internet delay space, we measured each subnetwork (defined according to the decomposition presented in Section 4.2) using embedding dimension, Correlation dimension, Hausdorff dimension, and PCA.

To confirm that the dimensionality reduction observed in this experiment is attributable to effects arising from the AS-level topology of the Internet, and not some simpler explanation, we tested two alternative hypotheses: that the dimensionality of the delay space can be reduced by decomposing it into pieces of smaller *cardinality*, or that it can be reduced by decomposing into pieces of smaller *diameter*.

The first hypothesis is partially justified by Bourgain's theorem [32] that the Euclidean distortion of a metric space grows logarithmically with its cardinality, in the worst case. To test the hypothesis that cardinality reduction leads to dimension reduction, we created 121 random subsets of the whole matrix, each containing 1454 hosts

(the median cardinality among all subnetworks). In Table 4.2, we denote by *random* $x\%$ the statistics derived from these random subsets corresponding to the x -th quartiles of these values. The values of both the Correlation and Hausdorff dimensions for the random networks, including the minimum of these values (random 0%), are consistently higher than those of every subnetwork (except for the pair count of network 1239) and a higher percent of these values are close to that of the whole matrix. As one increases the size of the original dataset and the number of representatives in the set of random networks, one expects even less deviation between the Correlation and Hausdorff dimensions of the original dataset and the median values measured in the random subnetworks.

To test the second hypothesis, that diameter reduction leads to dimensionality reduction, we created a new decomposition of the distance matrix by the following process. Starting from an element selected from a geographically diverse set of hosts, we grew a ball around it by selecting its 1454 closest neighbors, and examined the delay space consisting of all inter-distances among these 1454 hosts. We have constructed 12 of these subsets centered at hosts located in 9 different countries, 4 continents.

Table 4.2 summarizes the dimensionality of each the subnetworks and values of the statistic thereof for the random subsets⁵ in terms of the Correlation dimension, Hausdorff dimension, and PCA measures. The corresponding measures for the whole network are also displayed in the table for reference, under the label *Meridian*.

The first observation is that the power-law behavior observed in the whole matrix was preserved over the same range of distances in the submatrices, though not necessarily with the same exponent. Second, with the exception of the Correlation dimension

⁵The decomposition based on low-diameter subsets produced results which are incomparable with these results because the clustering rule significantly altered the nature of the power-law behavior, as explained further below. Hence the results of these experiments are not included in Table 4.2.

Network	Dimensionality		
	D_2	D_0	PCA
Meridian	1.783	1.510	42.66
1239	1.780	1.333	46.54
209	1.637	1.225	28.42
2914	1.618	1.161	44.25
3356	1.691	1.230	48.60
3549	1.634	1.265	48.96
3561	1.686	1.239	49.54
7018	1.710	1.304	49.68
701	1.701	1.244	49.68
random 0%	1.725	1.389	27.71
random 25%	1.762	1.465	45.76
random 50%	1.775	1.506	46.67
random 75%	1.789	1.554	53.10
random 100%	1.837	1.682	72.86

Table 4.2: Dimensionality measures: Correlation Dimension (D_2), Hausdorff Dimension (D_0) and PCA found for the different AS networks.

of the subnetwork rooted at AS 1239 (*Sprint*), all other networks exhibit *smaller* dimensionality than the whole matrix according to the two fractal measures, while no dimensionality reduction is observed in the measures of the random networks.

The decomposition based on clustering of growing balls resulted in the following observations. For all clusters, the pair-count plots present a power-law persisting over one decimal order of magnitude less than the whole matrix. For smaller values of r (in the range 3ms to 10ms) the number of pairs at distance r is significantly *higher* than the number of pairs predicted by the power-law approximation. This is perhaps not surprising: since the clusters were selected for their proximity to a single central host, small distances should be more prevalent within these clusters than in the dataset as a whole. Figure 4.7(c) shows one example of this outcome for one of the clusters.

Moreover, the pair-count exponents (i.e., Correlation dimensions) computed over the range in which the power-law persists indicate dimensionality consistently *greater* than that of the whole matrix. The deviation from a power-law is also observed in the range from 20ms to 30ms of set cover plots. In addition, the Hausdorff dimension measured over the range from 30 to 90ms is consistently greater than that of the whole matrix for all clusters.

Interestingly, delay space dimensionality reduction cannot be explained by a corresponding reduction in the geographic component. Upon applying the same decomposition analysis to the great-circle distances of the subnetworks, we observe no statistically significant difference in the power law exponent of the subnetworks as compared to the pair count and set cover exponents found for the combined network.

It is important to emphasize that the AS relationship graph is complex and, therefore, other forms of decomposition could result in pieces that display the optimal dimensionality reduction. Nevertheless, the analysis presented here, shows evidence, with statistical significance, that the presence of *Tier-1* transit links has a non-negligible effect on the Internet's delay space geometry. Both fractal measures are sensitive enough to capture this structural change, in the form of a reduction in fractal dimension when one restricts attention to a subset of the delay space consisting of a single Tier-1 provider and its downstream networks.

Table 4.2 also contains values for the percentage variance explained by the first three components (i.e., the three greatest in magnitude) found by PCA. Notice that, as opposed to the fractal measures, there is no clear distinction in dimensionality behavior between the subnetworks and the random sets as reported by PCA.

We have also computed the embedding dimension of each network in Euclidean

space using Vivaldi and did not observe that decomposing the network into subnetworks had any effect on the embedding dimensionality. In fact, similarly to PCA, in some cases, the dimensionality of subnetworks is reported as being greater than that of the entire network.

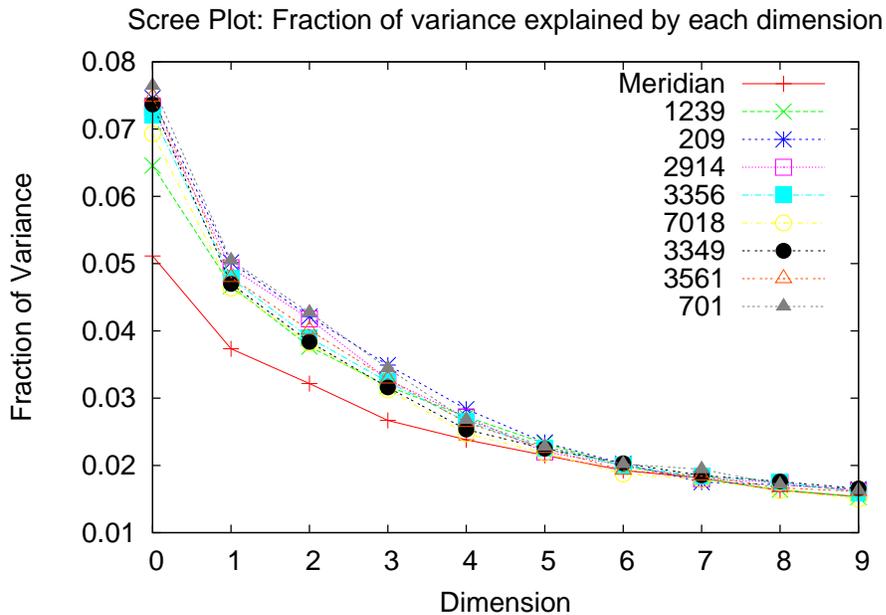


Figure 4.9: The scree plot outputted by Isomap.

We hypothesize three possible explanations for the insensitivity of PCA and Vivaldi to the network decomposition, which was otherwise very well captured by the fractal measures. First, PCA and Vivaldi are oblivious to non-linear relationships in delay space. Second, PCA and Vivaldi try to represent non-linear relationships using linear ones. Therefore, the process of accommodating these discrepancies causes the dimensionality of the host metric to inflate. Finally, PCA and Vivaldi are suited to reporting an integer which summarizes the dataset’s dimensionality; such methods are too coarse to detect a difference of 0.2 in the dimensionality.

In order to confirm our hypotheses we applied Isomap to the networks. Figure 4.9 shows the outcome of this experiment. In the plot, the fraction of the total variance

explained by the first five (also five most significant) components of all subnetworks is consistently greater than that of the entire network. As opposed to what PCA and Vivaldi tell us, and in accordance with the fractal measures, the dimensionality of the subnetworks is indeed consistently *smaller* than the dimensionality of the whole network, even though the dimensionality reported for all networks is still large. This result shows evidence of non-negligible non-linear structures in the delay space and suggests that methods sensitive to them might better reflect the structural properties of the Internet with a higher degree of accuracy than linear methods.

4.5 Discussion

Characterizing the geometry of the delay space is critical for designing and analyzing effective coordinate-based positioning systems and sheds light on the nature of the network. This work investigates the dimensionality observed in the Internet delay space, thus providing insight into some of the underlying causes.

We have observed that the Internet delay space adheres to a power law that extends over a significant range of Internet distances, namely the intra-continental distances. Therefore, its dimensionality can be characterized using fractal metrics sensitive to non-linear as well as linear structures in the delay space. Moreover, they are intrinsic properties of the delay space, independent of the target host metric space (e.g. Euclidean space). Therefore, the delay space dimensionality can be measured without computing coordinates for points, and is not subject to dimensionality inflation caused by embedding algorithms.

We have used the proposed fractal measures to quantify the extent to which geodesic distances are reflected in the dimensionality of the Internet, and we have also shown

that when decomposing the Internet into subnetworks consisting of hosts that share an upstream Tier-1 autonomous system in common, we observe a reduction in the intrinsic dimensionality of the pieces.

Moreover, both fractal measures were able to capture the role of the Internet’s AS-level topology in determining its delay space geometry, a factor not revealed by previously applied methods. Accordingly, linear methods, as well as embedding algorithms based on linear optimization problems, such as PCA and Vivaldi, are insensitive to subtle structural features of the network, thereby explaining the disappointing degree of accuracy observed in previous studies. Furthermore, our discovery of a dimensionality-reducing decomposition of the delay space lends support to the theory that embedding techniques based on hierarchical decompositions [147] may outperform existing techniques that attempt to embed the entire distance matrix in one shot.

An open question that we leave for future work concerns the applicability of non-linear dimensionality reduction techniques originally developed in the machine learning and pattern recognition communities, e.g. [80, 133]. These methods, including the Isomap method briefly studied in Section 4.4.2 as well as more recent diffusion-based techniques [80], are based on metric embeddings which can reflect the intrinsic geometry of datasets and allow multiscale analysis for solving dimensionality reduction, clustering and parametrization.

Finally, we expect that the future availability of more comprehensive datasets, containing a larger number of representatives, including hosts in edge networks, and path information (i.e., traceroutes) will allow us to better quantify the effectiveness of the fractal measures and discover subtle properties of the delay space geometry that cannot be fully contemplated via analyzing the King datasets.

CHAPTER 5

STATUS AND POWER IN ONLINE SOCIAL EXCHANGE

Exchanging resources constitutes a significant part of everyday social interactions. Whether material, symbolic or informational, resources flow between individuals either in overtly negotiated exchanges, as a result of some form of contract, or through informal exchanges, often sustained by reciprocity. Whatever the form of *social exchange*, actors frequently hold differential resource endowments, a situation that potentially results in power imbalances. Actors deprived of some resources find themselves dependent on resource owners, on whom they rely to achieve their desired goals.

Richard Emerson outlined in his foundational work the core tenets of Power-Dependence Theory [47]. Emerson posited that a fundamental tension emerges when one actor is more dependent on an exchange partner than vice-versa. In his theory, power relates to dependence via the equality $P_{AB} = D_{BA}$. In other words, A's power over B (P_{AB}) is a function of B's dependence on A (D_{BA}) in the relationship. Thus, an actor has power over another inasmuch as the other is dependent on him or her for valued resources for which alternative sources are scarce.

Because of the inherent tension in relations of asymmetrical dependence, Power-Dependence Theory predicts that actors will engage in behaviors to move the relationship to a more balanced state. Several processes may bring the relationship closer to power balance. For instance, the low-power actor may reduce her dependence by withdrawing from the relationship or by seeking other sources for the resource provided by the high-power actor. Alternatively, the dependent actor may increase power in the relationship by providing the partner with another valued resource in return.

In this work we focus on status giving [46] as a means by which a low-power actor

may lessen his or her dependence on a more powerful partner in an exchange. In this case, the dependent actors reward their powerful partners with status by granting them a higher level of esteem [119], thus producing a more balanced power relationship, at least as perceived by the actors involved¹. Experiments based on Status Characteristics Theory also support that status giving occurs in power-imbalanced situations. For instance, Ridgeway et al. show that status beliefs formed in the wake of a resource-unbalanced encounter tend to be more favorable to resource-advantaged individuals [119].

Status and power are often conflated conceptually, given their frequent interrelation in social settings [55, 142]. In spite of this frequent coincidence, there is no necessary relationship between the two concepts [142]. Power leads to status only when certain conditions are met. In particular, how power is used is an important mediator of the relationship between power and status. Our case study concerns a particular way – generosity – in which power obtained through resource access is exercised. Willer et al. shows that this behavior leads to status giving under laboratory conditions [142].

Despite the prominent role of status giving in everyday exchanges, little has been done to test Emerson’s theory beyond controlled laboratory experiments. In accordance with Emerson’s prediction, we observe status giving on a scale not addressed in previous work. To this end, we analyze the CouchSurfing.org network, a service that allows its worldwide user base of over 4 million to make contact with the aim of hosting one another.

CouchSurfing presents a particularly promising opportunity for testing Emerson’s predictions regarding status giving as a power-balancing mechanism when viewed as an organization that facilitates social exchange. In this context “couches” (places to spend the night) are the primary resource of interest to a traveler (henceforth “surfer”) in

¹Power-balance should not be construed to mean that power is no longer exercised in the relationship, however [45].

search of a place to stay in a particular city for a determined time period. Not only does opening one's house represent an act of generosity, but also even the most committed host's ability to provide others with hospitality is limited. Accordingly, we argue that the intrinsic value of hospitality, as well as its scarcity, leads to a high valuation of the host's resources by the surfer, making for a power imbalance in the relationship.

Even though CouchSurfing has been the object of a number of previous studies [28, 83, 134], to our knowledge no such study has investigated exchange-theoretic predictions. Here, we analyze the behavior of surfers with respect to status giving as a strategy for balancing the perceived power in their relationship with the host. Hosts control a resource, i.e., a place to sleep in a certain city, which surfers by definition do not possess. This makes for a typical power-unequal situation, with surfers dependent on hosts for obtaining accommodations. As a result, we expect to observe the employment of power-balancing strategies, such as status giving, as predicted by Emerson. The relationship between the surfer's dependence on the host and the surfer's status giving behavior resembles the exchange of status for advice, the example Peter Blau used for social exchange in an organizational setting [29].

The opportunity for surfers to engage in status giving emerges after the hosting interaction is completed, when users typically rate each other on the perceived strength of the tie and the perceived level of mutual trust. Tie strength ratings are made public and communicated to the partner, whereas trust ratings are stored privately (and never reported) in the web service's databases.

We test the intuition that power-balancing through status giving occurs in CouchSurfing using anonymized records of tie strength and trust ratings. If this strategy is adopted by surfers, then they will give status to their hosts by awarding them higher ratings than the surfers receive from their hosts. We examine this prediction in Study 1.

In Study 2, we consider additional exchange-theoretic predictions, by investigating the effect of the relative scarcity of hospitality resources on status giving.

5.1 Materials and Methods

In this section we describe the dataset and the Linear Mixed Effects model we used throughout this chapter.

5.1.1 The Couchsurfing Dataset

We analyze a sample of anonymized dyadic data with 80,194 hospitality interactions occurring between verified users (see Section 5.1.2) across the world and facilitated by CouchSurfing between January 2003 and November 2011. The data include tie strength and trust ratings. To establish a tie with other users in the network, the service presents users with two mandatory rating tasks whose outcomes constitute our dataset: (1) the strength of the tie and (2) the level of trust between the parties. Tie strength becomes visible to the other party, and we map its six ordered levels into integer values, namely “1: Acquaintance,” “2: Couchsurfing Friend,” “3: Friend,” “4: Good Friend,” “5: Close Friend,” and “6: Best Friend”. Conversely, trust ratings are recorded, but never reported. This rating spans a set of five ordinal values: “1: Do not Trust,” “2: Trust Somewhat,” “3: Generally Trust,” “4: Highly Trust,” and “5: Would Trust with Life.” Study 2 is conducted on a narrower sub-sample of hosting interactions initiated through a “CouchRequest.” We further impose the requirement that hosts included in this smaller sample live in cities with at least two other hosts, because we aim to compare hosts within the same city.

In our sample of dyadic hospitality interactions, 92.5% result in a mutual exchange of ratings, where 86.5% of the participants placed their ratings within three calendar months of the actual interaction date.

On CouchSurfing verification represents a process through which a user allows the organization to confirm their identity. The first step is making a purchasing power-adjusted payment to the organization from a credit card bearing the same name and address as one's profile. CouchSurfing then mails a postcard with a unique code to the given address. To gain fully verified status, the CouchSurfer then introduces the code they have received back to the website. We impose a minimum threshold of involvement with CouchSurfing by requiring that all users in our sample have undertaken the time and resource investment necessary to complete the verification process at the time of the interaction.

We present univariate statistics for the samples on which we based our dyadic analyses in Table 5.1.

5.1.2 Linear Mixed Effects

Because each of the raters and the rated users may be the source of multiple dyadic observations, we use Linear Mixed Effects (LME) model [109] (The regression was computed using the `lmer` function in the R package `lme4` [23]) to account for this structure of observational interdependence. LME is a multivariate linear regression where the error terms are assumed to be normally distributed. Nevertheless, the error term is broken down into multiple additive terms to account for variance at the group level (e.g., the ratings given by users are independent given the identity of users). Therefore, we can model the problem in LME in such a way as to net out the inter-user variance from the

residual, by considering the set of ratings from the same user as a group). The grouping of error terms is an attempt to account for possible bias resulting from non-independence of errors in the same group. In the LME vocabulary, the independent variables are called *fixed effects*. The variance at the group level is modeled as a random variable, one for each group, called *random effects*. Specific to our experiment, we model individual raters and rated users as random effects, to account for dyadic interdependencies.

5.2 Study 1: Status Giving From Surfer to Host

We first test for the existence of the status giving mechanism for balancing unequal power relationships, as described by Emerson [46]. We study 80,194 randomly selected pairs of ratings exchanged on CouchSurfing.² Our expectation is that surfer to host friendship and trust ratings will exceed the reverse ratings, given by hosts to surfers.

Table 5.2 plots the counts of dyads according to the value of friendship ratings given by the two exchange partners, within the same dyad. Plotted on the diagonal are rating pairs of equal magnitude: above the diagonal we show instances where surfer to host ratings were higher than host to surfer ratings, and below the diagonal we count cases where host to surfer ratings were higher. We can compare frequency counts between cells symmetrical to the diagonal. There were, for instance, 7,579 cases where the surfer nominated the host as a “friend,” and the host responded with a counter-nomination as “CouchSurfing friend,” a category closer to the bottom of the scale used by CouchSurfing. This count exceeds by 1,215 cases the 6,364 instances in the reverse pair of host to surfer ratings. This imbalance holds for most of the symmetric entries in the table, and a chi-square test ($\chi^2 = 29136.6, df = 25$) reveals a significant level of association

²An extended abstract that discusses the main results of Study 1 has appeared in [128].

	Study 1			Study 2		
	Mean (Freq.)	(S.D.)	Count	Mean (Freq.)	(S.D.)	Count
Host's friendship rating						
Acquaintance	.01		729	.01		288
CS friend	.70		56,333	.57		20,623
Friend	.15		12,081	.10		3,666
Good friend	.08		6,063	.05		1,796
Close friend	.02		1,329	.01		416
Best friend	<.01		364	.01		180
N.A.	.04		3,295	.25		9,187
Surfer's friendship rating						
Acquaintance	.01		541	<.01		194
CS friend	.70		56,043	.56		20,248
Friend	.17		13,396	.11		3,980
Good friend	.08		6,571	.05		1,962
Close friend	.02		1,444	.01		425
Best friend	<.01		347	<.01		176
N.A.	.02		1,852	.25		9,171
Host's trust rating						
Do not trust	<.01		160	<.01		48
Somewhat	.12		9,672	.09		3,277
Generally	.48		38,160	.37		13,455
Highly	.31		24,487	.24		8,657
with life	.03		2,347	.02		845
N.A.	.07		5,368	.27		9,874
Surfer's trust rating						
Do not trust	<.01		193	<.01		67
Somewhat	.09		7,287	.07		2,567
Generally	.43		34,089	.33		11,841
Highly	.40		31,678	.30		10,720
With life	.05		3,725	.04		1,319
N.A.	.04		3,222	.27		9,642
Host's prior CouchRequests						
Received				20.85	(37.05)	30,703
Accepted				6.06	(8.22)	30,703
Host's city prior CouchRequests						
Received				6,218	(12,179)	30,345
Accepted				876	(1,392)	30,345

Table 5.1: Means and standard deviations for continuous variables, and frequency and counts for discrete variables (datasets 1 and 2) Study 1: N=80,194 post hosting interactions from 34,755 verified surfers to 32,093 verified hosts. Study 2: N = 36,156 requests from 19,100 verified surfers to 19,192 verified hosts. Hosts in Study 2 were living in 4,533 cities having at least two other CouchSurfing hosts. On average 7.97 Study 2 hosts lived in any one city. Study 1 data collected between January 2003 and November 2011. Study 2 data collected between July 2010 and November 2011.

Host to surfer	Surfer to host						N.A.	Total
	Acq.	CS friend	Friend	Good	Close	Best		
Acq.	33	576	62	31	2	0	25	729
CS friend	403	43,804	7,579	2,736	362	65	1,384	56,333
Friend	51	6,364	3,844	1,312	229	31	250	12,081
Good	4	2,359	1,244	1,945	335	56	120	6,063
Close	2	327	203	282	432	56	27	1,329
Best	1	78	40	57	43	131	14	364
N.A.	47	2,535	424	208	41	8	32	3,295
Total	541	56,043	13,396	6,571	1,444	347	1,852	80,194

Table 5.2: Host and surfer's reports of friendship strength. $\chi^2 = 29136.6$, $df = 25$.
T-stat(H_a : Surfer > Host) = 10.885, $df=75078$.

Host to surfer	Surfer to host					N.A.	Total
	Do not	Somewhat	Generally	Highly	W/ life		
Do not trust	0	23	70	42	13	12	160
Somewhat	39	1,218	4,595	3,003	269	548	9,672
Generally	76	3,732	17,023	14,415	1,326	1,588	38,160
Highly	44	1,564	9,189	11,325	1,573	792	24,487
With life	4	111	679	1,121	358	74	2,347
N.A.	30	639	2,533	1,772	186	208	5,368
Total	193	7,287	34,089	31,678	3,725	3,222	80,194

Table 5.3: Host and surfer's reports of trust $\chi^2 = 2190.05$, $df = 16$. T-stat(H_a : Surfer > Host) = 44.131, $df=71811$.

between the matched ratings exchanged within a dyad. Furthermore, a one-sided t-test ($t = 10.885$, $df = 75,078$) shows a statistically significant difference between host to surfer and surfer to host friendship ratings. Overall 13,432 ratings were higher from surfer to host than vice-versa, whereas 11,458 rating pairs were symmetrical.

Finally, missing data suggest another status giving process at work. In 3,295 cases hosts did not award any ratings to their partners, a value 78% higher than the 1,852 instances in which surfers neglected to give any ratings. This is consistent with the status giving hypothesis. Given that rating other users on CouchSurfing requires at least a few minutes to answer the nine mandatory questions on the form used on the website,

we would expect surfers to be more likely than hosts to spend the time to give a rating to their partner. Thus, the host is not only more likely to rate the surfer lower on the friendship scale, but they are also more likely not to give any rating at all.

Hosts and surfers also produce trust ratings of each other that are unreported to the other party. Comparing the frequencies of anonymized public and unreported ratings helps us determine to what extent status giving persists when ratings are not shown to the recipient. In Table 5.3 we present the counts of trust ratings exchanged between host and surfer, using the same conventions as before. In this case the results are even more poignant: surfers say they trust hosts more than hosts declare they trust surfers in 25,329 of the cases, 53% higher than the 16,559 cases in which the reverse happens. As in the case of friendship ratings, the association is significant ($\chi^2 = 2,190.05$, $df = 16$), and surfers rate their partners higher than hosts ($t = 44.131$, $df = 71,811$). As we observed with friendship evaluations, missing data proves to be more prevalent in host to surfer ratings.

Some of the ratings presented in Tables 5.2 and 5.3 may be affected by anchoring effects [134], i.e., the first actor to give a rating in the dyad influences the rating of the second. To eliminate this effect, in Section 5.2.1 we present a complement of the preceding within-dyad bivariate analysis with a between-dyads comparison. We compile a sample containing only one rating from each dyad. Specifically, we consider the rating given in a dyad by the party who initiated the interaction (i.e., the rating process) chronologically. The results show that when surfers rate first and thus “set the tone” for the rating exchange, they will give higher ratings than when hosts first avail themselves of the opportunity.

FIXED EFFECTS			
Independent Variable	Coefficient	(S.E.)	T-value
Model 1: Response: friendship rating given to partner			
Intercept	3.341***	0.004	794.848
Rater was surfer	0.062***	0.005	11.527
Model 2: Response: trust rating given to partner			
Intercept	4.271***	0.004	1037.574
Rater was surfer	0.162***	0.005	30.732
Model 3: Response: friendship ratings given to partner			
Intercept	3.026***	0.055	55.023
Rater was surfer	0.002	0.005	0.464
Trust ratings (ref: "Do not trust"):			
... Trust somewhat	0.042	0.056	0.770
... Generally trust	0.187***	0.055	3.403
... Highly trust	0.553***	0.055	10.051
... Trust with life	1.250***	0.056	22.179
RANDOM EFFECTS			
	Variance	Std. dev.	
Rater			
Model 1	0.0704	0.2653	
Model 2	0.0674	0.2597	
Model 3	0.0645	0.2540	
Rated User			
Model 1	0.0681	0.2610	
Model 2	0.0693	0.2633	
Model 3	0.0612	0.2474	
Residual			
Model 1	0.4241	0.6512	
Model 2	0.3875	0.6224	
Model 3	0.3731	0.6108	

Table 5.4: Linear mixed-effects regressions comparing first ratings given in each dyad. *Source:* CouchSurfing dataset No. 1. Only one-directional ratings included. Sample sizes: 75,902; 74,311; 74,263. Scaled deviances: 168,841; 159,782; 156,170. Log-likelihoods: -84,429; -79,900; -78,107. AIC: 168,869; 159,810; 156,233. * $p < .10$, ** $p < .05$, *** $p < .01$. Two-tailed tests.

5.2.1 Between-dyads Analysis

To eliminate the effects of anchoring that may be present in the bivariate analysis in Study 1, here we present a complement of the within-dyad bivariate analysis with a between-dyads comparison. We compile a sample containing only one rating from each dyad. Specifically, we consider the rating given in a dyad by the party who initiated the interaction (i.e., the rating process) chronologically. Surfers were first to give ratings in 47% of cases, whereas hosts were first 53% of the time. We fit the data using three different Linear Mixed Effects (LME) models³, which we present in Table 5.4. Models 1 and 2 measure the friendship and trust ratings given to the partner as responses, respectively. To test our prediction that higher status is more likely to flow from surfer to host than vice-versa, we include the first rater's role (surfer or host) as a fixed effect. Model 3 adds the anonymized trust ratings given to the first person to be rated in the dyad as a control variable in the regression of the publicly displayed friendship ratings. Doing so allows us to ask whether the observability of ratings by the receiving party influences status giving behavior. By observing the change from Model 1 to Model 3 in the coefficient associated with the first rater's role ("Rater was surfer"), we can test which of two possible scenarios occurred. In the first, the surfer gives status to the host publicly while possibly holding other private beliefs. In the second, the surfer genuinely holds the host in higher esteem and displays status giving behavior both publicly and privately.

As expected, Models 1 and 2 show a positive effect on the first rating given, when the surfer is the rater (0.062 and 0.162, respectively). In Model 3 we notice the disappearance of a statistically significant effect of the first rater's being a surfer rather than a host, the coefficient decreasing to 0.002. Here the status giving effect of being a surfer

³See section 'Mixed Linear Models' for a brief description of the method.

appears to load exclusively on the private trust rating. That is, surfers are different from hosts in their private trust ratings, but once this private information is accounted for, being a surfer has no additional effect on the public friendship ratings. Given the coincidence of public and private ratings, rating disclosure does not have an added effect.

5.3 Study 2: Relative Valuation of Hospitality

The preceding study shows the existence of a status giving effect, whereby hosts are rated higher by surfers than vice-versa. Power-Dependence Theory posits a more elaborate relationship between power and status, however. The “scarcer” the host’s hospitality resource, the higher the valuation we expect surfers to place on it, and, therefore, the more status we expect surfers to give to their hosts [45].

The scarcity of a resource may be conceived in two ways. One is the host’s own popularity and inclination to host, relative to other hosts. Here we expect that more popular hosts receive more status, whereas hosts who tend to accept many requests (compared to the prevailing local norm) would be given less status. The second interpretation of scarcity deals with the availability of alternatives. Power-Dependence Theory predicts that a surfer in a city that is in high demand but where hosts, as a rule, accept few requests (e.g., Paris) should value the hospitality they receive more than if they had surfed in a low-demand, high-acceptance city. Thus, we expect status given by the surfer to be directly correlated with the number of requests received by hosts in the city, but inversely correlated with the number of accepted requests.

Our analysis confirms the assumed relationship between resource scarcity and power-imbalance: hosts who accept more requests receive less status, as do hosts living in cities that are in less demand where a request is more likely to be accepted. In a sense,

FIXED EFFECTS			
Independent Variable	Coefficient	(S.E.)	T-value
Intercept	2.809***	0.095	29.503
Surfer to host trust rating (ref.: “Do not Trust”)			
Trust somewhat	0.102	0.084	1.215
Generally trust	0.233***	0.084	2.783
Highly trust	0.494***	0.084	5.896
Would trust with life	1.049***	0.086	12.219
Host to surfer friendship rating (ref.: “Acquaintance”)			
CouchSurfing friend	0.080*	0.043	1.882
Friend	0.330***	0.044	7.505
Good friend	0.700***	0.046	15.374
Close friend	1.371***	0.055	25.159
Best friend	1.933***	0.069	27.904
Inclination to host			
Req. received (log)	0.002	0.007	0.299
Req. accepted (log)	-0.017**	0.008	-2.143
Availability of alternatives in the host’s city‡			
Req. received/host (log)	0.035***	0.013	2.585
Req. accepted/host (log)	-0.061**	0.025	-2.459
RANDOM EFFECTS			
	Variance	Std. dev.	
Host intercept	0.0288	0.1698	
Surfer intercept	0.1664	0.4079	
Host’s city intercept	0.0013	0.0364	
Residual	0.2285	0.4781	

Table 5.5: Linear mixed-effects regression. Response: friendship rating from surfer to host. *Source:* CouchSurfing dataset No.2. Sample size: 20,917 Scaled deviance: 39,322.29. Log-likelihood: -19,705.3. AIC: 39,446.6. 1 was added to all log-transformed quantities before taking the natural logarithm. * $p < .10$, ** $p < .05$, *** $p < .01$. Two-tailed tests. ‡Measured as the mean number of requests received and accepted by other hosts in the host’s city, during the 90-day interval prior to when the surfer made the initial request for the host’s hospitality.

hosts have the option of making their “couch” scarcer or more readily available than the prevailing behavior of other hosts. This distinction between the host’s behavior and the overall supply and demand conditions prevailing for hospitality in the city suggests a connection to the distinction drawn by Ekeh [44] between economic and social scarcity: sometimes it is social norms, and not implicit market-like conditions that make goods scarce.

To test these predictions we use a mechanism implemented by CouchSurfing starting in April 2010. From that point onward, surfers were given the option to issue hospitality requests to potential hosts through a standardized message, a “CouchRequest.” As a result, it is possible to observe whether or not a host agreed to a surfer’s request and to compute each host’s likelihood of accepting such a request. The focal measures in this study represent the 36,156 post-interaction ratings from Study 1 that could be matched with the pre-interaction CouchRequests sent by surfer to host. We likewise constrained our sample to include only those CouchRequests sent starting in July, 2010, as we computed indicators of each host’s prior responses during the ninety-day interval before the request was sent.

The fitted data are presented in Table 5.5 using a linear mixed-effects model (see Section 5.1.2) with the surfer’s friendship rating of the host as the response. To account for sources of unobserved variation, we added random effects for individual hosts and surfers participating in the interaction, as well as for the host’s city.

Friendship ratings reflect not only a status giving process, but also the outcome of an interpersonal interaction. For instance, both actors may rate each other as close friends after a particularly meaningful, prolonged conversation, or as mere acquaintances after a desultory interaction. Our analysis is concerned with the extent to which the actors’ post-interaction evaluations express status giving, rather than with the quality and depth

of the interaction itself. We remove to the extent possible variance due to the specific realization of the interaction by controlling for the surfer to host private trust rating, as well as for the public friendship “counter-rating” given by host to surfer. The control variables represent alternative baselines. The surfer to host private trust rating measures the surfers privately-held beliefs, without concern for social norms. Conversely, the host to surfer public friendship rating establishes another evaluation of the situation, from the point of view of the presumed recipient of status.

To test our hypotheses regarding the direct valuation of the host’s resource, we include in the regression a fixed-effect for the log-transformed prior number of requests received by the host during the ninety days before the request leading to the focal interaction. This effect is not statistically significant, however. We likewise include a fixed effect measuring the log-transformed number of prior requests the host accepted, yielding an average decrease of -0.017 in the received friendship rating, for each unit increase in the log-transformed number of previously-accepted requests.

Our predictions regarding the availability of alternatives are tested by including fixed effects for the log-transformed total number of received and accepted requests in the host’s city, by hosts other than the rated individual, with significant effects of magnitude 0.035 and -0.061 , respectively.

The results suggest a trade-off: those seeking to gain recognition in the organization may choose to exchange with more users but they will receive less status from any single interaction. In contrast, they may elect a more exclusive strategy, by accepting fewer exchanges but receiving more status from each exchange. Despite its statistical significance, the effect is small: the effect of the log-transformed number of acceptances (with respect to other hosts in the city) is three times smaller than that of the log-transformed mean number of accepted requests in the host’s city. The fact that we can even observe

the effect illustrates the richness of the CouchSurfing dataset, but the small magnitude argues for more specific research on the mechanisms through which scarcity is created in social exchange and its effects.

5.4 Discussion

Our investigation represents a first foray into the applications of Social Exchange Theory to the emerging wealth of detailed data resulting from trust-mediated social exchange supported by online platforms, of which CouchSurfing is a prime example.

In Study 1 we documented the flow of status from dependent surfers to powerful hosts in hospitality exchanges mediated by CouchSurfing. Study 2 suggested that resource scarcity mediates the amplitude of status giving as suggested by Social Exchange Theory: hosts who are “stingier” with their acceptances are more likely to get a higher rating, as are hosts in more desirable cities with fewer available couches. The signal of these effects detected in our dataset is rather small in magnitude, however, and more in-depth research is needed to further explore this aspect of power and status relations in similar contexts.

Another insight emerging from our research concerns the mediating role of the privately reported trust ratings in the flow of status between surfers and hosts. Even when asked privately, surfers seem to hold their hosts in genuinely higher esteem. This suggests that at least part of status giving is the result of private conviction rather than public performance. As Study 2 reveals, however, scarcity-dependent effects do emerge in public friendship ratings even when we net out the effect of privately expressed trust. This finding raises interesting questions about the complex role of observability and disclosure in the status giving process.

It is important to emphasize that CouchSurfing relies on an Internet platform and, therefore, our findings may have particularly interesting consequences for similar services. Indeed, many Web-based platforms have emerged to facilitate peer-to-peer exchange – monetized or not – of products and services. While the fundamental role of material incentives should remain in sight, our work argues for the need to consider alternative exchange mechanisms – such as status giving – through which economies may function.

Status giving as a mechanism for balancing a power-unequal relationships operates in relatively subtle ways. The fact that CouchSurfers can receive status by playing the role of host provides one possible explanation for a popular dilemma regarding CouchSurfing: its very existence. At first blush, it would appear as if there were no barrier to continuously exploiting the organization’s hospitality resources and never giving anything back to the common pool of “couches.” Similarly puzzling is the behavior of hosts, who usually do not receive anything tangible in return from their guests. If CouchSurfers were to follow their material incentives, the network would unravel: surfers would take advantage of hosts’ generosity, without ever contributing to the common pool, a classic free-rider problem [108]. Conversely, hosts’ incentives would guide them to withhold contributing, as they (seemingly) receive nothing in return.

Status giving comes into play as a possible explanation for why such “freeloading” behavior has not become a prevalent and dominant strategy. Even though few, if any, material benefits accrue to hosts from their generosity, hosts receive status in the organization instead. We advance this explanation as an instance in which status giving may serve to maintain a seemingly-tenuous process of generalized social exchange, a process which prior research has shown to operate in experimental settings [141].

We hope that the insights and perspectives we offer in this work will be confirmed

and further enlightened by future investigations of other social networking services representing the many forms of social exchange mediated by the Internet, a data source that have created unlimited opportunities for the study of fundamental questions in the social sciences at a scale not possible in the past.

BIBLIOGRAPHY

- [1] Hostip.info database. <http://www.hostip.info>.
- [2] P2PSim. <http://www.pdos.lcs.mit.edu/p2psim/>.
- [3] The new 5-year \$37.8 million initiative to data analysis by the Gordon and Betty Moore Foundation and Alfred P. Sloan Foundation. <http://www.moore.org/programs/science/data-driven-discovery/data-science-environments> (last accessed on 05/09/2014).
- [4] The Whitehouse blog: Big Data is a Big Deal. <http://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal> (last accessed on 05/09/2014).
- [5] University of Oregon RouteViews Project. <http://www.antc.uoregon.edu/route-views/>.
- [6] S. Abe and N. Suzuki. Omori's law in the Internet traffic. *Europhysics Letters*, 61(6), 2003.
- [7] Ittai Abraham, Yair Bartal, Hubert T-H. Chan, Khedar Dhamdhere, Anupam Gupta, Jon Kleinberg, Ofer Neiman, and Aleksandrs Slivkins. Metric embeddings with relaxed guarantees. In *Proc. of IEEE FOCS*, 2005.
- [8] Bruno Abrahao, Flavio Chierichetti, Robert Kleinberg, and Alessandro Panconesi. Trace complexity of network inference. In *Proc. of the 19th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2013.
- [9] Bruno Abrahao and Robert Kleinberg. On the internet delay space dimensionality. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, IMC '08, pages 157–168, New York, NY, USA, 2008. ACM.
- [10] Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. On the separability of structural classes of communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 624–632, New York, NY, USA, 2012. ACM.
- [11] Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. On the separability of structural classes of communities. In *Proc. of the Eighteenth ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, 2012.

- [12] Bruno Abrahao, Sucheta Soundarajan, John Hopcroft, and Robert Kleinberg. A separability framework for analyzing community structure. *ACM Trans. Knowl. Discov. Data*, 8(1):5:1–5:29, February 2014.
- [13] Bruno D. Abrahao and Robert D. Kleinberg. On the internet delay space dimensionality. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 419–419, New York, NY, USA, 2008. ACM.
- [14] Eytan Adar and Lada A. Adamic. Tracking information epidemics in blogspace. In *Proc. of the 2005 IEEE/WIC/ACM Int'l Conf. on Web Intelligence*, 2005.
- [15] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, January 1991.
- [16] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.
- [17] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proc. of the 12th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, 2006.
- [18] J. P. Bagrow and E. M. Bollt. A local method for detecting communities. *Phys. Rev. E*, 72:046108, 2005.
- [19] Norman Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London, 1975.
- [20] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proc. of the 4th ACM Int'l Conf. on Web search and Data Mining*, 2011.
- [21] Brian Ball, Brian Karrer, and M. E. J. Newman. Efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84:036103, Sep 2011.
- [22] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.
- [23] Douglas M. Bates. *lme4: Mixed-Effects Modeling with R*. Springer, 2010.

- [24] Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismael. Efficient identification of overlapping communities. In *Proc. of the 2005 IEEE Int'l. Conf. on Intelligence and Security Informatics*, pages 27–36, 2005.
- [25] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.
- [26] Alberto Belussi and Christos Faloutsos. Estimating the selectivity of spatial queries using the ‘correlation’ fractal dimension. In *Proc. of VLDB*, 1995.
- [27] Andrew C. Berry. The accuracy of the Gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49:122–136, 1941.
- [28] Paula Bialski and Dominik Batorski. From Online Familiarity to Offline Trust. In C. S. Ang and P. Zaphyris, editors, *Social Computing and Virtual Communities*. CRC, 2009.
- [29] Peter M. Blau. *Exchange and Power in Social Life*. Wiley, 1964.
- [30] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+, July 2008.
- [31] S. G. Bobkov and M. Ledoux. On modified logarithmic sobolev inequalities for bernoulli and poisson measures. *Journal of Functional Analysis*, 156(2):347 – 365, 1998.
- [32] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52, 1985.
- [33] Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proc. of ACM STOC*, 2005.
- [34] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer, 2005.
- [35] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, December 1996.
- [36] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community

- structure in very large networks. *Physical Review E*, 70(6):066111+, December 2004.
- [37] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.
- [38] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet coordinates for distance estimation. In *Proc. of ICDCS*, 2004.
- [39] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6), 1997.
- [40] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. In *Proc. of ACM SIGCOMM*, 2004.
- [41] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, 2008.
- [42] NAN DU, Le Song, Alex Smola, and Ming Yuan. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems 25*, pages 2789–2797. 2012.
- [43] Rick Durrett. *Probability: Theory and examples*. Cambridge Series in Statistical and Probabilistic Mathematics, 2011.
- [44] Peter P. Ekeh. *Social Exchange Theory*. Harvard University Press, 1974.
- [45] Richard M. Emerson. Exchange Theory Part I: A Psychological Basis for Social Exchange. In Joseph Berger, M. Zelditch, and B. Anderson, editors, *Sociological Theories in Progress*, pages 38–57. Houghton Mifflin, 1972.
- [46] Richard M. Emerson. Exchange Theory Part II: Exchange Relations and Network Structures. In Joseph Berger, M. Zelditch, and B. Anderson, editors, *Sociological Theories in Progress*, pages 58–87. Houghton Mifflin, 1972.
- [47] R.M. Emerson. Power-dependence relations. *Am. Sociol. Rev.*, 27(1):31–41, 1962.
- [48] P. Erdős and A Rényi. On the evolution of random graphs. In *Pub. of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

- [49] Carl-Gustav Esseen. A moment inequality with an application to the central limit theorem. *Skand. Aktuarietidskr.*, 39:160170, 1956.
- [50] T.S. Evans and R. Lambiotte. Line graphs, link partitions and overlapping communities. *Phys.Rev.E*, 80:016105, 2009.
- [51] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, August 1999.
- [52] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4), 1999.
- [53] N. Fatemi-Ghomi, P.L. Palmer, and M. Petrou. The two-point correlation function: A measure of interclass separability. *Journal of Mathematical Imaging and Vision*, 10:7–25, 1999.
- [54] Santo Fortunato. Community detection in graphs. *Phys. Reports*, 486:75–174, 2010.
- [55] Adam D. Galinsky, Joe C. Magee, Deborah H. Gruenfeld, Jennifer A. Whitson, and Katie A. Liljenquist. Power reduces the press of the situation. *Journal of Personality and Social Psychology*, 95(6):1450–1466, 2008.
- [56] M. Girvan and M. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, June 2002.
- [57] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, 2002.
- [58] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Stat. Assoc.*, 102:359–378, 2007.
- [59] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. of the 28th Int’l Conf. on Machine Learning*, 2011.
- [60] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proc. of the 16th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining*, 2010.

- [61] Steve Gregory. A fast algorithm to find overlapping communities in networks. In *Proc. of the 2008 European Conf. on Machine Learning and Knowledge Discovery in Databases - Part I*, pages 408–423, 2008.
- [62] Vincent Gripon and Michael Rabbat. Reconstructing a graph from path traces. *CoRR*, abs/1301.6916, 2013.
- [63] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. Information diffusion through blogspace. In *Proc. of the 13th Int'l Conf. on World Wide Web*, 2004.
- [64] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: estimating latency between arbitrary Internet end hosts. *SIGCOMM Comput. Commun. Rev.*, 32(3), 2002.
- [65] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. of IEEE FOCS*, 2003.
- [66] Mark A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato, 1999.
- [67] Jake M. Hofman and Chris H. Wiggins. Bayesian Approach to Network Modularity. *Physical Review Letters*, 100(25):258701+, June 2008.
- [68] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439, 2006.
- [69] Bradley Huffaker, Marina Fomenkov, Daniel J. Plummer, David Moore, and K Claffy. Distance metrics in the Internet. In *Proc. of IEEE Intl. Telecom. Symposium (ITS)*, 2002.
- [70] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proc. of FOCS*, 2001.
- [71] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.*, 26, 1984.
- [72] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, December 1998.

- [73] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2003.
- [74] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(1):291–307, 1970.
- [75] Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *Proc. of IEEE FOCS*, 2004.
- [76] Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es), December 1999.
- [77] Jon M Kleinberg. Navigation in a small world. *Nature*, August 2000.
- [78] Christian Komusiewicz, Falk Huffner, Hannes Moser, and Rolf Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, 410(38a-40):3640 – 3654, 2009.
- [79] Ioannis Kontoyiannis and Mokshay Madiman. Measure concentration for compound poisson distributions. *Electron. Commun. Probab.*, 11:no. 5, 45–57, 2006.
- [80] Stephane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.
- [81] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80:056117, Nov 2009.
- [82] Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.
- [83] Debra Lauterbach, Hung Truong, Tanuj Shah, and Lada Adamic. Surfing a web of trust: Reputation and reciprocity on couchsurfing.com. In *Proc. of the International Conference on Computational Science and Engineering*, pages 346–353, 2009.
- [84] E. Lebhar, P. Fraigniaud, and L. Viennot. The inframetric model for the Internet. In *Proc. of IEEE INFOCOM*, 2008.

- [85] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network coordinates in the wild. In *Proc. of USENIX NSDI*, 2007.
- [86] Sune Lehmann, Martin Schwartz, and Lars K. Hansen. Biclique communities. *Physical Review E*, 78(1):016108+, 2008.
- [87] Jure Leskovec, Lada Adamic, and Bernardo Huberman. The dynamics of viral marketing. In *Proc. of the 7th ACM Conf. on Electronic Commerce*, 2006.
- [88] Jure Leskovec, Kevin Lang, Anirban Dasgupta, and Michael Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. of the 17th Intl. Conf. on World Wide Web*, 2008.
- [89] Jure Leskovec, Kevin Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proc. of the 19th Intl. Conf. on World Wide Web*, 2010.
- [90] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proc. of the 15th ACM SIGKDD Int'l. Conf. on Knowledge Discovery and Data Mining*, pages 527–536, 2009.
- [91] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15, 1990.
- [92] Eng Keong Lua, Timothy Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft. On the accuracy of embeddings for Internet coordinate systems. In *Proc. of ACM/SIGCOMM Internet Measurement Conference*, 2005.
- [93] Russell Lyons and Yuval Peres. *Probability on Trees and Networks*. Cambridge University Press, 2012.
- [94] Harsha V. Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring, and Arun Venkataramani. A structural approach to latency prediction. In *Proc. of ACM/SIGCOMM Internet Measurement Conference*, 2006.
- [95] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane: An information plane for distributed services. In *Proc. of OSDI*, 2006.
- [96] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert Tarjan. Finding

- strongly knit clusters in social networks. *Internet Mathematics*, 5(1):155–174, January 2008.
- [97] Alan Mislove, Bimal Viswanath, Krishna Gummadi, and Peter Druschel. You are who you know: Inferring user profiles in online social networks. In *Proc. 3rd ACM Intl. Conf. on Web Search and Data Mining*, 2010.
- [98] Alan Mislove, Bimal Viswanath, Krishna Gummadi, and Peter Druschel. You are who you know: Inferring user profiles in online social networks. In *Proc. 3rd ACM Int’l. Conf. on Web Search and Data Mining*, 2010.
- [99] R. Motwani and P. Rahgavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [100] Klaus R. Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2), 2001.
- [101] Seth Myers and Jure Leskovec. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems 23*, pages 1741–1749. 2010.
- [102] Praneeth Netrapalli and Sujay Sanghavi. Learning the graph of epidemic cascades. In *SIGMETRICS*, pages 211–222, 2012.
- [103] M. Newman. Modularity and community structure in networks. *Proc. of the National Academy of Sciences*, 103(23):8577–8582, June 2006.
- [104] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003.
- [105] MEJ Newman. Detecting community structure in networks. *The European Phys. Journal B*, 38(2):321–330–330, March 2004.
- [106] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proc. of IEEE INFOCOM*, 2002.
- [107] Ricardo V. Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. In search of the elusive ground truth: the internet’s as-level connectivity structure. In *SIGMETRICS*, 2008.

- [108] Mancur Olson. *The logic of collective action: Public goods and the theory of groups*. Harvard University Press, Cambridge, 1965.
- [109] Lyman Ott and Michael Longnecker. *An introduction to statistical methods and data analysis*. Cengage Learning, December 2008.
- [110] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford InfoLab, Stanford, 1998. 17 p.
- [111] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.
- [112] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, June 2005.
- [113] Daniel Park, Rohit Singh, Michael Baym, Chung-Shou Liao, and Bonnie Berger. IsoBase: a Database of Functionally Related Proteins Across PPI Networks. *Nucleic Acids Research*, 39(suppl 1):D295–D300, 2011.
- [114] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3), 1995.
- [115] Vern Paxson and Sally Floyd. Why we don't know how to simulate the Internet. In *Proc. of Winter Simulation Conference*, 1997.
- [116] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In *Proc. of IPTPS*, 2003.
- [117] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. of Graph Algorithms and Applications*, 10(2):191–218, 2006.
- [118] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically aware overlay construction and server selection. In *Proc. of IEEE INFOCOM*, 2002.
- [119] C.L. Ridgeway, E.H. Boyle, K.J. Kuipers, and D.T. Robinson. How do status beliefs develop? *Am. Sociol. Rev.*, 63:331–350, 1998.

- [120] Everett M. Rogers and Everett Rogers. *Diffusion of Innovations*. Free Press, 5th edition, August 2003.
- [121] Martin Rosvall and Carl Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE*, 6(4):e18209, 04 2011.
- [122] Satu Elisa Schaeffer. Stochastic local clustering for massive graphs. In *Proc. of the 9th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining*, pages 354–360, 2005.
- [123] Manfred Schroeder. *Fractal, Chaos and Power Laws: Minutes from an Infinite Paradise*. W. H. Freeman and Co., NY, 1990.
- [124] Yuval Shavitt and Tomer Tankel. Big-bang simulation for embedding network distances in Euclidean space. *IEEE/ACM Transactions on Networking*, 12(6), 2004.
- [125] Yuval Shavitt and Tomer Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proc. of IEEE INFOCOM*, 2004.
- [126] Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388:1706–1712, 2009.
- [127] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proc. of ACM SIGCOMM*, 2003.
- [128] B. State, B. Abrahao, and K. Cook. From power to status in online social exchange. In *ACM Conference on Web Science*, pages 427–433, 2012.
- [129] Karen Stephenson and Marvin Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, March 1989.
- [130] G. Strang. *Linear Algebra and its Application, 2nd. Ed.* Academic Press, 1980.
- [131] Liying Tang and Mark Crovella. Virtual landmarks for the Internet. In *Proc. of ACM/SIGCOMM Internet Measurement Conference 2003*, 2003.
- [132] Liying Tang and Mark Crovella. Geometric exploration of the landmark selection problem. In *Proc. of Passive and Active Measurement Workshop*, 2004.

- [133] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2000.
- [134] C.Y. Teng, Debra Lauterbach, and L.A. Adamic. I rate you. You rate me. Should we do so publicly? In *Proc. of the 3rd USENIX conference on Online social networks*, 2010.
- [135] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition, November 2008.
- [136] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1st edition, September 1998.
- [137] Guohui Wang, Bo Zhang, and T. S. Eugene Ng. Towards network triangle inequality violation aware distributed systems. In *Proc. of ACM/SIGCOMM Internet Measurement Conference*, 2007.
- [138] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.
- [139] Fang Wei, Weining Qian, Chen Wang, and Aoying Zhou. Detecting overlapping community structures in networks. *World Wide Web*, 12(2):235–261, 2009.
- [140] E. Weinan, Tiejun Li, and Eric Vanden-Eijnden. Optimal partition and effective dynamics of complex networks. *Proc. of the National Academy of Sciences*, 105(23):7907–7912, 2008.
- [141] R. Willer. Groups reward individual sacrifice: The status solution to the collective action problem. *Am. Sociol. Rev.*, 74:23–43, 2009.
- [142] R. Willer, R. Youngreen, L. Troyer, and M.J. Lovaglia. How do the powerful attain status? *Managerial and Decision Economics*, 33:355–367, 33:355–367, 2012.
- [143] Walter Willinger, Will E. Leland, Murad S. Taqqu, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1), 1994.
- [144] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a lightweight network location service without virtual coordinates. *SIGCOMM Comput. Commun. Rev.*, 35(4), 2005.

- [145] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *12th IEEE Int'l. Conf. on Data Mining*, 2012.
- [146] Bo Zhang, T. S. Eugene Ng, Animesh Nandi, Rudolf Riedi, Peter Druschel, and Guohui Wang. Measurement based analysis, modeling, and synthesis of the Internet delay space. In *Proc. of ACM/SIGCOMM Internet Measurement Conference*, 2006.
- [147] Rongmei Zhang, Charlie Hu, Xiaojun Lin, and Sonia Fahmy. A hierarchical approach to Internet distance prediction. In *Proc. of IEEE International Conference on Distributed Computing Systems*, 2006.