



University of  
Massachusetts  
Amherst

## Enabling Accurate Analysis of Private Network Data

Item Type	Dissertation (Open Access)
Authors	Hay, Michael
DOI	<a href="https://doi.org/10.7275/1674270">10.7275/1674270</a>
Download date	2026-03-09 14:29:27
Link to Item	<a href="https://hdl.handle.net/20.500.14394/38754">https://hdl.handle.net/20.500.14394/38754</a>

**ENABLING ACCURATE ANALYSIS OF PRIVATE  
NETWORK DATA**

A Dissertation Presented

by

MICHAEL G. HAY

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

DOCTOR OF PHILOSOPHY

September 2010

Computer Science

© Copyright by Michael G. Hay 2010

All Rights Reserved

# ENABLING ACCURATE ANALYSIS OF PRIVATE NETWORK DATA

A Dissertation Presented

by

MICHAEL G. HAY

Approved as to style and content by:

---

Gerome Miklau, Co-chair

---

David Jensen, Co-chair

---

Don Towsley, Member

---

Andrew Papachristos, Member

---

Andrew G. Barto, Department Chair  
Computer Science

## ACKNOWLEDGMENTS

This work would not have been possible without the support and mentorship of David Jensen and Gerome Miklau.

David has been a generous and accommodating advisor. He gave me the latitude to explore the world of research freely and he taught me the methods that made the exploration fruitful. I have found his research methods invaluable in practice, but more importantly, they were foundational to my success, because they made the mysterious craft of research accessible to me. In addition to his teachings, I am grateful for his patience and encouragement. David also oriented and prepared me for a career in research, giving me valuable perspective on the challenges and rewards of such a career.

Gerome introduced me to the topic of data privacy and deserves much credit for guiding the effort that went into this dissertation. By working closely with me, he taught me how to formalize a problem and how to transform insights into substantive contributions. He helped me avoid doubtful thinking and embrace the intrinsic uncertainty of research with optimism. He also taught me the art of scholarly presentation: how to write and speak with precision and clarity, how to structure an argument, and how to embrace the mindset of the intended audience. I have been the grateful recipient of his advice: he is gentle, honest, and brings to every problem a clarity of mind that is enviable.

In addition to Gerome and David, I am grateful for the valuable input of the other members of my committee: Don Towsley contributed theoretical results to our early work and gave insightful comments on the rest; Andrew Papachristos guided me to

relevant sociology literature and supplied a valuable perspective as a potential end user.

The work in this dissertation is a product of the hard work and valuable insights of many people. I am grateful to my coauthors for their contributions: Gerome Miklau, David Jensen, Dan Suciu, Vibhor Rastogi, Chao Li, Don Towsley, Andrew McGregor, Philipp Weis, and Siddharth Srivastava. In addition, I am thankful to Vibhor for many thoughtful discussions and also for his friendship.

I have benefited greatly from interacting with the members, past and present, of the Knowledge Discovery Laboratory and the Database Group. Jen Neville and Andy Fast gave me hope and encouragement that a Ph.D. is within my reach. Brian Gallagher provided valuable friendship. Lisa Friedland entertained countless impromptu requests to act as a sounding board for my ideas, as well as my gripes. Chao Li has been a great collaborator and a memorable travel partner.

The Computer Science Department at UMass Amherst is a special place and I have learned so much as a student here. I am especially grateful to Micah Adler, Yanlei Diao, Neil Immerman, and Andrew McCallum for their instruction and to James Allan, Deb Bergeron, Rachel Lavery, LeeAnne LeClerc, and Sharon Mallory for their support.

Finally, I would not have completed this journey without the love and support of family and friends. Al and Buffy have been incredibly supportive, inviting us to enjoy several wonderful beach vacations, welcoming us into their home for a summer, and providing a safe haven for Carey and the boys when paper deadlines neared. My mom has helped keep a roof over our heads and yummy, organic food in our fridge, and she has been a sweet Nana to our boys and a lifeline to us. My Dad gave me encouragement to stick with it when I needed it most and has been incredibly helpful, especially with our transition to Ithaca. Nancy, Chris, and Clara have provided some memorable excursions, filled with tasty treats and big screen fun. Declan, Jesse,

Tanner, Dylan, Talia, Rosalie and their respective parents have been great friends and make us sad to leave Amherst. My boys motivated me to work hard and brought me joy at home. And Carey, you have been there from the beginning and by my side for every twist of the road. It has been a pleasant journey indeed and you made it so. Thank you.

## ABSTRACT

# ENABLING ACCURATE ANALYSIS OF PRIVATE NETWORK DATA

SEPTEMBER 2010

MICHAEL G. HAY

A.B., DARTMOUTH COLLEGE

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Jerome Miklau and Professor David Jensen

This dissertation addresses the challenge of enabling accurate analysis of network data while ensuring the protection of network participants' privacy. This is an important problem: massive amounts of data are being collected (facebook activity, email correspondence, cell phone records), there is huge interest in analyzing the data, but the data is not being shared due to concerns about privacy. Despite much research in privacy-preserving data analysis, existing technologies fail to provide a solution because they were designed for tables, not networks, and cannot be easily adapted to handle the complexities of network data.

We develop several technologies that advance us toward our goal. First, we develop a framework for assessing the risk of publishing a network that has been “anonymized.” Using this framework, we show that only a small amount of background knowledge about local network structure is needed to re-identify an “anonymous” individual. This motivates our second contribution: an algorithm that transforms the

structure of the network to provably lower re-identification risk. In comparison with other algorithms, we show that our approach more accurately preserves important features of the network topology. Finally, we consider an alternative paradigm, in which the analyst can analyze private data through a carefully controlled query interface. We show that the degree sequence of a network can be accurately estimated under strong guarantees of privacy.

# CONTENTS

	Page
<b>ACKNOWLEDGMENTS</b> .....	<b>iv</b>
<b>ABSTRACT</b> .....	<b>vii</b>
<b>LIST OF TABLES</b> .....	<b>xiii</b>
<b>LIST OF FIGURES</b> .....	<b>xiv</b>
 <b>CHAPTER</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Problem setting of prior work: privacy in tabular data .....	2
1.2 Our problem setting: privacy in network data .....	9
1.3 Overview of contributions .....	16
1.3.1 Assessing the risk of network data publication .....	16
1.3.2 Mitigating risk through network transformation .....	18
1.3.3 Estimating network statistics under strong privacy .....	19
<b>2. BACKGROUND</b> .....	<b>23</b>
2.1 $K$ -anonymity .....	23
2.2 Differential privacy .....	28
2.3 Differentially private query answering .....	31
2.4 Differential privacy for graphs .....	33
2.5 Network analyses and statistics .....	36
<b>3. ASSESSING RE-IDENTIFICATION RISK</b> .....	<b>40</b>
3.1 Modeling the adversary .....	42
3.1.1 Naive anonymization .....	42
3.1.2 Threats .....	43
3.1.3 Anonymity through structural similarity .....	46

3.1.4	Adversary model based on structural signatures .....	47
3.1.5	Alternative adversary models .....	49
3.2	Empirical risk assessment .....	51
3.2.1	Node re-identification .....	53
3.2.2	Edge disclosure .....	55
3.3	Theoretical risk assessment .....	56
3.3.1	Erdős-Rényi graphs .....	57
3.3.2	Power-law graphs .....	63
3.3.3	Discussion .....	63
3.4	Conclusion .....	64
<b>4.</b>	<b>MITIGATING RE-IDENTIFICATION RISK .....</b>	<b>65</b>
4.1	Structural anonymity .....	66
4.2	Graph generalization algorithm .....	69
4.2.1	Graph generalization .....	69
4.2.2	Anonymity of generalized graphs .....	70
4.2.3	Algorithm description .....	73
4.2.4	Capitalizing on limited adversaries .....	77
4.3	Evaluating graph anonymization algorithms .....	79
4.3.1	Compared anonymization algorithms .....	81
4.3.2	Overview of experiments .....	83
4.3.3	Results .....	84
4.3.3.1	Paths .....	84
4.3.3.2	Degree-related measures .....	87
4.3.3.3	Clustering .....	90
4.3.3.4	Runtime .....	91
4.3.3.5	Discussion .....	92
4.3.4	Utility of enhanced graph generalization algorithm .....	93
4.3.5	Assessing edge disclosure in generalized graphs .....	96
4.4	Conclusion .....	98
<b>5.</b>	<b>ACCURATE ESTIMATION OF THE DEGREE SEQUENCE UNDER DIFFERENTIAL PRIVACY .....</b>	<b>100</b>
5.1	Overview of the task and solution .....	104

5.1.1	Task: computing unattributed histograms	104
5.1.2	Solution: inference on queries with constraints	108
5.2	Query strategies	111
5.2.1	Baseline strategy: conventional histogram	111
5.2.2	First strategy: frequencies in rank order	114
5.2.3	Second strategy: frequency of frequencies	117
5.3	Inference	121
5.3.1	Inference for ordering constraints	122
5.3.2	Utility analysis	125
5.4	Inference algorithm	127
5.5	Experiments	131
5.5.1	Utility	132
5.5.2	Utility comparison: $\bar{\mathbf{S}}$ vs. $\bar{\mathbf{F}}$	138
5.5.3	Scalability of inference	138
5.6	Conclusion	139
5.7	Proofs	141
5.7.1	Proof of Theorem 5.1	141
5.7.2	Proof of Theorem 5.2	145
5.7.3	Proof of Theorem 5.3	149
<b>6.</b>	<b>ESTIMATING OTHER STATISTICS UNDER STRONG PRIVACY</b>	<b>155</b>
<b>7.</b>	<b>RELATED WORK</b>	<b>161</b>
7.1	Attacks	161
7.2	Network anonymization algorithms	162
7.2.1	Directed alteration	163
7.2.2	Generalization/clustering	164
7.2.3	Random alteration	165
7.3	Other privacy issues arising with network data	166
7.4	Related work in differential privacy	167
<b>8.</b>	<b>CONCLUSION</b>	<b>169</b>
8.1	Review of contributions	169
8.2	Future directions	172

**BIBLIOGRAPHY ..... 175**

## LIST OF TABLES

Table	Page	
2.1	Example to illustrate $k$ -anonymity. (a) a table $T$ of medical records; this table is 1-anonymous. (b) table $T$ after removing the <i>Name</i> attribute and coarsening attributes <i>Birth</i> , <i>Sex</i> , and <i>Zip</i> ; this table is 3-anonymous. . . . .	24
3.1	Descriptive statistics for the real and synthetic graphs studied. . . . .	51
4.1	On <b>NetTrace</b> , a comparison of runtimes (seconds). . . . .	92
4.2	A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and GraphGen at $k = 10$ . Numbers are normalized scores where less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than GraphGen. . . . .	95
4.3	A comparison of utility of GraphGen( $\mathcal{H}_2$ ) and GraphGen at $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_2$ ) is more accurate than GraphGen. . . . .	95
4.4	A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and LT at $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than LT. (A dash indicates that LT perfectly matched the original, so the normalized score is undefined. A 0* indicates that both LT and GraphGen( $\mathcal{H}_1$ ) perfectly matched the original.) . . . . .	96
5.1	Notational conventions for query sequences. . . . .	112
5.2	Examples of private outputs $\tilde{o} = \tilde{\mathbf{O}}(I)$ and their closest ordered sequence $\bar{o}$ . . . . .	124

## LIST OF FIGURES

Figure	Page
1.1	Traditional problem setting: each individual contributes a private record, and analysts wish to compute aggregate queries over the collection of records. Computational approaches to protecting privacy correspond to interventions in the flow of information: (A) input perturbation; (B) transformed data release; (C) query auditing and query answer perturbation; and (D) access control. . . . . 3
1.2	Our problem setting: individuals are connected by relations, forming a network. We explore three approaches to protecting privacy: simple anonymization, in which identifiers are replaced and attributes are coarsened or suppressed (a process we call <i>naive anonymization</i> ); transformed data release in which the network structure is altered; and query answer perturbation, in which noise is added to query answers. . . . . 10
3.1	A social network represented as a graph (left), the naive anonymization (center), and the anonymization mapping (right). . . . . 43
3.2	(a) A sample graph, (b) external information consisting of structural signatures $\mathcal{H}_0, \mathcal{H}_1$ and $\mathcal{H}_2$ computed for each individual in the graph, (c) the equivalence classes of nodes implied by the structural signatures. For the sample data, $\equiv_{\mathcal{H}_2}$ , corresponds to automorphic equivalence, $\equiv_A$ . . . . . 46
3.3	The relationship between candidate set size and structural signature knowledge $\mathcal{H}_i$ for $i = 1..4$ for four real graphs and three synthetic graphs. For each $\mathcal{H}_i$ , the bars show the percentage of nodes whose candidate sets have sizes in the following buckets: [1] (black), [2, 4], [5, 10], [11, 20], [21, $\infty$ ] (white). . . . . 53
3.4	The inferred edge probabilities resulting from attempted re-identification using structural signatures $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ . . . . . 55

3.5	For $\mathcal{H}_2$ and $\mathcal{H}_3$ the number of uniquely re-identified individuals in a classical random graph goes from zero to 100% quickly when there is sufficient edge density. But regardless of the density, the number of nodes with a unique degree is close to zero, showing that $\mathcal{H}_1$ is insufficient for unique re-identification. . . . .	60
4.1	The effect of anonymization on three graph measures related to paths. The results for three algorithms are compared, with varying privacy parameter $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region. . . . .	85
4.2	The effect of anonymization on four measures related to the degree distribution. Again, the results for three algorithms are compared, with varying privacy parameter $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region. . . . .	88
4.3	The effect of anonymization on clustering coefficient. . . . .	91
4.4	Risk of edge disclosure in generalized graphs across different datasets and settings of $k$ . . . . .	97
5.1	(a) Example table $R$ on which we compute histograms; (b) A conventional histogram on attribute <i>Condition</i> , whose domain is $dom = \{\text{Asthma, Cancer, Flu, Healthy}\}$ ; (c) An unattributed histogram on <i>Condition</i> ; (d) Definitions and sample values for alternative query sequences: $\mathbf{C}$ computes a conventional histogram, $\mathbf{S}$ returns the frequencies of the histogram in rank order, $\mathbf{F}$ computes a cumulative histogram on the frequencies. . . . .	105
5.2	(a) Example graph $G$ for which we compute degree sequences; (b) The edges of $G$ represented as a relation $R(\text{Src}, \text{Dest})$ ; (c) An attributed degree sequence; (d) An unattributed degree sequence. . . . .	107
5.3	(a) Example of how inference reduces the error: $\bar{s}$ is more accurate than $\tilde{s}$ ; (b) Error bars show expected error for $\bar{s}$ at each position, showing lower error in the middle of the uniform subsequence. 126	
5.4	Complementary CDFs of $\mathbf{S}(I)$ , $\tilde{s}$ and $\bar{s}$ (top). Bias of $\bar{\mathbf{S}}$ (bottom). . . . .	134

5.5	Accuracy (KS distance) across varying $\epsilon$ .	135
5.6	Accuracy (Mallows distance with $p = 2$ ) across varying $\epsilon$ .	135
5.7	Size vs. Accuracy for fixed $\epsilon = 0.01$ .	136
5.8	Accuracy of estimating power-law model using $\tilde{\mathbf{S}}, \bar{\mathbf{S}}$ .	137
5.9	On <b>Flickr</b> , a comparison of $\bar{\mathbf{S}}$ and $\bar{\mathbf{F}}$ .	138
5.10	Runtime of Algorithm 3 on real (left) and larger synthetic datasets (right).	139
6.1	Example graphs illustrating the high sensitivity of triangle and path queries.	157

# CHAPTER 1

## INTRODUCTION

We participate daily in a variety of networks. For example, through email correspondence, facebook friends, Internet activity, scholarly publications, and face-to-face communication, we interact with large, complex networks of peers. Due to technological advances, increasing amounts of our social interactions are being recorded, resulting in troves of detailed data about the structure of our networked society. Some believe the availability of this data has the opportunity to transform social science, much the way data transformed the biological and physical sciences, leading to a data-driven “computational social science” [68]. Already, the analysis of network data has advanced our understanding of diverse phenomena such as the robustness of the Internet, the spread of HIV, and the causes of financial fraud.

However, to achieve this vision, we must address concerns about privacy. The collected data can be highly sensitive, revealing the intimate details of our daily lives. Concerns about privacy make data managers reluctant to share it with the scientists best equipped to analyze it. At present, computational social science is happening, but only at the institutions that collected the data or by a select few who have negotiated access to the data. For science to flourish, we must alleviate the privacy concerns that prevent data sharing.

The goal of this dissertation is to develop algorithms to enable accurate analysis of network data while preventing the disclosure of sensitive information.

Despite a long history of research in privacy-preserving data publishing and privacy-preserving data analysis, existing techniques are not well suited for network data.

Most prior work assumes the private data can be represented as a table of records, where an individual’s private information is encapsulated in a single record. Network data does not fit this data model, and poses new challenges for protecting privacy. Our investigation uses the prior work in the tabular data setting as foundation to develop new approaches suited for the complexities of network data.

## 1.1 Problem setting of prior work: privacy in tabular data

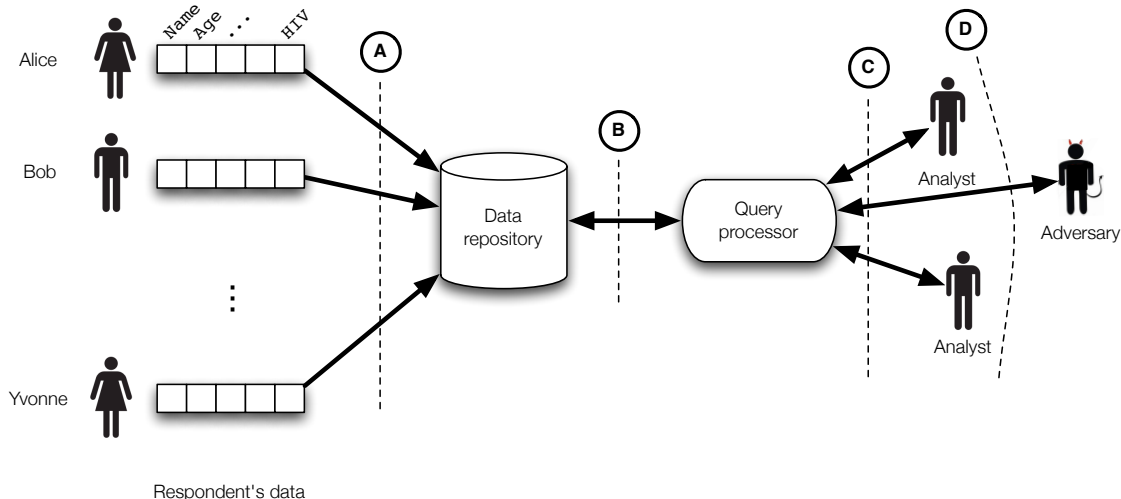
We introduce the main concepts in the traditional setting where the private data consists of a table of records. This is the setting for prior work. Many of the concepts introduced here carry over to the network setting and serve as a strong foundation on which to develop techniques for network data.

As a motivating example, consider the data collected and disseminated by the Demographic and Health Surveys (DHS) program<sup>1</sup> to assess HIV prevalence in developing countries throughout the world. The data is collected through surveys, in which respondents are asked for demographic and health information. Part of the survey includes voluntary HIV testing. Once collected, this data can be used by researchers to study, among other things, the factors influencing HIV prevalence.

The collected data contains both *sensitive* and *identifying* information. Identifying information can be used to associate the data to a real-world entity. For example, attributes such as gender, age, and geographic location can be identifying. Sensitive information is any information whose unauthorized disclosure could cause harm. For example, attributes that indicate HIV status, knowledge/practice of birth control, etc. could be considered sensitive.

---

<sup>1</sup><http://www.measuredhs.com/>



**Figure 1.1.** Traditional problem setting: each individual contributes a private record, and analysts wish to compute aggregate queries over the collection of records. Computational approaches to protecting privacy correspond to interventions in the flow of information: (A) input perturbation; (B) transformed data release; (C) query auditing and query answer perturbation; and (D) access control.

The challenge is to allow analysts to study the data in a way that prevents disclosure of sensitive information of survey respondents. There are several computational approaches to protecting privacy, illustrated in Figure 1.1 and described next.

Figure 1.1 illustrates the flow of information from survey respondents to analysts. Individual survey respondents (information providers) contribute their private data. In this setting, each respondent’s data can be encapsulated in a single record. The private records are collected into a database, which is controlled by the data manager. Analysts (information consumers) perform computations on it, interacting with the data through some kind of query processor.

We distinguish between two kinds of information consumers: *analysts* and *adversaries*. The analyst wants to study the population, by measuring aggregate statistics, fitting statistical models, etc. For example: an analyst might want to know what age groups have highest prevalence of HIV. The adversary, on the other hand, wants to learn facts about specific individuals. For example, an adversary may ask whether

Alice tested positive for HIV. There are two competing goals: to support the analyst and hinder the adversary.

To protect privacy we must control the flow of information from providers to consumers. Essentially, this requires intervening somewhere along the information pipeline. In Figure 1.1, there are labeled dashed lines at several points of intervention. We briefly describe some high-level strategies for intervention. Almost all prior work represents an instantiation of one of these strategies.

**Access control** A tempting, but ultimately inadequate solution to the problem is to try to protect privacy through *access control*. The goal of access control is to manage a resource so that it is accessible only to authorized users. In this setting, we may consider using access control to ensure that only “trusted” analysts can access the data and that access is limited to the parts of the data relevant for their analysis. In Figure 1.1, it corresponds to an intervention at the point marked *D*.

There are three challenges with using access control. First, it requires a mechanism for establishing trust, which may be difficult to design and costly to execute. In other words, it may be difficult to discern trustworthy analysts from malicious adversaries. Second, while there are technologies for supporting fine-grained access control policies, in this setting, the computations are aggregate statistics, which means each analyst requires access to *all of the records*. Access control would be a fairly blunt instrument, essentially providing access to either all or none of the data. Finally access control is not a complete solution because in many settings, the goal of analysis is to identify interesting trends and disseminate that knowledge broadly, into the public domain. So even if access control is used, there still must be a mechanism for safely releasing the analyst’s findings. For instance, at one data center that uses access control to restrict access to sensitive data, the findings of an analyst can be published only after a satisfactory *manual* review by the data custodian [1].

**Input perturbation** With *input perturbation*, each individual stochastically alters his/her data before revealing it to the data manager. In Figure 1.1, input perturbation corresponds to intervening at the point marked *A*. The classical example of this approach is Warner’s randomized response, in which the survey respondent flips a coin and answers truthfully or randomly based on the outcome [120]. The randomized answers are collected by the data manager and then published. While the randomness hides an individual response, thereby ensuring privacy, it remains possible, using the machinery of statistical inference, to derive accurate estimates of aggregate measures. This strategy is especially appropriate when even the data manager is untrusted. When the data manager is trusted, it is possible to achieve the same privacy (with respect to untrusted analysts) but higher accuracy results by intervening further “downstream,” with approaches such as transformed data release and query answer perturbation.

While input perturbation may be used in the tabular data setting, it seems poorly suited in the network data setting, where designing an effective perturbation mechanism seems challenging and often the data is already collected in a centralized repository, in which case input perturbation is not applicable.

**Transformed data release** An alternative strategy is *transformed data release*, where the individual records are collected by a trusted party, altered to protect privacy, and then released to the public. This approach is indicated by *B* in Figure 1.1. Randomization is one option for protecting privacy. However, compared with input perturbation, this setting offers greater flexibility in the design of the randomization operator. Rather than randomize each record independently, it is possible to design a randomizer that operates on the entire dataset and achieves greater accuracy (cf. [17, 106]).

In addition to randomization, transformed data release allows for an alternative strategy to protect privacy: *anonymization*. Before publishing the data, identifying

attributes (name, age, gender, location) are removed or sufficiently coarsened so that each an individual record cannot be distinguished by these features. This provides privacy through anonymity: the adversary can no longer associate an individual to a particular record – each person is “hidden in a crowd.” The analyst can still compute aggregations over the record groups.

This kind of privacy protection has been formalized as  $k$ -anonymity [110, 111, 116]. The objective of  $k$ -anonymity is to prevent *linkage attacks*, when an adversary joins the published “anonymized” data with some other external source based on some common attributes. For instance, a privacy researcher successfully linked Massachusetts voter registration records with an “anonymized” table of medical records published by the Massachusetts Group Insurance Commission based on birthdate, sex, and zip code, and was able to re-identify the medical record of the governor of Massachusetts [115].

$K$ -anonymity prevents linkage attacks by essentially requiring that potentially identifying attributes (such as birthdate, zip code, etc.) be coarsened so that each record becomes indistinguishable from at least  $k - 1$  other records. Research on  $k$ -anonymity has been substantial, and many lessons have been learned about benefits and potential risks of employing such fairly simple privacy protection strategies.

**Query auditing** Common to the above approaches is that the data is altered to protect privacy and then released. An alternative strategy is to not release transformed data but instead allow the analysts to query the private data through a controlled interface. The first of two approaches based on this strategy is *query auditing*. In Figure 1.1, query auditing corresponds to intervening at the point marked  $C$ .

In query auditing, the analysts’ queries are either answered exactly or denied if there is a chance that revealing the answer could lead to disclosure. The data manager is responsible for determining whether an answer is safe to release. The decision depends on what queries have been answered and assumptions about the

adversary’s *a priori* knowledge. Unfortunately, prior work suggests that there are substantial technical hurdles to implementing an effective strategy. In some cases, the auditing decision is computationally hard [64]. It is also subtle: for instance, a denial may itself leak private information [60]. But perhaps an even bigger issue is that protecting privacy may require many query denials, limiting its practical utility.

We describe a second strategy, query answer perturbation, that has a similar approach but seems to offer more flexibility. Like query auditing, the data is never released. The analyst interacts with the data through some kind of query interface and the data manager is responsible for query processing. However, rather than return exact answers, it returns approximate answers, using random noise to control the accuracy of the approximation.

**Query answer perturbation** In query answer perturbation, the data manager computes the true answer to the query, randomly perturbs it, and returns the perturbed answer to the analyst. Just like query auditing, it corresponds to intervening at the point marked *C* in Figure 1.1. A common assumption in this setting is that the answer to a query is a number, and therefore it makes sense to think about adding (appropriately scaled) random noise to a query answer. (However, there are alternative perturbation strategies when the queries have non-numerical answers, cf. [91].) Intuitively noise creates uncertainty about the exact query answer, but it is not obvious how to appropriately calibrate the noise to ensure privacy. Fortunately, recent work in  $\epsilon$ -differential privacy provides mechanisms for appropriately calibrating noise to ensure a very strong guarantee of privacy.

Differential privacy formally bounds the amount that an adversary can infer about a record given the noisy answer, even in the extreme case where the adversary has complete knowledge of the remaining records. If a query answering mechanism satisfies  $\epsilon$ -differential privacy, then the adversary’s posterior belief (about a person’s record) can be at most a factor of  $\exp(\epsilon)$  larger than his prior belief. From an indi-

vidual’s perspective differential privacy offers the following assurance: whether they “opt-in” and participate in the survey, or “opt-out” and exclude their data, the result will be virtually the same. The value of parameter  $\epsilon$  is selected by the data manager, with lower values conferring stronger privacy protection.

Differential privacy can be achieved by adding random noise to the query answer. The scale of the noise increases inversely with  $\epsilon$ . It also increases with the query’s *sensitivity*, informally the amount that a single record can influence the query answer. In the tabular data setting, many common analyses have low sensitivity, so the amount of noise is small and accurate answers are possible. If viewed as database queries, many analyses can be expressed using operations such as selection, projection, grouping, and aggregation, all of which are low sensitivity.

Of course, there are limits on how many queries can be answered accurately. If too many queries are answered too accurately, then disclosures will occur. In this sense, query answer perturbation is similar to auditing: at some point, the data manager must effectively refuse to answer queries.

However, query answer perturbation offers more flexibility. Differential privacy has nice composition properties: if query  $q_1$  is answered with  $\epsilon_1$ -differential privacy and  $q_2$  is answered with  $\epsilon_2$ -differential privacy, answering *both* queries satisfies  $(\epsilon_1 + \epsilon_2)$ -differential privacy. (Intuitively, if the answer to  $q_i$  increases the informed adversary’s posterior belief by at most a factor of  $\exp(\epsilon_i)$  over his (arbitrary) prior belief, then revealing both answers can increase belief by at most of a factor of  $\exp(\epsilon_1 + \epsilon_2)$ .) Therefore, the data manager has the flexibility of trading off lower accuracy in query answers (smaller  $\epsilon$ ) in exchange for being able to answer more queries. In fact, in some cases, answering both  $q_1$  and  $q_2$  requires only  $\max\{\epsilon_1, \epsilon_2\}$ . In addition, it is possible to restructure a workload of queries to get even more accurate answers for a fixed  $\epsilon$ . That said, optimal mechanisms that can answer a workload of queries with maximal accuracy do not yet exist. This is an important issue that currently limits

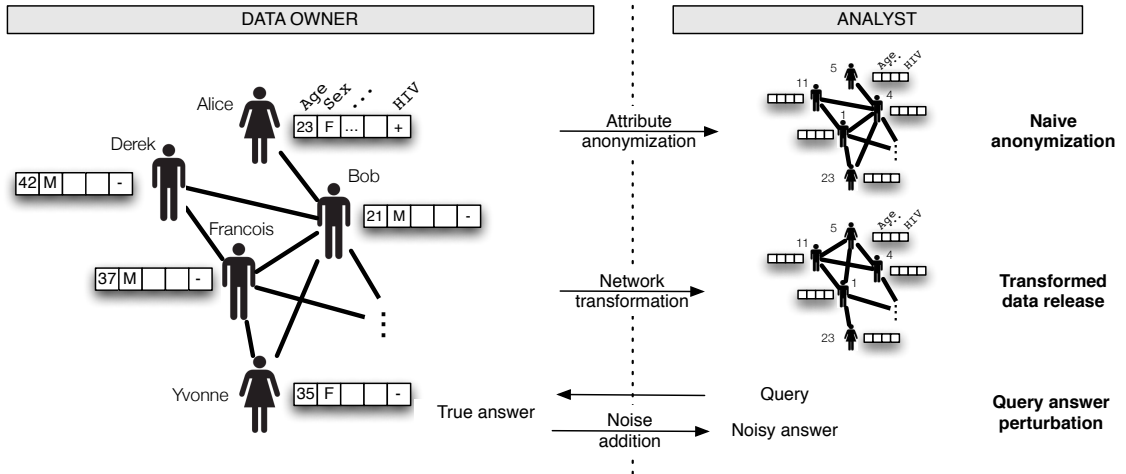
the practical applicability of differential privacy. In work outside the scope of this thesis, we have developed accurate (but non-optimal) mechanisms for answering range queries [55] and shown that for a given workload, the problem of finding the optimal mechanism can be formulated as a semi-definite program with rank constraints [76].

In summary, there are several possible ways to protect privacy in the tabular data setting. Most prior techniques can be viewed as an instance of either input perturbation, transformed data release, query auditing, query answer perturbation, or access control. For tables, it is possible to achieve strong privacy and yet also support accurate data analysis. This is because with tabular data, most analyses can be expressed as fairly simple aggregations of the data that have low sensitivity. This makes it possible to preserve aggregate properties, at least approximately, while at the same time ensuring that individual records are not disclosed.

## **1.2 Our problem setting: privacy in network data**

Network data encodes information about entities and the relationships between them. The existence of relationships profoundly alters many aspects of the privacy protection problem. The relationships are often sensitive and must be protected. In addition, the relationships are also an object of study: many analyses are concerned with understanding the topology of the network. Therefore approaches designed for tabular data fail to provide adequate solutions for network data, and we must develop new approaches to deal with the additional complexities introduced by relationships.

As a motivating example, consider the network data collected by Klov Dahl et al. [65] to study HIV transmission. The data was collected to study a population that is at high risk of contracting HIV: a population of prostitutes, injecting drug users, and their associates in a moderate-size city. In contrast to the HIV survey data collected by measureDHS, in this data, the individuals are connected by relationships



**Figure 1.2.** Our problem setting: individuals are connected by relations, forming a network. We explore three approaches to protecting privacy: simple anonymization, in which identifiers are replaced and attributes are coarsened or suppressed (a process we call *naive anonymization*); transformed data release in which the network structure is altered; and query answer perturbation, in which noise is added to query answers.

that represent pathways for disease transmission (intercourse or needle sharing). This allows for new kinds of analyses on the role that network structure can play in disease transmission. And it also raises new privacy concerns: the relationships are sensitive and the participants in the network have an expectation that this information will be kept private.

Network data can be naturally modeled as a graph where nodes represent entities and edges the relationships among them. In many settings, the entities are people, but they can also be other things, such as hosts in a computer network. There may be attributes on nodes as well as edges. An example network is illustrated in Figure 1.2.

The relationships in network data encode additional information about the entities which may be highly sensitive. Naturally, the existence of an edge may be sensitive (e.g., in the data collected by Klovdahl et al. [65], an edge reveals a sexual or shared needle relation). Other aspects of connections may be sensitive as well. For example, the degree of a vertex may be sensitive: academics in a scholarly collaboration net-

work may wish to hide their low degree, while participants in a network of romantic contacts may wish to hide their high degree. Even if the entities in a network are not individuals, network data may still be sensitive. For example, detailed topological information about the power grid may reveal vulnerabilities to potential terrorists, or records of information flow between host machines in a computer network may reveal applications running on those hosts or facts about host operators. Finally, if there are attributes on the entities, then just as in the tabular data setting, these may be also sensitive (e.g., HIV status).

The analyses of network data are more complex and varied than the analyses of tabular data. Many analyses focus solely on measuring the topology of the network. This includes computing various network statistics, such as degree distribution, diameter, and transitivity. It also includes searching for interesting structures – for instance, identifying frequently occurring subgraphs (motif analysis), finding highly influential nodes (network centrality), or recognizing modular structure (community discovery). Other analyses focus on correlations between topology and attributes, such as the study of homophily (the tendency for associations to form among similar individuals). Finally, a number of analyses are concerned with network dynamics, either dynamic processes operating over the network (navigation, diffusion) or the dynamics of the network’s formation. This incomplete summary illustrates the diversity of analyses that are performed on network data. It also illustrates the complexity: if viewed as database queries, many analyses require joins, in addition to the selection operations commonly used in tabular data.

In summary, the existence of relationships between individuals makes the problem considerably different from the traditional tabular data setting. There are differences in the structure of the data, in what makes the data sensitive, and in how it is analyzed. These changes may require novel strategies for protecting privacy. Building on the foundation of work developed in the tabular setting, this thesis explores how to

protect privacy in the network data setting. We explore three high level approaches to simultaneously protecting privacy and supporting accurate analysis. These approaches are illustrated in Figure 1.2.

The first part of this thesis explores whether attribute anonymization might also provide adequate protection for network data. As described earlier, attribute anonymization is a common strategy for protecting privacy in the tabular data setting. External identifiers such as name, social security number, are removed and replaced with synthetic identifiers. In addition, attributes that are potentially identifying (quasi-identifiers) are coarsened, perturbed, or suppressed.

Applied to a network, anonymization would require replacing node identifiers, such as names and IP addresses, with synthetic ones. In addition, if nodes possessed other identifying attributes, these could be coarsened, etc. using established techniques from tabular data anonymization. Such precautions would prevent attacks based on attribute knowledge. The network structure, however, would remain intact. We wonder whether the presence of relationships between the entities create new threats to privacy. Specifically, we explore the following question:

*Can the patterns of connections around an individual node act as an identifier, making the node vulnerable to a re-identification attack?*

To explore this question, we consider factors such as how much an adversary might know *a priori* about the graph structure surrounding a target; how much external information is necessary to successfully re-identify nodes; and what might be the consequences of re-identification. Our aim is to articulate the threats to privacy posed by the release of anonymized network data.

As we will show, there are settings where attribute anonymization provides insufficient protection. Thus, we are motivated to consider more complex techniques to protect privacy. In the second part of the thesis, we investigate the following question:

*If graph structure is in fact identifying, how can we alter the graph so that nodes are resistant to re-identification? What is the impact of the alterations on network topology?*

In addressing this question, it is necessary to consider a number of different issues. With tabular data, there are several operations, such as attribute coarsening and stochastic perturbation, that can be used to lower the risk of re-identification. With network data, it is not immediately obvious what kind of alterations should be applied to create anonymity. Nor is it clear how much alteration is necessary, or how to quantify the privacy benefit of alteration. Can privacy notions such as  $k$ -anonymity be adapted to the graph setting? How do the lessons learned in tabular data anonymization apply? Ultimately, with this approach, we will publish a graph that differs from the original. It will be important to understand the impact of the alterations on analyses of topology. How do we meaningfully compare the transformed graph with the original? How should the analyst account for the fact that the graph was altered?

We also explore the alternative paradigm of query answer perturbation. This is a substantially different approach than that considered in the earlier parts of this thesis, and there are important tradeoffs to consider. Compared with transformed data release, the obvious disadvantage of query answer perturbation is that a graph is never published. Instead, the analyst only receives answers to some statistics and there are limits as to how many queries can be accurately answered. However, the advantage of query answer perturbation is the potential to achieve much higher accuracy. With transformed data release, the transformations inevitably alter some

properties of the graph, and some analyses will be greatly distorted. With query answer perturbation, the mechanisms are tailored to the specific statistics of interest, and this creates an opportunity to obtain highly accurate results. In addition, we explore an approach based on differential privacy, which is a *much* stronger form of privacy protection than those based on anonymity.

The final component of this dissertation investigates the following question:

*Can we accurately measure network statistics under rigorous privacy standards, such as differential privacy?*

Our investigation takes advantage of the existing work in differential privacy. A number of useful techniques have already been developed for answering queries under differential privacy, and their application is not limited to queries over tabular data.

In thinking about applying differential privacy to statistics on network data, some interesting questions emerge. While differential privacy was originally defined for the tabular setting, it is in principle applicable to any data that can be (conceptually) broken up into “units” of private information. For network data, what is the appropriate “unit” of private information? For instance, is it an edge, a set of edges, or a node and its entire neighborhood? This is an important question whose answer determines what the technical condition of differential privacy actually means for the privacy of the individual. It also has a profound impact on the accuracy with which network statistics can be estimated. With tabular data, we saw that an individual’s data has a limited impact on common analyses—i.e., the sensitivity of common analyses is low. What is the sensitivity of common network analyses? Are there limits on what can be accurately computed under such a strong notion as differential privacy, and if so, what is an appropriate remedy?

As described above, we explore two of the high-level strategies for privacy protection, transformed data release and query answer perturbation. We chose not to consider query auditing and access control due to the limitations described earlier. We also chose not to consider input perturbation. In many settings, the data is already under the control of a trusted data manager. Network data is often collected by the data manager as a byproduct of providing some service (e.g., online social networks, email correspondence, phone call records, Internet traffic). In these settings, input perturbation is not applicable. That said, there may still be some settings where it applies, including for instance the previously mentioned study of sexual relationships by Klovdahl et al. [65]. This is a similar context as the one that motivated the original input perturbation mechanism, Warner’s randomized response. However, designing an appropriate input perturbation mechanism seems especially challenging in the context of network data, as the sensitive data is shared between individuals. We leave this as a consideration for future work.

While, in practice, network data may also have attribute information associated with nodes, the focus of our work is on network structure—that is, protecting sensitive structural features and supporting analyses of network topology. There are several reasons for this. The threats to privacy of sharing attribute information about individuals have been well-studied in the tabular data setting, and many techniques have been developed to reduce the threat. What makes the current problem challenging is the presence of relationships between the individuals. In addition, many network analyses are concerned exclusively with topology, so developing solutions that support the accurate analysis of network topology is an important goal by itself. Finally, enabling accurate analysis of structure is a necessary component of any solution that supports the combined analysis of attributes and structure. We view our work as the first steps along the critical path to a comprehensive solution. Future work will ex-

plore ways to support analyses that involve a combination of structural and attribute information.

### 1.3 Overview of contributions

Toward the goal of enabling accurate network analysis while protecting privacy, we make three main contributions:

- We develop a framework for evaluating the risk of publishing a network after removing obvious identifiers. Using this framework, we demonstrate that there is substantial risk: adversaries can use knowledge of graph structure to re-identify entities in the published network.
- To mitigate the risk, we design an algorithm that transforms the network prior to publication. The transformed network resists re-identification attacks and at the same time the transformations preserve important topological features of the original network.
- Using the paradigm of query answer perturbation, we explore whether it is possible to accurately compute network statistics under strong privacy protections. For some statistics, such as clustering coefficient, we identify some limitations. However, we show that the degree sequence can be accurately estimated under differential privacy.

These contributions are described in more detail below.

#### 1.3.1 Assessing the risk of network data publication

In Chapter 3, we study the threats to privacy of publishing a network after anonymizing node identifiers and removing identifying attributes. We refer to this strategy of privacy protection as *naive anonymization*. We consider how an adversary may use partial knowledge of the network structure surrounding a set of target entities

to re-identify them. Once re-identified, additional properties about the targets may be revealed in the published data. For instance, the adversary may learn that two targets are connected. We study both the general threat of *re-identification* as well as the specific threat of *edge disclosure*.

The risk of naive anonymization depends on how much the adversary knows and on how easily nodes can be distinguished by their network structure. We study the relationship between knowledge, structure, and risk and make the following contributions.

We present a framework for measuring the risk of naive anonymization. It has several innovative features. It includes a flexible model of adversary knowledge, allowing a data manager to assess risk with respect to a range of adversary capabilities. It is efficient to compute, making it applicable for large graphs. It can be used to assess both re-identification risk and edge disclosure. We believe this framework is a valuable resource for gaining insight into the relationship between adversary knowledge, graph structure, and re-identification risk. Furthermore, it is a practical tool that a data manager can use to assess risk prior to publication.

We use our framework to assess the risk of several real networks. We find diversity in risk: some networks are naturally more resistant to structural re-identification than others. Nevertheless, on some real-world networks, the risk is high, especially if an adversary has knowledge beyond a target entity’s immediate neighborhood. The significance of our study is that it demonstrates that there can be considerable risk in naive anonymization. Our findings establish a need for more sophisticated techniques.

Finally, we use theoretical analysis to gain insight into how properties such as density and degree distribution affect a network’s vulnerability to adversarial attack. We prove that in large Erdős-Rényi random graphs, density determines re-identification risk, with sparse graphs having low risk and dense graphs having very high risk. We also prove that power-law random graphs, despite their heavy-tailed degree distribu-

tions, have low risk in the limit of large graphs. We believe our theoretical analysis gives insight into how graph structure affects re-identification, which can inform the design of more sophisticated anonymization techniques.

This work first appeared as a technical report in 2007 [51], then in the proceedings of the International Conference on Very Large Data Bases, 2008 [54], and has been accepted for publication in the International Journal on Very Large Data Bases.

### 1.3.2 Mitigating risk through network transformation

When simple strategies such as removing obvious personally identifiable information fail to protect privacy, the data manager must consider alternative strategies for releasing network data. In Chapter 4, we consider the problem of how to transform a network to prevent entity re-identification. An effective transformation strategy will ensure that an adversary cannot re-identify entities in the published network, but at the same time preserve many important topological features of the network. We make the following contributions.

We introduce a new privacy definition that characterizes what it means for a transformed network to be resistant to re-identification attacks. Our definition is an adaptation of  $k$ -anonymity that is called *graph  $k$ -anonymity*. If a published graph is graph  $k$ -anonymous, it means that an adversary cannot successfully re-identify nodes: his confidence in the identity of a target will be at most  $1/k$ . The parameter  $k$  is set by the data manager, with higher  $k$  resulting in greater protection, though achieving it will also require more alterations to the graph. Our definition is configurable in the sense that privacy can be defined with respect to a particular class of adversary capability. It is also general: other recently proposed graph privacy definitions can be viewed as instantiations of our definition, with specific choices of adversary.

We introduce an algorithm for transforming a graph to achieve both privacy and accurate preservation of topology. The basic idea is to cluster the nodes of the graph

into groups and describe the network topology in terms of the groups. As we prove formally, it satisfies graph  $k$ -anonymity by enforcing that the minimum group size is at least  $k$ . To maximize the utility of the transformed graph, the algorithm searches for a clustering that preserves as much of the topological structure as possible, subject to the privacy condition.

We empirically evaluate the utility of the transformation algorithm and compare it with other graph anonymization techniques that have been recently proposed. On a number of graphs, both real and synthetic, we measure how transformation distorts common graph properties such as degree distribution, path lengths, and clustering. Our experiments reveal a number of findings. At small  $k$ , many graph properties are approximately preserved; as  $k$  increases, we observe a tendency toward random graphs. Compared with other techniques, our approach performs comparably and in some cases, preserves more utility. In particular, because its transformations are guided by the topology of the graph, it tends to do a much better job of preserving distinct “engineered” features. For example, on the **HOT** graph, which is an idealized model of the Internet, the graph’s low degree correlation is preserved under our technique.

This work appeared (together with the material in the previous chapter) in the proceedings of the International Conference on Very Large Data Bases, 2008 [54], and has been accepted for publication in the International Journal on Very Large Data Bases.

### 1.3.3 Estimating network statistics under strong privacy

The last component of this thesis considers an alternative paradigm—query answer perturbation—and investigates whether we can accurately measure common network statistics under strong privacy guarantees, such as differential privacy. Because of the shift in paradigm, the problems and techniques differ considerably from those

presented in Chapters 3 & 4. Towards the goal of accurate statistics and strong privacy, we make the following contributions.

A small but vital contribution is to determine how adapt differential privacy to graph structured data. In Chapter 2, after we present the original definition and its semantic interpretations, we present several alternative adaptations and describe the implications for the privacy-utility tradeoff inherent in them. One variant we introduce is called  $k$ -edge  $\epsilon$ -differential privacy. It prevents an adversary from learning not only about individual edges, but any set of edges of size up to  $k$ . For nodes of low degree (less than  $k$ ), this captures the “opt-in/opt-out” semantics of differential privacy. For nodes with large degree, some aggregate features about their neighborhood may be leaked—this is an inevitable compromise if we also want to enable accurate network analysis. The data manager chooses  $k$  as well as  $\epsilon$ .

Our main contribution is a differentially private algorithm for estimating the degree sequence of a graph. We choose to focus on the degree sequence because it is one of the most widely studied properties, and it has profound influence on a network’s structure and function [7, 16, 87, 99]. While existing differentially private techniques can be used to obtain a noisy estimate of the degree sequence, the accuracy is poor. In Chapter 5, we present an innovative algorithm that achieves privacy, accuracy, and scalability: The algorithm satisfies  $k$ -edge  $\epsilon$ -differential privacy. It produces an estimate of the degree sequence that is provably accurate, and shown empirically to be orders of magnitude more accurate than existing techniques. In addition, its linear time complexity allows it to scale to the massive graphs studied today. We believe this work is one of the first concrete positive results of applying differential privacy to network data.

The algorithm has broader applicability than estimating degree sequences. As part of this work, we define a new kind of histogram, called an *unattributed histogram*, and show how our technique can be used to estimate any unattributed histogram. While

less informative than a conventional histogram—because it hides the association between bin and frequency—an unattributed histogram is significant because it can be estimated much more accurately under differential privacy. And although it is less informative, there are some applications where the discarded information is irrelevant and so an unattributed histogram is sufficiently detailed. Estimating the degree sequence of a graph is one such application, and we describe others in Chapter 5.

To achieve its high accuracy, the algorithm uses an innovative strategy that we believe has applications beyond this particular instantiation. The strategy uses an existing differentially private mechanism to obtain a noisy answer and then uses statistical inference to reduce error. The inference stage exploits natural constraints that hold on query answers, yet are often violated by the random noise that was added to ensure privacy. For the degree sequence query, the inference stage produces a massive reduction in error. Our results demonstrate that the differentially private mechanism adds more noise than is strictly necessary, and inference is able to filter out much of the “extra” noise without weakening the privacy protection. We believe inference may be a computationally efficient way to effectively sample from a more complex, correlated noise distribution that is tailored to the particular query and input. We have explored its application to other kinds of queries: in work that lies outside the scope of this thesis, we show that it can be an effective component of a mechanism to support histograms [55] and arbitrary workloads of linear queries [76].

Given these positive results for the degree distribution, it is natural to wonder what other network properties can be computed under differential privacy. An additional contribution of this thesis is a review of known results and a discussion of limitations (Chapter 6). Unfortunately, we find that there are limits as to what can be accurately computed under differential privacy. For instance, we prove that clustering coefficient cannot be accurately estimated under differential privacy. One reason this query is challenging is that differential privacy protects against even a nearly

omniscient adversary, and for such adversaries, even an approximate answer can leak private information. We show that if the adversary is less informed, knowing only a small subgraph of the entire network, then clustering coefficient (and several other statistics) can be answered with relatively little noise while still ensuring rigorous privacy protection.

The work on estimating degree sequences presented in Chapter 5 combines results from two conference publications, the first appearing in the International Conference on Data Mining, 2009 [52] and the second to appear in International Conference on Very Large Data Bases, 2010 [55]. The work on computing clustering coefficient and protecting against weaker adversaries was done in collaboration with Rastogi and Suciú and appeared in the Symposium on Principles of Database Systems, 2009 [105].

We believe the contributions described above help advance our goal of enabling accurate network data analysis while simultaneously protecting privacy. Our work combines a mix of practical tools—methods for risk assessment and algorithms for network anonymization—with more foundational insights into the limits and possibilities for accurate analysis under rigorous privacy standards. In Chapter 2 we present background material. The above contributions are described in detail in Chapters 3-6. Concurrent with our work, a number of others have explored this topic and we review their contributions in Chapter 7 and conclude in Chapter 8.

## CHAPTER 2

### BACKGROUND

In this chapter, we review two privacy definitions,  $k$ -anonymity and differential privacy. Both of these definitions assume tabular data, or at least that an individual's private data is encapsulated in a single record. We discuss how to adapt differential privacy to graphs in Section 2.4, and in Chapter 4 we present a novel privacy definition for graphs that is inspired by  $k$ -anonymity. Also in this chapter, we review some common network analyses and define network statistics that will be used in this dissertation.

#### 2.1 $K$ -anonymity

$K$ -anonymity is a privacy definition that characterizes a table's resistance to re-identification attacks. Before giving the formal definition, we illustrate with an example.

In the original table (Table 2.1(a)), even if we were to remove names,  $t_2$  has a unique combination of *Birth*, *Sex*, and *Zip* attributes and so Beth, the individual corresponding to  $t_2$ , can be identified by an adversary with knowledge of these attributes. To prevent such linkage attacks, attributes *Birth*, *Sex*, and *Zip* have been coarsened in Table 2.1(b). This table has the property that each combination of these attributes occurs at least 3 times. Therefore Table 2.1(b) is 3-anonymous with respect to *Birth*, *Sex*, and *Zip*. Table 2.1(a) is only 1-anonymous because some tuples are uniquely identified by this combination of attributes.

**Table 2.1.** Example to illustrate  $k$ -anonymity. (a) a table  $T$  of medical records; this table is 1-anonymous. (b) table  $T$  after removing the *Name* attribute and coarsening attributes *Birth*, *Sex*, and *Zip*; this table is 3-anonymous.

	Name	Birth	Sex	Zip	Condition
$t_1$	Alice	1985	F	14850	Viral Inf.
$t_2$	Beth	1985	F	14853	Cancer
$t_3$	Carol	1985	F	14850	Cancer
$t_4$	Dave	1986	M	14851	Heart Dis.
$t_5$	Ellen	1986	F	14853	Flu
$t_6$	Fred	1986	M	14621	Heart Dis.
$t_7$	Greg	1981	M	14222	Flu
$t_8$	Hank	1962	M	14850	Heart Dis.
$t_9$	Ian	1944	M	14850	Heart Dis.
$t_{10}$	John	1959	M	14850	Heart Dis.

	Birth	Sex	Zip	Condition
$t_1$	1985	F	1485*	Viral Inf.
$t_2$	1985	F	1485*	Cancer
$t_3$	1985	F	1485*	Cancer
$t_4$	198*	*	14***	Heart Dis.
$t_5$	198*	*	14***	Flu
$t_6$	198*	*	14***	Heart Dis.
$t_7$	198*	*	14***	Flu
$t_8$	19**	M	14850	Heart Dis.
$t_9$	19**	M	14850	Heart Dis.
$t_{10}$	19**	M	14850	Heart Dis.

The formal definition of  $k$ -anonymity relies on the concept of a *quasi-identifier*. Let  $T$  be a table with relational schema  $T(A_1, A_2, \dots, A_m)$ . Let  $\mathcal{A}$  denote the set of attributes  $\{A_1, A_2, \dots, A_m\}$ .

**Definition 2.1** (Quasi-identifier). A set of attributes  $\mathcal{Q} = \{Q_1, \dots, Q_d\} \subseteq \mathcal{A}$  is a quasi-identifier if these attributes are sufficient to uniquely identify a tuple in  $T$  and may be externally available to an adversary.

In the previous example, the quasi-identifier is  $\mathcal{Q} = \{Birth, Sex, Zip\}$ . *Condition* is not included because this is private data that is unlikely to be available externally. However, the other attributes are readily available in external data sources. In fact, these attributes were used by a privacy researcher to re-identify the governor of Massachusetts in a table of medical records [115].

A table is  $k$ -anonymous if none of its records can be uniquely identified by their quasi-identifier. Some notation: For a tuple  $t \in T$  let  $t[A_i]$  denote the value of attribute  $A_i$  on tuple  $t$ , and similarly for a set of attributes  $\mathcal{A}' = \{A_{i_1}, \dots, A_{i_d}\} \subseteq \mathcal{A}$ , let  $t[\mathcal{A}']$  be equal to  $(t[A_{i_1}], \dots, t[A_{i_d}])$ .

**Definition 2.2** ( $k$ -anonymity [110, 111, 116]). Given a quasi-identifier  $\mathcal{Q}$ , a table  $T$  is  $k$ -anonymous with respect to  $\mathcal{Q}$  if for each  $t \in T$ , there exists at least  $k - 1$  other tuples  $t'_1, \dots, t'_{k-1}$  such that  $t[\mathcal{Q}] = t'_1[\mathcal{Q}] = \dots = t'_{k-1}[\mathcal{Q}]$ .

As suggested by the example,  $k$ -anonymity is usually achieved through the coarsening of attribute values. A naive algorithm for  $k$ -anonymity is to partition the tuples into groups of size  $k$  or larger, and then locally recode the attribute values so that tuples are indistinguishable within groups. This achieves anonymity but may result in considerable data loss because if tuples have vastly different values for  $\mathcal{Q}$ , then their values will be heavily coarsened. To maximize the utility of the published table, it is necessary to find the minimal coarsening that achieves the desired level of anonymity.

Several have looked at the complexity of finding the optimal coarsening. The first complexity result considers coarsening the table through attribute suppression. The optimal table is the one that achieves  $k$ -anonymity for a given  $k$  and minimizes the number of suppressions. This problem has been shown to be NP-hard [93]. Other variants of the problem, including ones that are not limited to suppression, have also been shown to be NP-hard [3, 4, 33, 127]. Note however, that the hardness comes from trying to maximize utility, so even a non-optimal solution will still satisfy the privacy condition. (With query auditing, it is computationally hard to even check that the privacy conditions are met.) Many heuristic algorithms have been proposed for making a table  $k$ -anonymous (cf. [4, 5, 15, 46, 69, 70, 71] and references therein).

$K$ -anonymity is a simple, intuitive definition, and it is easy to verify that an output satisfies the definition. Unfortunately, in the last few years, research has identified a number of limitations in  $k$ -anonymity. While it prevents re-identification, it does not necessarily prevent the disclosure of sensitive information. For example, suppose an adversary knows that Ian is 66 year-old male and lives in Ithaca, NY (zip code: 14850). If Table 2.1(b) is published, then the adversary, while unable to re-identify Ian, can nevertheless infer that he suffers from heart disease. This is an instance of a

*homogeneity attack*, in which there exists a set of tuples that not only have matching quasi-identifiers but are also homogenous with respect to a sensitive attribute.

Remedies have been proposed to address the homogeneity attack. However, even when groups are not homogenous, it may be possible for an adversary to make probabilistic inferences about sensitive attributes. In addition, some of the “remedies” introduce additional complexities into the anonymization process, which create new opportunities for adversarial attack. We highlight some of the vulnerabilities that have been identified in the literature.

- **Background knowledge** Even when the tuples within a group are not homogenous, the adversary may be able to use background knowledge to eliminate some values. To protect against both homogeneity and background knowledge attacks, a number of alternative privacy conditions have been developed. They are similar to  $k$ -anonymity but impose additional constraints on the distribution of sensitive attribute values within each group, essentially requiring some diversity in the distribution [19, 79, 84, 86, 88]. While these additional constraints protect against background knowledge attacks, they do not guard against the other attacks described below.
- **Learning** Privacy breaches can occur when the adversary is able to learn correlations from the anonymized data and then use these correlations to infer the sensitive attribute values of individual records [61]. For example, in a table of medical records, the adversary may observe a correlation between smoking and cancer. Suppose Bob is a smoker and his tuple is in a group with three other tuples, none of whom are smokers. Then, if the table reveals that one person in this group has cancer, the adversary may be able to confidently infer that Bob is the one with cancer. The only known remedy to this attack is differential privacy.

- **Composition** A composition attack [45] can occur when multiple agencies release anonymized data and some records occur in more than one dataset. This may allow the adversary to infer sensitive attributes by taking the intersection of values across datasets. Such a scenario might occur, for example, if two area hospitals release anonymized patient records and some patients have visited both hospitals. One way to address composition attacks is to require that agencies who manage overlapping datasets work in conjunction. There are some techniques for releasing multiple datasets in a way that protects privacy [125].
- **Minimality** The minimality attack [123] exploits the fact that many anonymization algorithms publish the table that satisfies the privacy requirement and also *minimizes* some measure of information loss. This allows the adversary to infer sensitive values by process of elimination: for some configurations of sensitive attribute values, the published table would not be minimal. There are some algorithms that are resistant specifically to minimality attacks [28, 123] and other algorithms were recently developed that are resistant to any attack based on knowledge of how the algorithm operates [126].

An attack may not always result in the exact disclosure of an individual's attribute value; it may only result in an educated guess about the value. Furthermore, it may not apply on all instances. Nevertheless, these attacks show that  $k$ -anonymity does not ensure privacy in a rigorous sense. It fails to account for how the adversary might reason, especially given background knowledge about the individuals represented in the data and knowledge about how the algorithm operates.

## 2.2 Differential privacy

In this section, we review differential privacy and existing techniques for answering queries under differential privacy. In Chapter 5, we use these techniques as a component of our solution.

The original definition of differential privacy modeled the private data as a set of records, where each record corresponds to an individual. To define differential privacy, we consider a database  $I$  that contains sensitive information about a set of individuals, and a randomized algorithm  $A$  that operates on  $I$ . Differential privacy ensures that the output of the algorithm does not disclose the presence or absence of an individual's data. Suppose an individual's data is removed from  $I$ , resulting in a new database  $I'$ . Differential privacy requires that whether the input is  $I$  or  $I'$ , the probability of a given output is nearly the same.

More formally, let  $I$  be a set of records where each record corresponds to an individual's private data. For sets  $A$  and  $B$ , we use  $\oplus$  to denote symmetric difference:  $A \oplus B = \{A \cup B\} - \{A \cap B\}$ . Database  $I'$  is a *neighbor* of  $I$  if  $|I \oplus I'| = 1$ . Let  $nbrs(I)$  denote the set of neighbors of  $I$ , i.e.,  $nbrs(I) = \{I' \mid |I \oplus I'| = 1\}$ . For algorithm  $A$ , let  $Range(A)$  be the set of possible outputs of  $A$ . For example, if  $A$  computes the number of records in the database satisfying a given predicate, then  $Range(A)$  is equal to set of non-negative integers. The following definition of differential privacy is due to Dwork [34].<sup>1</sup>

**Definition 2.3** ( $\epsilon$ -differential privacy). An algorithm  $A$  is  $\epsilon$ -differentially private if for all instances  $I$ , any  $I' \in nbrs(I)$ , and any subset of outputs  $S \subseteq Range(A)$ , the following holds:

$$Pr[A(I) \in S] \leq \exp(\epsilon) \times Pr[A(I') \in S]$$

---

<sup>1</sup>Differential privacy has been defined inconsistently in the literature. The original definition (called  $\epsilon$ -indistinguishability) defines neighboring databases in terms of Hamming distance [38]. Note that  $\epsilon$ -differential privacy (as defined above) implies  $2\epsilon$ -indistinguishability.

where probability  $Pr$  is over the randomness of  $A$ .

The parameter  $\epsilon$  measures the disclosure and is typically also an input to the algorithm. For example, the techniques used in this work add random noise to their outputs, where the noise is a function of  $\epsilon$ . The choice of  $\epsilon$  is a matter of policy, but typically  $\epsilon$  is “small,” say at most 1, making the probability “almost the same” whether the input is  $I$  or  $I'$ .

An example illustrates why this protects privacy. Suppose a hospital wants to analyze the medical records of their patients and publish some statistics about the patient population. A patient may wish to have his record omitted from the study, out of a concern that the published results will reveal something about him personally and thus violate his privacy. The above definition assuages this concern because whether the individual opts-in or opts-out of the study, the probability of a particular output is almost the same. Clearly, any observed output cannot reveal much about his particular record if that output is (almost) as likely to occur even when his record is excluded from the database.

**Semantic characterization of differential privacy** The example suggests that differential privacy prevents an adversary from learning private information about an individual. This intuition is formalized in the following result, adapted from Dwork et al. [38]. An *informed adversary* is one who knows all records in the database except for one. Let  $t$  be any tuple. Let  $\mathcal{A}_t$  denote an informed adversary who knows  $I - \{t\}$ .

**Definition 2.4** (Semantic security [38]). Let  $f$  be any boolean predicate over databases. Algorithm  $A$  is  $\epsilon$ -semantically secure if for all  $I$ , tuples  $t$ , informed adversaries  $\mathcal{A}_t$ , and any subset of outputs  $S \subseteq Range(A)$ , the following holds:

$$\exp(-\epsilon) \times \Pr[f(I) = 1] \leq \Pr[f(I) = 1 \mid A(I) \in S] \leq \exp(\epsilon) \times \Pr[f(I) = 1]$$

where the probability is taken with respect to the randomness in  $A$  and the prior beliefs of  $\mathcal{A}_t$ .

This semantic notion of privacy is equivalent to differential privacy.

**Proposition 2.1** (Equivalence [38]). *Algorithm  $A$  satisfies  $\epsilon$ -differential privacy if and only if  $A$  satisfies  $\epsilon$ -semantic security.*

**Hiding multiple records** Differential privacy is defined with respect to the addition or removal of a single record. However, it extends naturally to protecting the addition or removal of a *set* of records. When we consider adapting differential privacy to graphs, we will use this observation to protect the set of edges surrounding a node.

**Definition 2.5** ( $(k, \epsilon)$ -differential privacy). An algorithm  $A$  is  $(k, \epsilon)$ -differentially private if for all instances  $I$ , any  $I'$  such that  $|I \oplus I'| = k$ , and any subset of outputs  $S \subseteq \text{Range}(A)$ , the following holds:

$$\Pr[A(I) \in S] \leq \exp(\epsilon) \times \Pr[A(I') \in S]$$

where probability  $\Pr$  is over the randomness of  $A$ .

Note that  $\epsilon$ -differential privacy is the special case of  $(1, \epsilon)$ -differential privacy. By a transitivity,  $A$  satisfies  $(1, \epsilon)$ -differential privacy if and only if  $A$  satisfies  $(k, k\epsilon)$ -differential privacy.

**A relaxation of differential privacy** As we will later see, some properties cannot be accurately measured under the strong requirements of differential privacy. Several relaxations of differential privacy have been proposed, including the one defined below. As we later discuss, under this relaxation, it is possible to accurately compute some network statistics that cannot be accurately answered under strict  $\epsilon$ -differential privacy.

**Definition 2.6** ( $(\epsilon, \delta)$ -differential privacy [36]). An algorithm  $A$  is  $(\epsilon, \delta)$ -differentially private if for all instances  $I$ , any  $I' \in \text{nbrs}(I)$ , and any subset of outputs  $S \subseteq \text{Range}(A)$ , the following holds:

$$\Pr[A(I) \in S] \leq \exp(\epsilon) \times \Pr[A(I') \in S] + \delta$$

where probability  $\Pr$  is over the randomness of  $A$ .

### 2.3 Differentially private query answering

We review a technique by Dwork et al. [38] for answering numerical queries under differential privacy. Known as the *Laplace mechanism*, it achieves differential privacy by adding random noise to query answers, where the noise is carefully calibrated to the query. We will use this technique as a component of our solution.

The input to the Laplace mechanism is  $\mathbf{Q}$ , a sequence of queries where the answer to each query is a number in  $\mathbb{R}$ . The algorithm computes the true answer  $\mathbf{Q}(I)$  to the queries on the private data and then adds random noise to the answers. The noise depends on the query sequence’s *sensitivity*. For vectors  $\mathbf{x}$  and  $\mathbf{y}$ , let  $\|\mathbf{x} - \mathbf{y}\|_1$  denote their  $L_1$  distance:  $\|\mathbf{x} - \mathbf{y}\|_1 = \sum_i |x_i - y_i|$ .

**Definition 2.7** (Sensitivity). The sensitivity of  $\mathbf{Q}$ , denoted  $\Delta_{\mathbf{Q}}$ , is defined as

$$\Delta_{\mathbf{Q}} = \max_{I, I' \in \text{nbrs}(I)} \|\mathbf{Q}(I) - \mathbf{Q}(I')\|_1.$$

Intuitively, if a query has high sensitivity, then adding or removing an individual’s private data can have a large effect on the query answer. There are many queries with low sensitivity, such as the one in the following example.

**Example 1.** Let  $I$  be a database of employment records and let query sequence  $\mathbf{Q}$  be a salary histogram—specifically,  $\mathbf{Q}[1]$  returns the number of records whose salary is

less than \$10K,  $\mathbf{Q}[2]$  the number with salary between \$10K and \$20K, and so on, up to  $\mathbf{Q}[10]$ , the number with salary between \$90K and \$100K. A neighboring database instance,  $I'$ , has one additional or one less record, which affects exactly one of the histogram counts by 1. Therefore, the sensitivity of  $\mathbf{Q}$  is  $\Delta_{\mathbf{Q}} = 1$ .

To ensure privacy, query answers are randomly perturbed with noise proportional to their sensitivity.

The following algorithm ensures differential privacy for any  $\mathbf{Q}$ . Let  $\langle \text{Lap}(\sigma) \rangle^d$  denote a  $d$ -length vector of independent random samples from a Laplace distribution with mean zero and scale  $\sigma$ . (The Laplace distribution has density function  $f(y) \propto \exp(-|y|/\sigma)$ .)

**Proposition 2.2** (Laplace mechanism [38]). *Let  $\tilde{\mathbf{Q}}$  denote the randomized algorithm that takes as input a database  $I$ , a query  $\mathbf{Q}$  of length  $d$ , and some  $\epsilon > 0$ , and outputs*

$$\tilde{\mathbf{Q}}(I) = \mathbf{Q}(I) + \langle \text{Lap}(\Delta_{\mathbf{Q}}/\epsilon) \rangle^d$$

*Algorithm  $\tilde{\mathbf{Q}}$  satisfies  $\epsilon$ -differential privacy.*

We rely on Proposition 2.2 to ensure privacy for the query sequences we propose in this work. We emphasize that the proposition holds for *any* query sequence  $\mathbf{Q}$ , regardless of correlations or constraints among the queries in  $\mathbf{Q}$ . Such dependencies are accounted for in the calculation of sensitivity. (For example, consider the correlated sequence  $\mathbf{Q}$  that consists of the *same* query  $q$  repeated  $k$  times, then the sensitivity of  $\mathbf{Q}$  is  $k$  times the sensitivity of  $\langle q \rangle$ .)

One can also use the above mechanism to answer multiple query sequences. This is useful in interactive settings, where the analyst's query may depend on the answers to previous queries. Differentially private algorithms can be composed: the protocol that allows the analyst to issue  $\ell$  query sequences, each one using the  $\epsilon$ -differentially private mechanism above, is  $\ell\epsilon$ -differentially private [92].

**Hiding multiple records** The Laplace mechanism can be configured to provide  $(k, \epsilon)$ -differential privacy, thereby protecting a set of records. The Laplace mechanism can be instantiated with  $\epsilon' = \epsilon/k$ —i.e., the algorithm outputs the following:

$$\tilde{\mathbf{Q}}(I) = \mathbf{Q}(I) + \langle \text{Lap}(\Delta_{\mathbf{Q}}/\epsilon') \rangle^d$$

To make this concrete, suppose  $k = 10$  and  $\epsilon = 1.0$ . Then  $\epsilon' = 0.1$ . This instantiation of the Laplace mechanism satisfies both  $(1, 0.1)$ -differential privacy and  $(10, 1.0)$ -differential privacy, and even  $(100, 10.0)$ -differential privacy and so on.

## 2.4 Differential privacy for graphs

In the above definition, the database is a table whereas in the present work, the database is a graph. Below we adapt the definition of differential privacy to graphs.

The semantic interpretation of differential privacy rests on the definition of neighboring databases. Since differential privacy guarantees that the output of the algorithm cannot be used to distinguish between neighboring databases, what is being protected is precisely the difference between neighboring databases. In the above definition, a neighboring database is defined as the addition or removal of a single record. With the hospital example, the patient’s private information is encapsulated within a single record. So differential privacy ensures that the output of the algorithm does not disclose the patient’s medical history.

With network data, which is primarily about relationships among individuals, the correspondence between private data and database records is less clear. To adapt differential privacy to graphs, we must choose a definition for neighboring graphs and understand the privacy semantics of that choice. We propose three alternatives offering varying degrees of privacy protection.

We model the input as a graph,  $G = (V, E)$ , where  $V$  is a set of  $n$  entities and  $E$  is a set of edges. Edges are undirected pairs  $(u, v)$  such that  $u$  and  $v$  are members of

$V$ . While the meaning of an edge depends on the domain—it could connote friendship, email exchange, sexual relations, etc.—we assume that it represents a sensitive relationship that should be kept private. The focus of the present work is concerned with graph structure, so the inclusion of attributes on nodes or edges is left for future work.

The first adaptation of differential privacy to graphs is mathematically similar to the definition for tables. Neighboring graphs are defined as graphs that differ by one “record.” Given a graph  $G$ , one can produce a neighboring graph  $G'$  by either adding/removing an edge in  $E$ , or by adding/removing an *isolated* node in  $V$ . Restricting the definition to isolated nodes ensures that the change to  $V$  does not require additional changes to  $E$  to make it consistent with  $V$ .

**Definition 2.8** (Edge  $\epsilon$ -differential privacy). An algorithm  $A$  is *edge  $\epsilon$ -differentially private* if for all graphs  $G = (V, E)$  and  $G' = (V', E')$  such that  $|V \oplus V'| + |E \oplus E'| = 1$ , and any subset of outputs  $S \subseteq \text{Range}(A)$ , the following holds:

$$\Pr[A(G) \in S] \leq \exp(\epsilon) \times \Pr[A(G') \in S]$$

An edge-differentially private algorithm protects individual edges from being disclosed. For some applications, edge-differential privacy seems to be a reasonable privacy standard. For example, consider the study of Kossinets and Watts [67], in which they analyze a graph derived from the email communication among students and faculty of a large university. What makes this dataset sensitive is that it reveals who emails whom; edge-differential privacy protects email relationships from being disclosed.

However, in some applications, it may be desirable to extend the protection beyond individual edges. For example, Klov Dahl et al. [65] analyze the social network structure of “a population of prostitutes, injecting drug users and their personal associates.” In this graph, an edge represents a sexual interaction or the use of a shared

needle. Edges are clearly private information, but so too are other properties like node degree (the number of sexual/drug partners) and even membership in the network.

A second adaptation of differential privacy to graphs provides much stronger privacy protection. In *node-differential privacy*, two graphs are neighbors if they differ by at most one node and *all* of its incident edges.

**Definition 2.9** (Node  $\epsilon$ -differential privacy). An algorithm  $A$  is *node  $\epsilon$ -differentially private* if for all graphs  $G = (V, E)$  and  $G' = (V', E')$  such that  $|V \oplus V'| = 1$  and  $E \oplus E' = \{(u, v) | u \in (V \oplus V') \text{ or } v \in (V \oplus V')\}$ , and any subset of outputs  $S \subseteq \text{Range}(A)$ , the following holds:

$$\Pr[A(G) \in S] \leq \exp(\epsilon) \times \Pr[A(G') \in S]$$

Node-differential privacy mirrors the “opt-in/opt-out” notion of privacy from the hospital example. It assuages any privacy concerns, as a node-differentially private algorithm behaves almost as if the participant did not appear in at all.

While node-differential privacy is a desirable privacy objective, it may be infeasible to design algorithms that are both node-differentially private and enable accurate network analysis. A differentially private algorithm must hide even the worst case difference between neighboring graphs, and this difference can be large under node-differential privacy. For instance the empty graph ( $n$  isolated nodes) is a neighbor of the star graph (a hub node connected to  $n$  nodes).

To span the spectrum of privacy between edge- and node-differential privacy, we introduce an extension to edge-differential privacy that allows neighboring graphs to differ by more than a single edge. In  *$k$ -edge-differential privacy*, neighboring graphs can differ by up to  $k$  edges.

**Definition 2.10** ( $k$ -edge  $\epsilon$ -differential privacy). An algorithm  $A$  is  *$k$ -edge  $\epsilon$ -differentially private* if for all graphs  $G = (V, E)$  and  $G' = (V', E')$  such that  $|V \oplus V'| + |E \oplus E'| \leq k$ ,

and any subset of outputs  $S \subseteq \text{Range}(A)$ , the following holds:

$$\Pr[A(G) \in S] \leq \exp(\epsilon) \times \Pr[A(G') \in S]$$

A larger setting of  $k$  leads to greater privacy protection. If  $k = 1$ , then  $k$ -edge-differential privacy is equivalent to edge-differential privacy. If  $k = |V|$ , then  $k$ -edge-differential privacy is even stronger than node-differential privacy, as the set of neighboring graphs under  $k$ -edge-differential privacy is a superset of the neighbors under node-differential privacy. If  $1 < k < |V|$ , then  $k$ -edge-differential privacy prevents the disclosure of aggregate properties of any subset of  $k$  edges. Notice that for those nodes whose degree is less than  $k$ , it provides essentially equivalent protection as node-differential privacy. Nodes whose degree is  $k$  or larger face more exposure. However, nodes with large degree also have greater influence on the structure of the graph. If our goal is to also allow analysts to accurately measure the graph structure, then it may be necessary to expose high degree nodes to greater privacy risk.

## 2.5 Network analyses and statistics

We provide a brief overview of some common network analyses to illustrate the diversity of analyses that are done on network data. We also define several network statistics that are used later in the dissertation. In Chapter 4, we use these statistics to evaluate how the privacy-preserving transformations impact network topology. Also, in Chapters 5 and 6, we describe answer perturbation techniques for approximating some of these statistics.

Many analyses focus solely on measuring the topology of the network as defined by the edge table. Such analyses include the distribution of node degrees, the distribution of path lengths, and measures of clustering or transitivity. These basic structural properties of networks compensate for the difficulty in visualizing large networks.

Significant research effort has been devoted to models of network formation that generate graphs possessing the structural properties seen in the real world [10, 21, 26, 50, 73, 74, 72, 122].

Some analyses pinpoint specific structural features of the network. Analysis of network centrality [42] seeks to identify influential nodes. In addition, community discovery [100] divides the network into meaningful clusters. Motif analysis [94] identifies interesting structures that occur repeatedly in a network.

Another category of research focuses on understanding the function of the network by modeling processes that occur within the network. Such processes include search or navigation within networks [112, 121] and diffusion across networks (e.g., rumors or epidemics spreading in a group) [63].

While the above analyses focus on the structure of the graph, the presence of attributes on edges or nodes allow for some new analyses and variants of those above. For example, homophily, the tendency for associations to form among similar individuals, can be measured in a network with attributes on nodes [89, 104]. Network models have been developed that model the correlation between structural features and attributes [50]. Finally, network data can include temporal information, allowing the study of network dynamics. This includes the development of models of network formation and evolution [73] and models to accurately predict future links [67, 80].

This gives a brief overview of the diverse ways in which networks are analyzed. More complete surveys of network analysis appear in the literature [32, 99].

**Network statistics** Below we define some of the network statistics that we use in this dissertation. Let  $G = (V, E)$  be an undirected graph where  $|V| = n$  and  $|E| = m$ .

- **Density** The *edge density* of a graph is the number of edges divided by the number of possible edges:  $m/\binom{n}{2}$ .

- **Path statistics** A *path* from  $u$  to  $v$  is a sequence of edges that traverse from  $u$  to  $v$ . The *shortest path* between  $u$  and  $v$  is the path with the minimum number of edges, and its length is the number of edges. The *diameter* of a graph is the length of the longest shortest path.
- **Connected components** A *connected component* is a maximal subgraph such that for any pair of nodes in the subgraph, there is a path connecting them. Some statistics related to connected components include the *number of connected components* and the *relative size of the largest connected component*.
- **Distortion** *Distortion* is a statistic that captures how closely a graph resembles a tree [117]. To compute distortion of  $G$ , we first construct a spanning tree  $T$ . Then for each edge  $(u, v)$  in  $G$ , we compute the distance (length of shortest path) between  $u$  and  $v$  in  $T$ . The distortion is the average distance over all edges in  $G$ . Thus, it measures how path lengths of  $G$  are distorted (i.e., lengthened) if we are restricted to only traversing edges in tree  $T$ .
- **Degree** In an undirected graph, the *degree* of a node  $u$  is the number of neighbors of  $u$  (i.e., nodes  $v$  such that  $(u, v) \in E$ ). In a directed graph, the *in-degree* is the number of incoming edges, the *out-degree* is the number of outgoing edges. The *degree sequence* of a graph is a non-decreasing sequence of the degrees of the nodes in the graph.
- **Degree variability** For a degree sequence  $\mathbf{d} = (d_1, \dots, d_n)$ , the coefficient of variation  $C_V(\mathbf{d})$  is defined as  $C_V(\mathbf{d}) = \sigma(\mathbf{d}) / \langle \mathbf{d} \rangle$  where  $\langle \mathbf{d} \rangle$  is the average degree and  $\sigma(\mathbf{d}) = \sqrt{\sum_{i=1}^n (d_i - \langle \mathbf{d} \rangle)^2 / (n - 1)}$ . Graphs with homogenous degrees have low  $C_V$  and graphs with diverse degree sequences, such as power-law graphs, have high  $C_V$  [7].

- **Degree correlations** The *s-metric* is a measure of degree correlation [78]. It is defined as  $s(G) = \sum_{(u,v) \in E} d(u)d(v)$  where  $d(u)$  is the degree of node  $u$ . A high  $s(G)$  indicates that high degree nodes are connected to one another. The *normalized s-metric* is  $s(G)/s_{max}(G)$  where  $s_{max}(G)$  is the maximum possible  $s$  of any graph with the same degree sequence as  $G$ . (In practice, it is computationally intensive to find the true maximum, so we approximate it with the Havel-Hakimi graph [16], which is efficient to construct and tends to have very high  $s$ .)
- **Clustering** *Clustering coefficient* measures the likelihood that two neighbors of a node are themselves connected. It is defined as  $C(G) = \frac{1}{n} \sum_u \frac{2\Delta(u)}{d(u)(d(u)-1)}$  where  $\Delta(u)$  is the number of triangles (cliques of size 3) containing  $u$ .
- **Transitivity** A closely related measure to clustering coefficient is *transitivity*. It is defined three times the number of triangles in the graph divided by the number of paths of length two.
- **Motif** A *motif* is a subgraph pattern, and motif analysis is the process of finding “interesting” subgraph patterns, such as those that are unlikely to arise by chance given some model of network formation.

The above statistics are highlighted because they are used elsewhere in the dissertation. Newman [99] and da F. Costa [32] provide more comprehensive summaries of common network statistics.

## CHAPTER 3

### ASSESSING RE-IDENTIFICATION RISK

We consider the problem of publishing network data in such a way that permits useful analysis yet avoids disclosing sensitive information. Most existing work on privacy in data publishing has focused on tabular data, where each record represents a separate entity, and an individual may be re-identified by matching the individual’s publicly known attributes with the attributes of the anonymized table. Anonymization techniques for tabular data do not apply to network data because they fail to account for the interconnectedness of the entities (i.e., they destroy the network structure).

Because network analysis can be performed in the absence of entity identifiers (e.g., name, social security number), a natural strategy for protecting sensitive information is to replace identifying attributes with synthetic identifiers. We refer to this procedure as *naive anonymization*. It is a common practice and presumably, it protects sensitive information by breaking the association between the real-world identity and the sensitive data.

However, naive anonymization may be insufficient. A distinctive threat in network data is that an entity’s connections (i.e., the network structure around it) can be distinguishing, and may be used to re-identify an otherwise anonymous individual. We consider how a malicious individual (the *adversary*) might obtain partial knowledge about the network structure around targeted individuals and then use this knowledge to re-identify them in the anonymized network. Once re-identified, the adversary can learn additional properties about the targets; for instance, he may be able to infer

the presence or absence of edges between them. Since individual connections are often considered sensitive information, such *edge disclosure* constitutes a violation of privacy. Whether naive anonymization provides adequate protection depends on the structure of the network and the adversary’s capability. In this chapter, we provide a comprehensive assessment of the privacy risks of naive anonymization.

Although an adversary may also have information about the attributes of nodes, the focus of this chapter is on disclosures resulting from *structural* or *topological* re-identification, where the adversary’s information is about the structure of the graph only. The use of attribute knowledge to re-identify individuals in anonymized data has been well-studied, as have techniques for resisting it [85, 88, 110, 111, 116]. More importantly, many network analyses are concerned exclusively with structural properties of the graph, therefore safely publishing an unlabeled network is an important goal in itself. For example, the following common analyses examine only the network structure: finding communities, fitting power-law models, enumerating motifs, measuring diffusion, and assessing resiliency [99].

In this chapter, we make the following contributions:

- **Adversary Model** We propose a flexible model of external information used by an adversary to attack naively-anonymized networks. The model allows us to evaluate re-identification risk efficiently and for a range of different adversary capabilities. We also formalize the structural indistinguishability of a node with respect to an adversary with locally-bounded external information (Section 3.1).
- **Empirical Risk Assessment** We evaluate the effectiveness of structural attacks on real and synthetic networks, measuring successful re-identification and edge disclosures. We find that real networks are diverse in their resistance to attacks. Nevertheless, our results demonstrate that naive anonymization provides insufficient protection, especially if an adversary is capable of gathering knowledge beyond a target’s immediate neighbors (Section 3.2).

- **Theoretical Risk Assessment** In addition to the empirical study, we perform a theoretical analysis of random graphs. We show how properties such as a graph’s density and degree distribution affect re-identification risk. A significant finding is that in sufficiently dense graphs, nodes can be re-identified even when the graph is extremely large (Section 3.3).

The findings of this chapter tell us what makes nodes vulnerable to re-identification attacks. This understanding serves as a basis for designing effective strategies for anonymizing network data, a subject which is taken up in Chapter 4.

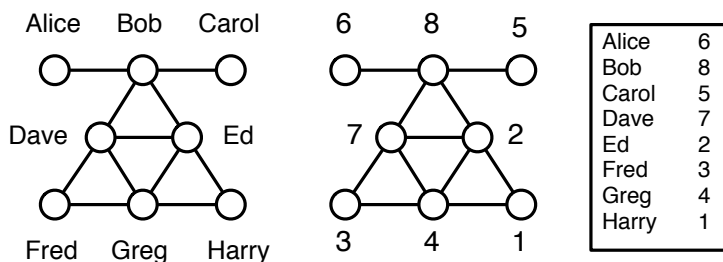
### 3.1 Modeling the adversary

In this section we describe the capabilities and motivations of the adversary in the context of network data. First, we describe the process of naive anonymization and how the adversary may attack it. Second, we define the threats of node re-identification and edge disclosure. Third, we explain how anonymity is achieved through structural similarity, which motivates a model of adversary knowledge based on degree signatures. Finally we review alternative models of the adversary.

#### 3.1.1 Naive anonymization

Formally, we model a network as an undirected graph  $G = (V, E)$ . The naive anonymization of  $G$  is an isomorphic graph,  $G_a = (V_a, E_a)$ , defined by a random bijection  $\Pi : V \rightarrow V_a$ . For example, Figure 3.1 shows a small network represented as a graph along with its naive anonymization. The *anonymization mapping*  $\Pi$ , also shown, is a random, secret mapping.

Naive anonymization prevents re-identification when the adversary has no information about individuals in the original graph. Formally stated, an individual  $x \in V$ , called the *target*, has a *candidate set*, denoted  $\text{cand}(x)$ , which consists of the nodes of  $G_a$  that could feasibly correspond to  $x$ . To assess the risk of re-identification, we



**Figure 3.1.** A social network represented as a graph (left), the naive anonymization (center), and the anonymization mapping (right).

assume each element of the candidate set is equally likely and use the size of the candidate set as a measure of resistance to re-identification. Since  $\Pi$  is random, in the absence of other information, any node in  $G_a$  could correspond to the target node  $x$ . Thus, given an uninformed adversary, each individual has the same risk of re-identification, specifically  $\text{cand}(x) = V_a$  for each target individual  $x$ .

However, if the adversary has access to external information about the entities, he may be able to reduce the candidate set and threaten the privacy of individuals.

### 3.1.2 Threats

In practice the adversary may have access to external information about the entities in the graph and their relationships. This information may be available through a public source beyond the control of the data owner, or may be obtained by the adversary’s malicious actions. For example, for the graph in Figure 3.1, the adversary might know that “*Bob has three or more neighbors,*” or that “*Greg is connected to at least two nodes, each with degree 2.*” Such information allows the adversary to reduce the set of candidates in the anonymized graph for each of the targeted individuals. For example, the first statement allows the adversary to partially re-identify Bob:  $\text{cand}(Bob) = \{2, 4, 7, 8\}$ . The second statement re-identifies Greg:  $\text{cand}(Greg) = \{4\}$ .

Re-identification can lead to additional disclosures under naive anonymization. If an individual is uniquely re-identified, then the entire structure of connections surrounding the individual is revealed. If two individuals are uniquely re-identified, then the presence or absence of an edge between them is revealed directly by the naively anonymized graph. Such an edge disclosure, in which an adversary is able to accurately infer the presence of an edge between two identified individuals, can be a serious privacy threat. In the present work, we consider the general threat of *re-identification* as well as the more specific threat *edge disclosure*.

Throughout this work, we model the adversary’s external information as access to a source that provides answers to a restricted *knowledge query* evaluated for a single target node of the original graph  $G$ .

An adversary attempts re-identification for a target node  $x$  by using  $Q(x)$  to refine the feasible candidate set. Since  $G_a$  is published, the adversary can easily evaluate *any* structural query directly on  $G_a$ , looking for matches. The adversary will compute the refined candidate set that contains all nodes in the published graph  $G_a$  that are consistent with answers to the knowledge query on the target node.

**Definition 3.1** (Candidate Set under  $Q$ ). For a knowledge query  $Q$  over a graph, the candidate set of target node  $x$  w.r.t  $Q$  is  $\text{cand}_Q(x) = \{y \in V_a \mid Q(x) = Q(y)\}$ .

**Example 2.** Referring to the example graph in Figure 3.1, suppose  $Q$  is a knowledge query returning the degree of a node. Then for targets  $Ed$ ,  $Fred$ ,  $Greg$  we have  $Q(Ed) = 4$ ,  $Q(Fred) = 2$ ,  $Q(Greg) = 4$ , and candidate sets  $\text{cand}_Q(Ed) = \text{cand}_Q(Greg) = \{2, 4, 7, 8\}$  and  $\text{cand}_Q(Fred) = \{1, 3\}$ .

Given two target nodes  $x$  and  $y$ , the adversary can use the naively anonymized graph to deduce the likelihood that the nodes are connected. In the absence of external information, the likelihood of any edge is simply the density of the graph (the fraction of all possible edges that exist in the graph).

If the candidate sets for  $x$  and  $y$  have been refined by the adversary's knowledge about  $x$  and/or  $y$ , then the adversary reasons about the likelihood  $x$  and  $y$  are connected based on the connections between the candidate sets for  $x$  and  $y$ . Thus we define the edge likelihood to be the Bayesian posterior belief assuming each candidate is an equally likely match for the targeted nodes.

**Definition 3.2** (Edge likelihood under  $Q$ ). For a knowledge query  $Q$  over a graph, and a pair of target nodes  $x$  and  $y$ , the inferred likelihood of edge  $(x, y)$  under  $Q$  is denoted  $\text{prob}_Q(x, y)$  and defined as:

$$\frac{|\{(u, v) \mid u \in X, v \in Y\}| + |\{(u, v) \mid u, v \in X \cap Y\}|}{|X| \cdot |Y| - |X \cap Y|}$$

where  $X = \text{cand}_Q(x)$  and  $Y = \text{cand}_Q(y)$ .

The denominator represents the total number of possible edges from a node of one candidate set to a node of the other candidate set, and accounts for the case where the intersection of the candidate sets is non-empty.

**Example 3.** Continuing the example above, the inferred likelihood of edge  $(Ed, Fred)$  is:

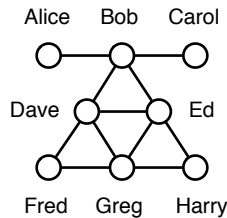
$$\text{prob}_Q(Ed, Fred) = (4 + 0)/(4 * 2) = 0.500$$

because there are 4 edges present in  $G_a$  between the disjoint candidate sets  $\text{cand}_Q(Ed)$  and  $\text{cand}_Q(Fred)$ . The inferred edge likelihood of edge  $(Ed, Greg)$  is:

$$\text{prob}_Q(Ed, Greg) = (5 + 5)/(4 * 4 - 4) = 0.833$$

because 5 edges are present in  $G_a$  between the identical candidate sets  $\text{cand}_Q(Ed)$  and  $\text{cand}_Q(Greg)$ . These edge likelihoods should be compared with the prior edge density of  $2 * 11/(8 * 7) = .393$ .

Node ID	$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_2$
Alice	$\epsilon$	1	{4}
Bob	$\epsilon$	4	{1, 1, 4, 4}
Carol	$\epsilon$	1	{4}
Dave	$\epsilon$	4	{2, 4, 4, 4}
Ed	$\epsilon$	4	{2, 4, 4, 4}
Fred	$\epsilon$	2	{4, 4}
Greg	$\epsilon$	4	{2, 2, 4, 4}
Harry	$\epsilon$	2	{4, 4}



(a) graph

(b) structural signatures

Equivalence Relation	Equivalence Classes
$\equiv_{\mathcal{H}_0}$	{A, B, C, D, E, F, G, H}
$\equiv_{\mathcal{H}_1}$	{A, C} {B, D, E, G} {F, H}
$\equiv_{\mathcal{H}_2}$	{A, C}{B}{D, E}{G}{F, H}
$\equiv_A$	{A, C}{B}{D, E}{G}{F, H}

(c) equivalence classes

**Figure 3.2.** (a) A sample graph, (b) external information consisting of structural signatures  $\mathcal{H}_0, \mathcal{H}_1$  and  $\mathcal{H}_2$  computed for each individual in the graph, (c) the equivalence classes of nodes implied by the structural signatures. For the sample data,  $\equiv_{\mathcal{H}_2}$ , corresponds to automorphic equivalence,  $\equiv_A$ .

In Section 3.2, we measure the threats of edge disclosure and node re-identification on real networks.

### 3.1.3 Anonymity through structural similarity

Intuitively, nodes that look structurally similar may be indistinguishable to an adversary, in spite of external information. A strong form of structural similarity between nodes is *automorphic equivalence*. Two nodes  $x, y \in V$  are automorphically equivalent (denoted  $x \equiv_A y$ ) if there exists an isomorphism from the graph onto itself that maps  $x$  to  $y$ .

**Example 4.** Fred and Harry are automorphically equivalent nodes in the graph of Figure 3.1. Bob and Ed are not automorphically equivalent: the subgraph around Bob is different from the subgraph around Ed and no isomorphism proving automorphic equivalence is possible.

Automorphic equivalence induces a partitioning on  $V$  into sets whose members have identical structural properties. It follows that an adversary — even with exhaustive knowledge of a target node’s structural position — cannot identify an individual beyond the set of entities to which it is automorphically equivalent. We say that two such nodes are *structurally indistinguishable* and observe that nodes in the graph achieve anonymity by being “hidden in the crowd” of its automorphic class members.

Some special graphs have large automorphic equivalence classes. For example, in a complete graph, or in a graph which forms a ring, all nodes are automorphically equivalent. But in most graphs we expect to find small automorphism classes, likely to be insufficient for protection against re-identification.

Though automorphism classes may be small in real networks, automorphic equivalence is an extremely strong notion of structural similarity. In order to distinguish two nodes in different automorphic equivalence classes, it may be necessary to use complete information about their positions in the graph. For a weaker adversary with limited knowledge, nodes that are not automorphically equivalent may in fact be indistinguishable. For example, for an adversary who only knows the degree of targeted nodes in the graph, Bob and Ed are indistinguishable (even though they are not automorphically equivalent). This motivates the notion of bounded structural knowledge we describe next.

#### **3.1.4 Adversary model based on structural signatures**

We now describe the adversary model. It is based on a class of knowledge queries, of increasing power, which report on the local structure of the graph around a node. These queries are inspired by iterative vertex refinement, a technique originally developed to efficiently test for the existence of graph isomorphisms [31]. In Section 3.1.5, we discuss alternative adversary models.

The queries are denoted  $\mathcal{H}_i$  for  $i = 0, 1, 2, \dots$ . The weakest knowledge query,  $\mathcal{H}_0$ , simply returns the label of the node. (We consider here unlabeled graphs, so  $\mathcal{H}_0$  returns  $\epsilon$  on all input nodes.) The queries are successively more descriptive:  $\mathcal{H}_1(x)$  returns the degree of  $x$ ,  $\mathcal{H}_2(x)$  returns the multiset of each neighbors' degree, and so on. The queries can be defined iteratively, where  $\mathcal{H}_i(x)$  returns the multiset of values which are the result of evaluating  $\mathcal{H}_{i-1}$  on the set of nodes adjacent to  $x$ :

$$\mathcal{H}_i(x) = \{\mathcal{H}_{i-1}(z_1), \mathcal{H}_{i-1}(z_2) \dots, \mathcal{H}_{i-1}(z_m)\}$$

where  $z_1 \dots z_m$  are the nodes adjacent to  $x$ .

**Example 5.** Figure 3.2 contains the same graph from Figure 3.1 along with the computation of  $\mathcal{H}_0$ ,  $\mathcal{H}_1$ , and  $\mathcal{H}_2$  for each node. For example:  $\mathcal{H}_0$  is uniformly  $\epsilon$ .  $\mathcal{H}_1(\text{Bob}) = \{\epsilon, \epsilon, \epsilon, \epsilon\}$ , which we abbreviate in the table simply as 4. Using this abbreviation,  $\mathcal{H}_2(\text{Bob}) = \{1, 1, 4, 4\}$  which represents Bob's neighbors' degrees.

In practice, we might expect that if an adversary can learn the degrees of the target's neighbors, he would also be able to learn about edges in the neighborhood. In this case, instead of learning  $\mathcal{H}_i$ , the adversary would learn a subgraph where the subgraph is induced by the edges adjacent to nodes that lie within at most  $i - 1$  edge traversals of the target. This additional knowledge would make the adversary more powerful, and thus the  $\mathcal{H}_i$  signature is a more conservative model. The  $\mathcal{H}_i$  signatures have the advantage that they are efficient to evaluate, whereas measuring subgraph knowledge requires checking for subgraph isomorphisms, an NP-Hard problem. Thus, the  $\mathcal{H}_i$  signature can be viewed as an efficient way to calculate a lower bound on the risk of the subgraph adversary. In Section 3.1.5, we discuss prior work, including our own, that has considered models based on knowledge of subgraphs surrounding the target.

For each query  $\mathcal{H}_i$ , we define an equivalence relation on nodes in the graph in the natural way.

**Definition 3.3** (Relative equivalence). Two nodes  $x, y$  in a graph are *equivalent relative to  $\mathcal{H}_i$* , denoted  $x \equiv_{\mathcal{H}_i} y$ , if and only if  $\mathcal{H}_i(x) = \mathcal{H}_i(y)$ .

**Example 6.** Figure 3.2(c) lists the equivalence classes of nodes according to relations  $\equiv_{\mathcal{H}_0}$ ,  $\equiv_{\mathcal{H}_1}$ , and  $\equiv_{\mathcal{H}_2}$ . All nodes are equivalent relative to  $\mathcal{H}_0$  (for an unlabeled graph). As  $i$  increases, the values for  $\mathcal{H}_i$  contain successively more precise structural information, and as a result, equivalence classes are divided.

To an adversary limited to knowledge query  $\mathcal{H}_i$ , nodes equivalent with respect to  $\mathcal{H}_i$  are indistinguishable. The following proposition formalizes this intuition:

**Proposition 3.1.** *Let  $x, x' \in V$ . If  $x \equiv_{\mathcal{H}_i} x'$  then  $\text{cand}_{\mathcal{H}_i}(x) = \text{cand}_{\mathcal{H}_i}(x')$ .*

Iterative computation of  $\mathcal{H}$  continues until no new vertices are distinguished. We call this query  $\mathcal{H}^*$ . In the example of Figure 3.2,  $\mathcal{H}^* = \mathcal{H}_2$ . The vertex refinement technique is the basis of efficient graph isomorphism algorithms which can be shown to work for almost all graphs [8]. In our setting, this means that equivalence under  $\mathcal{H}^*$  is very likely to coincide with automorphic equivalence.

### 3.1.5 Alternative adversary models

Throughout this work, we use the structural signatures described above as a parameterized model of external information that can capture the power of a range of adversaries. Our structural signatures have the advantage that they are efficient to evaluate even on large graphs, are amenable to theoretical analysis, and they are conservative model of structural knowledge.

One of our guiding principles is that adversary knowledge tends to be local to the targeted node, with more powerful adversaries capable of exploring the neighborhood around a node with increasing diameter.

In practice, external information about a published social network may be acquired through malicious actions by the adversary or from public information sources. In addition, a participant in the network, with some innate knowledge of entities and their relationships, may be acting as an adversary in an attempt to uncover unknown information. A legitimate privacy objective in some settings is to publish a graph in which participating individuals cannot re-identify themselves. For the participant-adversary, whose knowledge is based on their participation in the network, existing research about institutional communication networks suggests that there is a horizon of awareness of about distance two around most individuals [43].

Other work on network anonymity has also focused on adversaries whose structural knowledge is based on a local neighborhood around a target node [29, 82, 128, 132, 134]. An exception is the recent work by Narayanan et al. [97], which uses an auxiliary network to attack a target network, and work by Zou et al. [136], which protects against an adversary with unbounded structural knowledge.

In previous work [51], we considered alternative models of adversary knowledge, including partial subgraphs and signatures determined by connections to hubs. In evaluating adversaries with knowledge of partial subgraphs around a target, re-identification risk is generally lower than with degree signatures, but depends on how complete the known subgraph is. It is also computationally difficult to compute candidate sets because testing a potential candidate requires looking for a subgraph isomorphism.

Hubs are highly connected nodes observed in many network datasets. In a Web graph, a hub may be a highly visited website. In a graph of email connections, hubs often represent influential individuals. Because hubs are often outliers in a graph's degree distribution, the true identity of hub nodes is often apparent in a naively-anonymized graph. In addition, an individual's connections to hubs may be publicly known or easily deduced. We found that on real networks, the rate of re-identification using knowledge of hub connections was relatively low.

**Table 3.1.** Descriptive statistics for the real and synthetic graphs studied.

Statistic	Real Datasets			Synthetic Datasets			
	<b>HepTh</b>	<b>Enron</b>	<b>NetTrace</b>	<b>HOT</b>	<b>Power-Law</b>	<b>Tree</b>	<b>Mesh</b>
Nodes	2510	111	4213	939	2500	3280	2500
Edges	4737	287	5507	988	7453	3279	4900
Minimum degree	1	1	1	1	2	1	2
Maximum degree	36	20	1656	91	166	4	4
Median degree	2	5	1	1	4	1	4
Average degree	3.77	5.17	2.61	2.10	5.96	1.99	3.92
Edge density	0.0007	0.0235	0.0003	0.0022	0.0024	0.0006	0.0016
Avg. cand. set size ( $\mathcal{H}_1$ )	558.5	12.0	2792.1	635.5	549.7	1821.8	2138.1
Avg. cand. set size ( $\mathcal{H}_2$ )	25.4	1.5	608.6	81.1	1.4	1659.8	1818.1
Percent re-identified ( $\mathcal{H}_1$ )	0.2	2.7	0.5	0.9	0.9	< 0.1	< 0.1
Percent re-identified ( $\mathcal{H}_2$ )	40.4	73.9	11.1	5.9	82.5	< 0.1	< 0.1

As mentioned above, the focus of this work is on supporting the topological analysis of graphs. We therefore assume that attributes are not used to aid in re-identification, and our assessment of utility does not include analyses that depend on attribute values. Other authors have proposed anonymization schemes that protect against re-identification using attributes [29, 30, 134].

### 3.2 Empirical risk assessment

In this section we evaluate the risk of publishing the naive anonymization of a network through an empirical assessment on several real and synthetic network datasets.

For each dataset, we consider each node in turn as a target. We assume the adversary computes the structural signature of that node, and then we compute the corresponding candidate set. We report the distribution of candidate set sizes across the population of nodes to characterize how many nodes are protected and how many are identifiable.

We use the following seven datasets. The **HepTh** dataset is a graph of coauthors in theoretical high-energy physics. The dataset is derived from arXiv, an online

repository of papers. We extracted a subset of the authors and considered them connected if they wrote at least two papers together.

The **Enron** dataset is derived from a corpus of email sent to and from managers at Enron Corporation, made public by the Federal Energy Regulatory Commission during its investigation of the company. Two individuals are connected if they corresponded at least 5 times.

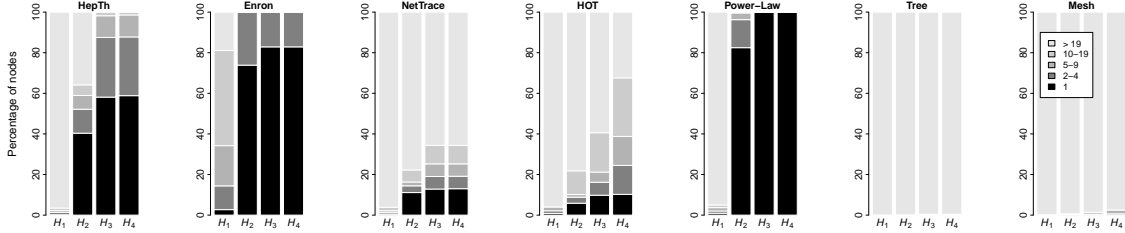
The **NetTrace** dataset was derived from an IP-level network trace collected at a major university. The trace monitors traffic at the gateway; it produces a bipartite graph between IP addresses internal to the institution, and external IP addresses. We restricted the trace to 187 internal addresses from a single campus department and the 4026 external addresses to which at least 20 packets were sent on port 80 (http traffic).

The **HOT** dataset is a model of the Internet of a single service provider (ISP). Its Heuristically Optimal Topology (HOT) is designed to reflect the economic and technological constraints that influence the topology. It has a hierarchical structure with a core of interconnected low degree (high-bandwidth) routers at its center and high-degree (low-bandwidth) routers at its periphery [77].

The **Power-Law** dataset is a random graph that is generated based on a model of growth and preferential attachment [10]. Its degree distribution follows a power-law. In some of the experiments, we also consider a slightly different dataset, **Clustered Power-Law**, which is constructed using the same model except that when edges are inserted into the graph, triangles are formed with some probability (we set  $p = 0.4$ ).

The **Mesh** dataset is a  $50 \times 50$  grid topology, where each node is connected to the four adjacent nodes in the grid. The **Tree** dataset is a balanced tree of arity 3.

All datasets have undirected edges, with self-loops removed. We eliminated a small percentage of disconnected nodes in each dataset, focusing on the largest connected component in the graph. Detailed statistics for the datasets are shown in Table 3.1.



**Figure 3.3.** The relationship between candidate set size and structural signature knowledge  $\mathcal{H}_i$  for  $i = 1..4$  for four real graphs and three synthetic graphs. For each  $\mathcal{H}_i$ , the bars show the percentage of nodes whose candidate sets have sizes in the following buckets: [1] (black), [2, 4], [5, 10], [11, 20], [21,  $\infty$ ] (white).

### 3.2.1 Node re-identification

Recall from Section 3.1.4 that nodes contained in the same candidate set for knowledge  $\mathcal{H}_i$  share the same value for  $\mathcal{H}_i$ , are indistinguishable according to  $\mathcal{H}_i$ , and are therefore protected if the candidate set size is sufficiently large.

Figure 3.3 is an overview of the likelihood of re-identification under  $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$  and  $\mathcal{H}_4$  knowledge queries. For each  $\mathcal{H}_i$ , the graph reports on the percentage of nodes whose candidate sets have sizes in the following buckets: [1], [2, 4], [5, 10], [11, 20], [21,  $\infty$ ]. Nodes with candidate set size 1 have been uniquely identified, and nodes with candidate sets between 2 and 4 are at high risk for re-identification. Nodes are at fairly low risk for re-identification if there are more than 20 nodes in their candidate set.<sup>1</sup> Each  $\mathcal{H}_i$  is represented as a different point on the  $x$ -axis.

Figure 3.3 shows that for the **HepTh** data,  $\mathcal{H}_1$  leaves nearly all nodes at low risk for re-identification, and it requires  $\mathcal{H}_3$  knowledge to uniquely re-identify a majority of nodes. For **Enron**, under  $\mathcal{H}_1$  about 15% of the nodes have candidate sets smaller than 5, while only 19% are protected in candidate sets greater than 20. Under  $\mathcal{H}_2$ , re-identification jumps dramatically so that virtually all nodes have candidate sets

<sup>1</sup>We do not suggest these categories as a universal privacy standard, but merely as divisions that focus attention on the most important part of the candidate set distribution where serious disclosures are at risk.

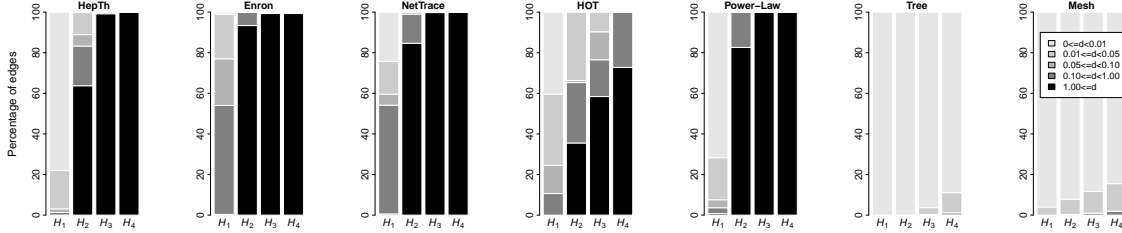
less than 5. These two real graphs are roughly similar in behavior to the synthetic **Power-Law** graph, as they display features similar to a power-law graph.

**NetTrace** and **HOT** have substantially lower disclosure overall, with very few identified nodes under  $\mathcal{H}_1$ , and even  $\mathcal{H}_4$  knowledge does not uniquely identify more than 10% of the nodes. For **NetTrace**, this results from the unique bipartite structure of the trace dataset: many nodes in the trace have low degree, as they are unique or rare web destinations contacted by only one internal host. The **HOT** graph has high structural uniformity because it contains many degree one nodes that are connected to the same high degree node, and thus structurally equivalent to one another.

The synthetic **Tree** and **Mesh** graphs display very low re-identification under all  $\mathcal{H}_i$ . This is obvious given that these graphs have highly uniform structure: the nodes in **Mesh** have either degree 2 or 4, the nodes in **Tree** have degree 1, 3 or 4. We include them here for completeness as these graphs are studied in Section 4.3.

A natural precondition for publication is a very low percentage of high-risk nodes under a reasonable assumption about adversary knowledge. Three datasets meet that requirement for  $\mathcal{H}_1$  (**HepTh**, **NetTrace**, **HOT**). Except for the extreme synthetic graphs **Tree** and **Mesh**, no datasets meet that requirement for  $\mathcal{H}_2$ .

Overall, we observe that there can be significant variance across different datasets in their vulnerability to different adversary knowledge. However, across all datasets, the most significant change in re-identification is from  $\mathcal{H}_1$  to  $\mathcal{H}_2$ , illustrating the increased power of adversaries that can explore beyond the target’s immediate neighborhood. Re-identification tends to stabilize after  $\mathcal{H}_3$ —more information in the form of  $\mathcal{H}_4$  does not lead to an observable increase in re-identification in any dataset. Finally, even though there are many re-identified nodes, a substantial number of nodes are *not* uniquely identified even with  $\mathcal{H}_4$  knowledge.



**Figure 3.4.** The inferred edge probabilities resulting from attempted re-identification using structural signatures  $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ .

### 3.2.2 Edge disclosure

We measure the risk of edge disclosure possible under adversaries with knowledge of degree signatures. Our sample datasets are sparse graphs – their edge densities are all quite low, as reported in Table 3.1. This means that the expectation of any particular edge existing in the graph is low.

To measure the risk of edge disclosure, we considered each edge present in the original graph and considered its inferred edge likelihood under various  $\mathcal{H}_i$ . That is, we imagine an adversary using  $\mathcal{H}_i$  knowledge to re-identify the individuals participating in each edge of the true graph, and report the inferred edge probability over the set of all true edges. For each  $\mathcal{H}_i$  we get a range of inferred edge probabilities, as illustrated in Figure 3.4.

The results show that with  $\mathcal{H}_1$  knowledge alone, the risk of edge disclosure is relatively limited. In the Hep-Th data, 80% of the edges have an inferred edge probability of less than 0.01, which constitutes a small shift in an adversary’s certainty about the presence of those edges. In the **Enron** and **NetTrace** data, roughly half the edges have inferred probabilities between 0.10 and 1, which represent a significant shift in the adversary’s expectation.

Of much more concern, however, is the fact that with  $\mathcal{H}_2$  knowledge (or greater) many edges are disclosed with certainty – the inferred edge probability is 1 for a majority of edges across all datasets. It is also important to note that even when

candidate sets tend to be large (such as in **NetTrace** and **HOT**), edges can be disclosed with high likelihood. In **NetTrace** and **HOT** this likely reflects a hub node with a unique degree connected to many degree-one nodes. Even though the candidate set of degree one nodes may be large, every node in that candidate set is connected to the hub, and density of connections between the candidate sets is one, resulting in certain edge disclosure.

### 3.3 Theoretical risk assessment

The results of the previous section show that re-identification risk varies across graphs. We want to understand and explain this variation. In some cases, such as **Tree** and **Mesh**, the low re-identification risk can be explained by the regular topology, which makes it hard to distinguish nodes by their local structure. However, across the other graphs, the reason for diversity in risk is unclear.

In this section, to gain insight into the factors affecting re-identification risk, we study random graphs. Random graphs are governed by parameters which control some aspect of the graph’s topology; by varying the parameters, we can measure how this property affects re-identification risk. Here, we study how re-identification risk is affected by two key graph properties, density and degree distribution. To study the relationship between graph density and anonymity, we analyze the Erdős-Rényi (ER) model [39, 40, 47], the simplest random graph model. Following that, we study random graphs with power-law degree distributions. These results help us to understand under what conditions distinctive structures arise in graphs, and thus provide insight into the foundations of anonymity for graphs.

### 3.3.1 Erdős-Rényi graphs

The ER model generates a graph by sampling each of the  $\binom{n}{2}$  edges independently with probability  $p$ . As the number of nodes,  $n$ , increases, these graphs exhibit different behaviors depending on how  $p$  scales with  $n$ .

We consider three cases. In a *sparse* random graph  $p = c/n$ , in a *dense* random graph  $p = c \log n/n$ , and in a *super-dense* random graph,  $p = c$  (where  $c$  is a constant). The first two cases are of interest because when  $c > 1$ , with high probability the graph includes a giant connected component of size  $\Theta(n)$  and a collection of smaller components (in the sparse case) or the graph is completely connected (in the dense case) [40].

To motivate the theoretical results that follow, Figure 3.5 shows experimental simulations on ER random graph of 100K nodes and varying edge probabilities. The trend lines measure the percentage of nodes *uniquely* identified by  $\mathcal{H}_1$ ,  $\mathcal{H}_2$ , and  $\mathcal{H}_3$  knowledge.

The figure shows that for sparse graphs, very few nodes are uniquely identified, even with the more powerful  $\mathcal{H}_3$  knowledge. Intuitively, nodes cannot be distinguished because a sparse graph lacks sufficient edge density to create diversity in structure. Because the edge probability is  $p = c/n$ , the expected node degree, which is  $p(n-1)$ , goes to  $c$  as  $n \rightarrow \infty$ . Because the expected degree is constant, for sufficiently large  $n$ , structural patterns must repeat, leading to complete structural uniformity in the limit. The following theorem formalizes this intuition, showing that no degree of  $\mathcal{H}_i$  knowledge can distinguish nodes in a large sparse ER random graph.

**Theorem 3.1** (Sparse ER random graphs). *Let  $G$  be an ER random graph containing  $n$  nodes with edge probability given by  $p = c/n$  for  $c > 1$ . (i) The expected sizes of the equivalence classes induced by  $\mathcal{H}_i$  are  $\Theta(n)$  for any  $i \geq 0$ ; (ii) with probability going to one, the sizes of the equivalence classes induced by  $\mathcal{H}_i$  are  $\Omega(n^\alpha)$ , for any  $i \geq 0$  and any  $0 < \alpha < 1$ .*

*Proof.* We begin with  $\mathcal{H}_1$ . Consider a graph of size  $n$ . Let  $N_i$  denote the degree of the  $i$ -th node,  $i \leq n$ . As  $n \rightarrow \infty$ ,

$$P(N_i = k) \rightarrow \frac{c^k}{k!} e^{-c}$$

Note that for any  $k = \omega(1)$ , the probability of  $N_i = k$  goes to zero as  $n \rightarrow \infty$ . Thus, it suffices only consider the case where  $k$  is a constant.

Let  $M_{1,k}(n)$  denote the expected size of the equivalence class of  $\mathcal{H}_1$  corresponding to node degree  $k$  when the graph is of size  $n$  and let  $M_{1,k} = \lim_{n \rightarrow \infty} M_{1,k}(n)$ . We have

$$\begin{aligned} M_{1,k} &= \lim_{n \rightarrow \infty} \sum_{i=1}^n P(N_i = k) \\ &= \Theta(n) \end{aligned}$$

In order to establish the second result, we restrict ourselves to a random subset of the  $n$  nodes of size  $n^\alpha$ , where  $\alpha < 1$ . Note that the fraction of nodes in this subset goes to zero as  $n \rightarrow \infty$ . This allows us to show that, as  $n \rightarrow \infty$ , the degrees of the nodes in this subset are independent random variables. Application of a Chernoff bound then produces  $P[L_k \leq (1 - \delta)n^\alpha c^k e^{-c}/k!] \leq e^{-(\delta^2 c^k e^{-c}/k!)n^\alpha}$  where  $L_k$  is the number of nodes in the subset having degree  $k$  as  $n \rightarrow \infty$ . As  $|M_{i,k}| \geq L_k$ , we conclude that  $|M_{1,k}(n)| = \Omega(n^\alpha)$  with probability going to one for all  $k$ .

Similar arguments hold for  $\mathcal{H}_i$ ,  $i = 2, \dots$ . Consider a node  $x$ . We first note that the  $\mathcal{H}_i$  equivalence class that  $x$  belongs to is determined by the subgraph rooted at  $x$  that includes all nodes within distance  $i$  of it. Now, as  $n \rightarrow \infty$ , with probability going to one, this subgraph is a tree. Moreover the probability of the above subgraph deviating from a tree is  $O(1/n)$ . Another observation is that every  $\mathcal{H}_i$  induced equivalence class contains at least one node, whose distance  $i$  subgraph is a tree in the limit as  $n \rightarrow \infty$ . This follows because any  $\mathcal{H}_i$  consistent multi-set can be used to construct a tree. Thus

any distance  $i$  subgraph centered at a node that is not a tree is hidden by commonly found trees.

Consider a tree,  $t$ , of height  $i$  or less. Let  $N(t)$  be a set containing the numbers of children for all nodes in the tree that are at distance  $j = 0, 1, \dots, i - 1$  from the root. Let  $G_i(x)$  denote the distance  $i$  subgraph centered at node  $x$  and let  $T_i$  denote the set of all possible height  $i$  or less trees. Then

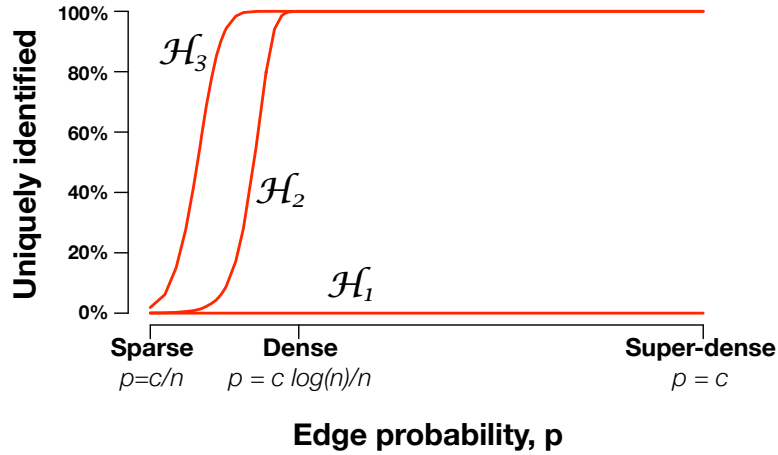
$$\begin{aligned} P(G_i(x) = t) &= \prod_{k \in N(t)} \frac{c^k}{k!} e^{-c} + O(1/n), \quad t \in T_i \\ &= \Theta(1) \\ P(G_i(x) \notin T_i) &= O(1/n) \end{aligned}$$

Note that as  $n$  grows, the distribution of the number of children that a node within the tree has is Poisson.

Since each equivalence class contains at least one height  $i$  or less tree in the limit as  $n \rightarrow \infty$ , it follows from the above expressions that the expected size of each equivalence class is  $\Theta(n)$ . Last a similar argument as used for  $\mathcal{H}_1$  establishes the second property.  $\square$

From the standpoint of protecting anonymity, this is an encouraging result for this class of graphs, assuming we are concerned with publishing large graphs. (In simulations, we found that some re-identification occurs in random graphs of less than  $10^6$  nodes.)

As we consider more dense ER random graphs, structural diversity increases and re-identification becomes a near certainty very quickly. Figure 3.5 suggests that as graphs become dense ( $p = c \log(n)/n$ ), while nodes remain well-hidden against  $\mathcal{H}_1$  adversaries,  $\mathcal{H}_2$  knowledge is sufficient to re-identify virtually all nodes in the graph. The following theorem supports the simulations.



**Figure 3.5.** For  $\mathcal{H}_2$  and  $\mathcal{H}_3$  the number of uniquely re-identified individuals in a classical random graph goes from zero to 100% quickly when there is sufficient edge density. But regardless of the density, the number of nodes with a unique degree is close to zero, showing that  $\mathcal{H}_1$  is insufficient for unique re-identification.

**Theorem 3.2** (Dense ER random graphs). *Let  $G$  be an ER random graph containing  $n$  nodes with edge probability given by  $p = c \log n/n$  for  $c > 1$ .*

1. *With high probability a node belongs to an equivalence class induced by  $\mathcal{H}_1$  that grows to infinity as  $n \rightarrow \infty$ .*
2. *The expected sizes of equivalence classes induced by  $\mathcal{H}_2$  goes to zero as  $n \rightarrow \infty$ .*

The second property indicates that for a given  $\mathcal{H}_2$  signature, the expected number of nodes having that signature grows more slowly than  $n$ . Given the simulation results, the most likely cause of this result is that the  $\mathcal{H}_2$  signatures are unique.

*Proof.* As  $n \rightarrow \infty$ , the degree distribution converges to the Poisson distribution with mean  $c \log n$ . Let  $N_i(n)$  denote the degree of node  $i$  in a graph of size  $n$  and consider degrees of the form  $N_i(n) = \delta c \log n$ ,  $0 < \delta$ . Then, as  $n \rightarrow \infty$ , we have

$$P[N_i = \delta c \log n] = \frac{1}{\sqrt{2\pi\delta c \log n} n^{c(1-\delta+\delta \log \delta)}}, \quad 0 < \delta$$

A Chernoff bound argument can be used to show that, whp, a node's degree lies within the range  $(\delta_0 c \log n, \delta_1 c \log n)$  where  $\delta_0$  is the largest value of  $\delta < 0$  such that  $c(1 - \delta + \delta \log \delta) = 1$  and  $\delta_1$  is the smallest value of  $\delta > 1$  to satisfy that equation. Note that  $P[N_i = \delta c \log n]$  decreases more quickly than  $1/n$  whenever  $\delta \notin [\delta_0, \delta_1]$  and more slowly otherwise. We focus now on the range of degrees  $(\delta_0 c \log n, \delta_1 c \log n)$ , and let  $L_\delta$  denote the number of nodes with degree  $\delta c \log n$ ,  $\delta \in [\delta_0, \delta_1]$ . Randomly select a set of nodes of size  $n^\alpha$ , where  $\alpha$  is chosen such that,  $c(1 - \delta + \delta \log \delta) < \alpha < 1$ . As in the previous theorem, we can show that the degrees of these nodes become independent random variables as  $n \rightarrow \infty$ . Apply now a Chernoff bound (as  $n \rightarrow \infty$ ) to obtain

$$P[L_\delta < (1 - \beta)n^\alpha (2\pi\delta c \log n)^{-1} n^{-c(1-\delta+\delta \log \delta)}] \leq e^{-\beta^2 n^\alpha (2\pi\delta c \log n)^{-1} n^{-c(1-\delta+\delta \log \delta)}/2}$$

Because of the choice of  $\alpha$ , the right hand side goes to zero. Thus  $L_\delta \rightarrow \infty$  as  $n \rightarrow \infty$  whp and therefore the size of the equivalence class corresponding to degree  $\delta c \log n$  goes to infinity as  $n \rightarrow \infty$ . Since a node takes its degree from the range  $(\delta_0 c \log n, \delta_1 c \log n)$  whp, it belongs to an equivalence class whose size goes to infinity whp as  $n \rightarrow \infty$ .

The proof of the second property is more involved. We sketch the proof. Consider a node with degree  $k$ , we need only consider  $k \in (\delta_0 c \log n, \delta_1 c \log n)$ . Moreover, we need only consider degrees of the neighbors in the same range. Furthermore, we can assume that the degrees of the neighbors are independent of each other as  $n \rightarrow \infty$ . Application of a straightforward generalization of Theorem 5.7 in [96] to the case of a non-uniformly random balls and urns problem allows us to write

$$P[X_1 = k_1, \dots, X_s = k_s] \leq e^{\sqrt{\delta c \log n} \prod_{i=1}^s \frac{(p_i \delta c \log n)^{k_i}}{k_i!}} e^{-p_i \delta c \log n}$$

where  $p_i$  is the probability that a neighbor selects degree  $i$ . Here  $(X_1, \dots, X_s)$  constitutes the  $\mathcal{H}_2$  signature of the node. Now, it is easy to argue using Chernoff bounds that neighbors only choose degrees clustered around  $c \log n$  ( $c \log n + l$ ,  $l = 0, \pm 1, \pm 2, \dots$ ). Hence

$$\begin{aligned} P[X_1 = k_1, \dots, X_s = k_s] &\leq \\ e\sqrt{\delta c \log n} \frac{(p c \log n \delta c \log n)^{\delta c \log n}}{\prod_{i=1}^s k_i!} e^{-p c \log n \delta c \log n} &\leq \\ a((\log n)^{-1/2})^{\delta c \log n} (e^{-b(\log n)^{-1/2}})^{\delta c \log n} & \end{aligned}$$

Where  $a$  is a constant. The second inequality follows from  $\prod_i k_i > 1$ . Now consider the expected number of nodes with signature  $(k_1, \dots, k_s)$ ,  $M_{k_1, \dots, k_s}$ . It is upper bounded by

$$M_{k_1, \dots, k_s} \leq a n ((\log n)^{-1/2})^{\delta c \log n} (e^{-b(\log n)^{-1/2}})^{\delta c \log n}$$

which goes to zero as  $n \rightarrow \infty$ . □

Lastly, we include a known result for the case of a super-dense graph where  $p = 1/2$ . The following theorem, originally due to Babai and Kucera [8] and rephrased below, shows that with high probability every node will be uniquely identified using  $\mathcal{H}_3$  knowledge:

**Theorem 3.3** (Super-dense ER random graphs). *Let  $G$  be an ER random graph on  $n$  nodes with edge probability  $p = 1/2$ . The probability that there exist two nodes  $x, y \in V$  such that  $x \equiv_{\mathcal{H}_3} y$  is less than  $2^{-cn}$  for constant value  $c > 0$ .*

This result provides a sufficient condition for unique re-identification of the entire population in a graph.

Theorems 3.2 and 3.3 are disappointing from an anonymity perspective. However, most social and communication networks appear to be sparse, and so Theorem 3.1 may be more applicable. Furthermore, real networks often have heavy-tailed degree

distributions, which is not the case for ER graphs. To capture the heavy-tailed degree distribution, we also study re-identification risk in power-law graphs.

### 3.3.2 Power-law graphs

Several graph models have been proposed that exhibit the heavy-tailed degree distributions often observed in real networks, including the power law random graph (PLRG) model [6]. In this model, a graph is constructed by first assigning a degree to each node, where the degree is sampled from a power law distribution. Edges are inserted by randomly choosing endpoints until every node has as many edges as its specified degree. (This can result in self-loops or multiple edges between a pair of nodes, which are often removed to form a simple graph that closely approximates the original degree distribution.)

The PLRG, and other power-law models, generate graphs with constant average degree as the number of nodes increases. Thus the edge density is low, and despite the skew in node degree, we find that the structural diversity is insufficient for re-identification. We state this formally for PLRG because it is the easiest power-law graph model to analyze.

**Theorem 3.4** (Power-law random graphs). *Let  $G$  be a PLRG on  $n$  nodes. With probability going to one, the expected sizes of the equivalence classes induced by  $\mathcal{H}_i$  is  $\Theta(n)$ , for any  $i \geq 0$ .*

*Proof.* The proof of Theorem 3.4 proceeds in a similar manner to the proof of Theorem 3.1 except that the Poisson distribution is replaced by  $P(N_i = k) = ak^{-\alpha} > 0$ ,  $k = 0, 1, \dots$  where  $a$  is a constant such that  $\sum_{k=0}^{\infty} P(N_i = k) = 1$ .  $\square$

### 3.3.3 Discussion

The theoretical results of this section complement the empirical results of the previous section. We see that re-identification risk depends on graph size: the empirical

results for the 2500 node **Power-Law** graph show high re-identification risk; however, Theorem 3.4 shows that once a power-law graph is sufficiently large, nodes will be anonymous.

In fact, the critical factor determining re-identification risk in large random graphs is not the degree distribution, but density. Sparse graphs (including power law graphs) have low re-identification risk, whereas dense graphs have high re-identification risk. This is an important finding as it shows that even in extremely large graphs, nodes are not necessarily well hidden. It depends on the topological properties of the graph. This one reason why the  $\mathcal{H}_i$  structural signatures can be a valuable tool for data owners, as they allow them to efficiently assess re-identification risk even on large graphs.

### 3.4 Conclusion

We have focused on what we believe to be one of the most basic and distinctive challenges for protecting privacy in network datasets—understanding the extent to which graph structure acts as an identifier. We have formalized adversary knowledge and evaluated their impact on real and synthetic networks as well as models of random graphs. Our findings suggest that there is considerable risk in publishing the naive anonymization of a graph. In the next chapter, we investigate strategies for mitigating re-identification risk through more complex transformations of the graph.

## CHAPTER 4

### MITIGATING RE-IDENTIFICATION RISK

The previous chapter was about risk assessment. The main finding was that there is considerable risk in publishing the naive anonymization of a graph because informed adversaries can use their knowledge to re-identify nodes and in some cases, infer particular edges. In this chapter, we focus on risk mitigation. We make the following contributions:

- **Privacy Definition** First, we propose a privacy condition, which formally specifies a limit on how much the adversary can learn about a node’s identity. We compare it with other definitions that have been proposed in the literature and discuss its limitations (Section 4.1).
- **Anonymization Algorithm** Then we propose a novel algorithm to achieve this privacy condition. The algorithm produces a generalized graph, which describes the structure of the original graph in terms of node groups called *supernodes*. The generalized graph retains key structural properties of the original graph yet ensures anonymity (Section 4.2).
- **Algorithm Evaluation** We perform a comprehensive evaluation of the utility of the generalized graphs. This includes a comparison with other state-of-the-art graph anonymization algorithms (Section 4.3).

## 4.1 Structural anonymity

In the previous sections, the size of the candidate set is used as a measure of re-identification risk. This is a natural measure for naive anonymization. A node can be a candidate only if its local graph structure is an exact match to the adversary’s knowledge. Therefore each candidate is an equally plausible guess for the target. However, as we move beyond naive anonymization to consider strategies that alter the graph structure, the size of the candidate set is no longer an appropriate measure of risk.

If the graph structure has been altered by the anonymization process, the alterations may have changed the structure around the target. Therefore a candidate may include not only exact matches in the published graph, but also partial matches. In addition, not all matches are equally likely. The probability of a candidate depends on the adversary’s prior belief about the structure around the target, and on the likelihood that the algorithm altered that structure to produce the observed output.

We introduce a new privacy condition to account for these differences. Invariably, the first step of any algorithm is to perform naive anonymization to create uncertainty about the true identities of the nodes. Recall  $\Pi : V \rightarrow V_a$ , the secret mapping between identifiers in the original graph and the synthetic identifiers in the anonymized graph. The adversary’s goal is to learn this mapping; the data owner’s goal is to sufficiently alter the graph so that the adversary fails to achieve its goal.

Our privacy definition is a condition on the adversary’s posterior belief after having seen the published graph. The posterior belief depends on the published graph, the algorithm that produced the published graph, and the adversary’s prior belief. A successful anonymization is one that meets the following definition:

**Definition 4.1** (Graph  $k$ -anonymity under  $Q$ ). Let  $Q$  be a structural knowledge query. An anonymized graph  $G_a$  satisfies graph  $k$ -anonymity with respect to  $Q$  if

$$\forall x \in V, \forall y \in V_a : Pr[\Pi(x) = y \mid G_a] \leq 1/k$$

where the probability depends on the randomness of the algorithm that produced  $G_a$  and the adversary’s prior probability over input graphs  $G$ .

If we make the natural assumption that the adversary has no other external information other than  $Q$ , then the adversary’s prior probability is uniform over all graphs  $G$  such that in  $G$ , the structure around  $x$  agrees with  $Q(x)$ .

Revisiting naive anonymization, there is a relationship between graph  $k$ -anonymity and our previously used measure of risk, the size of the candidate set. If the probability distribution over candidates is uniform, this condition simply requires at least  $k$  candidates: a naive anonymization satisfies graph  $k$ -anonymity under  $Q$  if for any  $x$ ,  $|\text{cand}_Q(x)| \geq k$ .

Finally, as we will see in Section 4.2, some anonymizations are graph  $k$ -anonymous with respect to any  $Q$ . We simply say in this case that the output satisfies graph  $k$ -anonymity.

**Relation to alternative privacy conditions and limitations** The above condition of graph  $k$ -anonymity is similar to, and in some sense encompasses other definitions recently proposed for graph data. Liu and Terzi [82] propose a condition which requires that in the published graph each degree in the graph occurs at least  $k$  times. Such an output satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$  (i.e., degree). Zhou and Pei [134] require that in the published graph each neighborhood (the subgraph induced by a node and its neighbors) be isomorphic to at least  $k - 1$  others. Such an output satisfies graph  $k$ -anonymity with respect to  $N$  where  $N$  is the knowledge query that returns the neighborhood subgraph of a node. Note it also satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$  since query  $N$  also reveals node degree.

The above definitions are graph analogues of  $k$ -anonymity [110, 111, 116], a privacy condition defined for tables. Each assumes the adversary has some knowledge about

a target entity (analogous to knowledge of the quasi-identifier) and the anonymity condition requires that this knowledge cannot be used to distinguish entities in the published data. The graph data privacy conditions differ on how much knowledge the adversary is assumed to have (node degree, neighborhood, etc.); analogous to differences in the choice of quasi-identifier.

Like  $k$ -anonymity, the above definitions also have limitations. In a homogeneity attack, while the adversary is not able to distinguish among a set of candidates, all of the candidates share a common property. Because the candidates are homogenous, the adversary has learned something about the target, even though re-identification did not occur. In tabular data, definitions such as  $\ell$ -diversity [85] and  $t$ -closeness [79] have been introduced to counter the threat of homogeneity attacks.

An instance of the homogeneity attack is edge disclosure (Section 3.1). A published graph which is graph  $k$ -anonymous may still be vulnerable to edge disclosure. To address the threat of edge disclosure, Cormode et al. [29] introduce an edge safety condition (described in Section 4.3.1 of this work). While this prevents edge disclosure, it appears to do so at a significant expense to utility, based on the experimental results in Section 4.3.3. In addition, we measure the risk of edge disclosure of our proposed algorithm and find in practice it is low for reasonable  $k$  (Section 4.3.5).

Other attacks have been proposed on tabular data anonymizations, and analogues of these attacks may apply to graph anonymization. Attacks include the composition attack [45], the minimality attack [123], and the deFinetti attack [61]. While some of these attacks can be remedied by imposing additional conditions (e.g.,  $m$ -invariance [125] defends against the composition of multiple releases of a dynamic table), developing data publication techniques that resist all of them is an open problem, not only for graph data, but for tabular data as well. Differential privacy [38] ensures protection from all of the above attacks, but it remains unclear whether efficient and accurate data publication is possible under differential privacy [34, 35].

As discussed in Section 7, some differentially private algorithms for graph data have been developed, but they output answers to particular queries and do not publish a graph.

## 4.2 Graph generalization algorithm

In this section we describe an anonymization technique that protects against re-identification by generalizing the input graph. We generalize a graph by grouping nodes into partitions, and then publishing the number of nodes in each partition, along with the density of edges that exist within and across partitions. The adversary attempts re-identification in the generalized graph, while the analyst uses it to study properties of the original graph.

### 4.2.1 Graph generalization

To generalize a naively-anonymized graph  $G_a = (V_a, E_a)$ , we partition its nodes into disjoint sets. The elements of a partitioning  $\mathcal{V}$  are subsets of  $V_a$ . They can be thought of as *supernodes* since they contain nodes from  $G_a$ , but are themselves the nodes of a undirected generalized graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . The *superedges* of  $\mathcal{E}$  include self-loops and are labeled with non-negative weights by the function  $d : \mathcal{E} \rightarrow \mathbb{Z}^*$ .  $\mathcal{G}_{\mathcal{V}}$  is a generalization of  $G_a$  under a partitioning  $\mathcal{V}$  if the edge labels report the density of edges (in  $G_a$ ) that exist within and across the partitions:

**Definition 4.2** (Generalization of graph). Let  $\mathcal{V}$  be the supernodes of  $V_a$ .  $\mathcal{G}$  is a *generalization of  $G_a$  under  $\mathcal{V}$*  if, for all  $X, Y \in \mathcal{V}$ ,  $d(X, Y) = |\{(x, y) \in E_a \mid x \in X, y \in Y\}|$ .

$\mathcal{G}$  summarizes the structure of  $G_a$ , but the accuracy of that summary depends on the partitioning. For any generalization  $\mathcal{G}$  of  $G_a$ , we denote by  $\mathcal{W}(\mathcal{G})$ , the set of possible worlds (graphs over  $V_a$ ) that are consistent with  $\mathcal{G}$ . Intuitively, this set of graphs is generated by considering each supernode  $X$  and choosing exactly  $d(X, X)$

edges between its elements, then considering each pair of supernodes  $(X, Y)$  and choosing exactly  $d(X, Y)$  edges between elements of  $X$  and elements of  $Y$ . The size of  $\mathcal{W}(\mathcal{G})$  is a measure of the accuracy of  $\mathcal{G}$  as a summary of  $G_a$ .

The partitioning of nodes is chosen so that the generalized graph satisfies privacy goals and maximizes utility, as explained in Sections 4.2.2 and 4.2.3 respectively. In the extreme case that all partitions contain a single node, then the graph generalization  $\mathcal{G}$  does not provide any additional anonymity:  $\mathcal{W}(\mathcal{G})$  contains just the graph  $G_a$  (the function  $d$  encodes its adjacency matrix). At the other extreme, if all nodes are grouped into a single partition, then  $\mathcal{G}$  consists of a single supernode with a self-loop labeled with  $|E_a|$  (the total number of edges in the original graph).  $\mathcal{W}(\mathcal{G})$  is thus the set of all graphs over  $V_a$  with  $|E_a|$  edges. In this case the generalization provides anonymity, but is unlikely to be useful to the analyst since it reflects only the edge density of the original graph.

In studying a generalized graph, the analyst can sample a single random graph from  $\mathcal{W}(\mathcal{G})$  and then perform standard graph analysis on this synthetic graph. Repeated sampling can improve the accuracy of analysis. We study in Section 4.3 the bias and variance of estimates of graph properties based on graphs sampled from  $\mathcal{W}(\mathcal{G})$ .

#### 4.2.2 Anonymity of generalized graphs

To ensure anonymity we require that the adversary have a minimum level of uncertainty about the identity of any target node in  $V$ . We use the size of a partition to provide a basic guarantee against re-identification and require that each partition have size at least  $k$ . This ensures that the output satisfies graph  $k$ -anonymity with respect to any structural query  $Q$ .

**Proposition 4.1.** *Let  $\mathcal{G}$  be a generalized graph such that each supernode  $X$  has at least  $k$  nodes. Then  $\mathcal{G}$  satisfies graph  $k$ -anonymity.*

*Proof.* The intuition for this claim is that the generalized graph summarizes the graph in terms of supernodes and contains no information that allows the adversary to distinguish between two nodes in the same supernode. Therefore, each of the  $k$  or more nodes in the same supernode must be equally likely candidates and the probability of any one node being the target is at most  $1/k$ .

We now give a formal proof. Given an input graph  $G$ , there are two key steps to producing a generalized graph: (a) first the nodes of the graph are relabeled, as with naive anonymization; and then (b) the nodes are partitioned into groups. We assume the algorithm that chooses the partition does not depend on the particular labels on the nodes; since it receives a naive anonymization, the labels are arbitrary. Therefore we can commute these two operations without affecting the final output. Without loss of generality, we can assume that the nodes are relabeled after the partition is chosen.

Let  $\Pi : V \rightarrow V_a$  denote the function which relabels nodes. Let  $P$  denote the partition of  $V$  into groups. The output  $\mathcal{G}$  is completely determined by  $G$ ,  $\Pi$ , and  $P$ . For convenience, let  $f$  be the function that takes as input  $G, \Pi, P$  and outputs  $\mathcal{G}$ .

To show graph  $k$ -anonymity, we must show that an adversary cannot use  $\mathcal{G}$  to re-identify a target node  $x$ . Formally, we must show that for any  $x \in V$  and any  $y \in V_a$ ,  $Pr[\Pi(x) = y \mid \mathcal{G}] \leq 1/k$  where the probability comes from the randomness in the algorithm and the adversary's prior belief.

To prove this, we will show that

$$Pr[\Pi(x) = y \mid \mathcal{G}] = Pr[\Pi(x) = y' \mid \mathcal{G}]$$

for any two nodes  $y$  and  $y'$  that are in the same supernode of  $\mathcal{G}$ . Since there at least  $k$  nodes in each supernode, this implies  $Pr[\Pi(x) = y \mid \mathcal{G}] \leq 1/k$  for any  $y$ .

Since the conditional probability  $Pr[\Pi(x) = y|\mathcal{G}] = Pr[\Pi(x) = y, \mathcal{G}]/Pr[\mathcal{G}]$  and the denominator does not depend on  $y$ , it suffices to show that  $Pr[\Pi(x) = y, \mathcal{G}] = Pr[\Pi(x) = y', \mathcal{G}]$ .

We can write  $Pr[\Pi(x) = y, \mathcal{G}]$  as:

$$\begin{aligned}
& Pr[\Pi(x) = y, \mathcal{G}] \\
&= \sum_{\pi: \pi(x)=y} Pr[\Pi = \pi, \mathcal{G}] \\
&= \sum_{\substack{\pi, g, p: \\ \mathcal{G}=f(g, \pi, p) \\ \text{and } \pi(x)=y}} Pr[\Pi = \pi, G = g, P = p] \\
&= \sum_{\substack{\pi, g, p: \\ \mathcal{G}=f(g, \pi, p) \\ \text{and } \pi(x)=y}} Pr[P = p|G = g]Pr[\Pi = \pi]Pr[G = g]
\end{aligned}$$

where  $Pr[P = p|G = g]$  is the probability the algorithm outputs partition  $p$  given the input graph  $g$ ;  $Pr[\Pi = \pi]$  is the probability of a particular relabeling, which is equal to  $1/|V|!$  for any  $\pi$ ; and  $Pr[G = g]$  is the adversary's prior belief that the input graph is  $g$ .

Consider one term in the above summation by fixing the input graph  $g$ , the partition  $p$ , and the map  $\pi$ . Let  $x'$  denote the node that maps to  $y'$  under  $\pi$ , i.e.,  $\pi(x') = y'$ . Construct an alternate mapping  $\pi_{alt}$  such that the mapping for  $x$  and  $x'$  are flipped and all other mappings are unchanged:  $\pi_{alt}(x) = \pi(x')$  and  $\pi_{alt}(x') = \pi(x)$  and  $\pi_{alt}(x'') = \pi(x'')$  for all  $x'' \notin \{x, x'\}$ . There is a corresponding term in the summation for  $Pr[\Pi(x) = y', \mathcal{G}]$  where  $\pi$  is replaced with  $\pi_{alt}$ . Since  $x$  and  $x'$  appear in the same partition, we can permute their relabelings without changing the generalized graph; i.e.,  $f(g, \pi, p) = f(g, \pi_{alt}, p)$ . Since each term in the above summation for  $Pr[\Pi(x) = y, \mathcal{G}]$  can be paired with an equal term in the summation for  $Pr[\Pi(x) = y', \mathcal{G}]$ , then  $Pr[\Pi(x) = y, \mathcal{G}] = Pr[\Pi(x) = y', \mathcal{G}]$  and this completes the proof.  $\square$

Requiring a minimum supernode size of  $k$  only imposes an upper bound on the adversary’s confidence in the true identity of his target. For some graphs and some adversaries, the adversary’s confidence may be much less than  $1/k$ .

For example, consider an adversary who knows only the degree of its target. The candidates for the target include any node such that in some possible world, its degree matches the target’s degree. For each supernode, we can determine a range of degrees such that for each degree in that range and each node in that supernode, there exists a possible world where that node obtains that degree. For supernode  $X$ , the range is determined by mindegree and maxdegree, which are defined as  $\text{mindegree}(X) = \max(0, d(X, X) - \binom{|X|-1}{2}) + \sum_{Y \in \mathcal{V}} \max(0, d(X, Y) - (|X|-1)|Y|)$  and  $\text{maxdegree}(X) = \min(|X| - 1, d(X, X)) + \sum_{Y \in \mathcal{V}} \min(|Y|, d(X, Y))$ .

The degree range of each supernode determines the candidates, however, not all candidates are equally likely. Intuitively, a node is more likely if there are more possible worlds in which its degree matches the target.

In general, it may be computationally hard to determine the adversary’s posterior probability of a candidate being the target. The brute force solution—enumerating all possible worlds and computing candidate set in each one—requires exponential time. We conservatively require  $k$ -sized partitions but observe that in practice this may provide much stronger protection than that implied by the value of  $k$ .

### 4.2.3 Algorithm description

We now present the graph generalization algorithm, which we call GraphGen. The input to the GraphGen is  $G_a$  and privacy parameter  $k$ . The output is a generalized graph  $\mathcal{G}$ . Pseudocode for the algorithm is given in Algorithm 1.

Subject to the privacy constraint, which requires the supernodes of  $\mathcal{G}$  to be of size at least  $k$ , we would like to find the generalized graph that best fits the input graph. We estimate fitness via a maximum likelihood approach. We consider a uniform

---

**Algorithm 1** GraphGen, an algorithm that generalizes a graph to ensure anonymity.

---

**Input:**  $G_a = (V_a, E_a)$ , graph to generalize

$k$ , minimum supernode size

**Output:**  $\mathcal{G}$ , a generalized graph such that each supernode contains at least  $k$  nodes

```

1:  $\mathcal{G} \leftarrow \text{Initialize}(G_a)$  {All nodes in one partition.}
2:  $t_{cycle} \leftarrow 5|V_a|$ 
3: for  $t \leftarrow 1$  to  $\infty$  do
4:    $T \leftarrow \text{Schedule}(t)$  {Temperature  $T$  cools as  $t$  increases.}
5:    $S \leftarrow \text{Successors}(\mathcal{G}, k)$ 
6:    $\mathcal{G}' \leftarrow \arg \max_{\mathcal{G}' \in S} \frac{1}{|\mathcal{W}(\mathcal{G}')|}$  {Find max likelihood successor}
7:    $\Delta L \leftarrow \frac{1}{|\mathcal{W}(\mathcal{G}')|} - \frac{1}{|\mathcal{W}(\mathcal{G})|}$  {Change in likelihood}
8:   if  $\Delta L > 0$  then
9:      $\mathcal{G} \leftarrow \mathcal{G}'$ 
10:  else
11:     $\mathcal{G} \leftarrow \mathcal{G}'$  with probability  $e^{\Delta L/T}$ 
12:  end if
13:  if  $\mathcal{G}$  updated less than 0.02% of last  $t_{cycle}$  steps then
14:    return  $\mathcal{G}$ 
15:  end if
16: end for

```

**Successors** subroutine returns a set of generalized graphs that can be derived from  $\mathcal{G}$  by making a small change, such as splitting or merging a supernode in  $\mathcal{G}$ .

**Input:**  $\mathcal{G}$ , current generalized graph

$k$ , minimum supernode size

**Output:** a set of generalized graphs, the successors to  $\mathcal{G}$

```

1:  $S \leftarrow \emptyset$  {The set of successors}
2:  $u \leftarrow \text{Choose random node}$ 
3:  $X \leftarrow \text{Find supernode that contains } u$ 
4: if  $|X| > 2k$  then
5:    $\mathcal{G}' \leftarrow \text{Split}(X, \mathcal{G})$  {Choose greedy split of  $X$ }
6:    $S \leftarrow S \cup \{\mathcal{G}'\}$ 
7: end if
8: for  $Y$  such that  $X, Y$  are neighbors or share a neighbor do
9:   if  $|X| > k$  then
10:     $\mathcal{G}' \leftarrow \text{MoveNode}(u, X, Y, \mathcal{G})$ 
11:     $S \leftarrow S \cup \{\mathcal{G}'\}$ 
12:   end if
13:    $\mathcal{G}' \leftarrow \text{MergeAndSplit}(X, Y, \mathcal{G})$ 
14:    $S \leftarrow S \cup \{\mathcal{G}'\}$ 
15: end for
16: return  $S$ 

```

---

probability distribution over the possible worlds  $\mathcal{W}(\mathcal{G})$ . For a graph  $g \in \mathcal{W}(\mathcal{G})$  we define  $\Pr_{\mathcal{G}}[g] = 1/|\mathcal{W}(\mathcal{G})|$  where the number of possible worlds is:

$$|\mathcal{W}(\mathcal{G})| = \prod_{X \in \mathcal{V}} \binom{\frac{1}{2}|X|(|X| - 1)}{d(X, X)} \prod_{X, Y \in \mathcal{V}} \binom{|X||Y|}{d(X, Y)}$$

Without regard to the anonymity condition, the generalized graph that maximizes likelihood is the one with each node in a separate partition. Then, as explained above,  $|\mathcal{W}(\mathcal{G})| = 1$  and  $\Pr_{\mathcal{G}}[G_a] = 1$ . In general, likelihood is greater with more supernodes because each supernodes introduces more parameters to fit a fixed amount of data. But subject to the minimum size constraint, generalized graphs can vary greatly in their fit to the input graph. GraphGen uses local search to explore the exponential number of generalized graphs.

The design of the search GraphGen is based on techniques for solving a related social network analysis problem: *stochastic block-modeling* [99]. The objective of stochastic block-modeling is to cluster the nodes of the graph so that nodes in the same group play a similar “social role” in the graph. While the high-level idea is the same, there are a few key distinctions from our work. First, our differing motivations result in different likelihood functions. In stochastic block-modeling, the goal is to build a predictive model of the data and so the likelihood includes a penalty term for model complexity; in contrast, our goal is to fit the original graph as closely as possible given the anonymity condition. Second, the anonymity condition imposes a new constraint on the search space, which makes search more complex.

To find the generalized graph that maximizes the likelihood function, GraphGen searches using simulated annealing [109]. Each valid generalized graph (i.e., those such that each supernode at least  $k$  nodes) is a state in the search space. Starting with a generalized graph that has a single partition (i.e., supernode) containing all nodes, GraphGen proposes a change of state, by splitting a partition, merging two partitions, or moving a node to a different partition. The proposal of changing the

current state from generalized graph  $\mathcal{G}$  to some new generalized graph  $\mathcal{G}'$  is evaluated based on the change in likelihood that results. The proposal is always accepted if it improves the likelihood and accepted with some probability if it decreases the likelihood. The acceptance probability starts high and is cooled slowly until, as it approaches zero, a move is accepted only if it increases the likelihood. We terminate search when fewer than 0.02% of proposals are accepted.

GraphGen may return a partitioning that is only locally maximal. Whether this happens depends in part on the cooling schedule of simulated annealing; if cooled slowly enough, it will return the global maximum with high probability [109]. Nevertheless, finding the globally optimal partition is an intractable problem, and we cannot quantify how close the output is to the optimum. In experimental results not shown, we did a more systematic exploration of the search space using random restarts. On the **Enron** graph with  $k = 3$ , the log-likelihood of the output partition ranged from  $-362.6$  to  $-353.3$ ; in contrast, a greedy algorithm returns a partition with log-likelihood of only  $-511.5$ .

To make search more efficient, we cache the statistics needed to compute likelihood. We maintain a cache of edge counts  $d(X, Y)$  to facilitate computing the likelihood. Furthermore, when considering a move in search space, it is only necessary to compute the change in likelihood, which is more efficient since a move only affects a subset of terms in the likelihood equation. For example, to split supernode  $X$  into  $X'$  and  $X''$ , the only affected terms are the ones involving  $X$ . There is a term for each neighbor  $Y$  of  $X$  (i.e.,  $Y$  such that  $d(X, Y) > 0$ ). Since the input graphs are typically sparse,  $X$  has few neighbors, resulting in only a small number of affected terms. In the worst-case, computing the change in likelihood requires time that is linear in the size of the input graph.

We also made a few design choices that make search more efficient. A supernode is split in a greedy fashion: a randomly chosen node is moved from  $X$  to a new group

$X'$ , and then for each of the next  $k - 1$  nodes, we select the node that maximizes the likelihood when moved from  $X$  to  $X'$ . Second, when we consider merging two supernodes or moving a node between supernodes, we only consider supernodes  $X, Y$  that are neighbors or share a neighbor. This is locally optimal, in that if  $Y$  does not satisfy this condition, then merging  $X$  and  $Y$  can only decrease the likelihood of the current generalized graph. While these choices may exclude the optimal assignment, results indicate that they are effective heuristics: they greatly reduce runtime without any decrease in likelihood.

#### 4.2.4 Capitalizing on limited adversaries

The GraphGen algorithm places each node in a supernode with at least  $k - 1$  other nodes. This is a conservative approach in that it ignores the fact that some nodes may be structurally well-hidden in the original graph. Nodes may be automorphically equivalent, or so similar that only an adversary with substantial structural knowledge can distinguish them.

Such a conservative approach has consequences for utility, as graph structure is coarsened to the supernode level. We would like an approach that can take advantage of situations in which the adversary is known to have limited knowledge of graph structure or where the graphs contain many structurally homogenous nodes.

We propose an extension of GraphGen that anonymizes the graph with respect to a fixed model of adversary knowledge. The idea is to only anonymize nodes that are *vulnerable* to re-identification by the given adversary. By focusing the anonymization on the vulnerable nodes, it may be possible to preserve more of the structure of the input graph.

To incorporate into the algorithm, the first step is to identify the vulnerable nodes. Given adversary model  $Q$  and group size  $k$ , a node  $x$  is vulnerable if  $|\text{cand}_Q(x)| < k$ . For example, if  $Q$  is  $\mathcal{H}_1$ , then the only nodes that are vulnerable are the ones whose

degree occurs less than  $k$  times. Then, the privacy condition on the generalized graph is altered so that the only requirement is that if a supernode contains a vulnerable node, then its size must be at least  $k$ . This means that a invulnerable node can be placed in a supernode of size 1.

This relaxed privacy condition can be incorporated into the search procedure by allowing state changes that place invulnerable nodes into supernodes of size less than  $k$ . Alternatively, the search can execute as described above, and then supernodes that contain only invulnerable nodes can be replaced with individual supernodes for each invulnerable node. (Supernodes containing a mixture of vulnerable and invulnerable nodes must remain intact to ensure that the vulnerable nodes are protected.) In Section 4.3.4, we evaluate the latter approach for the  $\mathcal{H}_1$  and  $\mathcal{H}_2$  adversary models and measure the improvement in utility that results. We refer to these variants of the algorithm as  $\text{GraphGen}(\mathcal{H}_1)$  and  $\text{GraphGen}(\mathcal{H}_2)$  respectively. The pseudocode is shown in Algorithm 2.

---

**Algorithm 2**  $\text{GraphGen}(Q)$  a modification of Algorithm 1 that protects against  $Q$  adversaries.

---

**Input:**  $G_a = (V_a, E_a)$ , graph to generalize

$k$ , minimum supernode size

$Q$  knowledge query representing adversary capability

**Output:**  $\mathcal{G}$ , a generalized graph that satisfies graph  $k$ -anonymity with respect to  $Q$  adversaries.

- 1:  $S \leftarrow \{u \in V_a \mid |\text{cand}_Q(u)| < k\}$  {Vulnerable nodes}
  - 2:  $\mathcal{G} \leftarrow \text{GraphGen}(G_a, k)$   
    {Replace supernodes that contain only invulnerable nodes}
  - 3: **for** supernode  $X$  in  $\mathcal{G}$  **do**
  - 4:     **if**  $X \cap S = \emptyset$  **then**
  - 5:         replace  $X$  with a supernode for each  $u \in X$
  - 6:     **end if**
  - 7: **end for**
  - 8: **return**  $\mathcal{G}$
- 

These alternative anonymization algorithms satisfy graph  $k$ -anonymity, but for restricted adversaries.

**Corollary 1.** *The output of  $\text{GraphGen}(\mathcal{H}_1)$  satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_1$ . Similarly, the output of  $\text{GraphGen}(\mathcal{H}_2)$  satisfies graph  $k$ -anonymity with respect to  $\mathcal{H}_2$ .*

This follows from Proposition 4.1: vulnerable nodes remain in groups of size  $k$  and are therefore protected, and invulnerable nodes are by definition nodes that the adversary cannot re-identify with confidence greater than  $1/k$  and therefore it is not necessary to generalize them.

### 4.3 Evaluating graph anonymization algorithms

We now present an extensive empirical evaluation of the GraphGen algorithm. We evaluate its utility, compare it to competing techniques, and measure the effectiveness of the utility enhancements proposed in Section 4.2.4.

The first goal of our experimental evaluation is to assess the overall utility of anonymized graphs. We would like to quantify the extent to which the anonymized graphs produced by GraphGen (and competing techniques) can serve as an accurate approximation of the original private graph. This is challenging because there are no well-defined metrics to determine the similarity of two graphs. As methods for producing anonymized networks emerge, it is becoming increasingly important to develop a reliable means for assessing their utility.

Our basic approach is to consider a suite of graph properties, measure both the original graph and the anonymized graph and compare the difference. If the anonymized graph differs from the original for some graph property, as it often does, an essential question is whether the difference is substantial. To help answer this question, we include, as a reference point, a random graph of the same size and density as the original graph. With respect to a particular measure, if the original graph looks very different from a random graph, then it is useful to compare the anonymized graph to both the original and the random graph. The more closely the anonymized

graph resembles a random graph, the less useful it is. With the GraphGen approach, as group size  $k$  increases, the anonymized graph converges on a random graph, and we can measure the rate of convergence by varying  $k$ . On the other hand, when the original graph and a random graph appear similar, then the measured property does not distinguish the original from a random graph and thus cannot be used to assess whether anonymization has preserved the structure of the original graph.

As another yardstick for measuring the loss in utility, we evaluate the anonymization algorithms on some carefully chosen combinations of metrics and synthetic graphs. Inspired by research in the networking community [7, 117], we consider a few graphs that have a deliberately engineered structure and then use metrics that capture how well this structure is preserved in the anonymized graph. For instance, we consider a graph that is a tree and measure the extent to which the graph remains tree-like after anonymization. While some of these graphs are unlikely to arise in practice, we find the experiments give useful insights into the effect of anonymization and help distinguish the behavior of competing techniques. It is also important given that real technological networks are often highly structured and poorly approximated by random graphs [78].

The second goal of the experimental evaluation is to compare GraphGen against competing techniques. One challenge is that the privacy guarantees are not always compatible and so an “apples to apples” comparison is not straightforward. We attempt to address these disparities in privacy guarantees by aligning our technique with others so that privacy conditions are comparable (Section 4.3.4), and by assessing the extent to which our approach is vulnerable to attacks (Section 4.3.5). Despite the incompatible privacy semantics in some cases, we believe that comparisons of the algorithms are still useful: their strengths and weaknesses are exposed and their tendency to bias graph measures is revealed.

We note that the goal of publishing an anonymized graph is not only to support the specific graph properties studied here. The hope is that the released dataset can be used for a wide range of investigations determined by graph topology. If measuring a specific graph property is the final objective of an analyst, alternative mechanisms for releasing that property alone should be considered (see discussion of some techniques in Section 7). At any rate, many analyses cannot be distilled into simple graph properties, and analysts often require sample datasets to refine their algorithms or interpret results.

#### 4.3.1 Compared anonymization algorithms

In the first set of experiments, we compare the GraphGen algorithm described in Section 4.2 against two other algorithms for graph anonymization: the algorithm of Cormode et al. [29], denoted BCKS, and the algorithm of Liu and Terzi [82], denoted LT.

The BCKS algorithm is similar to GraphGen in that it partitions nodes into supernodes and outputs a generalized graph. However, in addition to preventing re-identification, the resulting generalized graph is also guaranteed to prevent edge disclosure. The privacy condition ensures that each supernode contains at least  $k$  nodes and that edge disclosure is bounded by  $1/k$ . This is done by requiring that the supernodes satisfy an additional *safety* condition, which states that if two nodes share a neighbor, they must be placed in separate supernodes. The GraphGen algorithm may not prevent edge disclosure, especially at small  $k$  (see Section 4.3.5).

Another important difference is that the BCKS algorithm’s strategy for choosing supernodes is guided by privacy concerns—partitions are chosen to ensure low edge disclosure risk—whereas the strategy of GraphGen is guided by utility. As one might expect, we find that GraphGen achieves higher utility than BCKS.

It should also be mentioned that the approaches proposed by Cormode et al. [29] can handle richer graph data representations, including attributes on nodes and edges and multiple edge types. The focus of the empirical evaluation in [29] is on queries that involve attributes and short path queries. The focus of our study is to measure the effects of anonymization on graph topology.

The LT algorithm alters the graph through the insertion and removal of edges with the goal of making nodes more structurally uniform. The output is a single graph, not a generalized graph. The algorithm alters the graph until each node degree occurs at least  $k$  times. This prevents re-identification by an adversary whose knowledge is limited to node degree (i.e., an  $\mathcal{H}_1$  adversary). It may not protect against a more powerful adversary (e.g., an  $\mathcal{H}_2$  adversary). Given the weaker privacy condition, the LT can achieve better utility than BCKS and GraphGen on some measures.

The LT algorithm anonymizes the graph in a two-stage process. First, it finds the minimum change to the degree sequence such that the privacy condition is satisfied (each degree must appear at least  $k$  times), and the degree sequence can be realized (the sequence of integers must satisfy certain graph theoretic constraints). Then, it attempts to transform the original graph into a new graph that matches this degree sequence.

This second stage is non-trivial and Liu and Terzi [82] consider several alternative algorithms. We implement and compare against SimultaneousSwap. This algorithm is the only one that allows both edge insertions and deletions and appears to perform better than some of the alternative approaches proposed in [82] that only allow edge insertions. It is a greedy algorithm that starts with a canonical graph conforming to the anonymized degree sequence and rewires it in such a way that preserves its degree sequence but increases the edge overlap with the original graph.

### 4.3.2 Overview of experiments

To assess how anonymization impacts utility, we compare the original graph to the anonymized output based on several important graph properties (described below). For each property, we measure it on the original graph and on the anonymized output. For the algorithms that output a single graph, we simply measure the property on the output graph. For the algorithms that output a generalized graph  $\mathcal{G}$ , we estimate the graph property by drawing 100 sample graphs from  $\mathcal{W}(\mathcal{G})$ , measuring the property of each sample, and then aggregating measurements across samples. We report the average and show the standard deviation using error bars. The error bars give a sense of how much variation there is among the graphs in  $\mathcal{W}(\mathcal{G})$ .

If the samples are drawn uniformly from  $\mathcal{W}(\mathcal{G})$ , this models an analyst who believes that each graph in  $\mathcal{W}(\mathcal{G})$  is equiprobable. In these experiments, we perform biased sampling taking samples uniformly from  $\mathcal{W}(\mathcal{G})$  subject to the constraint that the minimum degree is one. This makes it more likely that the sampled graph will contain a large connected component. All of the input graphs contain a single connected component, and we assume this fact is revealed to the analyst.

As a baseline, we also measure the property on a sample of 100 random graphs that are the same density as the original graph. We refer to this baseline as Random. Note this baseline is equivalent to applying a graph generalization algorithm where  $k = |V|$ . It has maximum privacy, but low utility as the only property of the original revealed is the number of nodes and edges.

We repeat this procedure for each graph and each setting of  $k \in \{2, 5, 10, 20\}$ . Note that while  $k$  is a common parameter across the algorithms that controls the size of the group, the resulting privacy is not the same: while GraphGen and BCKS ensure graph  $k$ -anonymity, LT ensures only graph  $k$ -anonymity with respect to degree ( $\mathcal{H}_1$ ).

We report results on the datasets that were described earlier in Section 3.2.

### 4.3.3 Results

We now present a comparison of the algorithms across several different graph metrics. Results are presented one metric at a time. We conclude with a general discussion of the findings in Section 4.3.3.5.

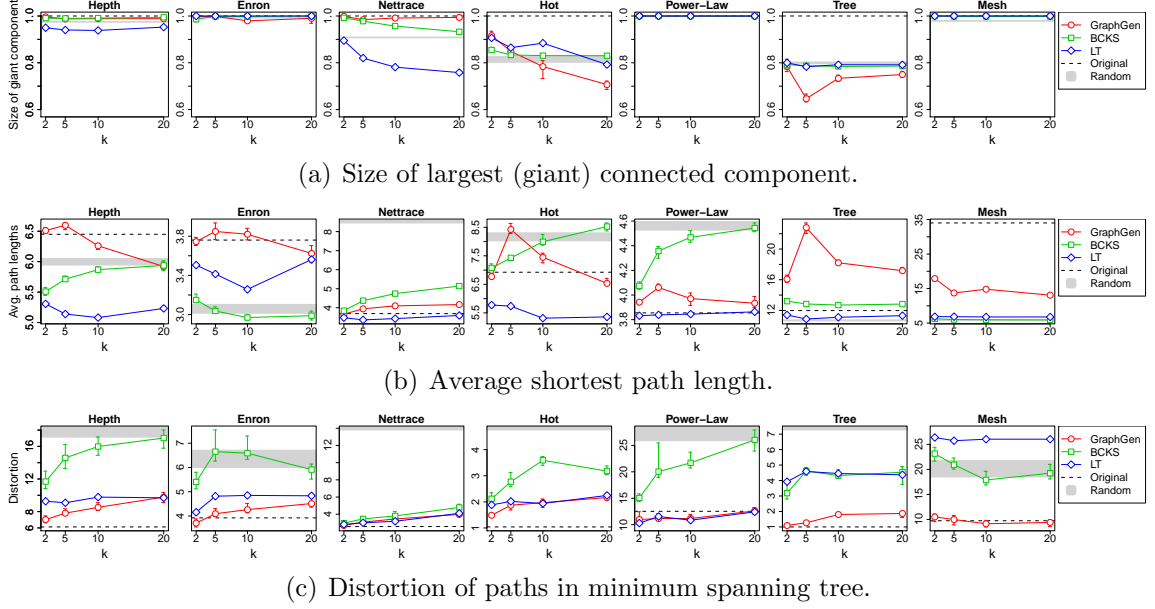
The results of the experiments are shown in Figures 4.1-4.3. Each figure presents the results for a single graph metric. The value of the metric for the true graph is shown as a dashed black line. As a reference point, the light gray region shows the value of the metric for a random graph. It is a region because it depicts a range of  $\pm 1$  standard deviation around the average value over conforming random graphs. Note that for each measure, the scales of the y-axis vary across datasets, so in some cases, while the gap between lines is large, the numerical difference is quite small.

#### 4.3.3.1 Paths

We consider several measures related to paths.

**Connectedness** Each of the anonymization algorithms may alter the connectivity of the graph, either dividing a connected component or merging two components. Each of the input graphs contains a single connected component, so we evaluate whether anonymization divides it. Figure 4.1(a) shows the results. Generally, the anonymized graphs contain a single large component, encompassing about 95% or more of the nodes. However, on the sparsest graphs—**NetTrace**, **HOT**, and **Tree**—the largest connected component of the anonymized graphs can contain as few as 70% of the nodes.

**Shortest Path Lengths** We evaluate how anonymization affects path lengths in the graph. We measure the length of a shortest path between a pair of randomly chosen nodes and compute the average length over 200 random pairs. When the graph contains multiple connected components, we only sample pairs from the largest



**Figure 4.1.** The effect of anonymization on three graph measures related to paths. The results for three algorithms are compared, with varying privacy parameter  $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region.

connected component. Since the measure itself is random, there can be variation due to sampling. We measured this variation and found it small compared to the bias introduced by anonymization and so for presentation purposes we only report the average from a single sample.

Figure 4.1(b) shows the results. The effect of anonymization varies greatly across datasets. The greatest change occurs on **Mesh** where path lengths are dramatically shortened. In fact, for LT and BCKS, path lengths are much closer to those of a random graph than to the original graph. With the GraphGen graphs, while paths are shortened, they remain considerably longer. GraphGen tends to group neighboring nodes together, thus it does not introduce as many shortcut paths that can connect distant regions of the mesh graph.

The distortion of path lengths on **Mesh** is perhaps not too surprising. For highly-structured graphs such as a mesh or a lattice, even a small amount of perturbation can

greatly shorten paths by introducing a few shortcuts paths that can connect distant regions of the mesh with only a few hops [122] and meshes [62].

Generally, across all input graphs, the average path lengths of an BCKS graph appears to converge to those of Random as  $k$  increases. Convergence sometimes occurs at small  $k$  (e.g., **Mesh**, **Enron**, **HepTh**). Convergence occurs whether or not path lengths are shorter or longer in random graphs than with the original.

LT produces graphs with shorter path lengths than the original graph. It is very accurate on some graphs (**NetTrace**, **Power-Law**).

There are no consistent trends for GraphGen. Sometimes paths are shorter, sometimes longer. Increasing  $k$  does not have a consistent effect on path lengths. On some graphs, particularly **Tree**, the path lengths can be considerably longer than on the original graph.

**Tree-like shortest paths** We also include a graph theoretic measure called *distortion*, which in some sense captures how closely a graph resembles a tree [117]. To compute distortion of  $G$ , we first construct a spanning tree  $T$ . Then for each edge  $(u, v)$  in  $G$ , we compute the distance between  $u$  and  $v$  in  $T$ . The distortion is the average distance over all edges in  $G$ . Thus, it measures how path lengths of  $G$  are distorted (i.e., lengthened) if we are restricted to only traversing edges in tree  $T$ . If  $G$  is a tree, then distortion is 1. A random graph has a distortion of approximately  $\log n$ .

Figure 4.1(c) shows the distortion of the anonymized graphs. We focus on **Tree**, because the original graph is in fact a tree and so its distortion is 1. Anonymized graphs have a distortion measure exceeding 1, indicating the anonymized graphs are no longer tree-like. Distortion is high for LT and BCKS across all  $k$ . In fact, the distortion measure of the anonymized graphs is often closer to a random graph than the original tree. For GraphGen graphs, while distortion increases with  $k$ , it is very

low at small  $k$ . Thus, it appears as though GraphGen more accurately preserves the tree-like structure of **Tree**.

In the other graphs, anonymization tends to produce graphs with higher distortion than the original graph. LT performs comparably to GraphGen, except on **Mesh**, where the distortion of GraphGen is much lower and closer to the original graph. On **HOT**, which has low distortion indicating tree-like structure, both GraphGen and LT preserve its tree-like structure at small  $k$ .

#### 4.3.3.2 Degree-related measures

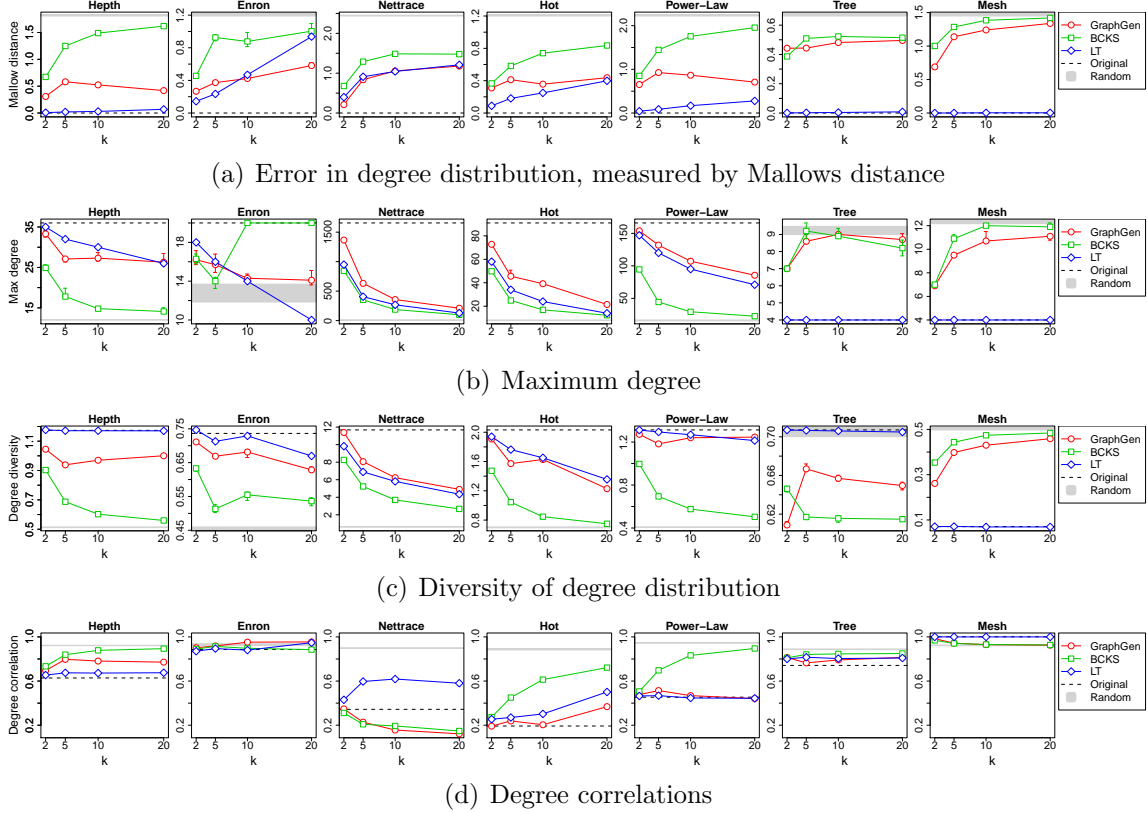
The degree distribution of a graph is an important property of a graph. We look at several different metrics that capture how anonymization affects degree distributions.

**Mallows distance** First, we compare the distributions using Mallows distance, a standard metric for comparing two distributions. Let  $d = d_1, \dots, d_n$  be the degree sequence of the original graph  $G$  where  $d_i$  corresponds to the  $i^{\text{th}}$  largest node degree in  $G$ . Let  $d'$  be the degree sequence of an anonymized graph. Mallows distance (also known as Earth Mover’s distance [75]) is the  $L_p$  distance between the two sequences

$$\text{Mallows}_p(d, d') = \left( \frac{1}{n} \sum_{i=1}^n |d_i - d'_i|^p \right)^{1/p}$$

We use  $p = 1$ . Thus, the Mallows distance captures how much, on average, each node degree is altered by anonymization. E.g., a distance of 1 means each node’s degree is changed on average by  $\pm 1$ .

Figure 4.2(a) shows some trends across datasets and  $k$ . Mallows distance tends to increase with  $k$ , though sometimes inconsistently for GraphGen. The anonymized graphs tend to have lower Mallows distance than Random, indicating that the degree sequence of the anonymized graph preserves some of the “structure” of the original degree sequence.



**Figure 4.2.** The effect of anonymization on four measures related to the degree distribution. Again, the results for three algorithms are compared, with varying privacy parameter  $k$ , on seven different graphs. The value of the given measure on the true graph is shown as a black dotted line. The value of the measure for sampled random graphs matching the density of the original is shown as a gray region.

In comparing algorithms, BCKS performs worse than the other approaches, with Mallows distance rapidly approaching that of Random with increasing  $k$ . LT almost always has the lowest Mallows distance, which is expected given that the LT algorithm explicitly tries to minimize the change to the degree sequence. On graphs where the original graph has nearly uniform degree—**Mesh** and **Tree**—the LT alters the degree sequence only slightly to satisfy its privacy condition, resulting in a Mallows distance of zero or near zero on these graphs. GraphGen is typically between LT and BCKS.

**Maximum degree** Figure 4.2(b) compares the maximum degree of the original graph with the maximum degree in the anonymized graph. The figure shows a clear

trend: as  $k$  increases, the maximum degree of each anonymized graph converges to the maximum degree of Random. On **Mesh** and **Tree**, the max degree is higher in Random and the max degree of anonymized graphs increase (except for LT which stays constant). For the other graphs, the max degree of Random is lower than that of the original, sometimes *much* lower. For example, on **NetTrace**, the maximum degree is 1656 but Random has a max degree of around 10. For all approaches, anonymization reduces the max degree by more than half at  $k = 5$ . When the maximum degree is an outlier, such distortion is in some sense inevitable given the privacy condition: each node degree must be homogenous with at least  $k - 1$  other node degree. Nevertheless, such a significant change in degree suggests that the graph structure has been significantly altered.

While all approaches converge to Random, their rates of convergence differ. The max degree of BCKS changes the most rapidly with  $k$ . Surprisingly, on the graphs where the maximum degree is larger than that of a random graph, the max degree of GraphGen decreases less rapidly than LT.

**Degree variability** In addition to measuring the maximum degree, we also measure the variation in the degree distribution. The coefficient of variation  $C_V(d)$  measures the diversity of degree distribution  $d$ . It is defined as  $C_V(d) = \sigma(d)/\langle d \rangle$  where  $\langle d \rangle$  is the average degree and  $\sigma(d) = \sum_{i=1}^n (d_i - \langle d \rangle)^2 / (n - 1)$ . Graphs with homogenous degrees have low  $C_V$  and graphs with diverse degree sequences, such as power-law graphs, have high  $C_V$  [7].

Figure 4.2(c) shows that, like maximum degree, the  $C_V$  of anonymized graphs converges towards random graphs as  $k$  increases, except on **Power-Law**, where diversity remains high at  $k = 20$ . The comparison between algorithms is similar as it is with max degree.

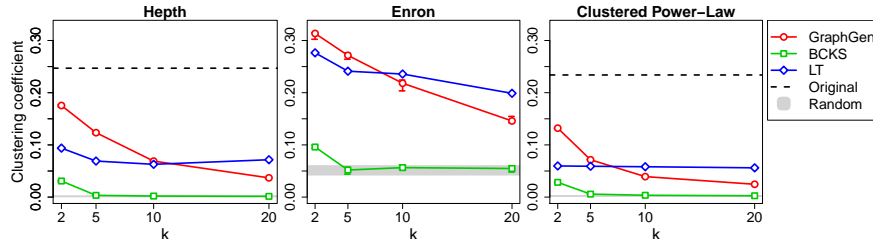
**Degree correlations** We also measure degree correlations—i.e., the correlation between a node’s degree and the degrees of its neighbors. It is an important property that influences processes on networks [32]. We measure correlations using the  $s$  metric. For graph  $G = (V, E)$  it is defined as  $s(G) = \sum_{(u,v) \in E} d(u)d(v)$  where  $d(u)$  is the degree of node  $u$ . A high  $s(G)$  indicates that high degree nodes are connected to one another. We report a normalized  $s$  measure  $s(G)/s_{max}(G)$  where  $s_{max}(G)$  is the maximum possible  $s$  of any graph with the same degree sequence as  $G$ . (In practice, it is computationally intensive to find the true maximum, so we approximate it with the Havel-Hakimi graph [16], which is efficient to construct and tends to have very high  $s$ .)

This measure is particularly interesting on the **HOT** graph. The **HOT** graph is explicitly engineered so that high degree nodes are at the periphery of the graph connected to low degree nodes, resulting in a low  $s(G)$  measure. In contrast, in a random graph, high degree nodes are likely to be connected to each other, resulting in a high  $s$  measure [7].

Figure 4.2(d) shows that in the anonymized version of **HOT**, increasing  $k$  results in an increased  $s$  measure. GraphGen preserves the low  $s$  measure better than LT and substantially better than BCKS. On the other graphs, the performance varies considerably, with correlations sometimes tending to Random (e.g., **HepTh**), sometimes diverging from it (e.g., **NetTrace**), and sometimes remaining constant (e.g., **Tree**).

### 4.3.3.3 Clustering

Clustering coefficient measures the likelihood that two neighbors of a node are themselves connected (in a social network, whether a friend of a friend is also a friend). It is defined as  $C(G) = \frac{1}{n} \sum_u \frac{\Delta(u)}{(d(u)(d(u)-1))/2}$  where  $\Delta(u)$  is the number of triangles (cliques of size 3) containing  $u$  and  $d(u)$  is the degree of  $u$ .



**Figure 4.3.** The effect of anonymization on clustering coefficient.

We report on graphs that have substantial clustering ( $C(G) > 0.15$ ). For the graphs where clustering coefficient is low, the anonymization tends to preserve the low clustering coefficient (they never exceeded 0.15). The graphs with high clustering include **Enron** and **HepTh**. We also include a synthetic graph, **Clustered Power-Law**, which is similar to **Power-Law** except that the random graph generation process is biased to introduce triangles [57]. We set the probability of triangle formation to be 0.4.

Figure 4.3 shows how anonymization reduces the clustering coefficient of clustered graphs. Even at  $k = 2$ , the BCKS has substantially lower clustering coefficient than the original graph. At larger  $k$ , all anonymized graphs have substantially reduced clustering. At small  $k$ , GraphGen preserves the greatest amount of clustering.

With GraphGen, it is difficult to preserve clustering coefficient, especially at large  $k$ . The process of randomly sampling from  $\mathcal{W}(\mathcal{G})$  tends to destroy clustering coefficient. The sampled structure within each supernode is simply a random graph with a density determined by the weight of the supernode’s self edge. Unless they are very dense, random graphs have low clustering coefficient. Real world graphs, are typically very sparse, and so as  $k$  increases the density within a supernode decreases.

#### 4.3.3.4 Runtime

We also measure the runtime of the different algorithms. We report results on one of the largest graphs, **NetTrace**; runtimes on the other graphs are qualitatively

**Table 4.1.** On **NetTrace**, a comparison of runtimes (seconds).

Algorithm	$k = 2$	$k = 5$	$k = 10$	$k = 20$
BCKS	0.2	0.3	0.2	0.2
LT	43.4	29.6	74.2	52.7
GraphGen	3628.8	3171.9	15311.1	1560.1

similar. While GraphGen is considerably slower than the alternative algorithms, runtime is a secondary concern as the algorithms are run “offline” by the data owner.

Table 4.1 shows that the runtime of BCKS does not depend on group size, agreeing with previous theoretical analysis [29]. The runtime of the LT algorithm varies across  $k$ : its runtime is dominated by the graph construction process, which depends on the number of rewiring iterations, something that varies considerably depending on the particular instance, leading to variation in runtime. Finally, the runtime of GraphGen appears to decrease with  $k$ . This is due to the fact that when groups are large, the supergraph is comparably more sparse. Therefore, the number of the successors (see Algorithm 1) is smaller, and so each step in the search runs faster.

#### 4.3.3.5 Discussion

The experiments give insight into how the topological properties of graphs are affected by anonymization. Path lengths tend to more closely resemble path lengths in a random graph, whether they are shorter or longer than the original graph. Highly variable degree distributions (as occurs in power-law graphs) tend to become more uniform and high degree nodes have their degrees reduced. In graphs that are highly clustered, the effect of anonymization is to substantially reduce the clustering coefficient. However, the results also show that it is possible to provide privacy and still preserve some aspects of the original graph.

For graphs with a deliberately engineered structure (such as **Mesh**, **Tree**, and also **HOT**), anonymization can introduce significant distortion. The GraphGen algorithm, because it explicitly accounts for structure in its anonymization, preserves

these qualities relatively well. For example, paths remain long in **Mesh**, **Tree** remains tree-like, and degree correlation of **HOT** remains low.

In terms of comparing the different algorithms, we find that LT and GraphGen perform consistently better than BCKS. While LT clearly has an advantage over GraphGen on some metrics, the performance of GraphGen is often comparable and sometimes better than the performance of LT. In the next section, we resolve the difference in the privacy standards between these algorithms and present an more apples-to-apples comparison.

Recall that the error bars around the measures for GraphGen and BCKS measure the variability across samples from the  $\mathcal{W}(\mathcal{G})$ . Since the original graph  $G$  is a member  $\mathcal{W}(\mathcal{G})$ , one might expect that the error bars would overlap the measure recorded on  $G$ . This does not always occur, suggesting that while  $G$  is a possible world that is consistent with  $\mathcal{G}$ , it is unlikely to be sampled by chance. It may be possible to bias the sampling to make  $G$  more likely, but it is not clear how this impacts privacy.

As mentioned earlier, the GraphGen and BCKS approaches differ in how the generalized graph is constructed; in GraphGen it is guided by utility concerns and in BCKS it is guided by privacy concerns. The edge safety condition of BCKS requires two neighbors of a node to be placed into separate supernodes. However, the GraphGen often places a node’s neighbors together and it appears to lead to better utility. It may be that the edge safety condition, while it ensures that the output does not allow edge disclosures, may conflict with some of the utility metrics considered here.

#### 4.3.4 Utility of enhanced graph generalization algorithm

In this section, we evaluate the proposed enhancements to GraphGen described in Section 4.2.4. By focusing the anonymization only on the nodes that are vulnerable to re-identification, we hypothesize that we can improve the utility of GraphGen, which conservatively generalizes all nodes. We compare GraphGen against two alternatives:

GraphGen( $\mathcal{H}_1$ ) which guards against  $\mathcal{H}_1$  adversaries, and GraphGen( $\mathcal{H}_2$ ) which protects against the stronger  $\mathcal{H}_2$  adversary. Since GraphGen( $\mathcal{H}_1$ ) provides the same privacy guarantee as LT, we also include a direct comparison of those approaches.

Based on our earlier assessment in Section 3.2, we expect that GraphGen( $\mathcal{H}_1$ ) will alter the input graph much less than GraphGen, as most nodes are naturally well-hidden against an  $\mathcal{H}_1$  adversary. For GraphGen( $\mathcal{H}_2$ ), it will depend on the dataset. Many nodes are vulnerable in **HepTh** and almost all nodes are vulnerable in **Enron** and **Power-Law**, so we may not expect much improvement on those datasets. For the other datasets, many nodes are well hidden at  $\mathcal{H}_2$  and so GraphGen( $\mathcal{H}_2$ ) may generalize these graphs much less than GraphGen.

We summarize the performance difference between GraphGen and its variants using a suitably normalized measure of each of the properties described in in Section 4.3.3. We normalize each measure in terms of the distance between between GraphGen and the original graph  $G$ . Let  $P$  denote a graph property and  $P(g)$  denote the evaluation of  $P$  on graph  $g$ . The normalized score of anonymized graph  $A$  is defined as  $\frac{|P(A)-P(G)|}{|P(\text{GraphGen})-P(G)|}$ . A score of less than 1 indicates that algorithm  $A$  preserves the property more accurately than GraphGen.

Since GraphGen( $\mathcal{H}_1$ ) and GraphGen( $\mathcal{H}_2$ ) guard against weaker adversaries than GraphGen, the expectation is that the normalized score will be closer to zero, indicating closer agreement with the original graph.

Table 4.2 shows the results for GraphGen( $\mathcal{H}_1$ ). The results show in general that by targeting the anonymization to protect against  $\mathcal{H}_1$  adversaries, it is possible to improve utility. The magnitude of the improvement is not consistent across datasets, with datasets such **Tree** and **Mesh** seeing large gains and **Enron** seeing relatively small gains. Sometimes utility degrades (a normalized score exceeding one). Generally this is when the original GraphGen algorithm is a very accurate approximation of the original graph (e.g., distortion on **Mesh**), so the denominator of the normalized

**Table 4.2.** A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and GraphGen at  $k = 10$ . Numbers are normalized scores where less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than GraphGen.

Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Giant comp. size	0.014	0.348	0.19	0.695	0	0.333	0.086
Avg. path lengths	1.369	6.174	1.016	0.919	0.002	0.431	0.046
Distortion	0.648	0.973	0.972	0.624	0.665	0.006	0.483
Mallows distance	0.54	0.959	0.946	0.653	0.256	0.005	0.009
Max. degree	1.023	0.982	1.001	0.994	0.997	0.349	0.398
Degree diversity	0.584	1.015	1	0.95	0.911	0.006	0.034
Degree correlation	0.712	0.972	1.039	1.013	0.403	0.065	0.015
Clustering coeff.	0.508	0.869	0.223	0.318	0.954	0.002	0.023

**Table 4.3.** A comparison of utility of GraphGen( $\mathcal{H}_2$ ) and GraphGen at  $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_2$ ) is more accurate than GraphGen.

Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Giant comp. size	0.921	1.03	0.926	0.923	1.079	0.772	0.086
Avg. path lengths	0.947	0.793	1.031	0.818	1.098	0.671	0.046
Distortion	0.964	1.31	1.02	0.822	1.24	0.018	0.483
Mallows distance	0.996	1.005	0.996	0.841	0.998	0.014	0.009
Max. degree	1.018	0.998	0.999	0.983	1.004	0.481	0.398
Degree diversity	0.997	0.999	1	0.973	1.002	0.004	0.034
Degree correlation	1.004	1.018	1.002	1.013	0.97	0.075	0.015
Clustering coeff.	0.995	1.009	0.93	0.643	0.926	0.006	0.023

measure is small. Table 4.3 shows that utility improves with GraphGen( $\mathcal{H}_2$ ), but the improvement is much less than with GraphGen( $\mathcal{H}_1$ ).

**Comparison between GraphGen( $\mathcal{H}_1$ ) and LT** While the utility of LT was compared against BCKS and GraphGen in Section 4.3.3, these algorithms are not directly comparable in terms of their privacy guarantees because LT places restrictions on the adversary’s knowledge. However, we can directly compare LT with GraphGen( $\mathcal{H}_1$ ) because they both provide equal privacy protection.

**Table 4.4.** A comparison of utility of GraphGen( $\mathcal{H}_1$ ) and LT at  $k = 10$ . Numbers are normalized scores where a number less than 1 indicates GraphGen( $\mathcal{H}_1$ ) is more accurate than LT. (A dash indicates that LT perfectly matched the original, so the normalized score is undefined. A 0\* indicates that both LT and GraphGen( $\mathcal{H}_1$ ) perfectly matched the original.)

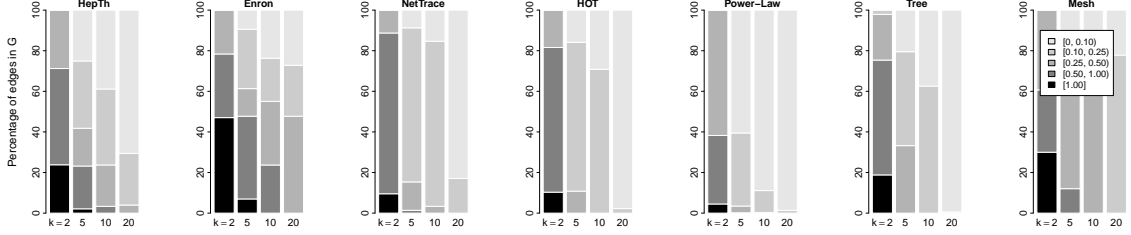
Statistic	HepTh	Enron	NetTrace	HOT	Power-Law	Tree	Mesh
Giant comp. size	0.003	-	0.007	1.195	0*	0.423	0*
Avg. path lengths	0.242	0.151	1.484	0.296	0.016	3.438	0.032
Distortion	0.473	0.55	1.425	0.661	0.84	0.002	0.039
Mallows distance	9.555	0.872	0.958	0.934	1.323	0.643	3.465
Max. degree	1.48	0.92	0.937	0.775	0.807	-	-
Degree diversity	46.66	8.311	0.928	1.016	1.482	0.273	8.151
Degree correlation	2.483	12.204	0.71	0.112	0.803	0.051	0.871
Clustering coeff.	0.492	1.025	0.002	1.684	0.579	0.138	0.556

Table 4.4 compares LT and GraphGen( $\mathcal{H}_1$ ) using a measure which is normalized to LT. Thus a score less than 1 indicates that GraphGen( $\mathcal{H}_1$ ) more accurately approximates the original graph, and a score exceeding 1 indicates that LT is more accurate. (A dash indicates that LT matches the original, so the normalized score is undefined; and a 0\* indicates that both LT and GraphGen( $\mathcal{H}_1$ ) perfectly match the original.) The results suggest that the approaches perform somewhat comparably. There is only one measure (distortion) in which one algorithm is consistently more accurate across the datasets, and there is no dataset where one algorithm is consistently more accurate.

#### 4.3.5 Assessing edge disclosure in generalized graphs

Recall our assessment (Section 3.2.2) of edge disclosure under naive anonymization, which showed that it is possible for a knowledgeable adversary to accurately determine whether two nodes are connected. We revisit edge disclosure here, measuring the extent to which graph generalization reduces the risk of edge disclosure.

While graph generalization prevents re-identification (Section 4.2.2), edge disclosure may still be possible. For example, if an adversary can determine which supern-



**Figure 4.4.** Risk of edge disclosure in generalized graphs across different datasets and settings of  $k$ .

ode contains Alice and which supernode contains Bob he can estimate the likelihood of an edge between Alice and Bob based on the weight of the superedge between their respective supernodes. The weight reveals the number of edges in the original graph between the nodes in Alice’s supernode and the nodes in Bob’s supernode. A higher weight increases the likelihood they are connected.

To assess the risk of edge disclosure, we conservatively assume that the adversary can successfully identify the supernode of each target node. In practice, we expect that this will be difficult for an adversary with limited knowledge, so our results may exaggerate the risk. Given two target nodes  $u$  and  $v$  in  $G$ , the adversary computes the likelihood of edge between  $u$  and  $v$  by first identifying their supernodes in  $\mathcal{G}$ , denoted  $X$  and  $Y$  respectively, and then observing the superedge weight,  $d(X, Y)$ . The likelihood of edge  $(u, v)$  is  $d(X, Y)/|X||Y|$  or, in the case when  $X = Y$ —i.e., the targets share a supernode—the edge likelihood is  $2d(X, X)/|X|(|X| - 1)$ .

Our experiment is as follows. Given a graph  $G$  and a setting of  $k$ , we produce a generalized graph  $\mathcal{G}_k$  and measure its edge disclosure risk. For each edge in the original graph  $G$ , we measure its likelihood given  $\mathcal{G}_k$ . Each edge likelihood  $\ell$  is a number in  $[0, 1]$  which we discretize into five categories from “low” ( $\ell \in [0, 0.10)$ ) to “high” ( $\ell = 1.0$ ). We report the percentage of edges in each category. This is similar to the experiments in Section 3.2.2 except rather than vary adversary knowledge, we assume a powerful adversary who knows the mapping of nodes to supernodes.

Figure 4.4 shows the results across several input graphs and settings of  $k$ . (Note the grayscale used here differs from the one used in Figure 3.4.) The results show that when  $k = 2$ , some edges are disclosed in all datasets. This is not surprising because at  $k = 2$ , whenever two neighbors are placed into the same supernode, the edge between them is disclosed—the weight of the self-superedge is either 1 (if they are connected) or 0 (if they are not).

At  $k = 5$ , a small portion of edges is disclosed in two graphs, **HepTh** (2.1%) and **Enron** (6.9%), but for the other graphs no edges are disclosed. Overall, edge disclosure diminishes rapidly with increasing  $k$ . By  $k = 20$ , edge likelihoods are less than half across all graphs.

The experiments show that for reasonable settings of  $k$ , the process of graph generalization greatly reduces the threat of edge disclosure. Our assessment is conservative and may overstate the threat. To prevent disclosure even at small  $k$ , one must explicitly place neighboring nodes in separate supernodes. This is done in the BCKS algorithm, which uses a safety condition to ensure that superedge weights are bounded by  $1/k$ . However, this additional safety condition has considerable cost in utility as shown in Section 4.3.3.

## 4.4 Conclusion

We proposed anonymizing a graph by generalizing it: partitioning the nodes and summarizing the graph at the partition level. This approach is shown to satisfy graph  $k$ -anonymity under any structural query. We show that a wide range of important graph analyses can be performed accurately on the generalized graphs published. An important area for future investigation is to develop bounds on the distortion introduced by anonymization. Analytical bounds could be developed through analysis of the generalized graphs, or empirical bounds could be inferred through careful sampling of the possible worlds implied by the generalized graphs. We also hope to investigate

techniques that will safely permit the analyst to sample higher quality representatives from the set of possible worlds, for example, by biasing sampling towards the true graph.

## CHAPTER 5

### ACCURATE ESTIMATION OF THE DEGREE SEQUENCE UNDER DIFFERENTIAL PRIVACY

The previous chapters explored transformed data release as a paradigm for sharing sensitive network data in a way that protects privacy. In this chapter, we consider an alternative paradigm: query answer perturbation. In this setting, the analyst poses queries and receives noisy answers. To protect privacy, the noise must be large enough to hide the contributions of an individual’s private data, however, the hope is that for statistics of interest, the scale of the noise is small relative to the scale of the statistic, and the noisy answer is an accurate approximation of the truth. We explore whether this paradigm is a viable solution for computing common network statistics.

Compared with transformed data release, the obvious disadvantage of query answer perturbation is that a graph is never published. Instead, the analyst only receives answers to some queries. However, with transformed data release the analyst receives a *transformed* graph, and as we saw in the previous chapter, the transformations can distort important properties of the graph. For instance, we saw that transformations often diminished the degree of high degree nodes, a bias that could cause analysts to underestimate node centrality. The advantage of query answer perturbation is the potential to get accurate and unbiased answers by tailoring the perturbation techniques to the specific statistics of interest.

This chapter and the one that follows investigate whether it is possible to accurately compute network statistics under rigorous guarantees of privacy.

In this chapter, we focus on a specific utility goal—estimating the degree sequence of a graph. The degree sequence of a graph is a monotonic non-decreasing

sequence of the degrees of its nodes. A directed graph has two degree sequences: an out-degree sequence and an in-degree sequence. For example, for the graph in Figure 5.2(a), its out-degree sequence is  $\langle 0, 1, 1, 1, 2, 2, 3 \rangle$  and its in-degree sequence is  $\langle 0, 0, 1, 1, 1, 1, 2, 4 \rangle$ .

We choose to focus on the degree sequence because it is one of the most widely studied properties of a graph. It influences the structure of a graph and processes that operate on a graph, and a diverse line of research has studied properties of random ensembles of graphs consistent with a known degree sequence [78, 87, 101].

The simple strategy of releasing the exact degree sequence fails to provide adequate privacy protection. Some graphs have unique degree sequences (i.e., all graphs matching this degree sequence are isomorphic) making the release of the degree sequence no safer than naive anonymization. In general, it is unclear how to determine what the degree sequence reveals about the structure of the graph. The problem is compounded when either the adversary has partial knowledge of graph structure, or the degree sequence is only one of several statistics published. Our goal is to design an approach that provides robust privacy protection against powerful adversaries and is compatible with releasing multiple statistics.

Prior work in *differential privacy* provides an excellent foundation for exploring this question. Differential privacy protects against any adversary, even one with nearly complete knowledge of the private data. While originally described in settings where an individual's private data is encapsulated in a single record (such as the tabular setting), it is adaptable to the graph setting (Section 2.2). It also composes well: one can release multiple statistics under differential privacy, so long as the algorithm for each statistic satisfies differential privacy. Thus, a differentially private algorithm for the degree sequence can be combined with differentially private algorithms for other statistics into a single privacy framework. While existing differentially private algorithms, such as the Laplace mechanism (Section 2.2), can be easily adapted to

obtain noisy answers to queries about the degree sequence, the added noise introduces considerable error.

The main contribution of this chapter is an algorithm for accurately estimating the degree sequence of a network in a way that protects the privacy of network participants. The algorithm has several desirable properties:

- **PRIVACY** The algorithm satisfies  $\epsilon$ -differential privacy, which means that the output of the algorithm does not disclose the presence or absence of an edge, even when the adversary has nearly complete knowledge of the graph. The proof of privacy is spread across Sections 5.2 & 5.3 as our technique has multiple components.
- **ACCURACY** Using experiments on real data, we show that the technique is extremely accurate and orders of magnitude more accurate than existing techniques (Section 5.5). In theoretical analysis, we prove that error scales with number of distinct degrees whereas existing techniques have error that scales with the number of nodes (Section 5.3).
- **SCALABILITY** We analyze the complexity of the algorithm in Section 5.4 and show the algorithm runs in linear time. In experiments, we show the algorithm is fast and can scale to large inputs: it computes a private estimate of a 200 million node degree sequence in under 6 seconds.
- **CONFIGURABILITY** As described in Section 5.2, the algorithm can be instantiated in two different ways and the choice impacts how noise is introduced into the degree sequence: experiments show that one approach tends to add relatively more noise to low degrees; the other adds relatively more noise to high degrees. The analyst can choose among the approaches depending on which part of the sequence is most significant. More importantly, we believe this is an

interesting finding that leaves open questions for future work on whether the two strategies can be combined to have lower error throughout the sequence.

- **INNOVATION** A key component of our approach is using statistical inference to filter out some of the noise that was added for privacy. While prior work in differential privacy has considered the idea of post-processing the noisy answers, our work is (to the best of our knowledge) unique in that this is the first instance where post-processing is shown to improve accuracy. The boost in accuracy depends on properties of the input sequence, and we quantify the conditions that lead to high accuracy results (Section 5.3).

A second contribution of this work is to recognize that our technique has broader applicability: a degree sequence is an instance of a more general measurement, which we call an *unattributed histogram*. We define unattributed histograms in the next section, but for now it suffices to say that while they are less informative than a conventional histogram—because they hide some information, namely the association between bin and frequency—they are significant in the context of data privacy because we will show that we can estimate unattributed histograms much more accurately than conventional histograms under differential privacy. Importantly, unattributed histograms have many useful applications, including, but not limited to, measuring the degree sequence of a graph.

We present our algorithm in this more general setting of estimating an unattributed histogram under differential privacy. (Some issues arise for the special case of estimating degree sequences, which we highlight.) In the next section, we define unattributed histograms, give examples, and contrast them with conventional histograms. We also give an overview of our algorithm, conveying the main ideas behind our approach.

Admittedly, the degree sequence is just one property of a graph, and there is evidence that a number of other properties are not constrained by the degree sequence

alone [78, 87]. Nevertheless, because of the degree sequence’s profound influence on the structure of the graph, we believe it is important to know how accurately it can be estimated, independent of other properties. In the following chapter, we look at other network statistics, review known results, including some additional work of our own, and discuss limitations.

## 5.1 Overview of the task and solution

We start with an overview of the problem—computing an unattributed histogram under differential privacy—and our solution. An unattributed histogram is a new concept, so we describe it, give examples, and discuss its practical application. Our solution uses an innovative technique in differential privacy: it involves post processing the output of a differentially private mechanism to obtain a more accurate answer. A key component of our solution is choosing a query strategy where constraints hold among the answers. The constraints are what drives the post-processing phase.

### 5.1.1 Task: computing unattributed histograms

We introduce the concept of an unattributed histogram by contrasting it with a conventional (attributed) histogram.

A histogram on attribute  $A$  in relational schema  $R(A, B, \dots)$  summarizes the distribution of values of  $A$  occurring in  $R$ . We assume the domain of  $A$ ,  $dom$ , is ordered. A histogram is computed with respect to a partition of  $dom$  into disjoint intervals, called *bins*. For each bin, the histogram reports the number of tuples in  $R$  whose value of  $A$  falls in the interval specified by the bin. In examples, we partition the domain into unit-length intervals, i.e., each bin contains only a single value from the domain.

For example, Figure 5.1(a) shows a table of fictitious medical records having the schema  $R(Name, Age, Gender, Condition)$ . We compute a histogram on the attribute

Name	Age	Sex	Condition
Alice	23	F	Healthy
Bob	45	M	Asthma
Carol	59	F	Flu
Dave	25	M	Flu
Ed	70	M	Asthma

(a) Relation  $R$

Condition	Frequency
Asthma	2
Cancer	0
Flu	2
Healthy	1

(b) Conventional histogram

Frequency
0
1
2
2

(c) Unattributed histogram

Query Definitions:  $\mathbf{C} : \langle c([x_1]), c([x_2]), \dots, c([x_n]) \rangle$  for  $x_i \in dom$   
 $\mathbf{S} : \langle c([x_{\pi(1)}]), c([x_{\pi(2)}]), \dots, c([x_{\pi(n)}]) \rangle$   
 $\mathbf{F} : \langle f_1, \dots, f_N \rangle$  where  $f_i = \sum_{i=1}^n \mathbb{I}[c([x_i, y_i]) \geq k]$

<i>True answer</i>	<i>Private output</i>	<i>Inferred answer</i>
$\mathbf{C}(I) = \langle 2, 0, 2, 1 \rangle$	$\tilde{\mathbf{C}}(I) = \langle 1.9, -0.3, 2.1, 0.8 \rangle$	
$\mathbf{S}(I) = \langle 0, 1, 2, 2 \rangle$	$\tilde{\mathbf{S}}(I) = \langle -0.1, 1.1, 2.2, 1.8 \rangle$	$\bar{\mathbf{S}}(I) = \langle -0.1, 1.1, 2.0, 2.0 \rangle$
$\mathbf{F}(I) = \langle 3, 2, 0, 0, 0 \rangle$	$\tilde{\mathbf{F}}(I) = \langle 3.1, 2.1, 0.1, 0.1, 0.4 \rangle$	$\bar{\mathbf{F}}(I) = \langle 3.1, 2.1, 0.2, 0.2, 0.2 \rangle$

(d) Query strategies

**Figure 5.1.** (a) Example table  $R$  on which we compute histograms; (b) A conventional histogram on attribute  $Condition$ , whose domain is  $dom = \{Asthma, Cancer, Flu, Healthy\}$ ; (c) An unattributed histogram on  $Condition$ ; (d) Definitions and sample values for alternative query sequences:  $\mathbf{C}$  computes a conventional histogram,  $\mathbf{S}$  returns the frequencies of the histogram in rank order,  $\mathbf{F}$  computes a cumulative histogram on the frequencies.

*Condition*. The domain of  $Condition$  is  $\{Asthma, Cancer, Flu, Healthy\}$  and ordered lexicographically. A conventional histogram on  $Condition$  is shown in Figure 5.1(b).

In contrast to a conventional histogram, an *unattributed histogram* reports only the frequencies, and omits the association between bin and frequency. For example, Figure 5.1(c) shows an unattributed histogram on the  $Condition$  attribute. This result reveals the frequencies but hides which condition goes with which frequency. For instance, it shows that some condition never occurs (frequency is zero) but not which condition this is. We call it an unattributed histogram because the frequencies are not attributed to their values. The result of an unattributed histogram is a

multiset of frequencies, which can be returned in any order. We adopt a convention of reporting the frequencies in ascending order.

We can further contrast conventional and unattributed histograms by comparing their expression in SQL. Both histograms can be expressed as `GROUP BY` queries in SQL, the only differences being the column name selection and the order of results. The following is an SQL expression for a conventional histogram that returns results ordered by bin:

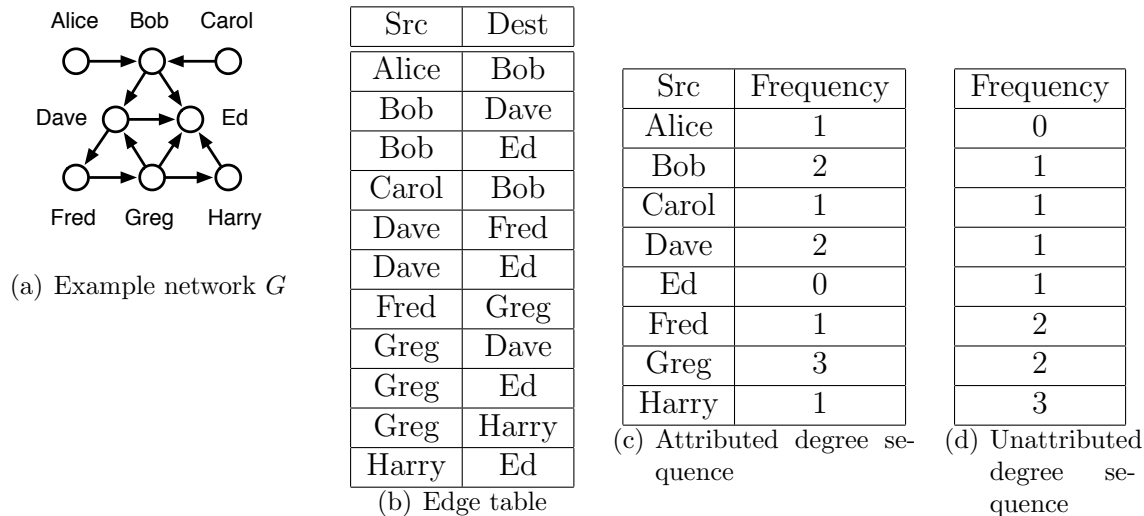
```
SELECT A, COUNT(*) AS Frequency FROM R GROUP BY A ORDER BY A
```

The expression for an unattributed histogram omits  $A$  from the results and orders the results by frequency:

```
SELECT COUNT(*) AS Frequency FROM R GROUP BY A ORDER BY Frequency
```

One can think of an unattributed histogram as a projection on the result of a conventional histogram that retains only the frequency column, removing the bin column from the result. A technical detail: the above SQL expressions compute histograms over the active domain of  $A$ —values from the domain of  $A$  that do not appear in  $R$  are omitted from the results. Our differentially private techniques will compute unattributed histograms over  $dom$  and therefore bins with a frequency of zero are included in the result.

While unattributed histograms may be unconventional, they have practical application. There are some settings where the purpose of computing a histogram is not to learn the frequency of a particular bin or bins, but to simply learn the overall distribution of frequencies. For example, if the tuples of  $R$  represent queries submitted to a search engine, and  $A$  is the search term, then an unattributed histogram shows



**Figure 5.2.** (a) Example graph  $G$  for which we compute degree sequences; (b) The edges of  $G$  represented as a relation  $R(\text{Src}, \text{Dest})$ ; (c) An attributed degree sequence; (d) An unattributed degree sequence.

the frequency of occurrence of all terms (but not the terms themselves). This can be used, for instance, in predicting cache performance.

Another important application, especially in the context of this thesis, is measuring the degree sequence of a graph. Given an undirected graph  $G$ , the degree sequence is a monotonic non-decreasing sequence of the degrees of its nodes. A directed graph has two degree sequences, a sequence of in-degrees and a sequence of out-degrees. For example, for the graph in Figure 5.2(a), its out-degree sequence is  $\langle 0, 1, 1, 1, 2, 2, 3 \rangle$  and its in-degree sequence is  $\langle 0, 0, 1, 1, 1, 1, 2, 4 \rangle$ .

**Degree sequences as unattributed histograms** Degree sequences can be viewed as instances of unattributed histograms. To compute a degree sequence as an unattributed histogram, we first represent the edges of a (directed) graph as a binary relation  $R(\text{Src}, \text{Dest})$ . For example, the edges of the graph in Figure 5.2(a) are shown as a relation in Figure 5.2(b). The out-degree sequence is an unattributed histogram on  $\text{Src}$ ; the in-degree sequence is an unattributed histogram on  $\text{Dest}$ .

For example, the out-degree sequence of the directed graph in Figure 5.2(a) is equal to the unattributed histogram on *Src* shown in Figure 5.2(d).

For an undirected graph, if we represent an edge  $(u, v)$  as two symmetric tuples in  $R$  – tuples  $(u, v)$  and  $(v, u)$  are in  $R$  if and only if edge  $(u, v)$  is in the graph – then the degree sequence is an unattributed histogram on *Src* in  $R$ . (Each edge in the graph is counted twice in the histogram.)

A conventional histogram would be an attributed degree sequence and would provide an additional association of each degree with a named individual. For example, the attributed out-degree sequence of the graph shown in Figure 5.2(a) is equal to the conventional histogram on *Src* shown in Figure 5.2(c).

Most uses of the degree sequence are concerned with the distribution of degrees in the graph, and not the degrees of particular individuals, so the unattributed degree sequence is sufficient.

While an unattributed histogram can be derived from a conventional histogram, we distinguish it because it is possible to estimate unattributed histograms much more accurately under differential privacy. Because the association between bin and frequency is not needed, we have greater flexibility in designing query strategies. Next, we describe innovative strategies for estimating unattributed histograms under differential privacy.

### **5.1.2 Solution: inference on queries with constraints**

We developed two related strategies for computing an unattributed histogram under differential privacy. Both strategies share a common three step procedure for computing the histogram. First, we carefully formulate a query such that its answer, a sequence of numbers, is constrained so that relationships hold among the numbers. The query is answered using the Laplace mechanism (Section 2.2) to obtain noisy answer that is differentially private. The addition of random noise may result in a

sequence of numbers that violate the constraints. So, the final step refines the noisy answer using statistical inference to produce an answer that is consistent with the constraints. The inferred sequence can be much more accurate. The two strategies differ in their choice of query.

We formally define the two strategies in the next section. In the remainder of this section, we convey the main ideas behind the first strategy, by describing the query and showing how enforcing constraints through inference can result in a more accurate answer. The second strategy and the details of the first strategy are described in Section 5.2.

In the first strategy, we ask for the frequencies of the unattributed histogram in rank order. For a histogram over  $n$  bins, the answer is a sequence of  $n$  numbers, where the  $i^{\text{th}}$  number is the  $i^{\text{th}}$  smallest frequency in the histogram. The true answer to the query is a monotonic non-decreasing sequence of numbers. To achieve differential privacy, we show it is sufficient to add (appropriately scaled) independent random noise to each number. Thus, the analyst receives a sequence of  $n$  numbers, where the  $i^{\text{th}}$  number is the  $i^{\text{th}}$  smallest frequency plus noise. Since noise has been added, the numbers may no longer be monotonically non-decreasing. But we can use the monotonicity constraint to infer a potentially more accurate answer. We propose an inference procedure that takes the noisy sequence and derives an estimate for the unknown monotonic sequence. The output of the inference is the closest monotonic sequence to the noisy sequence.

Inference can reduce error, but it will depend on the frequencies in the histogram. The greatest reduction occurs when all bins have the same frequency. Here is an example for a histogram over  $n = 10$  bins where all bins have the same frequency, 12. The Laplace mechanism adds independent random noise to each count.

True answer	$\langle 12, 12, 12, 12, 12, 12, 12, 12, 12, 12 \rangle$
Noisy answer	$\langle 12.3, 13.2, 12.9, 11.4, 12.0, 11.6, 9.5, 11.4, 12.4, 11.5 \rangle$
Inferred answer	$\langle 11.8, 11.8, 11.8, 11.8, 11.8, 11.8, 11.8, 11.8, 11.9, 11.9 \rangle$

In this case, inference effectively averages the noisy observations, thereby reducing variance and producing a more accurate estimate of the count. At the other extreme, if all bins have vastly different frequencies, then the noisy sequence may not violate the monotonicity constraints. For example, consider this histogram over  $n = 5$  bins.

True answer	$\langle 12, 25, 50, 97, 123 \rangle$
Noisy answer	$\langle 11.5, 25.2, 49.6, 95.2, 124.8 \rangle$
Inferred answer	$\langle 11.5, 25.2, 49.6, 95.2, 124.8 \rangle$

The noisy answers are already monotonically increasing and therefore there is nothing to be done during inference. The inferred answer is identical to the noisy answer and there is no improvement in accuracy. Fortunately, in real world applications such as estimating the degree sequence of a private graph, the histograms tend to more closely resemble the former example, in which case inference greatly boosts accuracy.

There are several challenges in executing the above strategy, which we explore in the rest of this chapter. First, we must formally define our query strategies and prove that differential privacy is satisfied. This requires describing the mechanism for adding random noise and proving it is differentially private. This is the focus of Section 5.2. Second, we must formalize the inference process and demonstrate its effectiveness. The inference process is common to both strategies and described in Sections 5.3 & 5.4, and its performance is evaluated theoretically in Section 5.3.2 and experimentally in Section 5.5.

## 5.2 Query strategies

We describe three query strategies. The first is a baseline strategy, in which our strategy for computing an unattributed histogram is to simply compute a conventional histogram and ignore attribution. This strategy does not work well: the noise introduced to ensure privacy greatly distorts the answer. We then describe our two innovative strategies. Both strategies ask queries where ordering constraints hold among the answers, which will then be exploited by the inference mechanism described in Section 5.3.

While our query strategies are simply alternative representations of the same information, we find that representation matters when answers can be revealed only approximately. If we were able to return exact answers, then there is no distinction between alternative representations: the answer to one query can be used to derive the answer for the other. But under differential privacy, we must add noise and the choice of representation determines where the noise is introduced. We will show in experiments that the consequence is that different representations exhibit different performance.

### 5.2.1 Baseline strategy: conventional histogram

We first describe the baseline strategy of asking a conventional histogram query, introducing common notational conventions along the way.

All of the query strategies considered in this paper are formulated as *query sequences* where each element of the sequence is a query on the database. We use  $\mathbf{Q}$  to denote a generic query sequence. We refer to query sequences using bold letters ( $\mathbf{Q}$ ,  $\mathbf{C}$ ,  $\mathbf{S}$ ). When  $\mathbf{Q}$  is evaluated on a database instance  $I$ , the output,  $\mathbf{Q}(I)$ , includes one answer to each numerical query, so  $\mathbf{Q}(I)$  is a vector of numbers. The  $i^{\text{th}}$  query in  $\mathbf{Q}$  is  $\mathbf{Q}[i]$ . Table 5.1 reviews notational conventions.

**Table 5.1.** Notational conventions for query sequences.

<b>Q</b>	Sequence of queries
<b>C</b>	Conventional histogram
<b>S</b>	Unattrib. histogram via <b>S</b> orting
<b>F</b>	Unattrib. histogram via <b>F</b> requency
$\gamma_{\mathbf{Q}}$	Constraint set for query <b>Q</b>
$\tilde{\mathbf{Q}}, \tilde{\mathbf{C}}, \tilde{\mathbf{S}}, \tilde{\mathbf{F}}$	Randomized query sequence
$\overline{\mathbf{S}}, \overline{\mathbf{F}}$	Randomized query sequence, returning minimum $L_2$ solution
$I$	Private database instance
$\mathbf{C}(I), \mathbf{S}(I), \mathbf{F}(I)$	Output sequence (truth)
$\tilde{d} = \tilde{\mathbf{C}}(I), \tilde{s} = \tilde{\mathbf{S}}(I), \tilde{f} = \tilde{\mathbf{F}}(I)$	Output sequence (noisy)
$\bar{s} = \overline{\mathbf{S}}(I), \bar{f} = \overline{\mathbf{F}}(I)$	Output sequence (inferred)

We can express a histogram as a sequence of counting queries on range attribute  $A$ . We write intervals as  $[x, y]$  for  $x, y \in \text{dom}$ , and abbreviate  $[x, x]$  as  $[x]$ . For a given interval  $[x, y]$ , a counting query  $c([x, y])$  reports the number of tuples whose value of  $A$  is contained by the interval:

$$c([x, y]) = \text{SELECT COUNT(*) FROM R WHERE } x \leq \text{R.A} \leq y$$

A histogram is computed with respect to a user-specified partition of  $\text{dom}$  into  $n$  disjoint intervals  $[x_1, y_1], [x_2, y_2], \dots, [x_n, y_n]$  whose union is equal to  $\text{dom}$ . We refer to each interval as a bin. We use  $\mathbf{C}$  to denote the query sequence of a conventional histogram. It is defined as:

$$\mathbf{C} = \langle c([x_1, y_1]), c([x_2, y_2]), \dots, c([x_n, y_n]) \rangle$$

For example, consider a histogram on attribute *Condition* with unit-length intervals (each contains a single condition). The histogram expressed as a query sequence is:

$$\mathbf{C} = \langle c([\text{Asthma}]), c([\text{Cancer}]), c([\text{Flu}]), c([\text{Healthy}]) \rangle$$

The answer to  $\mathbf{C}$  is a vector of  $n$  numbers, corresponding to the number of tuples in each bin. For example, if  $I$  is the relation in Figure 5.1(a) and  $\mathbf{C}$  is the histogram on attribute *Condition*, then the answer is.

$$\mathbf{C}(I) = \langle 2, 0, 2, 1 \rangle$$

To ensure differential privacy, we do not return exact answers to queries. Instead, random noise is added. For all of our query strategies, we rely on the Laplace mechanism. This mechanism adds independent noise to each answer, where the noise is drawn from a zero mean, appropriately scaled Laplace distribution. As described in Section 2.2, the scale depends on the *sensitivity* of the query—informally, the amount by which the query answer can change due to the addition or removal of an individual’s private data. This mechanism can be applied to *any* query sequence and satisfies the rigorous privacy guarantee of differential privacy.

To apply the Laplace mechanism to  $\mathbf{C}$ , we must determine the sensitivity of  $\mathbf{C}$ , denoted  $\Delta_{\mathbf{C}}$ .

**Proposition 5.1** (Sensitivity of  $\mathbf{C}$ ). *The sensitivity of  $\mathbf{C}$ , denoted  $\Delta_{\mathbf{C}}$ , is 1.*

*Proof.* Let  $I$  and  $I'$  be neighboring databases that differ by the tuple  $t$ . This tuple affects the answer to a single query in  $\mathbf{C}$ , specifically it changes the count for the interval that contains  $t$ .  $A$  by exactly 1. Therefore,  $\|\mathbf{C}(I) - \mathbf{C}(I')\|_1 = 1$ .  $\square$

We use  $\tilde{\mathbf{C}}$  to denote the application of the Laplace mechanism to the query sequence  $\mathbf{C}$ . Let  $\langle \text{Lap}(b) \rangle^n$  denote a vector of independent Laplace random variables each with scale  $b$ .

**Definition 5.1** ( $\tilde{\mathbf{C}}$ ). The randomized algorithm  $\tilde{\mathbf{C}}$  is defined as the application of the Laplace mechanism to the query sequence  $\mathbf{C}$ . It returns the following randomized vector:

$$\tilde{\mathbf{C}}(I) = \mathbf{C}(I) + \langle \text{Lap}(\Delta_{\mathbf{C}}/\epsilon) \rangle^n$$

It follows from Proposition 2.2 that this is  $\epsilon$ -differentially private.

### 5.2.2 First strategy: frequencies in rank order

The first query strategy returns the frequencies of the unattributed histogram in rank order. Let  $\mathbf{S}$  denote the  $n$ -length query sequence where the  $i^{\text{th}}$  query returns the frequency of the  $i^{\text{th}}$  lowest frequency bin. Formally, for input  $I$  let  $\pi : [1, n] \rightarrow [1, n]$  be a permutation of the bins specific to  $I$  such that if  $i < j$ , then  $c([x_{\pi(i)}, y_{\pi(i)}]) \leq c([x_{\pi(j)}, y_{\pi(j)}])$  on input  $I$ . Then  $\mathbf{S}[i] = c([x_{\pi(i)}, y_{\pi(i)}])$  for  $i = 1$  to  $n$ .

For example, if  $\mathbf{S}$  is an unattributed histogram on attribute *Condition*, then  $\mathbf{S}[1]$  is the frequency of the rarest condition and  $\mathbf{S}[4]$  is the frequency of the most common condition (out of four possible conditions).

The answer to  $\mathbf{S}$  is a vector of length  $n$ , corresponding to the bin frequencies. The numbers in the vector are monotonically non-decreasing. Following the previous example, if  $I$  is a database consisting of the relation in Figure 5.1(a) and  $\mathbf{S}$  is an unattributed histogram on attribute *Condition*, then  $\mathbf{S}(I)$  is:

$$\mathbf{S}(I) = \langle 0, 1, 2, 2 \rangle$$

Query sequences  $\mathbf{C}$  and  $\mathbf{S}$  are similar, but with one key distinction. Both compute the same set of  $n$  counts over the intervals  $[x_1, y_1]$  to  $[x_n, y_n]$ , and so when evaluated, they return the same multiset of numbers. The distinction between them is how those numbers are indexed: in  $\mathbf{C}$  the query answers are indexed by domain—the  $i^{\text{th}}$  answer is the frequency of  $i^{\text{th}}$  smallest attribute value in the domain—and in  $\mathbf{S}$  the answers are indexed by their rank—the  $i^{\text{th}}$  answer is the  $i^{\text{th}}$  smallest frequency.

To answer  $\mathbf{S}$  using the Laplace mechanism, we must determine its sensitivity. Perhaps surprisingly, the sensitivity of  $\mathbf{S}$  is no larger than the sensitivity of  $\mathbf{C}$ . (In fact,  $\Delta_{\mathbf{S}} = \Delta_{\mathbf{C}} = 1$ .) This follows from the following claim.

**Proposition 5.2** (Sensitivity of  $\mathbf{S}$ ). *For any query sequence  $\mathbf{Q}$ , let  $\mathbf{S}_{\mathbf{Q}}$  be the query sequence that returns the answers of  $\mathbf{Q}$  in rank order. Then  $\Delta_{\mathbf{S}_{\mathbf{Q}}} \leq \Delta_{\mathbf{Q}}$ .*

*Proof.* Let  $x$  and  $y$  be two vectors of length  $n$ . Let  $\pi_x : [1, n] \rightarrow [1, n]$  be a permutation that sorts  $x$ , i.e., if  $i < j$ , then  $x_{\pi_x(i)} \leq x_{\pi_x(j)}$ . Similarly, let  $\pi_y$  be a permutation that sorts  $y$ . First, we claim that

$$\left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi_y(1)}, \dots, y_{\pi_y(n)} \rangle \right\|_1 \leq \|x - y\|_1$$

If this claim is true, it implies for any databases  $I$  and  $I'$ ,  $\|\mathbf{S}_{\mathbf{Q}}(I) - \mathbf{S}_{\mathbf{Q}}(I')\|_1 \leq \|\mathbf{Q}(I) - \mathbf{Q}(I')\|_1$ .

We now prove the claim. While  $\pi_x$  is specific to  $x$ , we can still apply the same permutation to  $y$ , to obtain  $\langle y_{\pi_x(1)}, \dots, y_{\pi_x(n)} \rangle$ . If we apply the same permutation to both vectors, this does not change the  $L_1$  distance between them since the alignment between  $x$  and  $y$  is preserved under the permutation. Thus, we have:

$$\left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi_x(1)}, \dots, y_{\pi_x(n)} \rangle \right\|_1 = \|x - y\|_1$$

Now consider swapping any out-of-order pair in the permuted  $y$ : i.e., if there exists a pair  $y_{\pi(i)}, y_{\pi(j)}$  such that  $i < j$  but  $y_{\pi(i)} > y_{\pi(j)}$ , then let  $\pi'$  be a new permutation identical to  $\pi_x$  except  $\pi'(i) = \pi_x(j)$  and  $\pi'(j) = \pi_x(i)$ . This can only decrease  $L_1$  distance, as the smaller (resp. larger) number in  $y$  is now aligned with the smaller (resp. larger) number in  $x$ . That is,

$$\begin{aligned} & \left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi'(1)}, \dots, y_{\pi'(n)} \rangle \right\|_1 \\ & \leq \left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi_x(1)}, \dots, y_{\pi_x(n)} \rangle \right\|_1 \end{aligned}$$

If we repeat this step, swapping any out-of-order pair in  $y$  and updating  $\pi'$  accordingly, the  $L_1$  can only decrease at each step. Eventually, no out-of-order pairs will remain, and  $\pi'$  will be a permutation that sorts  $y$ ; i.e.,  $\pi' = \pi_y$  and we have shown that

$$\begin{aligned} & \left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi_y(1)}, \dots, y_{\pi_y(n)} \rangle \right\|_1 \\ & \leq \left\| \langle x_{\pi_x(1)}, \dots, x_{\pi_x(n)} \rangle - \langle y_{\pi_x(1)}, \dots, y_{\pi_x(n)} \rangle \right\|_1 \\ & = \|x - y\|_1 \end{aligned}$$

completing the proof. □

Again, we can obtain a differentially private estimate of  $\mathbf{S}$  using the Laplace mechanism.

**Definition 5.2** ( $\tilde{\mathbf{S}}$ ). The randomized algorithm  $\tilde{\mathbf{S}}$  is defined as the application of the Laplace mechanism to the query sequence  $\mathbf{S}$ . It returns the following randomized vector:

$$\tilde{\mathbf{S}}(I) = \mathbf{S}(I) + \langle \text{Lap}(\Delta_{\mathbf{S}}/\epsilon) \rangle^n$$

It follows from Proposition 2.2 that this is  $\epsilon$ -differentially private:

Since the same magnitude of noise is added to  $\mathbf{S}$  as to  $\mathbf{C}$ , it appears as though the accuracies of  $\tilde{\mathbf{S}}$  and  $\tilde{\mathbf{C}}$  are the same. However,  $\mathbf{S}$  implies a powerful set of constraints. Notice that the ordering occurs *before* noise is added. Thus, the returned counts are ordered according to the true rank order. If the returned answer contains out-of-order counts, this must be caused by the addition of random noise.

We use  $\gamma$  to denote the set of constraints associated with a query. For the query  $\mathbf{S}$ ,  $\gamma_{\mathbf{S}}$  contains the set of inequalities  $\mathbf{S}[i] \leq \mathbf{S}[i + 1]$  for  $1 \leq i < n$ . (The query  $\mathbf{C}$  has no constraints between the answers, so  $\gamma_{\mathbf{C}}$  is empty.) In Section 5.3, we show how to exploit these constraints to boost accuracy.

### 5.2.3 Second strategy: frequency of frequencies

The main idea behind the second strategy is as follows. We can view the frequencies of an unattributed histogram as simply a collection of  $n$  numbers. Each number is an integer in the range  $[0, N]$  where  $N$  is the number of tuples in the database. For now, we assume  $N$  is known. Treating these  $n$  numbers as data, we can measure the data distribution, for example by computing a histogram. In other words, we are computing the frequency of frequencies: the number of times that each frequency occurs in the unattributed histogram.

Our second approach computes a particular kind of histogram over these numbers, a *cumulative* histogram. The query sequence is denoted  $\mathbf{F}$  and has length  $N$ . For  $k = 1$  to  $N$ , the  $k^{\text{th}}$  query in the sequence is defined as

$$\mathbf{F}[k] = \sum_{i=1}^n \mathbb{I}[c([x_i, y_i]) \geq k]$$

where  $\mathbb{I}[\cdot]$  is the indicator function, equal to 1 when its argument is true and 0 otherwise. In words,  $\mathbf{F}[k]$  reports the number of bins (in the unattributed histogram) whose frequency is  $k$  or larger. The answer is an integer between 0—meaning that no bins have at least this frequency—and  $n$ —meaning that all bins have at least this frequency.

For example, if  $\mathbf{F}$  is computed on attribute *Condition*, then  $\mathbf{F}[1]$  is the number of conditions that occur at least once,  $\mathbf{F}[2]$  is the number of conditions that occur at least twice, and so on. The length of the query depends on  $N$ , the number of tuples in the relation. Observe that given an answer to  $\mathbf{F}$ , we can derive the answer  $\mathbf{S}$ , and vice versa.

Evaluating this query on the input  $I$  shown in Figure 5.1(a), we have  $N = 5$  and therefore  $\mathbf{F}(I)$  has length 5 and the answer is as follows

$$\mathbf{F}(I) = \langle 3, 2, 0, 0, 0 \rangle$$

This answer indicates that three conditions occur at least once, two conditions occur twice, and zero conditions occur three or more times.

Like the  $\mathbf{S}$  query, the answers of  $\mathbf{F}$  are constrained. As  $k$  increases, the value of  $\mathbf{F}[k]$  can only decrease. Thus,  $\gamma_{\mathbf{F}}$  consists of the set of inequalities  $\mathbf{F}[i] \geq \mathbf{F}[i + 1]$  for  $1 \leq i < N$ .

To apply the Laplace mechanism, we must determine the sensitivity of  $\mathbf{F}$ .

**Proposition 5.3** (Sensitivity of  $\mathbf{F}$ ). *The sensitivity of  $\mathbf{F}$  is  $\Delta_{\mathbf{F}} = 1$ .*

*Proof.* Let  $I$  and  $I'$  be neighboring databases such that  $I'$  has an additional tuple  $t$ . Without loss of generality, let  $t$  fall into interval  $[x_i, y_i]$ . Further, suppose that on  $I$ ,  $c([x_i, y_i]) = k$ . On database  $I'$ , the inclusion of  $t$  makes  $c([x_i, y_i]) = k + 1$ . Therefore if  $\mathbf{F}[k + 1] = a$  on  $I$ , then  $\mathbf{F}[k + 1] = a + 1$  on  $I'$ ; note that  $\mathbf{F}[k]$  is the same on both  $I$  and  $I'$ , as are all other positions. Therefore  $\|\mathbf{F}(I) - \mathbf{F}(I')\|_1 = 1$ .  $\square$

Again, we can obtain a differentially private estimate of  $\mathbf{F}$  using the Laplace mechanism.

**Definition 5.3** ( $\tilde{\mathbf{F}}$ ). The randomized algorithm  $\tilde{\mathbf{F}}$  is defined as the application of the Laplace mechanism to the query sequence  $\mathbf{F}$ . It returns the following randomized vector:

$$\tilde{\mathbf{F}}(I) = \mathbf{F}(I) + \langle \text{Lap}(\Delta_{\mathbf{F}}/\epsilon) \rangle^N$$

It follows from Proposition 2.2 that this is  $\epsilon$ -differentially private.

When  $N$  is unknown, a slightly more complex query strategy is necessary. Our strategy is an interactive approach in which we incrementally compute  $\mathbf{F}$  by iteratively asking  $\mathbf{F}[k]$  for increasing  $k$  until a stopping condition is reached. Observe that for  $k \geq N + 1$ , the true query answer,  $\mathbf{F}[k]$ , is zero. While the returned answers may not be equal to zero due to the addition of random noise, the expected value of each answer is zero. We can continue increasing  $k$  until the noisy answers appear to have

“converged” to 0. For example, we can apply the central limit theorem to derive a confidence interval around the mean of the last  $N_0$  observations and terminate when the upper bound of the interval is less than 1. The number  $N_0$  is a function of the noise magnitude  $\Delta_{\mathbf{F}}/\epsilon$  and our desired level of confidence. The following proposition claims that this interactive approach is differentially-private.

**Proposition 5.4** (Interactive  $\mathbf{F}$ ). *Let  $\tilde{\mathbf{F}}_k$  denote the randomized algorithm that returns the following noisy answer:*

$$\tilde{\mathbf{F}}_k(I) = \mathbf{F}(I)[k] + \text{Lap}(\Delta_{\mathbf{F}}/\epsilon)$$

*The process that interactively asks  $\tilde{\mathbf{F}}_i$  for  $i = 1$  up to any  $N'$  (where  $N'$  may depend on the answers received) satisfies  $\epsilon$ -differential privacy.*

*Proof.* Consider a transcript of noisy answers Let  $\langle f_1, \dots, f_{N'} \rangle$  output by the interactive process. We claim that for any two neighboring databases  $I$  and  $I'$  the probability of returning  $\langle f_1, \dots, f_{N'} \rangle$  differs by at most a factor of  $\exp(\epsilon)$ . There exists a single  $k$  such that  $\mathbf{F}(I)[k] \neq \mathbf{F}(I')[k]$ , for all  $i \neq k$ ,  $\mathbf{F}(I)[i] = \mathbf{F}(I')[i]$ . Thus, for any  $i \neq k$ , we have

$$\Pr[\tilde{\mathbf{F}}_i(I) = f_i | f_1, \dots, f_{i-1}] = \Pr[\tilde{\mathbf{F}}_i(I') = f_i | f_1, \dots, f_{i-1}]$$

and for  $k$ , the Laplace noise ensures that

$$\Pr[\tilde{\mathbf{F}}_k(I) = f_k | f_1, \dots, f_{k-1}] \leq \exp(\epsilon) \Pr[\tilde{\mathbf{F}}_k(I') = f_k | f_1, \dots, f_{k-1}]$$

Therefore

$$\prod_{i=1}^{N'} \Pr[\tilde{\mathbf{F}}_i(I) = f_i | f_1, \dots, f_{i-1}] \leq \exp(\epsilon) \prod_{i=1}^{N'} \Pr[\tilde{\mathbf{F}}_i(I') = f_i | f_1, \dots, f_{i-1}]$$

and we have proved the claim. □

**Special considerations for computing degree sequences** We briefly remark on some differences in computing sensitivity for the special case where the unattributed histogram is the degree sequence of an undirected graph. The issue is that in the graph setting, neighboring instances may differ by more than a single tuple. Neighboring graphs can differ by multiple edges under  $k$ -edge differential privacy for  $k > 1$  and under node-differential privacy (Section 2.4).

For  $k$ -edge differential privacy, it is sufficient to analyze the special case of  $k = 1$  because existing results (Section 2.2) show that differential privacy for  $k = 1$  extends naturally to  $k > 1$ . Therefore, we focus on the case of  $k = 1$ . For directed graphs, removing one edge is equivalent to removing one tuple, so our analysis of unattributed histograms applies immediately to the (in or out) degree sequence of a directed graph. For an undirected graph, each edge is counted twice in the unattributed histogram, once for each endpoint. Therefore the sensitivity must be doubled.

For node differential privacy, it is not hard to see that the sensitivity is large, specifically  $O(n)$ . In other words, the magnitude of the noise added by the Laplace mechanism matches the range of the query answers and therefore, the answers are so noisy, they are useless. For this reason, in the remainder of this chapter, we adopt  $k$ -edge differential privacy as the privacy standard.

A final issue for degree sequences is that  $n$ , the number of bins in the unattributed histogram, depends on the graph and may not be public knowledge. This impacts the **S** query, because its length is  $n$ . Recall from Section 5.1.1, there is a bin for each node in the graph. Releasing the exact number of nodes in the graph does not technically satisfy  $k$ -edge differential privacy. This is not a significant issue, and can be remedied similarly to the way the issue of unknown  $N$  is handled with the **F** query. Specifically, first we slightly modify the definition of **S**: we return frequencies (degrees) in descending, rather than ascending order, and the  $i^{\text{th}}$  smallest degree for

any  $i > |V| = n$  is defined as  $-1$ . Then we apply the incremental approach described for the  $\mathbf{F}$  query, stopping when the answers appear to “converge” at  $-1$ .

### 5.3 Inference

Here we describe how we can use statistical inference to refine noisy answers to obtain estimates that are consistent with the query constraints. For the query sequences we consider here –  $\mathbf{S}$  and  $\mathbf{F}$  – the primary constraint is an ordering constraint. However, inference can be applied to other query sequences, resulting in consistent answers and, in some cases, increased accuracy [55, 76]. We first describe the general inference framework and then describe the specific approach for handling order constraints.

Given a query sequence  $\mathbf{Q}$ , let  $\gamma_{\mathbf{Q}}$  denote the set of constraints which must hold among the answers. The inference process takes the randomized output of the query, denoted  $\tilde{q} = \tilde{\mathbf{Q}}(I)$ , and finds the sequence of query answers  $\bar{q}$  that is “closest” to  $\tilde{q}$  and also satisfies the constraints of  $\gamma_{\mathbf{Q}}$ . Here closest is determined by  $L_2$  distance, and the result is the *minimum  $L_2$  solution*:

**Definition 5.4** (Minimum  $L_2$  solution). Let  $\text{MIN}L_2(\cdot, \cdot)$  be a function that takes noisy sequence  $\tilde{q}$  and constraints  $\gamma_{\mathbf{Q}}$  and outputs a vector  $\bar{q}$  that satisfies the constraints  $\gamma_{\mathbf{Q}}$  and at the same time minimizes  $\|\tilde{q} - \bar{q}\|_2$ . We say that  $\bar{q}$  is the minimum  $L_2$  solution.

Examples of minimum  $L_2$  solutions are shown in Figure 5.1 for two different queries that have ordering constraints.

Finding the minimum  $L_2$  solution requires no access to the private database, only  $\tilde{q}$ , the output of a differentially private algorithm, and  $\gamma_{\mathbf{Q}}$ , a property of the query. In fact, the computation can be carried out by the analyst after receiving  $\tilde{q}$ . Nevertheless, we now formally prove that  $\bar{q}$  can be computed under differential privacy. We use  $\overline{\mathbf{Q}}$  to denote the two step randomized process in which the data owner first computes  $\tilde{q} = \tilde{\mathbf{Q}}(I)$  and then computes  $\bar{q} = \text{MIN}L_2(\tilde{q}, \gamma_{\mathbf{Q}})$ .

**Proposition 5.5** (Inference preserves privacy). *If  $\tilde{\mathbf{Q}}$  satisfies  $\epsilon$ -differential privacy, then  $\overline{\mathbf{Q}}$  satisfies  $\epsilon$ -differential privacy.*

*Proof.* For any set of outputs  $\mathcal{O}$  of the function  $\text{MIN}_{L_2}(\cdot, \cdot)$ , let  $\mathcal{I}_{\mathcal{O}}$  denote  $\mathcal{O}$ 's pre-image:  $\mathcal{I}_{\mathcal{O}} = \{\tilde{q} \mid \text{MIN}_{L_2}(\tilde{q}, \gamma_{\mathbf{Q}}) \in \mathcal{O}\}$ . For any  $I$  and  $I' \in \text{nbrs}(I)$ , the following shows that  $\overline{\mathbf{Q}}$  is  $\epsilon$ -differentially private:

$$\begin{aligned} \Pr[\overline{\mathbf{Q}}(I) \in \mathcal{O}] &= \Pr[\tilde{\mathbf{Q}}(I) \in \mathcal{I}_{\mathcal{O}}] \\ &\leq \exp(\epsilon) \Pr[\tilde{\mathbf{Q}}(I') \in \mathcal{I}_{\mathcal{O}}] \\ &= \exp(\epsilon) \Pr[\overline{\mathbf{Q}}(I') \in \mathcal{O}] \end{aligned}$$

where the inequality is due to the fact that  $\tilde{\mathbf{Q}}$  is  $\epsilon$ -differentially private.  $\square$

We will show that for some queries, inference produces more accurate answers. In other words, by moving the noisy answer to the closest answer that is consistent with the constraints, we also move it closer to the true answer. One can show that when the constraints define a solution set that is convex, as is the case with the queries we study here, then inference can only reduce error [114]. However, we find that inference not only does not harm, but it can substantially reduce error, as was suggested previously by the examples in Section 5.1.2

Next we show how to compute the minimum  $L_2$  solution for queries with ordering constraints.

### 5.3.1 Inference for ordering constraints

We now present the minimum  $L_2$  solution for the case of ordering constraints. It can be applied to any query sequence where ordering constraints hold among the answers. This includes  $\mathbf{S}$  and  $\mathbf{F}$ . Let  $\mathbf{O}$  be any  $n$ -length query sequence where *ascending* order constraints  $\gamma_{\mathbf{O}}$  hold among the answers, i.e.,  $\gamma_{\mathbf{O}}$  contains the inequalities  $\mathbf{O}[i] \leq \mathbf{O}[i + 1]$  for all  $1 \leq i < n$ . Of course, we can apply our solution to queries

with *descending* order constraints—such as  $\mathbf{F}$ —by reversing the vector before applying inference. Let  $\tilde{o}$  denote a random output from the Laplace mechanism applied to  $\mathbf{O}$ .

For ordering constraints, finding the minimum  $L_2$  solution can be cast as a convex optimization problem. A convex optimization problem involves minimizing (or maximizing) a convex objective function subject to convex constraints on its arguments [18].

**Problem 1** (Inference under order constraints). Given  $\tilde{o}$  and  $\gamma_{\mathbf{O}}$ , the problem of finding the minimum  $L_2$  solution,  $\bar{o} = \text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$  is equivalent to solving the following convex optimization problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^n (\tilde{o}[i] - \bar{o}[i])^2 \\ \text{subject to} \quad & \bar{o}[i] \leq \bar{o}[i+1], \quad 1 \leq i < n. \end{aligned} \tag{5.1}$$

The solution to this problem has an elegant, closed form as shown the following theorem. The proof is in Section 5.7.1.

**Theorem 5.1.** [*Minimum  $L_2$  Solution for ordering constraints*] For a given  $\tilde{o}$ , let  $\tilde{o}[i, j]$  denote the subsequence  $\langle \tilde{o}[i], \tilde{o}[i+1], \dots, \tilde{o}[j] \rangle$  and  $M[i, j]$  the average of this subsequence:  $M[i, j] = \sum_{k=i}^j \tilde{o}[k] / (j - i + 1)$ . For each position  $k$ , let  $L_k$  and  $U_k$  be defined as

$$L_k = \min_{k \leq j \leq n} \max_{1 \leq i \leq n} M[i, j]$$

and

$$U_k = \max_{1 \leq i \leq k} \min_{i \leq j \leq n} M[i, j]$$

The minimum  $L_2$  solution,  $\bar{o} = \text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$ , is unique and is equal to  $\bar{o}[k] = L_k = U_k$ .

After we developed our solution to Problem 1, we learned that this problem is an instance of *isotonic regression*, a problem that has been analyzed in statistics. The statistics literature has several characterizations, including the max-min formula we present above (cf. Barlow et al. [14]). To our knowledge, our extension to include additional constraints (below) and our utility analysis (next section) are novel.

We describe a linear time algorithm for computing  $\text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$  in Section 5.4. It is in fact a variant of an existing algorithm for isotonic regression, though we developed it independently.

Here are a few examples of short sequences  $\tilde{o}$  and their minimal  $L_2$  solutions  $\bar{o}$ .

**Table 5.2.** Examples of private outputs  $\tilde{o} = \tilde{\mathbf{O}}(I)$  and their closest ordered sequence  $\bar{o}$ .

original output $\tilde{o}$	inferred output $\bar{o}$	$L_2$ distance $\ \tilde{o} - \bar{o}\ _2$
$\langle 9, 10, 14 \rangle$	$\langle 9, 10, 14 \rangle$	0
$\langle 9, 14, 10 \rangle$	$\langle 9, 12, 12 \rangle$	4
$\langle 14, 9, 10, 15 \rangle$	$\langle 11, 11, 11, 15 \rangle$	14

**Example 7.** Table 5.2 gives three examples of  $\tilde{o}$  and its closest ordered sequence  $\bar{o}$ . First, suppose  $\tilde{o} = \langle 9, 10, 14 \rangle$ . Since  $\tilde{o}$  is already ordered,  $\bar{o}$  is equal to  $\tilde{o}$ . In the second example,  $\tilde{o} = \langle 9, 14, 10 \rangle$ , the last two elements are out of order. The closest ordered sequence is  $\bar{o} = \langle 9, 12, 12 \rangle$ . Finally, let  $\tilde{o} = \langle 14, 9, 10, 15 \rangle$ . The sequence is in order except for  $\tilde{o}[1]$ . While changing the first element from 14 to 9 would make it ordered, its distance from  $\tilde{o}$  would be  $(14 - 9)^2 = 25$  which is further away than  $\bar{o} = \langle 11, 11, 11, 15 \rangle$ , which is 14 from  $\tilde{o}$ .

Since the Laplace mechanism introduces random noise to each numerical answer in the sequence, both the output  $\tilde{o}$  and  $\bar{o}$  may include numbers that are non-integral and/or negative. In many applications, there may be additional constraints of integrality and non-negativity. This motivates the following problem.

**Problem 2** (Inference under order, integrality, and bound constraints). Let  $\gamma'_{\mathbf{O}}$  be the constraint set  $\gamma_{\mathbf{O}}$  augmented with the additional constraints that each answer be integral and between given lower and upper bounds, respectively denoted  $L$  and  $U$ . The problem of finding the minimum  $L_2$  solution  $\text{MIN}L_2(\tilde{\delta}, \gamma'_{\mathbf{O}})$  is equivalent to solving the following optimization problem.

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n (\tilde{\delta}[i] - \bar{\delta}[i])^2 \\ & \text{subject to} && L \leq \bar{\delta}[i] \leq \bar{\delta}[i+1] \leq U, \quad 1 \leq i < n. \\ & && \bar{\delta}[i] \in \mathbb{Z}, \quad 1 \leq i \leq n. \end{aligned} \tag{5.2}$$

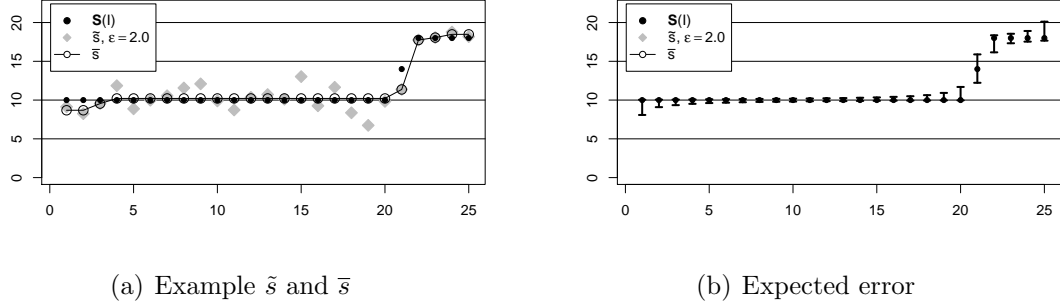
Despite the fact that integrality constraints make the problem non-convex, Theorem 5.2 shows that including these constraints can be easily incorporated. The resulting solution is equal to  $\bar{\delta}$  rounded to the nearest non-negative integers. The proof is in Section 5.7.2.

**Theorem 5.2.** *[Minimum  $L_2$  Solution for order, integrality, and bound constraints]* Given  $\bar{\delta} = \text{MIN}L_2(\tilde{\delta}, \gamma_{\mathbf{O}})$ , let  $\bar{\delta}'$  denote the sequence derived from  $\bar{\delta}$  in which each element  $\bar{\delta}'[k]$  is rounded to the nearest integer in  $[L, U]$ . Then  $\bar{\delta}' = \text{MIN}L_2(\tilde{\delta}, \gamma'_{\mathbf{O}})$ .

### 5.3.2 Utility analysis

We analyze  $\bar{\mathbf{O}}$  and show under what conditions inference can lead to improved accuracy. To our knowledge, prior work in isotonic regression has only shown that inference cannot increase error [58]. Before presenting a theoretical statement of such conditions, we first give an illustrative example of how inference can reduce error for an instance of the  $\mathbf{S}$  query.

**Example 8.** Figure 5.3(a) shows a sequence  $\mathbf{S}(I)$  along with a sampled  $\tilde{s}$  and inferred  $\bar{s}$ . While the values in  $\tilde{s}$  deviate considerably from  $\mathbf{S}(I)$ ,  $\bar{s}$  lies very close to the true answer. In particular, for subsequence  $[1, 20]$ , the true sequence  $\mathbf{S}(I)$  is uniform



**Figure 5.3.** (a) Example of how inference reduces the error:  $\bar{\mathbf{s}}$  is more accurate than  $\tilde{\mathbf{s}}$ ; (b) Error bars show expected error for  $\bar{\mathbf{s}}$  at each position, showing lower error in the middle of the uniform subsequence.

and the constrained inference process effectively averages out the noise of  $\tilde{\mathbf{s}}$ . At the twenty-first position, which is a unique count in  $\mathbf{S}(I)$ , and constrained inference does not refine the noisy answer, i.e.,  $\bar{\mathbf{s}}[21] = \tilde{\mathbf{s}}[21]$ . Figure 5.3(b) shows the expected error per position; error is lowest in the middle of the uniform subsequence, and larger at the ends.

Figure 5.3(a) suggests that error will be low for sequences in which many counts are the same. To analyze the accuracy of the randomized query sequences proposed in this work we quantify their error.  $\tilde{\mathbf{O}}$  can be considered an estimator for the true value  $\mathbf{O}(I)$ . We use the common Mean Squared Error as a measure of accuracy.

**Definition 5.5** (Error). For a randomized query sequence  $\tilde{\mathbf{Q}}$  whose input is  $\mathbf{Q}(I)$ , the *error*( $\tilde{\mathbf{Q}}$ ) is  $\sum_i \mathbb{E}(\tilde{\mathbf{Q}}[i] - \mathbf{Q}[i])^2$ . Here  $\mathbb{E}$  is the expectation taken over the possible randomness in generating  $\tilde{\mathbf{Q}}$ .

For example,  $error(\tilde{\mathbf{S}}) = \sum_i \mathbb{E}(\tilde{\mathbf{S}}[i] - \mathbf{S}[i])^2$  which simplifies to:  $n \mathbb{E}[Lap(\Delta_{\mathbf{S}}/\epsilon)^2] = 8n/\epsilon^2$ .

The following theorem quantifies the accuracy of  $\bar{\mathbf{O}}$  precisely. Let  $n$  and  $d$  denote the number of values and the number of distinct values in  $\mathbf{O}(I)$  respectively. Let  $n_1, n_2, \dots, n_d$  be the number of times each of the  $d$  distinct values occur in  $\mathbf{O}(I)$  (thus  $\sum_i n_i = n$ ). The proof appears in Section 5.7.3.

**Theorem 5.3.** *[Utility of inference] There exist constants  $c_1$  and  $c_2$  independent of  $n$  and  $d$  such that*

$$\text{error}(\overline{\mathbf{O}}) \leq \sum_{i=1}^d \frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} (c_1 \log^3 n_i + c_2)$$

Thus  $\text{error}(\overline{\mathbf{O}}) = O(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} d \log^3 n)$  whereas  $\text{error}(\tilde{\mathbf{O}}) = \Theta(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} n)$ .

The above theorem shows that constrained inference can boost accuracy, and the improvement depends on properties of the input  $\mathbf{O}(I)$ . In particular, if the number of distinct elements  $d$  is 1, then  $\text{error}(\overline{\mathbf{O}}) = O(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} \log^3 n)$ , while  $\text{error}(\tilde{\mathbf{O}}) = \Theta(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} n)$ . On the other hand, if  $d = n$ , then  $\text{error}(\overline{\mathbf{O}}) = O(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} n)$  and thus both  $\text{error}(\overline{\mathbf{O}})$  and  $\text{error}(\tilde{\mathbf{O}})$  scale linearly in  $n$ . For many practical applications,  $d \ll n$ , which makes  $\text{error}(\overline{\mathbf{O}})$  significantly lower than  $\text{error}(\tilde{\mathbf{O}})$ . In Sec. 5.5, experiments on real data demonstrate that the error of  $\overline{\mathbf{O}}$  can be orders of magnitude lower than that of  $\tilde{\mathbf{O}}$ .

Theorem 5.3 also demonstrates the existence of a  $\epsilon$ -differentially private where the magnitude of the error depends on the instance. While there are other algorithms that add instance dependent noise, the privacy definition must be relaxed to allow for the (very low probability) event that the output discloses too much about the instance [102].

## 5.4 Inference algorithm

We now describe an efficient algorithm for computing  $\bar{o} = \text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$ . A straightforward approach for computing  $\bar{o}$  is to construct a dynamic program based on the solution given in Theorem 5.1. However, it requires linear time to compute each  $\bar{o}[k]$ , making the total runtime quadratic, infeasible for large sequences. We present an algorithm that requires only linear time.

We first described this algorithm Hay et al. [52]. Since that time, we learned that Algorithm 3 is in fact a variant of the Pool Adjacent Violators Algorithm (cf. Barlow et al. [13]).

The design of the algorithms stems from the following observation. If  $\tilde{o}$  violates the ordering constraints, then there exists at least one adjacent pair that is out of order. We can replace each of these observations with their average without affecting the minimum  $L_2$  solution.

**Lemma 5.1** (Solution Invariant). *Let  $\tilde{o}[k]$  and  $\tilde{o}[k + 1]$  be any adjacent pair such that  $\tilde{o}[k] \geq \tilde{o}[k + 1]$ . Let  $\tilde{o}'$  be equivalent to  $\tilde{o}$  except that  $\tilde{o}'[k] = \tilde{o}'[k + 1] = (\tilde{o}[k] + \tilde{o}[k + 1])/2$ . Then  $\text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}}) = \text{MIN}L_2(\tilde{o}', \gamma_{\mathbf{O}})$ .*

*Proof.* First, we show that  $\bar{o}[k] = \bar{o}[k + 1]$  whenever  $\tilde{o}[k] \geq \tilde{o}[k + 1]$ . We know that  $\bar{o}[k] \leq \bar{o}[k + 1]$  because of the ordering constraints. Suppose  $\bar{o}[k] < \bar{o}[k + 1]$ . We are free to increase  $\bar{o}[k]$  or reduce  $\bar{o}[k + 1]$  without violating the order constraints. There are two cases. If  $\tilde{o}[k + 1] < \bar{o}[k + 1]$  then decreasing  $\bar{o}[k + 1]$  reduces the  $L_2$  distance, contradicting the fact that  $\bar{o}$  has minimal  $L_2$  distance to  $\tilde{o}$ . Otherwise, we have  $\tilde{o}[k] \geq \tilde{o}[k + 1] \geq \bar{o}[k + 1] > \bar{o}[k]$  and increasing  $\bar{o}[k]$  reduces  $L_2$  distance, contradicting minimality. Either way, we reach a contradiction and therefore  $\bar{o}[k] = \bar{o}[k + 1]$  whenever  $\tilde{o}[k] \geq \tilde{o}[k + 1]$ .

The above implies  $\bar{o}[k] = \bar{o}[k + 1]$  and  $\bar{o}'[k] = \bar{o}'[k + 1]$ . We now show that  $\bar{o}[k] = \bar{o}'[k]$ . Let  $y$  denote the value of the solution at positions  $k$  and  $k + 1$ . Let us write the objective function of (5.1) as a function of  $y$ . On input  $\tilde{o}$ , the objective is to minimize:

$$f(y) = (\tilde{o}[k] - y)^2 + (\tilde{o}[k + 1] - y)^2 + \sum_{j \notin \{k, k+1\}} (\tilde{o}[j] - \bar{o}[j])$$

and on input  $\tilde{o}'$  the objective is to minimize:

$$g(y) = 2((\tilde{o}[k] + \tilde{o}[k+1])/2 - y)^2 + \sum_{j \notin \{k, k+1\}} (\tilde{o}[j] - \bar{o}[j])$$

Algebraic manipulation reveals that  $g(y) = f(y) + c$  where  $c$  is constant with respect to  $y$ . Therefore, both  $f$  and  $g$  are minimized at the same value of  $y$  and we have shown that the minimal  $L_2$  solution is the same for  $\tilde{o}$  and  $\tilde{o}'$ .  $\square$

The algorithm exploits this observation as follows. Starting at the end of the sequence, it descends the sequence until an out-of-order pair is found, say  $\tilde{o}[k]$  and  $\tilde{o}[k+1]$ . This pair of observations is replaced with two observations equal to their average. This may result in a new out-of-order pair – the revised  $\tilde{o}'[k+1]$  may now be larger than  $\tilde{o}[k+2]$  – so the algorithm extends the average to include later observations until it is less than or equal to the next observation in the sequence. This entire subsequence of observations is replaced by a uniform subsequence equal to their average. The algorithm then descends the sequence to find the next out-of-order pair. When all pairs are in order, the algorithm has found the minimum  $L_2$  solution.

Algorithm 3 is a pseudocode implementation. The altered sequence is stored on a stack, denoted  $S$ , where each item in the stack is a uniform subsequence, represented by its length and value. In lines 2-16, the sequence is descended to identify out-of-order observations. Line 4 checks whether the current observation is out-of-order. If so, then the current observation is averaged with the observations stored in the stack until the average is less than or equal to the next observation in the stack (lines 7-11), and this average is placed on the stack (line 12). If it is not out-of-order, then the current observation is simply pushed onto the stack (line 14). Lines 18-24 transform the stack representation into a sequence.

The time and space complexity of Algorithm 3 is  $O(n)$ . First, iterating through the observations (lines 2-16) requires linear time: in each step, the run time of the while loop (lines 7-11) is proportional to the number of items popped from  $S$  during

---

**Algorithm 3** An algorithm for computing  $\bar{o} = \text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$

---

```

1:  $first \leftarrow \langle length : 1, value : \tilde{o}[n] \rangle$ 
2: for  $k$  from  $n - 1$  to 1 do
3:    $next \leftarrow S.top()$ 
4:   if  $\tilde{o}[k] > next.value$  then
5:      $sum \leftarrow \tilde{o}[k]$ 
6:      $len \leftarrow 1$ 
7:     while  $S \neq \emptyset$  and  $\frac{sum}{len} > next.value$  do
8:        $sum \leftarrow sum + next.value \times next.length$ 
9:        $len \leftarrow len + next.length$ 
10:       $next \leftarrow S.pop()$ 
11:    end while
12:     $S.push(\langle length : len, value : sum \rangle)$ 
13:  else
14:     $S.push(\langle length : 1, value : \tilde{o}[k] \rangle)$ 
15:  end if
16: end for
17:
18:  $\bar{o} \leftarrow \langle \rangle$ 
19: while  $S \neq \emptyset$  do
20:    $next \leftarrow S.pop()$ 
21:   for 1 to  $next.length$  do
22:      $\bar{o}.append(next.value)$ 
23:   end for
24: end while
25: return  $\bar{o}$ 

```

---

this step. Since each iteration pushes at most one item onto the stack (line 12 or 14), the total number of pops is at most  $n$ . Therefore the amortized cost of a single step is  $O(1)$ . Second, once the stack  $S$  is completed, reconstructing  $\bar{o}$  (lines 18-24) takes only linear time. In the worst-case the stack  $S$  can require  $O(n)$  space. However, only the top of the stack is accessed during computations and the rest can be written to disk as needed.

**Theorem 5.4.** [*Correctness*] Algorithm 3 computes  $\text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$ .

*Proof.* The proof follows from Lemma 5.1. The algorithm takes the input sequence and constructs an alternate sequence (lines 4-11) such that, according to Lemma 5.1, its minimum  $L_2$  solution is the same as the input sequence.

At step  $k$ , let  $\tilde{o}_k$  denote the alternate sequence  $\tilde{o}_k = \langle \tilde{o}[1], \dots, \tilde{o}[k], \tilde{o}_S[k+1], \dots, \tilde{o}_S[n] \rangle$  where  $\tilde{o}_S[k+1, \dots, n]$  corresponds to the subsequence stored on the stack. The invariant of the algorithm is that the minimum  $L_2$  solution for  $\tilde{o}_k$  is equal to the minimum  $L_2$  solution for  $\tilde{o}$ .

Once  $k = 1$ ,  $\tilde{o}_1$  is stored entirely on the stack  $S$  and it must satisfy the ordering constraints, as the while loop ensures that subsequences are only pushed onto the stack if they obey the ordering constraints. Thus, the minimum  $L_2$  solution for  $\tilde{o}_1$  is itself and lines 18-24 simply transforms  $\tilde{o}_1$  from its stack representation to a sequence representation.  $\square$

## 5.5 Experiments

The primary goal of the experiments is to assess the utility of our proposed techniques for estimating unattributed histograms. We compare three strategies: the baseline strategy of computing a conventional histogram and ignoring attribution, plus our two techniques based on carefully formulated queries followed by inference ( $\bar{\mathbf{S}}$  and  $\bar{\mathbf{F}}$ ).

We compare the techniques through several means. Each technique produces an unattributed histogram, which can be viewed as simply a collection of numbers, and we can measure its distribution. For each technique, we measure how closely its distribution resembles the true distribution (the distribution of an unattributed histogram computed on the private data). We do this several ways: through visual comparison, using metrics such as KS distance and Mallows distance, comparing to sampling error, and by carrying out a common analyses on the distribution, such as modeling. We fit the parameters of a power-law model to each distribution and compare their fit.

Asking a conventional histogram and then ignoring attribution yields the same answer as asking the  $\mathbf{S}$  query. Therefore, in the experiments,  $\tilde{\mathbf{S}}$  represents both the

input to the inference-based approach  $\bar{\mathbf{S}}$  and the strategy of asking a conventional histogram. For most of the experiments, we focus on  $\bar{\mathbf{S}}$  and  $\tilde{\mathbf{S}}$  and but we compare  $\bar{\mathbf{S}}$  with  $\bar{\mathbf{F}}$  in Section 5.5.2.

Our experiments show that  $\bar{\mathbf{S}}$  and  $\bar{\mathbf{F}}$  are very accurate techniques for computing unattributed histograms under differential privacy. They are much more accurate than the baseline strategy of using the Laplace mechanism to compute a conventional histogram ( $\tilde{\mathbf{S}}$ ). The accuracy of  $\bar{\mathbf{S}}$  and  $\bar{\mathbf{F}}$  is not uniform across the entire distribution, and in fact, the techniques behave differently, achieving highest accuracy in different parts of the distribution.

In addition to studying utility, we also assess the scalability of the inference algorithm. Our evaluation shows that inference is very fast and can scale to large inputs.

**Datasets** We focus on one the most compelling application of unattributed histograms: measuring the degree sequence of a network. We use data derived from real social networks and apply our technique to estimating the network’s degree sequence. The datasets are derived from crawls of four online social networking sites: **Flickr** ( $\approx 1.8\text{M}$  nodes), **LiveJournal** ( $\approx 5.3\text{M}$ ), **Orkut** ( $\approx 3.1\text{M}$ ), and **YouTube** ( $\approx 1.1\text{M}$ ) [95]. To the best of our knowledge, these are the largest publicly available social network datasets. We also evaluate the technique synthetic network including **Random**, a classical random graph, which has a Poisson degree distribution ( $\lambda = 10$ ), and **Power**, a random graph with a power-law degree distribution ( $\alpha = 1.5$ ).

### 5.5.1 Utility

We use two measures to measure the accuracy of the estimated distribution. First, we use the Kolmogorov-Smirnoff (KS) statistic, a measure used to test whether two samples are drawn from the same distribution. Let the empirical cumulative distribu-

tion function (CDF) of sample  $X = X_1, \dots, X_n$  be defined as  $F_X(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{I}[X_i \leq x]$ . Then the KS statistic between  $X$  and  $Y$  is  $KS(X, Y) = \max_x |F_X(x) - F_Y(x)|$ .

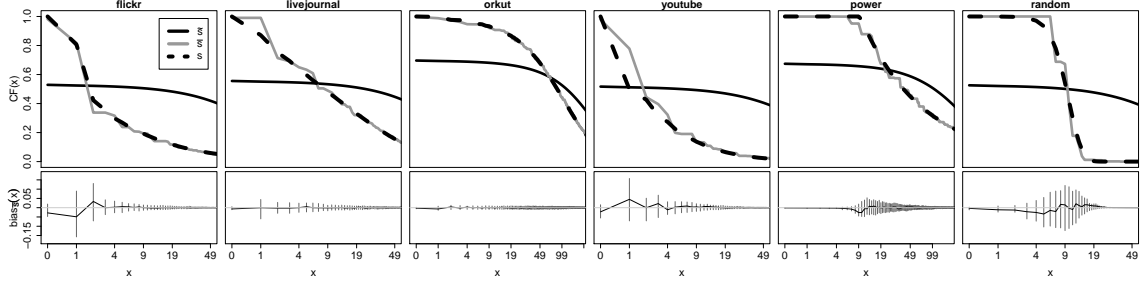
The KS statistic is insensitive to differences in the tails of the two distributions, so we also use the Mallows distance (aka Earth Mover's distance) to capture deviations in the tail. Given samples  $X$  and  $Y$  each of size  $n$ , with  $X_{(i)}$  denoting the  $i^{\text{th}}$  largest sample in  $X$ , the Mallows  $p$ -distance is

$$\text{Mallows}_p(X, Y) = \left( \frac{1}{n} \sum_{i=1}^n |X_{(i)} - Y_{(i)}|^p \right)^{1/p}$$

An example shows how Mallows distance is more sensitive than the KS statistic to the tail of the distribution. Consider three graphs  $A$ ,  $B$ , and  $C$  in which all nodes have degree 1, except in  $B$  one node has degree 2 and in  $C$  one node has degree  $n - 1$ . The KS statistic between  $A$  and either  $B$  or  $C$  is  $O(n^{-1})$ . The Mallows distance ( $p = 1$ ) between  $A$  and  $B$  is  $O(n^{-1})$ , but between  $A$  and  $C$ , the Mallows distance is  $O(1)$ , capturing the difference between their largest degrees.

**A visual comparison of distributions** Figure 5.4 shows the true degree distribution along with the differentially private approximations, revealing that  $\bar{\mathbf{S}}$  produces a very accurate approximation while  $\tilde{\mathbf{S}}$  does not. The distributions are represented using the complementary CDF (CCDF), denoted  $CF$  and defined as  $CF_X(x) = 1 - F_X(x)$ . Thus, each line shows what fraction of nodes have a degree greater than the given value on the x-axis. Abusing notation, we use  $\mathbf{S}(I)$ ,  $\tilde{s}$ , and  $\bar{s}$ , which are all degree sequences, to refer to their corresponding degree distributions. Thus, the line labeled  $\mathbf{S}(I)$  refers to the true degree distribution and the lines labeled  $\tilde{s}$  and  $\bar{s}$  refer to the degree distributions derived from differentially private sequences  $\tilde{s}$  and  $\bar{s}$  (here  $\epsilon = 0.01$ ).

Figure 5.4 shows that noise added to produce  $\tilde{s}$  substantially distorts the degree distribution. In contrast,  $\bar{s}$  is a much more accurate approximation of  $\mathbf{S}(I)$ . While



**Figure 5.4.** Complementary CDFs of  $\mathbf{S}(I)$ ,  $\tilde{s}$  and  $\bar{s}$  (top). Bias of  $\bar{\mathbf{S}}$  (bottom).

$\bar{s}$  exhibits some deviations from the true distribution, the deviations appear to oscillate around the true distribution. This demonstrates that, by exploiting the sort constraints, constrained inference can filter out much of the noise in  $\tilde{s}$ .

**Bias & variance analysis** In addition to showing individual samples  $\tilde{s}$  and  $\bar{s}$ , we also analyze the bias and variance of randomized algorithms  $\tilde{\mathbf{S}}$  and  $\bar{\mathbf{S}}$ . More precisely, we measure bias of  $\bar{\mathbf{S}}$  as the expected difference between the CCDFs of  $\bar{\mathbf{S}}$  and  $\mathbf{S}(I)$  for each degree—i.e.,  $\text{bias}_{\bar{\mathbf{S}}}(x) = \mathbb{E}[CF_{\bar{\mathbf{S}}}(x) - CF_{\mathbf{S}(I)}(x)]$  where the expectation is over the randomness in  $\bar{\mathbf{S}}$ . The variance of  $\bar{\mathbf{S}}$  is  $\text{var}_{\bar{\mathbf{S}}}(x) = \mathbb{E}[(CF_{\bar{\mathbf{S}}}(x) - \mathbb{E}[CF_{\bar{\mathbf{S}}}(x)])^2]$ . We focus on  $\bar{\mathbf{S}}$  because it is evident from Figure 5.4 that  $\tilde{\mathbf{S}}$  exhibits substantial bias.

We evaluate the bias/variance of  $\bar{\mathbf{S}}$  empirically thru repeated sampling. The results are shown in the bottom panel of Figure 5.4. The y-axis is the difference in cumulative probability between  $\mathbf{S}$  and  $\bar{\mathbf{S}}$ ,  $CF_{\bar{\mathbf{S}}}(x) - CF_{\mathbf{S}(I)}(x)$ . The line shows the average difference (bias) and the error bars depict the standard deviation from the average (square root of variance). The line remains near 0, suggesting that  $\bar{\mathbf{S}}$  may be an unbiased or nearly unbiased estimator of  $\mathbf{S}(I)$ . The variance peaks wherever the CCDF exhibits steepest change.

**Accuracy vs.  $\epsilon$**  Figures 5.5 and 5.6 show the relationship between  $\epsilon$  and accuracy for two measures of accuracy—KS in 5.5, Mallows in 5.6. We report the average

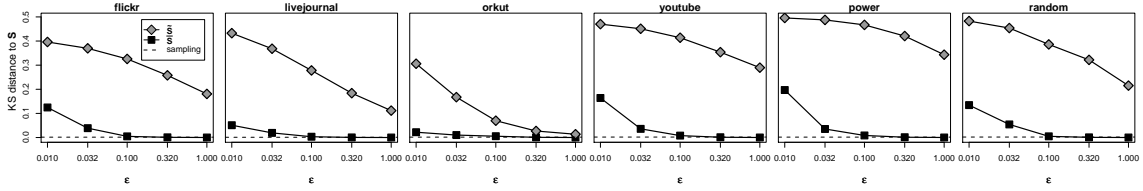


Figure 5.5. Accuracy (KS distance) across varying  $\epsilon$ .

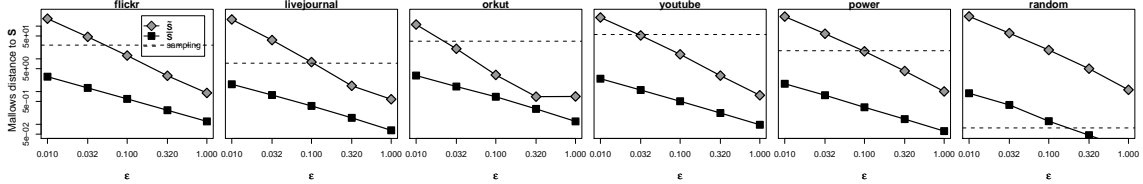
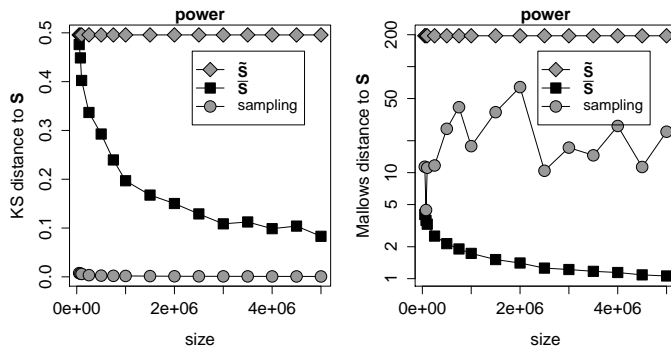


Figure 5.6. Accuracy (Mallows distance with  $p = 2$ ) across varying  $\epsilon$ .

accuracy over 10 trials (random samplings of  $\tilde{s}$ ). In both figures, the parameter  $\epsilon$  varies along horizontal axis—smaller  $\epsilon$  corresponds to greater privacy protection.

The results show that  $\bar{\mathbf{S}}$  is uniformly more accurate than  $\tilde{\mathbf{S}}$ , across all datasets, settings of  $\epsilon$ , and both measures of accuracy. Furthermore, for low settings of  $\epsilon$  (stronger privacy), the difference in accuracy is greater, suggesting that the benefit of constrained inference increases with privacy.

Also shown in the figure is the accuracy of an estimate based on random sampling (10% of the degrees are sampled uniformly at random). While sampling does not provide differential privacy, it can serve as a useful reference point. Sampling has very low KS distance (as expected), but higher Mallows distance because random sampling is unlikely to select the high degree nodes in the tail. In fact, sampling has higher Mallows distance than  $\bar{\mathbf{S}}$  (except on **Random**, which is a distribution without long tails). Since analysts often cannot obtain complete graphs and must rely on samples, this result suggests that the additional error due to privacy can be small compared to the sampling error.

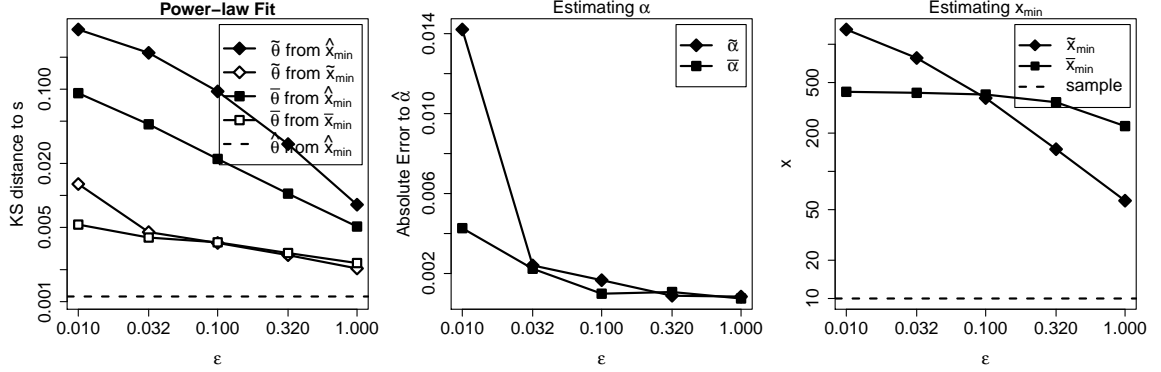


**Figure 5.7.** Size vs. Accuracy for fixed  $\epsilon = 0.01$ .

**Accuracy vs. size** Figure 5.7 shows how accuracy of  $\bar{S}$  improves as the graph increases in size. The figure reports accuracy on **Power** graphs of varying size, from 10K to 5M nodes. The results show a clear separation between  $\tilde{S}$  and  $\bar{S}$ : as the size of the graph increases, the accuracy of  $\tilde{S}$  remains constant whereas the accuracy of  $\bar{S}$  improves. Thus, with  $\bar{S}$ , larger datasets yield either more privacy (given a fixed accuracy target, we can lower  $\epsilon$ ) or better utility (higher accuracy for fixed  $\epsilon$ ).

The accuracy of  $\tilde{S}$  does not improve with graph size because random noise is added to each degree, thus the average error per degree does not change with the size of the graph. However, as Example 8 showed,  $\bar{S}$  can be very accurate when the degree sequence contains long subsequences of uniform degrees. As the graph size increases, accuracy improves because the subsequences of uniform degree grow longer (in a power-law graph, the expected proportion of nodes with a given degree is a constant independent of  $n$ ).

**Modeling power-law distributions** Our final experiment assesses how accurately the analyst can estimate the parameters of a power-law model using  $\tilde{S}$  or  $\bar{S}$ . The experiment is designed as follows. First, we sample a **Power** graph with parameters  $\theta = (\alpha = 1.5, x_{min} = 10)$ . We fix this as the true degree distribution. Then we sample  $\tilde{s}$  and  $\bar{s}$  and derive corresponding distributions. To each of these three degree distributions, we fit a power-law model using maximum likelihood [27]. The result is



**Figure 5.8.** Accuracy of estimating power-law model using  $\tilde{\mathbf{S}}, \bar{\mathbf{S}}$ .

three different estimates for the parameters  $\theta$ , which we denote  $\hat{\theta}$ ,  $\tilde{\theta}$ , and  $\bar{\theta}$  respectively. We are interested in comparing the model fit to the true degree distribution,  $\hat{\theta}$ , to the models fit under differential privacy,  $\tilde{\theta}$  and  $\bar{\theta}$ .

The individual parameter estimates are shown in the middle and right plot of Figure 5.8, but the leftmost plot provides a holistic assessment of the model fit. It assesses model fit using the  $D$  statistic of Clauset et al. [27] which measures the KS statistic on the power-law tail of the distribution. We consider two variants of this measure: in one, the tail is defined by the estimate of  $x_{min}$  under  $\tilde{s}$  or  $\bar{s}$ ; in the other,  $x_{min}$  is based on the true  $x_{min}$ .

The plots reveal that using either  $\tilde{\mathbf{S}}$  or  $\bar{\mathbf{S}}$ , the analyst will estimate a model that has a close fit to the tail of the original (power-law) distribution, when the tail is defined by the  $x_{min}$  estimated on the noisy distribution. However, it also shows that the size of the tail is under-estimated (the power-law behavior becomes apparent only for large degrees). If we compare the models based on how well they fit the true tail of the power-law distribution (solid lines of leftmost plot), we see that  $\tilde{\mathbf{S}}$  has considerable distortion (note the log-scale) while  $\bar{\mathbf{S}}$  is reasonably accurate even at small  $\epsilon$ .

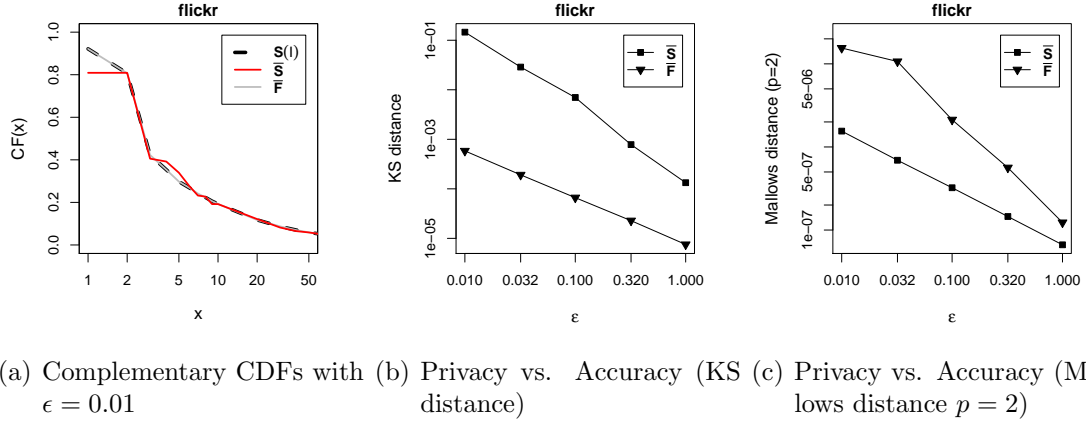


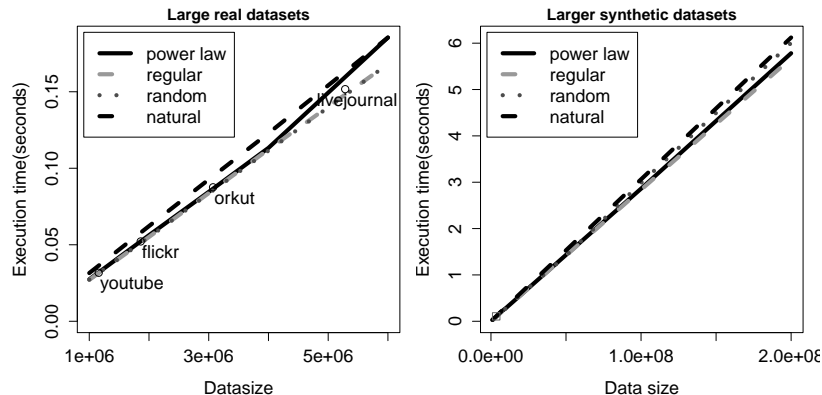
Figure 5.9. On Flickr, a comparison of  $\bar{S}$  and  $\bar{F}$ .

### 5.5.2 Utility comparison: $\bar{S}$ vs. $\bar{F}$

We now compare the query strategy  $\bar{S}$  with the alternative strategy  $\bar{F}$ . The results are shown in Figure 5.9. We show results on **Flickr**, but the results are representative of the trends across all datasets. Figure 5.9(a) shows the complementary CDFs of a typical sample at  $\epsilon = 0.01$ . Both techniques approximate the true distribution quite well; but  $\bar{F}$  appears to be much more accurate, at least for the degrees shown ( $\leq 50$ ). This higher accuracy also reflected in the comparison of KS Distance: Figure 5.9(b) shows that  $\bar{F}$  has much lower KS distance than  $\bar{S}$ . However, Figure 5.9(c) shows that  $\bar{F}$  is less accurate when it comes to Mallows distance. The higher Mallows distance is due to the fact that  $\bar{F}$  is not as accurate in the high degree nodes at the tail of the degree distribution.

### 5.5.3 Scalability of inference

Figure 5.10 shows that the runtime of Algorithm 3 scales linearly and is extremely fast. The left figure shows the runtime on the real datasets and the right figure shows the runtime on even larger synthetic datasets of up to 200M nodes. In addition to **Random** and **Power**, we include two non-random synthetic distributions, corresponding to the best- and worst-case inputs for the runtime of the algorithm.



**Figure 5.10.** Runtime of Algorithm 3 on real (left) and larger synthetic datasets (right).

The best-case is **Regular**, a uniform degree distribution (all nodes have degree 10), the worst-case is **Natural**, a distribution having one occurrence of each degree in  $\{0, \dots, n - 1\}$ .

The small variation in runtime across datasets shows that it is not particularly sensitive to the type of degree distribution. Furthermore, it is extremely fast: processing a 200 million node graph takes less than 6 seconds. The efficiency of the algorithm makes the constrained inference approach practical for large graphs.

## 5.6 Conclusion

We introduced the concept of an unattributed histogram and showed how to compute it accurately under differential privacy. An unattributed histogram is in some sense less informative than a conventional histogram, but nevertheless has several practical applications. For example, the degree sequence of a graph is an instance of an unattributed histogram. Its significance lies in the context of the analysis of private data: under differential privacy, it is possible, using our technique, to estimate an unattributed much more accurately than a conventional histogram.

Our technique for estimating an unattributed histogram satisfies differential privacy, scales to large inputs, and produces extremely accurate approximations. The

guarantee of differential privacy means that, unlike approaches based on anonymization, it provides extremely robust protection, even against powerful adversaries. Our algorithm runs in linear time and is demonstrably fast. The output is accurate and has provably bounded error that scales with the number of distinct frequencies of a histogram, rather than the number of bins.

Our technique achieves high accuracy by post-processing the noisy output of the Laplace mechanism, a differentially private mechanism for answering an arbitrary query sequence. The fact that we are able to reduce error comes from slack in the system: the Laplace mechanism adds more noise than necessary to achieve differential privacy. While it adds independent and identically distributed noise to each answer in the sequence, intuitively, it seems that less noise is needed in the middle of uniform subsequences, because changes in the database can only induce changes at the ends of uniform subsequences. Our inference mechanism effectively reduces noise in the regions of uniformity, while the noise at the regions' ends remains sufficiently large to protect privacy.

A natural goal is to describe directly the improved noise distributions implied by our inference technique, and build a privacy mechanism that samples from it. This could, in theory, avoid the inference step altogether. But it seems quite difficult to discover, describe, and sample these improved noise distributions, which will be highly dependent on a particular query of interest. Our approach suggests that constraints and statistical inference can be an effective path to discovering new, more accurate noise distributions that satisfy differential privacy. As a practical matter, our approach does not necessarily burden the analyst with the inference process because the server can implement the post-processing step. In that case it would appear to the analyst as if the server was sampling from the improved distribution.

We proposed two query strategies for unattributed histograms,  $\bar{\mathbf{S}}$  and  $\bar{\mathbf{F}}$ , and showed experimentally how each is impacted by the added noise: with  $\bar{\mathbf{S}}$ , the largest

distortion occurs at low degrees, with  $\overline{\mathbf{F}}$ , the largest distortion occurs in the tail of high degrees. A successful strategy may be to ask *both*  $\overline{\mathbf{S}}$  and  $\overline{\mathbf{F}}$ , and combine the answers using inference. Asking two queries requires increasing the magnitude of the noise, but the accuracy boost from inference may offset the extra noise.

Finally, given the importance of the degree sequence to the structure of a graph, we believe that our techniques are a critical first step towards the ultimate goal of publishing synthetic graphs that are both accurate and ensure differential privacy.

## 5.7 Proofs

### 5.7.1 Proof of Theorem 5.1

We first restate the theorem below.

**Theorem 5.1.** *[Minimum  $L_2$  Solution for ordering constraints] For a given  $\tilde{o}$ , let  $\tilde{o}[i, j]$  denote the subsequence  $\langle \tilde{o}[i], \tilde{o}[i + 1], \dots, \tilde{o}[j] \rangle$  and  $M[i, j]$  the average of this subsequence:  $M[i, j] = \sum_{k=i}^j \tilde{o}[k] / (j - i + 1)$ . For each position  $k$ , let  $L_k$  and  $U_k$  be defined as*

$$L_k = \min_{k \leq j \leq n} \max_{1 \leq i \leq n} M[i, j]$$

and

$$U_k = \max_{1 \leq i \leq k} \min_{i \leq j \leq n} M[i, j]$$

The minimum  $L_2$  solution,  $\bar{o} = \text{MIN}L_2(\tilde{o}, \gamma_{\mathbf{O}})$ , is unique and is equal to  $\bar{o}[k] = L_k = U_k$ .

*Proof.* In the proof, we abbreviate the notation and implicitly assume that the range of  $i$  is  $[1, n]$  or  $[1, j]$  when  $j$  is specified. Similarly, the range of  $j$  is  $[1, n]$  or  $[i, n]$  when  $i$  is specified.

We start with the easy part, showing that  $U_k \leq L_k$ . Define an  $n \times n$  matrix  $A^k$  as follows:

$$A_{ij}^k = \begin{cases} M[i, j] & \text{if } i \leq j \\ \infty & \text{if } j < i \leq k \\ -\infty & \text{otherwise} \end{cases}$$

Then  $\min_j \max_i A_{ij}^k = L_k$  and  $\max_i \min_j A_{ij}^k = U_k$ . In any matrix  $A^k$ ,  $\max_i \min_j A_{ij}^k \leq \min_j \max_i A_{ij}^k$ : this is a simple fact that can be checked directly, or see [103], hence  $U_k \leq L_k$ .

We show next that if  $\bar{o}$  is the minimum  $L_2$  solution, then  $L_k \leq \bar{o}[k] \leq U_k$ . If we show this, then the proof of the theorem is completed, as then we will then have  $\bar{o}[k] = L_k = U_k$ . The proof relies on the following lemma.

**Lemma 5.2.** *Let  $\bar{o}$  be the minimum  $L_2$  solution. Then (i)  $\bar{o}[1] \leq U_1$ , (ii)  $\bar{o}[n] \geq L_n$ , (iii) for all  $k$ ,  $\min(\bar{o}[k+1], \max_i M[i, k]) \leq \bar{o}[k] \leq \max(\bar{o}[k-1], \min_j M[k, j])$ .*

The proof of the lemma appears below, but now we use it to complete the proof of Theorem 5.1. First, we show that  $\bar{o}[k] \leq U_k$  using induction on  $k$ . The base case is  $k = 1$  and it is stated in the lemma, part (i). For the inductive step, assume  $\bar{o}[k-1] \leq U_{k-1}$ . From (iii), we have that

$$\begin{aligned} \bar{o}[k] &\leq \max(\bar{o}[k-1], \min_j M[k, j]) \\ &\leq \max(U_{k-1}, \min_j M[k, j]) = U_k \end{aligned}$$

The last step follows from the definition of  $U_k$ . A similar induction argument shows that  $\bar{o}[k] \geq L_k$ , except the order is reversed: the base case is  $k = n$  and the inductive step assumes  $\bar{o}[k+1] \geq L_{k+1}$ . □

The only remaining step is to prove the lemma.

of Lemma 5.2. For (i), it is sufficient to prove that  $\bar{o}[1] \leq M[1, j]$  for all  $j \in [1, n]$ . Assume the contrary. Thus there exists a  $j$  such that for  $\bar{o}[1] > M[1, j]$ . Let  $\delta = \bar{o}[1] - M[1, j]$ . Thus  $\delta > 0$ . Further, for all  $i$ , denote  $\delta_i = \bar{o}[i] - \bar{o}[1]$ . Consider the sequence  $\bar{o}'$  defined as follows:

$$\bar{o}'[i] = \begin{cases} \bar{o}[i] - \delta & \text{if } i \leq j \\ \bar{o}[i] & \text{otherwise} \end{cases}$$

It is obvious to see that since  $\bar{o}$  is a sorted sequence, so is  $\bar{o}'$ .

We now claim that  $\|\bar{o}' - \tilde{o}\|_2 < \|\bar{o} - \tilde{o}\|_2$ . For this note that since the sequence  $\bar{o}'[j+1, n]$  is identical to the sequence  $\bar{o}[j+1, n]$ , it is sufficient to prove  $\|\bar{o}'[1, j] - \tilde{o}[1, j]\|_2 < \|\bar{o}[1, j] - \tilde{o}[1, j]\|_2$ . To prove that, note that  $\|\bar{o}[1, j] - \tilde{o}[1, j]\|_2$  can be expanded as

$$\begin{aligned} \|\bar{o}[1, j] - \tilde{o}[1, j]\|_2 &= \sum_{i=1}^j (\bar{o}[i] - \tilde{o}[i])^2 = \sum_{i=1}^j (\bar{o}[1] + \delta_i - \tilde{o}[i])^2 \\ &= \sum_{i=1}^j (M[1, j] + \delta + \delta_i - \tilde{o}[i])^2 \end{aligned}$$

Suppose for a moment that we fix  $M[1, j]$  and  $\delta_i$ 's, and treat  $\|\bar{o}[1, j] - \tilde{o}[1, j]\|_2$  as a function  $f$  over  $\delta$ . The derivative of  $f(\delta)$  is:

$$\begin{aligned} f'(\delta) &= 2 \sum_{i=1}^j (M[1, j] + \delta + \delta_i - \tilde{o}[i]) \\ &= 2(jM[1, j] - \sum_{i=1}^j \tilde{o}[i]) + 2j\delta + 2 \sum_{i=1}^j \delta_i \\ &= 2j\delta + 2 \sum_{i=1}^j \delta_i \end{aligned}$$

Since  $\delta_i \geq 0$  for all  $i$ , then the derivative is strictly greater than zero for any  $\delta > 0$ , which implies that  $f$  is a strictly increasing function of  $\delta$  and has a minimum at

$\delta = 0$ . Therefore,  $\|\bar{o}[1, j] - \tilde{o}[1, j]\|_2 = f(\delta) > f(0) = \|\bar{o}'[1, j] - \tilde{o}[1, j]\|_2$ . This is a contradiction since it was assumed that  $\bar{o}$  was the minimum solution. This completes the proof for (i).

For (ii), the proof of  $\bar{o}[n] \geq \max_i M[i, n]$  follows from a similar argument: if  $\bar{o}[n] < M[i, n]$  for some  $i$ , define  $\delta = M[i, n] - \bar{o}[n]$  and the sequence  $\bar{o}'$  with elements  $\bar{o}'[j] = \bar{o}[j] + \delta$  for  $j \geq i$ . Then  $\bar{o}'$  can be shown to be a strictly better solution than  $\bar{o}$ , proving (ii).

For the proof of (iii), we first show that  $\bar{o}[k] \leq \max(\bar{o}[k-1], \min_j M[k, j])$ . Assume the contrary, i.e. there exists a  $k$  such that  $\bar{o}[k] > \bar{o}[k-1]$  and  $\bar{o}[k] > \min_j M[k, j]$ . In other words, we assume there exists a  $k$  and  $j$  such that  $\bar{o}[k] > \bar{o}[k-1]$  and  $\bar{o}[k] > M[k, j]$ . Denote  $\delta = \bar{o}[k] - \max(\bar{o}[k-1], M[k, j])$ . By our assumption above,  $\delta > 0$ . Define the sequence

$$\bar{o}'[i] = \begin{cases} \bar{o}[i] - \delta & \text{if } k \leq i \leq j \\ \bar{o}[i] & \text{otherwise} \end{cases}$$

Note that by construction,  $\bar{o}'[k] = \bar{o}[k] - \delta = \bar{o}[k] - (\bar{o}[k] - \max(\bar{o}[k-1], M[k, j])) = \max(\bar{o}[k-1], M[k, j])$ . It is easy to see that  $\bar{o}'$  is sorted (indeed the only inversion in the sort order could have occurred if  $\bar{o}'[k-1] > \bar{o}'[k]$ , but doesn't as  $\bar{o}'[k-1] = \bar{o}[k-1] \leq \max(\bar{o}[k-1], M[k, j]) = \bar{o}'[k]$ ).

Now a similar argument as in the proof of (i) for the sequence  $\tilde{o}[k, j]$ , yields that the error  $\|\bar{o}'[k, j] - \tilde{o}[k, j]\|_2 < \|\bar{o}[k, j] - \tilde{o}[k, j]\|_2$ . Thus  $\|\bar{o}' - \tilde{o}\|_2 < \|\bar{o} - \tilde{o}\|_2$  and  $\bar{o}'$  is a strictly better solution than  $\bar{o}$ . This yields a contradiction as  $\bar{o}$  is the minimum  $L_2$  solution. Hence  $\bar{o}[k] \leq \max(\bar{o}[k-1], \min_j M[k, j])$ .

A similar argument in the the reverse direction shows that  $\bar{o}[k] \geq \min(\bar{o}_{k+1}, \max_i M[i, k])$  completing the proof of (iii).  $\square$

### 5.7.2 Proof of Theorem 5.2

We prove the following result.

**Theorem 5.2.** *[Minimum  $L_2$  Solution for order, integrality, and bound constraints]*  
 Given  $\bar{o} = \text{MIN}L_2(\bar{o}, \gamma_{\mathbf{O}})$ , let  $\bar{o}'$  denote the sequence derived from  $\bar{o}$  in which each element  $\bar{o}[k]$  is rounded to the nearest integer in  $[L, U]$ . Then  $\bar{o}' = \text{MIN}L_2(\bar{o}, \gamma'_{\mathbf{O}})$ .

**Lemma 5.3.** *If  $\bar{o}_k = \bar{o}_{k+1} = \dots = \bar{o}_{j^*} = M[k, j^*]$ , for any  $k \leq i \leq j^*$ ,  $M[i, j^*] \leq M[k, j^*]$ .*

*Proof.* According to Theorem 5.1,  $M[k, j^*] = \min_{j \geq k} M[k, j]$ . If there exists an  $k \leq i^* \leq j^*$  such that  $M[i^*, j^*] > M[k, j^*]$ , we have  $M[k, i^*] < M[k, j^*]$ , which comes to a contradiction.  $\square$

**Theorem 5.5** (Minimum  $L_2$  with boundary constraints). *Let  $\{\gamma_{\mathbf{S}}, [L, U]\}$  be the constraint set  $\gamma_{\mathbf{S}}$  augmented with the additional constraint that each count be in range  $[L, U]$ . Given  $\bar{o} = \text{MIN}L_2((, \bar{o}), \gamma_{\mathbf{S}})$ , let  $\bar{o}^{L,U}$  denote the sequence derived from  $\bar{o}$  in which*

$$\bar{o}_k^{L,U} = \begin{cases} L & \bar{o}_k \leq L \\ \bar{o}_k & L < \bar{o}_k < U \\ U & \bar{o}_k \geq U \end{cases}$$

*Then  $\bar{o}^{L,U} = \text{MIN}L_2((, \bar{o}), \{\gamma_{\mathbf{S}}, [L, U]\})$ .*

*Proof.* Without loss of generality, here we just consider the case  $L = -\infty$  and prove it by contradiction. Assume that  $\bar{o}' = \text{MIN}L_2((, \bar{o}), \{\gamma_{\mathbf{S}}, [L, U]\})$ . Let  $k_0 = \min_k \{\bar{o}_k \geq U\}$ . Since  $\bar{o}_{k_0} \geq U > \bar{o}_{k_0-1}$ , according to Theorem 5.1,  $\bar{o}_{k_0} = \min_{j \geq k_0} M[k_0, j]$ . If  $\bar{o}'_{k_0} < U$ , let  $k_1 = \max_k \{\bar{o}'_k = \bar{o}'_{k_0}\}$ . Consider function  $f(x) = \sum_{k=k_0}^{k_1} (\bar{o}_k - x)^2$ ,

$f'(x) = 2(k_1 - k_0 + 1)(x - M[k_0, k_1])$ . Since  $\bar{o}'_{k_0} < \bar{o}'_{k_1+1} \leq U \leq \bar{o}_{k_0} \leq M[k_0, k_1]$ , we know

$$\sum_{k=k_0}^{k_1} (\tilde{o}_k - \bar{o}'_{k_0})^2 = f(\bar{o}'_{k_0}) > f(\bar{o}'_{k_1+1}) = \sum_{k=k_0}^{k_1} (\tilde{o}_k - \bar{o}'_{k_1+1})^2,$$

which contradicts with the fact that  $\bar{o}' = \text{MIN}L_2((, \tilde{o}), \{\gamma_{\mathbf{S}}, [L, U]\})$ . Therefore  $\bar{o}'_{k_0} = U$ .

Since  $\bar{o}'_{k_0} = U$  and  $\bar{o}_{k_0-1} < U$ , from the definition of  $\bar{o}$  and  $\bar{o}'$ , we know both  $\bar{o}_1, \bar{o}_2, \dots, \bar{o}_{k_0-1}$  and  $\bar{o}'_1, \bar{o}'_2, \dots, \bar{o}'_{k_0-1}$  are the minimum  $L_2$  solution of  $\tilde{o}_1, \tilde{o}_2, \dots, \tilde{o}_{k_0-1}$  that satisfies the constraint set  $\{\gamma_{\mathbf{S}}, [-\infty, U]\}$ . Since  $\bar{o}$  is unique,  $\bar{o}_1, \bar{o}_2, \dots, \bar{o}_{k_0-1}$  and  $\bar{o}'_1, \bar{o}'_2, \dots, \bar{o}'_{k_0-1}$  are identical. Above all,

$$\bar{o}^{-\infty, U} = \text{MIN}L_2((, \tilde{o}), \{\gamma_{\mathbf{S}}, [-\infty, U]\}).$$

□

According to Theorem 5.5, one can easily get the following corollaries:

**Corollary 2.** *Let  $L \leq U_1 \leq U_2$ . If  $\bar{o}_n^{L, U_1} < U_1$ ,  $\bar{o}^{L, U_1} = \bar{o}^{L, U_2}$ .*

**Corollary 3.** *Let  $L_1 \leq L_2 \leq U$ . If  $\bar{o}_1^{L_2, U} > L_2$ ,  $\bar{o}^{L_1, U} = \bar{o}^{L_2, U}$ .*

Now we can further generate Theorem 5.5 to integer case.

**Theorem 5.6** (Minimum  $L_2$  with integer and boundary constraints). *Given integer  $L, U$ . Let  $\{\gamma_{\mathbf{S}}, [L, U], \text{int}\}$  be the constraint set  $\{\gamma_{\mathbf{S}}, [L, U]\}$  augmented with the additional constraint that each count be an integer. Given  $\bar{o}^{L, U} = \text{MIN}L_2((, \tilde{o}), \{\gamma_{\mathbf{S}}, [L, U]\})$ , let  $\hat{s}^{L, U}$  denote the sequence derived from  $\bar{o}^{L, U}$  in which each element  $\bar{o}_k^{L, U}$  is rounded to its nearest integer. Then  $\hat{s}^{L, U}$  is a minimum  $L_2$  solution that satisfies the constraint set  $\{\gamma_{\mathbf{S}}, [L, U], \text{int}\}$ .*

*Proof.* Proof by induction over  $n$ . The conclusion is trivial for  $n = 1$ . Suppose the theorem hold for all  $n \leq k$ , and consider that  $n = k + 1$ .

**Case 1:**  $\bar{o}^{L,U}$  isn't a constant sequence, which means there exists  $1 \leq j^* \leq k$  such that  $L \leq \bar{o}_{j^*}^{L,U} < \bar{o}_{j^*+1}^{L,U} \leq U$ . Let's consider  $\bar{o}^{L,U}$  as two subsequences  $\bar{o}_1^{L,U}, \bar{o}_2^{L,U}, \dots, \bar{o}_{j^*}^{L,U}$  and  $\bar{o}_{j^*+1}^{L,U}, \bar{o}_{j^*+2}^{L,U}, \dots, \bar{o}_n^{L,U}$ .

Now let's consider the first subsequence. Since the subsequence  $\bar{o}_1^{L,U}, \bar{o}_2^{L,U}, \dots, \bar{o}_i^{L,U}$  is the minimum  $L_2$  solution of  $\tilde{o}_1, \tilde{o}_2, \dots, \tilde{o}_i$  that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, \tilde{o}_{i+1}]\}$ , according to Corollary 2, it is also the minimum  $L_2$  solution of  $\tilde{o}_1, \tilde{o}_2, \dots, \tilde{o}_i$  that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, U]\}$ . Therefore, according to the induction hypothesis, by rounding each element of  $\bar{o}_1^{L,U}, \bar{o}_2^{L,U}, \dots, \bar{o}_i^{L,U}$  to its nearest integer, which are  $\hat{s}_1^{L,U}, \hat{s}_2^{L,U}, \dots, \hat{s}_i^{L,U}$ , we get a minimum  $L_2$  solution of  $\tilde{o}_1, \tilde{o}_2, \dots, \tilde{o}_i$  that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, U], int\}$ .

Similarly,  $\hat{s}_{i+1}^{L,U}, \hat{s}_{i+2}^{L,U}, \dots, \hat{s}_{k+1}^{L,U}$  is a minimum  $L_2$  solution of  $\tilde{o}_{i+1}, \tilde{o}_{i+2}, \dots, \tilde{o}_{k+1}$  that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, U], int\}$ . Thus  $\hat{s}^{L,U}$  is a minimum  $L_2$  solution that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, U], int\}$ .

**Case 2:** Otherwise,  $\bar{o}^{L,U}$  is a constant sequence. If they all equal to an integer (i.e.  $L, U$  or in case that  $M[1, n]$  is an integer), then the conclusion is obvious. Otherwise, let's consider a minimum  $L_2$  solution  $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_n$  that satisfies the constraint set  $\{\gamma_{\mathbf{s}}, [L, U], int\}$ . If  $\hat{s}_1 < M[1, n]$ , let  $i_1 = \max_i \{\hat{s}_i = \hat{s}_1\}$ . Since  $M[1, i_1] \geq M[1, n]$ , to minimize  $\sum_{i=1}^{i_1} (\tilde{o}_i - \hat{s}_i)^2$ , we know  $\hat{s}_1 > \lfloor M[1, n] \rfloor$ . Here  $\lfloor x \rfloor$  is the maximum integer that smaller than or equal to  $x$ . Otherwise we can assign  $\hat{s}_1 = \dots = \hat{s}_{i_1} = \min\{\lfloor M[1, n] \rfloor, \hat{s}_{i_1+1}\}$  to get a even smaller  $L_2$  solution. Similarly, we can prove that if  $\hat{s}_n > M[1, n]$ ,  $\hat{s}_n \leq \lceil M[1, n] \rceil$ . Here  $\lceil x \rceil$  is the minimum integer that larger than or equal to  $x$ . Above all, we know  $\lfloor M[1, n] \rfloor \leq \hat{s}_1, \hat{s}_2, \dots, \hat{s}_n \leq \lceil M[1, n] \rceil$ . Let  $m = M[1, n] - \lfloor M[1, n] \rfloor$ , remind  $i_1 = \max_i \{\hat{s}_i = \hat{s}_1\}$ . Therefore:

$$\sum_{i=1}^n (\tilde{o}_i - \hat{s}_i)^2 \quad (5.3)$$

$$= \sum_{i=1}^{i_1} (\tilde{o}_i - M[1, n] + m)^2 + \sum_{i=i_1+1}^n (\tilde{o}_i - M[1, n] - (1-m))^2$$

$$= \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 \quad (5.4)$$

$$+ 2mi_1(M[1, i_1] - M[1, n]) + 2(1-m)(n-i_1)(M[1, n] - M[i_1+1, n])$$

$$+ i_1m^2 + (n-i_1)(1-m)^2 \quad (5.5)$$

According to Lemma 5.3 and Theorem 5.1,  $M[i, n] \leq M[1, n] \leq M[1, i]$ . Thus (5.5) becomes

$$(5.5) \geq \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + i_1m^2 + (n-i_1)(1-m)^2$$

$$\geq \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + n \min\{m^2, (1-m)^2\} \quad (5.6)$$

Notice that for any  $t$ ,

$$\begin{aligned} \sum_{i=1}^n (\tilde{o}_i - M[1, n] + t)^2 &= \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + nt^2 + t \sum_{i=1}^n (\tilde{o}_i - M[1, n]) \\ &= \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + nt^2 + t \left( \sum_{i=1}^n \tilde{o}_i - nM[1, n] \right) \\ &= \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + nt^2 \end{aligned}$$

we know

$$(5.6) = \min\left\{ \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + nm^2, \sum_{i=1}^n (\tilde{o}_i - M[1, n])^2 + n(m-1)^2 \right\}$$

$$= \min\left\{ \sum_{i=1}^n (\tilde{o}_i - M[1, n] + m)^2, \sum_{i=1}^n (\tilde{o}_i - M[1, n] + (m-1))^2 \right\}$$

$$= \min\left\{ \sum_{i=1}^n (\tilde{o}_i - (M[1, n] - m))^2, \sum_{i=1}^n (\tilde{o}_i - (M[1, n] - m + 1))^2 \right\}$$

Thus, assigning  $\hat{s}_i$  to the nearest integer of  $M[1, n]$  gives a minimum  $L_2$  solution.

Above all, we finish the induction and the theorem is proved.  $\square$

In particular, with the boundary condition  $[0, n - 1]$ , Theorem 5.6 becomes the Theorem 5.2.

### 5.7.3 Proof of Theorem 5.3

We first restate the theorem below. Denote  $n$  and  $d$  as the number of values and the number of distinct values in  $\mathbf{O}(I)$  respectively. Let  $n_1, n_2, \dots, n_d$  be the number of times each of the  $d$  distinct values occur in  $\mathbf{O}(I)$  (thus  $\sum_i n_i = n$ ).

**Theorem 5.3.** *[Utility of inference] There exist constants  $c_1$  and  $c_2$  independent of  $n$  and  $d$  such that*

$$\text{error}(\bar{\mathbf{O}}) \leq \sum_{i=1}^d \frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} (c_1 \log^3 n_i + c_2)$$

Thus  $\text{error}(\bar{\mathbf{O}}) = O(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} d \log^3 n)$  whereas  $\text{error}(\tilde{\mathbf{O}}) = \Theta(\frac{\Delta_{\mathbf{O}}^2}{\epsilon^2} n)$ .

Before showing the proof, we prove the following lemma.

**Lemma 5.4.** *Let  $o = \mathbf{O}(I)$  be the input sequence. Call a translation of  $o$  the operation of subtracting from each element of  $o$  a fixed amount  $\delta$ . Then  $\text{error}(\bar{\mathbf{O}}[i])$  is invariant under translation for all  $i$ .*

*Proof.* Denote  $Pr(\bar{o}|o)$  ( $Pr(\tilde{o}|o)$ ) the probability that  $\bar{o}$  ( $\tilde{o}$ ) is output on the input sequence  $o$ . Denote  $o'$ ,  $\bar{o}'$ , and  $\tilde{o}'$  the sequence obtained by translating  $o$ ,  $\bar{o}$ , and  $\tilde{o}$  by  $\delta$ , respectively.

First observe that  $Pr(\tilde{o}|o) = Pr(\tilde{o}'|o')$  as  $\tilde{o}$  and  $\tilde{o}'$  are obtained by adding the same Laplacian noise to  $o$  and  $o'$ , respectively. Using Theorem 5.1 (since all  $U_k$ 's and  $L_k$ 's shift by  $\delta$  on translating  $\tilde{o}$  by  $\delta$ ), we get that if  $\bar{o}$  is the minimum  $L_2$  solution given  $\tilde{o}$ , then  $\bar{o}'$  is the minimum  $L_2$  solution given  $\tilde{o}'$ . Thus,  $Pr(\bar{o}|o) = Pr(\bar{o}'|o')$  for

all sequences  $\bar{o}$ . Further, since  $\bar{o}[i]$  and  $\bar{o}'[i]$  yield the same  $L_2$  error with  $o[i]$  and  $o'[i]$  respectively, we get that the expected  $error(\bar{\mathbf{O}}[i])$  is same for both inputs  $o$  and  $o'$ .  $\square$

**Lemma 5.5.** *Let  $X$  be any positive random variable that is bounded ( $\lim_{x \rightarrow \infty} xPr(X > x)$  exists). Then*

$$\mathbb{E}(X) \leq \int_0^{\infty} Pr(X > x) dx$$

*Proof.* The proof follows from the following chain of equalities.

$$\begin{aligned} \mathbb{E}(X) &= \int_0^{\infty} x \frac{\partial}{\partial x} (Pr(X \leq x)) \\ &= - \int_0^{\infty} x \frac{\partial}{\partial x} (Pr(X > x)) \\ &= -[xPr(X > x)]_0^{\infty} + \int_0^{\infty} (Pr(X \leq x) - 1) dx \quad (\text{by parts}) \\ &= - \lim_{x \rightarrow \infty} xPr(X > x) + \int_0^{\infty} Pr(X > x) dx \\ &\leq \int_0^{\infty} Pr(X > x) dx \end{aligned}$$

Here the last equality follows as  $X$  is bounded and therefore the limit exists and is positive. This completes the proof.  $\square$

We next state a theorem that was shown in [25].

**Theorem 5.7** (Theorem 3.4 [25]). *Suppose that  $X_1, X_2, \dots, X_n$  are independent random variables satisfying  $X_i \leq \mathbb{E}(X_i) + M$ , for  $1 \leq i \leq n$ . We consider the sum  $X = \sum_{i=1}^n X_i$  with expectation  $\mathbb{E}(X) = \sum_{i=1}^n \mathbb{E}(X_i)$  and  $Var(X) = \sum_{i=1}^n Var(X_i)$ . Then, we have*

$$Pr(X \geq \mathbb{E}(X) + \lambda) \leq e^{\frac{-\lambda^2}{2(Var(X) + M\lambda/3)}}$$

For a random variable  $X$ , denote  $\mathbb{I}[X]$  the indicator function that  $X \geq 0$  (thus  $\mathbb{I}[X] = 1$  if  $X \geq 0$  and 0 otherwise). Using Theorem 5.7, we prove the following lemma.

**Lemma 5.6.** *Suppose  $i, j$  are indices such that for all  $k \in [i, j]$ ,  $o[k] \leq 0$ . Then there exists a constant  $c$  such that for all  $\tau \geq 1$  the following holds.*

$$Pr \left( \tilde{M}[i, j]^2 \mathbb{I}[\tilde{M}[i, j]] \geq c \left( \frac{\log^2((j-i+1)\tau)}{(j-i+1)\epsilon^2} \right) \right) \leq \frac{1}{(j-i+1)^2 \tau^2}$$

*Proof.* We apply Theorem 5.7 on  $\tilde{o}[k]$  for  $k \in [i, j]$ . First note that  $\mathbb{E}(\tilde{o}[k]) = o[k] \leq 0$ . Further  $Var(\tilde{o}[k]) = \frac{2}{\epsilon^2}$  as  $\tilde{o}[k]$  is obtained by adding Laplace noise to  $o[k]$  which has this variance. We also know that  $\tilde{o}[k] \geq M + o[k]$  happens with probability at most  $e^{-\epsilon M}/2$ .

For simplicity, call  $n$  to be  $j - i + 1$ . Denoting  $X = \sum_{k \in [i, j]} \tilde{o}[k]$ , we see that  $\mathbb{E}(X) \leq 0$  and  $Var(X) = \frac{2n}{\epsilon^2}$ . Further, set  $M = 3 \log(n\tau)/\epsilon$ . Denote  $B$  the event that for some  $k$ ,  $\tilde{o}[k] \geq M + o[k]$ . Thus  $Pr(B) \leq ne^{-\epsilon M}/2 \leq \frac{1}{2n^2\tau^3}$ . If  $B$  does not happen, we know that  $\tilde{o}[k] \leq M + o[k]$  for all  $k \in [i, j]$ . Thus we can then apply Theorem 5.7 to get:

$$\begin{aligned} Pr(X \geq \mathbb{E}(X) + \lambda) &\leq e^{\frac{-\lambda^2}{2(2n/\epsilon^2 + \lambda \log(n\tau)/\epsilon)}} + Pr(B) \\ &= e^{\frac{-\lambda^2}{2(2n/\epsilon^2 + \lambda \log(n\tau)/\epsilon)}} + \frac{1}{2n^2\tau^3} \end{aligned}$$

Setting  $\lambda = \frac{8}{\epsilon} \sqrt{n} \log(n\tau)$  gives us that

$$Pr \left( X \geq \mathbb{E}(X) + \frac{8}{\epsilon} \sqrt{n} \log(n\tau) \right) \leq \frac{1}{n^2\tau^2}$$

Since  $\mathbb{E}(X) \leq 0$ , we get

$$Pr \left( X \geq \frac{8}{\epsilon} \sqrt{n} \log(n\tau) \right) \leq \frac{1}{n^2\tau^2}$$

Also we observe that  $\tilde{M}[i, j] = X/n$ , which yields

$$\Pr \left( \tilde{M}[i, j] \geq \frac{8 \log(n\tau)}{\sqrt{n\epsilon}} \right) \leq \frac{1}{n^2\tau^2}$$

Finally, observe that  $\tilde{M}[i, j] \leq c$  implies that  $\tilde{M}[i, j]^2 \mathbb{I}[\tilde{M}[i, j]] \leq c^2$ . Thus we get

$$\Pr \left( \tilde{M}[i, j]^2 \mathbb{I}[mm[i, j]] \geq \frac{64 \log^2(n\tau)}{n\epsilon^2} \right) \leq \frac{1}{n^2\tau^2}$$

Putting  $n = j - i + 1$  and using  $c = 64$  gives us the required result.  $\square$

Now we can give the proof of Theorem 5.3. In the proof we assume without loss of generality that  $\Delta_{\mathbf{O}} = 1$ .

of Theorem 5.3. The proof of  $error(\tilde{\mathbf{O}}) = \Theta(n/\epsilon^2)$  is obvious since:

$$error(\tilde{\mathbf{O}}) = \sum_{k=1}^n error(\tilde{o}[k]) = n \left( \frac{2}{\epsilon^2} \right)$$

In the rest of the proof, we shall show bound  $error(\overline{\mathbf{O}})$ . Let  $o = \mathbf{O}(I)$  be the input sequence. We know that  $o$  consists of  $d$  distinct elements. Denote  $o_r$  as the  $r^{th}$  distinct element of  $o$ . Also denote  $[l_r, u_r]$  as the set of indices corresponding to  $o_r$ , i.e.  $\forall_{i \in [l_r, u_r]} o[i] = o_r$  and  $\forall_{i \notin [l_r, u_r]} o[i] \neq o_r$ . Let  $M[i, j]$  record the mean of elements in  $o[i, j]$ , i.e.  $M[i, j] = \sum_{k=i}^j o[k] / (j - i + 1)$ .

To bound  $error(\overline{\mathbf{O}})$ , we shall bound  $error(\overline{\mathbf{O}}[i])$  separately for each  $i$ . To bound  $error(\overline{\mathbf{O}}[i])$ , we can assume W.L.O.G that  $s[i]$  is 0. This is because if  $o[i] \neq 0$ , then we can translate the sequence  $o$  by  $o[i]$ . As shown in Lemma 5.4 this preserves  $error(\overline{\mathbf{O}}[i])$ , while making  $o[i] = 0$ .

Let  $k \in [l_r, u_r]$  be any index for the  $r^{th}$  distinct element of  $o$ . By definition,  $error(\overline{\mathbf{O}}[k]) = \mathbb{E}(\bar{o}[k] - o[k])^2 = \mathbb{E}(\bar{o}[k]^2)$  (as we can assume W.L.O.G  $o[k] = 0$ ).

From Theorem 5.1, we know that  $\bar{o}[k] = U_k$ . Thus  $error(\bar{\mathbf{O}}[k]) = \mathbb{E}(U_k^2)$ . Here we treat  $U_k = \max_{i \leq k} \min_j \tilde{M}[i, j]$  as a random variable. Now by definition of  $\mathbb{E}$ , we have

$$\mathbb{E}(U_k^2) = \mathbb{E}(U_k^2 \mathbb{I}[U_k]) + \mathbb{E}(U_k^2 (1 - \mathbb{I}[U_k])) = A + B \text{ (say)}$$

We shall bound  $A$  and  $B$  separately. For bounding  $A$ , denote  $\mathcal{U}_k = \max_{i \leq k} \tilde{M}[i, u_r]$ . It is apparent that  $\mathcal{U}_k \geq U_k$  and thus  $\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k] \geq U_k^2 \mathbb{I}[U_k]$ . To bound  $A$ , we observe that

$$A = \mathbb{E}(U_k^2 \mathbb{I}[U_k]) \leq \mathbb{E}(\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k])$$

Further, since  $\mathcal{U}_k = \max_{i \leq k} \tilde{M}[i, u_r]$ , we know that  $\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k] = \max_{i \leq k} \tilde{M}[i, u_r]^2 \mathbb{I}[\tilde{M}[i, u_r]]$ . Thus we can write:

$$A \leq \mathbb{E}(\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k]) = \mathbb{E} \left( \max_{i \leq k} \tilde{M}[i, u_r]^2 \mathbb{I}[\tilde{M}[i, u_r]] \right)$$

Let  $\tau > 1$  be any number and  $c$  be the constant used in Lemma 5.6. Let us denote  $e_i$  the event that:

$$\tilde{M}[i, u_r]^2 \mathbb{I}[\tilde{M}[i, u_r]] \geq c \left( \frac{\log^2((u_r - i + 1)\tau)}{(u_r - i + 1)\epsilon^2} \right)$$

We can apply lemma 5.6 to compute the probability of  $e_i$  as  $o[j] \leq 0$  for all  $j \leq u_r$  (as we assumed W.L.O.G  $s[k] = 0$ ). Thus we get  $Pr(e_i) \leq \frac{1}{(u_r - i + 1)^2 \tau^2}$ .

Define  $e = \bigvee_{i=1}^{u_r} e_i$ . Then  $Pr(e) \leq \sum_{i=1}^{u_r} Pr(e_i) = 2/\tau^2$  (as  $\sum_{i=1}^{u_r} 1/i^2 \leq 2$ ). If the event  $e$  does not happen, then it is easy to see that

$$\begin{aligned}
\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k] &= \max_{i \leq k} \tilde{M}[i, u_r]^2 \mathbb{I}[\tilde{M}[i, u_r]] \\
&\leq c \left( \frac{\log^2((u_r - k + 1)\tau)}{(u_r - k + 1)\epsilon^2} \right)
\end{aligned}$$

Thus with at least probability  $1 - 2/\tau^2$  (which is  $Pr(\neg e)$ ), we get  $\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k]$  is bounded as above. This yields that there exist constants  $c_1$  and  $c_2$  such that  $\mathbb{E}(\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k]) \leq \frac{c_1 \log^2(u_r - k + 1) + c_2}{(u_r - k + 1)\epsilon^2}$ . The proof is by the application of Lemma 5.5 (as  $\mathcal{U}_k$  is bounded) and a simple integration over  $\tau$  ranging from 1 to  $\infty$ . Finally we get that  $A \leq \mathbb{E}(\mathcal{U}_k^2 \mathbb{I}[\mathcal{U}_k]) \leq \frac{c_1 \log^2(u_r - k + 1) + c_2}{(u_r - k + 1)\epsilon^2}$ .

Recall that  $B = \mathbb{E}(U_k^2(1 - \mathbb{I}[U_k]))$ . We can write  $B$  as  $\mathbb{E}(L_k^2(1 - \mathbb{I}[L_k]))$  as  $L_k = U_k$ . Using the exact same arguments as above for  $L_k$  but on sequence  $-\mathbf{O}$  yields that  $B \leq \frac{c_1 \log^2(k - l_r + 1) + c_2}{(k - l_r + 1)\epsilon^2}$ .

Finally, we get that  $\overline{\mathbf{O}}[k] = A + B$  which is less than  $\frac{c_1 \log^2(u_r - k + 1) + c_2}{(u_r - k + 1)\epsilon^2} + \frac{c_1 \log^2(k - l_r + 1) + c_2}{(k - l_r + 1)\epsilon^2}$ .

To obtain a bound on the total  $error(\overline{\mathbf{O}})$ .

$$\begin{aligned}
error(\overline{\mathbf{O}}) &= \sum_{r=1}^d \sum_{k \in [l_r, u_r]} error(\overline{\mathbf{O}}[k]) \\
&\leq \sum_{r=1}^d \sum_{k \in [l_r, u_r]} \frac{c_1 \log^2(u_r - k + 1) + c_2}{(u_r - k + 1)\epsilon^2} + \\
&\quad \sum_{r=1}^d \sum_{k \in [l_r, u_r]} \frac{c_1 \log^2(k - l_r + 1) + c_2}{(k - l_r + 1)\epsilon^2} \\
&\leq \sum_{r=1}^d \frac{c_1 \log^3(u_r - l_r + 1) + c_2}{\epsilon^2}
\end{aligned}$$

Finally noting that  $u_r - l_r + 1$  is just  $n_r$ , the number of occurrences of  $o_r$  in  $o$ , we get  $error(\overline{\mathbf{O}}) = \sum_r \frac{c_1 \log^3 n_r + c_2}{\epsilon^2} = O(d \log^3 n / \epsilon^2)$ . This completes the proof of the theorem.  $\square$

## CHAPTER 6

### ESTIMATING OTHER STATISTICS UNDER STRONG PRIVACY

While being able to compute the degree distribution under differential privacy is an important result, it is only one of many network properties that an analyst may wish to measure. Network analysis is an often mentioned goal in the differential privacy literature, but relatively few concrete results exist that demonstrate the feasibility of differential privacy for network data. Below we highlight a few results and discuss some of the challenges.

As described previously (Chapter 2), the Laplace mechanism can be used to approximately answer any query or query sequence. The accuracy of the answer depends the query’s sensitivity, with lower sensitivity yielding greater accuracy. Some analyses of networks can be computed with queries that are low sensitivity. For example, one measure of network resiliency can be approximated with a low sensitivity query. The query asks how many edges must be removed until the network becomes, say, disconnected, and it has a sensitivity of one [38]. In addition, for weighted graphs with edge weights in  $[0, 1]$ , the weight of a minimum edge-cut or a minimum spanning tree are both low-sensitivity queries [38].

However, the fact that an analysis can be computed using a query, or sequence of queries, with low sensitivity does not necessarily imply that the analysis will be accurate under differential privacy. Our investigation of the degree distribution is an illustration of this point. The query sequences described earlier – **C**, **S**, **F** – all have low sensitivity. Yet because the query sequence is linear in the size of the

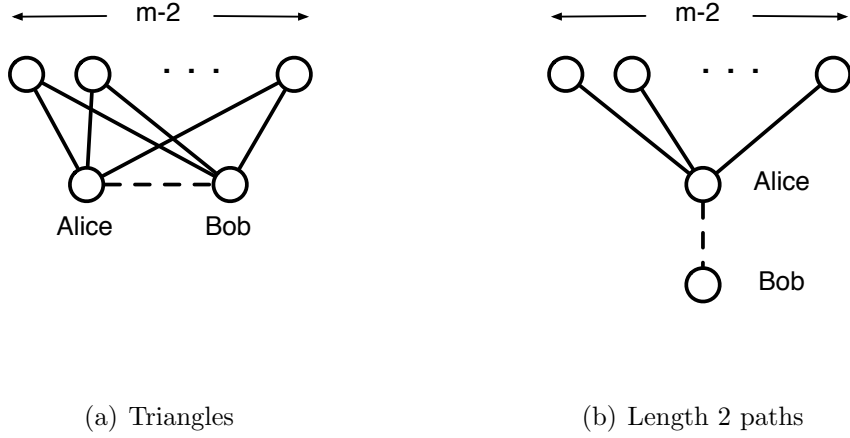
graph, the small amount of noise on individual queries accumulates, resulting in severe distortion of the degree distribution. To achieve acceptable accuracy, we had to design query strategies where constraints held between query answers and then exploit those constraints to reduce noise.

While these are promising results, open questions remain about the accuracy obtainable for many common network analyses. It is not known if the strategy used for the degree sequence can be applied to other high-dimensional degree statistics, such as the paired in-out degree sequence of the  $p1$  model [56] or degree correlations [87].

For other important analyses, the prospects for accurate analysis under differential privacy seem poor. Computations such as transitivity, clustering coefficient, centrality, and path-lengths involve joins on the edge table. It is not hard to show that the sensitivity of such statistics is extremely high. We give illustrative examples for two counting queries: the number of triangles (i.e., cycles of length 3) and the number of paths of length two. Among other things, these two statistics can be used to compute transitivity.

Figure 6.1 shows two graphs that correspond to worst-case inputs for the two respective queries. In Figure 6.1(a), when the edge  $(Bob, Alice)$  is present, there are  $m - 2$  triangles; when it is absent, there are 0 triangles. Therefore the sensitivity of the query that returns the number of triangles is at least  $m - 2$  where  $m$  is the number of nodes in the graph. Therefore, using the Laplace mechanism, it would be necessary to add Laplace noise with scale  $O(m/\epsilon)$ . Since real graphs are typically sparse, the total number of triangles is typically at most  $O(m)$ , so the large noise renders the answer useless. Similarly, Figure 6.1(b) shows the sensitivity of counting the number of length two paths is also  $O(m)$ .

These are not limitations of the Laplace mechanism, specifically, but of any differentially private mechanism. This is formalized in the following proposition which says that any  $\epsilon$ -differentially private algorithm will likely have error at least  $\Delta_{\mathbf{Q}}/2$  on



**Figure 6.1.** Example graphs illustrating the high sensitivity of triangle and path queries.

some inputs. The probability of this event depends on  $\epsilon$ , but it is non-negligible for reasonable  $\epsilon$ .

**Proposition 6.1** (Limits of differential privacy). *Let  $\mathbf{Q}$  be any query sequence and  $\mathcal{A}$  be any  $\epsilon$ -differentially private algorithm. There exists inputs  $I$  such that with probability at least  $\frac{e^{-\epsilon}}{1+e^{-\epsilon}}$ ,*

$$\|\mathbf{Q}(I) - \mathcal{A}(\mathbf{Q}(I))\|_1 \geq \frac{\Delta_{\mathbf{Q}}}{2}$$

*Proof.* Let  $I$  and  $I'$  be worst-case neighboring instances such that  $\|\mathbf{Q}(I) - \mathbf{Q}(I')\|_1 = \Delta_{\mathbf{Q}}$ . Let  $\mathcal{O}_I$  be the set of outputs “near”  $I$ , meaning for any  $o \in \mathcal{O}_I$ , we have  $\|o - \mathbf{Q}(I)\|_1 < \Delta_{\mathbf{Q}}/2$ . Let  $\delta$  be such that for all  $I$ ,  $\Pr[\mathcal{A}(I) \in \mathcal{O}_I] \geq 1 - \delta$ . Therefore we have the following chain of inequalities:

$$\delta \geq \Pr[\mathcal{A}(I) \notin \mathcal{O}_I] \geq \Pr[\mathcal{A}(I) \in \mathcal{O}_{I'}] \geq e^{-\epsilon} \Pr[\mathcal{A}(I') \in \mathcal{O}_{I'}] \geq e^{-\epsilon}(1 - \delta)$$

which implies  $\delta \geq \frac{e^{-\epsilon}}{1+e^{-\epsilon}}$ . □

The reason for these limits is that differential privacy is a guarantee under two worst-case assumptions. Recall from Section 2.2 the semantic interpretation of differential privacy as a bound on the ratio between an adversary’s prior and posterior beliefs. The first assumption is that the adversary is nearly omniscient and has complete knowledge of the private data except for one tuple (or, more generally, one differential object). Second, the guarantee holds for all inputs, regardless of how likely they are to occur in practice. The combined result is that for an analysis like the number of triangles, it is possible to construct a graph where the number of triangles depends entirely on a single edge. On this input, against an adversary who knows every edge save this one, it is impossible to simultaneously produce accurate answers and prevent the adversary from learning about the edge.

Circumventing these limitations requires addressing at least one of these worst-case assumptions. In Rastogi et al. [105], we propose an approach based on relaxing the former assumption; Nissim et al. [102] propose an approach that is based on addressing the latter assumption. We briefly describe each approach, and summarize their application to network statistics.

The basic idea behind Nissim et al. [102] is that while privacy should hold on all inputs, the accuracy of the query answer may depend on the instance. When the input is a worst-case input, accuracy is necessarily low. But when the input is “far” from a worst-case input, it is possible to achieve much higher accuracy. A tempting solution is to use the *local* sensitivity of the given input  $I$ —i.e., the maximum change between  $\mathbf{Q}(I)$  and  $\mathbf{Q}(I')$  for any  $I' \in \text{nbrs}(I)$ . However, this can leak information because the local sensitivity itself can change substantially between neighboring instances and thus an approach that uses it directly would fail to satisfy differential privacy. Their approach is based on *smooth sensitivity*, an upper bound on local sensitivity that varies smoothly over the space of possible databases. Using the Laplace mechanism with smooth sensitivity in place of global sensitivity satisfies  $(\epsilon, \delta)$ -differential pri-

vacy. The weakening to  $(\epsilon, \delta)$ -differential privacy is necessary because the smooth sensitivity can differ on two neighboring instances (albeit by a bounded amount) and so some outputs become much more likely on one instance as compared to its neighbor; however, the probability of observing such an output is very rare ( $\delta$  is typically exponentially small in  $n$ ).

Nissim et al. [102] apply the smooth sensitivity idea to the problem of computing the number of triangles. They show how to efficiently compute the smooth sensitivity for a given graph and also show that random graphs are likely to have low smooth sensitivity. Since the smooth sensitivity is bounded below by the local sensitivity, a particular input will have low smooth sensitivity only if it has low local sensitivity. For the triangle query, local sensitivity is low when high degree nodes share few common neighbors. In real-world graphs, this may not be the case, and so the local sensitivity (and therefore the smooth sensitivity) may be high.

In Rastogi et al. [105], we designed an alternative approach by “relaxing” differential privacy. Recall that differential privacy implies protection against adversaries who are practically omniscient. We considered more realistic adversaries who have greater uncertainty about the input. We developed an algorithm that is a variant on the Laplace mechanism, however it is not differentially private. Instead it satisfies *adversarial privacy*, a definition based on bounding the prior and posterior beliefs of adversaries within a restricted class. Against adversaries who know a small subgraph (of size  $O(\log n)$ ) and whose prior belief about the rest of the graph is based on a sparse random graph model, it is possible to accurately release the number of triangles. In addition, several other statistics can be accurately released, including queries about path lengths, cycles, and cliques and other subgraph motifs. Error scales poly-logarithmically with the size of the graph. Details are provided in [105].

Finally, another potential solution for high sensitivity queries is to reformulate them into lower sensitivity queries. High sensitivity means that for some networks,

the change of a single edge can profoundly alter the query answer. Given that network data is often incomplete and noisy, analysts need measures that are robust to minor perturbations of network structure. In fact, there has been some work looking at how some common analyses are affected by small perturbations [59]. The connection between robust statistics and differentially private algorithms has been explored, but existing results are limited to high sensitivity queries of tabular data [37].

Despite the poor prospects for some network analyses, it is nevertheless important to identify analyses that can be accurately computed under differential privacy, such as the degree distribution. Differential privacy is so rigorous that even if it is ultimately necessary to adopt weaker privacy definitions, any differentially private techniques can be easily integrated into a complete solution.

## CHAPTER 7

### RELATED WORK

While the broader topics of privacy-preserving data publication and privacy-preserving data analysis have a long and well-documented history (cf. [2, 23, 35]), addressing these issues for network data is a topic that has only recently gained attention. Work on this topic includes attacks of naive anonymization and alternative approaches to network anonymization. We review relevant work below, and there are also several recent surveys [53, 81, 124, 135].

When it comes to network data, particularly the data stored in online social networks, there are a number of privacy issues beyond those that arise with analysis and publication. We also briefly review some of the work addressing other privacy issues with network data.

We also discuss work in differential privacy that is related to the techniques proposed in Chapter 5.

#### 7.1 Attacks

Backstrom et al. [9] propose an *active* attack on anonymized networks, where the adversary is capable of adding nodes and edges *prior* to anonymization. The attack re-identifies an arbitrary set of targets by inserting a random subgraph that will be unique with high probability (independent of the input graph) and then connecting the subgraph to the targets. Active attacks, while relevant to online social networks, are difficult or impossible to carry out in many other networks (such as

contact networks used to study disease transmission or email networks internal to an organization).

Passive attacks—where the adversary attacks an already published network—have been more extensively studied. We first introduced the passive attack based on  $\mathcal{H}_i$  degree signatures in Hay et al. [51]. We also studied adversaries with knowledge of partial subgraph patterns around a target, and knowledge of connections to hubs in the network. Narayanan and Shmatikov [97] propose a passive attack in which the adversary exploits access to an auxiliary network whose membership overlaps with the anonymized network. Such an attack can lead to breaches of privacy if for instance the anonymized network includes sensitive attributes or additional edges absent from the auxiliary network. Singh and Zhan [113] measure the vulnerability to attack as a function of well known topological properties of the graph, and Wang et al. [119] propose a measure of anonymity based on description logic.

## 7.2 Network anonymization algorithms

There are three primary approaches to network anonymization: directed alteration, generalization, and random alteration. With *directed alteration*, the graph structure is altered, using operations such as edge insertions, to create common structural patterns. Nodes in the output graph are more likely to look more similar to one another, but the graph may be missing data or contain spurious information. With *generalization/clustering*, the structure of the graph is generalized at a granularity that is coarse enough to provide some privacy but fine enough to reveal the essential features of the network’s topology. Most approaches to generalization, including the one described in Chapter 4, are based on clustering nodes into groups and then describing the graph at the group level. Finally, with *random alteration*, the graph is altered stochastically, through random edge additions and deletions. Structural patterns in the original graph are disguised by the random alteration.

### 7.2.1 Directed alteration

Liu and Terzi [82] propose several algorithms for anonymizing a graph through the insertion and removal of edges, altering the graph so that nodes cannot be distinguished by degree. We compare against their SimultaneousSwap algorithm in Section 4.3.

Zhou and Pei [134] propose a similar problem formulation as Liu and Terzi [82] but with a stronger privacy condition. The condition requires that for each node in the graph, its *neighborhood*—the subgraph induced by the node and its neighbors—is isomorphic with at least  $k - 1$  other neighborhoods. Any graph satisfying this condition will also satisfy the condition of Liu and Terzi [82] because if two nodes have isomorphic neighborhoods, then they must have equal degrees. Another difference is that the data model includes labels on the nodes, which must also be anonymized. They show that the problem of determining the minimal set of edge insertions that satisfy the privacy condition is NP-Hard. They propose a greedy algorithm, however, due to a subgraph matching step, its runtime remains exponential in the worst-case. Thompson and Yao [118] propose a more efficient algorithm for this problem.

Zou et al. [136] further strengthen the privacy condition, requiring that each node be *automorphically equivalent* with at least  $k - 1$  others. Since nodes in the same automorphic equivalence class are structurally indistinguishable, no adversary can successfully re-identify a target node beyond a set of at least  $k$  nodes. They propose an algorithm in which the graph is partitioned into subgraphs, and then the subgraphs are placed into groups containing at least  $k$  subgraphs. The subgraphs within each group are made automorphic with each other through a process of vertex/edge insertions and deletions. They compare the utility of this approach against the approaches described above as well as our generalization approach described in Chapter 4. Recent work by Cheng et al. [24] adopts a similar approach of altering

the graph to ensure automorphic equivalence, but do so in a way that also limits edge disclosure.

### 7.2.2 Generalization/clustering

In their initial work on graph anonymization, Cormode et al. [30] consider bipartite graph data—representing, for example, associations between people and products they purchase—and propose an anonymization algorithm that breaks the association between identifying attributes and nodes in the graph. The main threat considered is an adversary with knowledge of node attributes, and so under these assumptions, it is safe to release a naive anonymization of the graph. They propose an algorithm that groups nodes so that those within a group cannot be distinguished by their attributes. Further, they impose a safety condition that two nodes in the same group do not have common neighbors, thereby preventing edge disclosure.

In subsequent work, Cormode et al. [29] extend their approach to handle a richer class of data, such as social networks with multiple edge types and attributes on nodes and edges. They also consider an approach which protects against an adversary with knowledge of graph structure. They propose a partitioning based approach that we compare against in Section 4.3.

Campan and Truta [20] propose an approach similar to the generalization approach described in Chapter 4. In terms of graph structure, their *masked social network* appears to be equivalent to our generalized graph. They also include identifying attributes on the nodes, handling them using standard techniques from tabular data. To generalize the graph, Campan and Truta apply an iterative, greedy algorithm that anonymizes nodes in batches of  $k$ , similar to the algorithm of Zhou and Pei [134].

Zheleva et al. [132] consider a related but distinct problem. They study an adversary who might be capable of predicting sensitive edges (e.g., friendship relations) given knowledge of non-sensitive attributes and non-sensitive edges (e.g., classmate

relations). The sensitive data is never published, not even in an anonymized form. The purpose of the study is to measure to what extent non-sensitive properties leak information about sensitive properties, and they consider several alternative strategies for coarsening the non-sensitive properties to prevent accurate inference.

### 7.2.3 Random alteration

Rastogi et al. [106] present a mechanism for tables that has a natural interpretation for graphs. They randomly remove a fraction of original edges and randomly add a fraction of new edges. The resulting edge table is released in its entirety. They show that the parameters of the random process can be chosen to ensure strong protection against edge disclosure while allowing a class of counting queries to be estimated accurately. Unfortunately it does not address queries that require joins on the edge table, which are crucial to network analysis.

Hay et al. [54] consider randomly permuting some fraction of the network's edges. They show this is effective at limiting an adversary's ability to re-identify a target based on node degree. However, the graph structure is changed considerably: a 10% change in the edge structure results in a roughly 33% change in the value of some important graph metrics. Although the analyst may be able to reduce the error through statistical post-processing, the results suggest that the gain in privacy is offset by a substantial loss in utility.

To address the loss of utility, Ying and Wu [128] consider a more complex randomization strategy that is guided by the graph structure, choosing a random alteration that preserves key properties of the network. The technique is based on the observation that many important network properties are related to the graph's spectrum—i.e., the set of eigenvalues of the graph's adjacency matrix or other matrices derived from it. Thus, they develop a random-alteration algorithm where edges are randomly added and deleted, but the random choice is guided based on how the change affects

the graph’s spectrum. They show that the utility of the randomly altered network—measured both in terms of common metrics and spectral properties—is much improved. However, they do not assess what impact spectrum-based randomization has on privacy. The protection must necessarily be weaker than pure randomization: the noise is influenced by the structure of the graph which means the adversary may be able to use his knowledge of graph structure to infer likely edge swaps. It is unclear how much this improves the adversary’s ability to breach privacy.

Additional work by the same authors considers imposing additional constraints [129] and a more in depth study of the risk of edge disclosure with randomization based approaches [131]. Ying et al. [130] compare random alteration and directed alteration approaches.

### **7.3 Other privacy issues arising with network data**

The anonymization of existing networks is not the only privacy problem that arises with network data.

Some work has looked at the problem of reconstructing a private graph under various access restrictions. Frikken and Golle [44] designed a protocol for privately assembling a graph that is distributed among a large number of parties. The output of the protocol is a naively-anonymized graph. Korolova et al. [66] consider an adversary who tries to re-assemble the graph from a set of views of local neighborhoods (obtained, for example, by breaking into user accounts in an online social network setting).

There are many privacy concerns with online social networks. At least two works have looked at how a user’s private information can sometimes be inferred from the public information available on friends’ profiles or from knowledge of group memberships [49, 133]. Others have looked at ways to mitigate user exposure. Liu and Terzi [83] propose a way of computing a privacy “score” for an online profile that

measures a user’s risk, with higher scores indicating the user profile discloses more sensitive information. Fang and LeFevre [41] propose a privacy “wizard,” an application that aids users in configuring their privacy settings, identifying communities of friends that should be granted the same access privileges. Carminati et al. [22] look at access-control models for protecting privacy in online social networks.

## 7.4 Related work in differential privacy

Differential privacy has been an active area of research. Dwork has written comprehensive reviews of differential privacy [34, 35]. In Chapter 6, we reviewed applications of differential privacy to network analyses. Differential privacy also has applications in network security, where it has been proposed as a tool for network operators to conduct collaborative intrusion detection without disclosing sensitive information about individual client’s network traffic [107]. Recent work looks at carrying out common network trace analyses under differential privacy [90].

The technique introduced in Chapter 5 involved post-processing the answers of a differentially private mechanism to ensure consistency. This idea of using statistical inference to post-process answers and boost accuracy has broader applications. We used this technique to design a mechanism for publishing synthetic data that is accurate for range queries [55]. It asks a hierarchical tree of range queries and uses summation constraints to infer accurate answers for all range queries. We extended this approach to support inference over a set of linear counting queries where arbitrary linear constraints may hold among the answers. Using this framework, we showed that it is possible to design optimal strategies for support workloads of queries, using tools from convex optimization [76].

The basic idea of post-processing was introduced in Barak et al. [11], who proposed a linear program for making a set of marginals consistent, non-negative, and integral. However, unlike the present work, the post-processing is not shown to improve accu-

racy. (While it is true that their Fourier transformations can increase accuracy for low-order marginals, this is a *pre*-processing step that reformulates queries to avoid issues of consistency and does not improve accuracy on the queries considered in the present paper.)

## CHAPTER 8

### CONCLUSION

This dissertation addresses the challenge of enabling accurate analysis of network data while ensuring the protection of network participants' privacy. This is an important problem: massive amounts of data are being collected, there is huge interest in analyzing the data, but the data are not being shared due, in large part, to concerns about privacy. Recent incidents have demonstrated that releasing data without properly addressing these concerns can be a damaging exercise, with negative consequences for both data manager and data participant [12, 98, 115]. Although there have been decades of research in privacy-preserving data publication and privacy-preserving data analysis, existing technologies provide an inadequate solution because they were designed for tables, not networks, and cannot be easily adapted to handle the complexities of network data.

We develop several technologies that advance us toward our goal. Our contributions represent important first steps in addressing the problem and we see opportunities to extend the work in new and promising directions.

#### **8.1 Review of contributions**

In Chapter 3, we demonstrated that simple strategies fail to provide adequate protection. We studied the risk of publishing a network after applying naive anonymization, in which external identifiers are replaced and identifying attributes are suppressed. In carrying out this task, our workhorse was the vertex refinement query: it is efficient to evaluate and capable of modeling a range of adversaries. Using vertex

refinement queries, we found that real networks are diverse in their risk but in some networks, nodes are highly vulnerable to re-identification attack, especially if the adversary has knowledge beyond a target node’s immediate neighborhood (e.g., this is true of the **HepTh** network). Re-identification risk depends on the network’s structure and we showed, through the analytical study of Erdős-Rényi random graphs, that density has a profound influence on risk, with sparse graphs being at low risk and dense graphs being at high risk. Our study of risk supports our overarching goal because it provides data managers with algorithms that they can use to assess risk prior to publication, and because it demonstrates that there is a need for technologies to mitigate risk. The rest of the dissertation was devoted to the design of such technologies.

In Chapter 4 we presented an algorithm for transforming a network to prevent node re-identification. Our algorithm transforms the network by generalizing it, replacing groups of nodes with supernodes and sets of edges with superedges. Effectively, it produces a coarse-grained summary of the network’s topology, describing it in terms of the edge density between and within groups. The process of summarization probably lowers re-identification risk; and the data manager can choose the appropriate setting of  $k$  to tradeoff lower risk against greater information loss. Through experiments on real and synthetic networks, we evaluate how well the transformed network preserves the topology of the original network, and compare our approach with other techniques. In comparison with the LT and BCKS algorithms, we find our technique preserves the input comparably or better. One notable distinction between techniques is that the other techniques do not account for topology when transforming the network—for instance, LT minimizes the number of edge insertions, but edges can be inserted anywhere, even between disconnected components. Our approach is explicitly guided by the network structure, using search to place nodes into clusters of “similar” nodes, similar to stochastic block-modeling. This has its advantages: we

find that our technique more accurately preserves distinctive features, such as mesh or tree patterns, and the low degree correlations of the **HOT** network. This work contributes to the goal of this dissertation: we provide a practical tool that a data manager can use to publish a transformed network that has provably bounded risk. Also, our study gives insights into the kinds of distortions that are inevitably caused by the transformations.

In Chapters 5 & 6, we address the problem of computing accurate network statistics under strong privacy protections, such as those afforded by differential privacy. Our main contribution is a technique for estimating the degree sequence of a network (Chapter 5). The algorithm satisfies  $k$ -edge  $\epsilon$ -differential privacy, produces extremely accurate estimates, and is computationally efficient, capable of running on networks with hundreds of millions of nodes. A data manager can now release (an approximation of) the degree sequence of a private network with assurances that the error in the approximation is provably low and that the output does not leak private information, even to powerful, informed adversaries. In addition, the technique, which relies on applying inference to the noisy answers of a differentially private mechanism, is innovative and has broader applications. Our work shows that existing mechanisms can add more noise than is strictly necessary and inference can be an appealing strategy for reducing excess noise. We have investigated other applications of inference in subsequent work [55, 76].

Admittedly, the degree sequence of a network is only one statistic (though an important one), and questions remain about what other statistics can be accurately learned under differential privacy. We summarize known results, including additional work of our own, in Chapter 6. Unfortunately, we find limitations with differential privacy: network statistics like clustering coefficient involve joins on the edge table and have high sensitivity. This provably limits the accuracy with which they can be estimated under differential privacy. As a potential remedy to this conflict between

privacy and accuracy, we show that if the adversary has limited information about the network—such as knowing only a small subgraph of it—then a relatively small amount of noise is sufficient to ensure strong privacy (albeit not differential privacy). The work of Nissim et al. [102] provides another possible remedy and relies on a modest relaxation of differential privacy.

## 8.2 Future directions

We see a number of directions for future work. We briefly highlight a few of them:

- **Empirical comparison** At this point, many network transformation algorithms have been developed (see Chapter 7) and their relative benefits are not clearly understood. We could extend the empirical evaluation we undertook in Chapter 4 to include a broader array of techniques and thereby provide a more comprehensive evaluation.
- **Vulnerabilities** Most algorithms for network data transformation define privacy using an adaptation of  $k$ -anonymity to networks. In the tabular data setting,  $k$ -anonymity (and variants of it) has been shown to be vulnerable to attack (Chapter 2). It is unclear whether those attacks, or new ones, apply to the network anonymization algorithms.
- **Attributes** Our work has focused on network structure. It would be interesting to investigate supporting analyses of structure and attributes. For instance, can we approximately measure homophily under differential privacy?
- **High Sensitivity** High sensitivity queries are problematic under differential privacy. Many network analyses have high sensitivity often because of the existence of some pathological worst-case input that is highly unlikely to arise in practice. How can we incorporate background knowledge about realistic network structure to rule out pathological instances?

One of the directions for future work that we find most intriguing is *model-based synthetic data*. The idea behind model-based synthetic data is to take an existing statistical parametric model and use query answer perturbation techniques to accurately estimate the model parameters. An instantiated model defines a probability distribution over networks from which one can sample synthetic networks. Because the model was fit under differential privacy, both the model and the networks can be safely released to the public.

This approach represents, in some sense, a synthesis between transformed data release and query answer perturbation, the two approaches that have been considered in this dissertation and in other work on this topic. But it has the potential to alleviate some of the limitations of each of the current approaches.

Query answer perturbation provides very strong privacy, and for some queries, it achieves very high accuracy. However, it does have some practical limitations. Some analyses cannot be easily reduced to a small set of queries. For instance, some analyses are not statistics, but involve running algorithms over the network or carrying out simulations. It can be hard to determine the sensitivity of such complex procedures. Also, the user may find it frustrating to only be able to access the data through a query interface, especially during the initial phase of analysis, which is often exploratory, open-ended, and iterative. In addition, the data manager is responsible for processing queries. For the manager, this means he must invest resources in query processing and for the analyst, he must reveal his analyses to the data manager. For these reasons, there is a significant benefit to having techniques for publishing data.

While there are now several algorithms for publishing transformed network data (Chapter 4 and the related work described in Chapter 7), these too have limitations. Compared to the query answer perturbation techniques, the privacy protections are weak. In addition, while the transformations inevitably distort some properties of the

network, these approaches provide no theoretical bounds on how large that distortion may be.

Model-based synthetic data has the potential to provide the best of both worlds. The analyst can use the model to produce synthetic networks and then run analyses on the sampled networks. The approach ensures rigorous privacy, as all interactions with the private data are through a differentially private mechanism. In addition, this approach may provide a clearer picture of how noise distorts the network. For instance, it may be possible to derive error bounds on the accuracy of the parameter estimates, allowing the analyst to account for the distortion introduced by the privacy mechanism. In addition, the biases of the model are often made explicit as the structure of the model encodes independence assumptions and the analyst can reason about their realism and the consequences of their invalidity.

To pursue this approach it will be necessary to identify existing statistical models that may fit well with the query answer perturbation approach. Many network models have been developed in the last couple of decades [48], and a few stand out as potentially well-suited for the task. The  $dK$  family of models [87] is based on node degrees and degree correlations, and in fact our current techniques can be immediately applied to fit the  $1K$  model, the simplest model in the  $dK$  family. Also, the exponential random graph model (ERGM) family [50, 89, 108] is intriguing because the model is defined by a set of sufficient statistics and so it fits naturally with the framework of using query answer perturbation. ERGMs can also handle attributes. Of the many models in the ERGM family, the  $p_1$  model [56] is particularly interesting because its sufficient statistics include the in- and out-degree sequence of the network.

There are of course a number of challenges to address to understand whether model-based synthetic data will yield the hoped for benefits. We believe that the contributions of this dissertation provide lessons and insights that will be valuable in addressing these challenges.

## BIBLIOGRAPHY

- [1] Abowd, John. Personal communication regarding the practices at the Restricted Data Access Center at Cornell University, 2009.
- [2] Adam, Nabil R., and Worthmann, John C. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.* 21, 4 (1989), 515–556.
- [3] Aggarwal, Charu C. On  $k$ -anonymity and the curse of dimensionality. In *VLDB* (2005).
- [4] Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., and Zhu, A. Approximation algorithms for  $k$ -anonymity. *Journal of Privacy Technology* (2005).
- [5] Aggarwal, Gagan, Feder, Tomás, Kenthapadi, Krishnaram, Khuller, Samir, Panigrahy, Rina, Thomas, Dilys, and Zhu, An. Achieving anonymity via clustering. In *PODS* (2006), pp. 153–162.
- [6] Aiello, William, Chung, Fan, and Lu, Linyuan. A random graph model for massive graphs. In *STOC* (2000).
- [7] Alderson, David, and Li, Lun. Diversity of graphs with highly variable connectivity. *Phys. Rev. E* (2007).
- [8] Babai, L., and Kucera, L. Canonical labeling of graphs in linear average time. In *FOCS* (1979).
- [9] Backstrom, Lars, Dwork, Cynthia, and Kleinberg, Jon. Wherefore art thou R3579X? Anonymized social networks hidden patterns and structural steganography. In *WWW* (2007).
- [10] Barabási, A.L., and Albert, R. Emergence of scaling in random networks. *Science* 286 (1999).
- [11] Barak, Boaz, Chaudhuri, Kamalika, Dwork, Cynthia, Kale, Satyen, McSherry, Frank, and Talwar, Kunal. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *PODS* (2007).
- [12] Barbaro, Michael, and Zeller Jr., Tom. A face is exposed for AOL searcher no. 4417749. *New York Times* (2006).

- [13] Barlow, R E, Bartholomew, D J, Bremner, J M, and Brunk, H D. *Statistical Inference Under Order Restrictions*. John Wiley and Sons Ltd, 1972.
- [14] Barlow, R. E., and Brunk, H. D. The isotonic regression problem and its dual. *Journal of the American Statistical Association* 67, 337 (1972), 140–147.
- [15] Bayardo, R., and Agrawal, R. Data privacy through optimal  $k$ -anonymization. In *ICDE* (2005).
- [16] Blitzstein, J, and Diaconis, Persi. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Annals of Applied Probability* (2006).
- [17] Blum, Avrim, Ligett, Katrina, and Roth, Aaron. A learning theory approach to non-interactive database privacy. In *STOC* (2008), pp. 609–618.
- [18] Boyd, Stephen, and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2009.
- [19] Brickell, Justin, and Shmatikov, Vitaly. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD* (2008), pp. 70–78.
- [20] Campan, Alina, and Truta, Traian Marius. A clustering approach for data and structural anonymity in social networks. In *PinKDD* (2008).
- [21] Cancho, Ramon Ferrer, and Sole, Ricard V. Optimization in complex networks. In *ArXiv cond-mat/0111222* (2001).
- [22] Carminati, Barbara, Ferrari, Elena, and Perego, Andrea. Enforcing access control in web-based social networks. *ACM Trans. Inf. Syst. Secur.* 13, 1 (2009), 1–38.
- [23] Chen, Bee-Chung, Kifer, Daniel, LeFevre, Kristen, and Machanavajjhala, Ashwin. Privacy-preserving data publishing. *Foundations and Trends in Databases* (2009).
- [24] Cheng, James, Fu, Ada Wai-Chee, and Liu, Jia.  $k$ -isomorphism: Privacy preserving network publication against structural attacks. In *SIGMOD* (2010).
- [25] Chung, Fan, and Lu, Linyuan. Concentration inequalities and martingale inequalities: A survey. *Internet Mathematics* (2006).
- [26] Clauset, Aaron, Moore, Cristopher, and Newman, Mark. Hierarchical structure and the prediction of missing links in networks. *Nature* (2008).
- [27] Clauset, Aaron, Shalizi, Cosma Rohilla, and Newman, Mark. Power-law distributions in empirical data. *SIAM Review* (2009).

- [28] Cormode, G., Li, N., Li, T., and Srivastava, D. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. In *VLDB, to appear* (2010).
- [29] Cormode, Graham, Srivastava, Divesh, Bhagat, Smriti, and Krishnamurthy, Balachander. Class-based graph anonymization for social network data. In *VLDB* (2009).
- [30] Cormode, Graham, Srivastava, Divesh, Yu, Ting, and Zhang, Qing. Anonymizing bipartite graph data using safe groupings. In *VLDB* (2008).
- [31] Corneil, D. G., and Gottlieb, C. C. An efficient algorithm for graph isomorphism. *J. ACM* 17, 1 (1970), 51–64.
- [32] da F. Costa, Luciano, Rodrigues, Francisco A., Travieso, Gonzalo, and Boas, P. R. Villas. Characterization of complex networks: A survey of measurements. *Advances In Physics* (2007).
- [33] Du, Yang, Xia, Tian, Tao, Yufei, Zhang, Donghui, and Zhu, Feng. On multi-dimensional  $k$ -anonymity with local recoding generalization. In *ICDE* (2007), pp. 1422–1424.
- [34] Dwork, Cynthia. Differential privacy: A survey of results. In *TAMC* (2008).
- [35] Dwork, Cynthia. A firm foundation for private data analysis. *Communications of the ACM, to appear* (2010).
- [36] Dwork, Cynthia, Kenthapadi, Krishnaram, McSherry, Frank, Mironov, Ilya, and Naor, Moni. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT* (2006).
- [37] Dwork, Cynthia, and Lei, Jing. Differential privacy and robust statistics. In *STOC* (2009), pp. 371–380.
- [38] Dwork, Cynthia, Nissim, Frank McSherry Kobbi, and Smith, Adam. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006).
- [39] Erdős, Paul, and Rényi, Alfréd. On random graphs. *Publicationes Mathematicae Debrecen* 6 (1959), 290.
- [40] Erdős, Paul, and Rényi, Alfréd. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5 (1960), 17–61.
- [41] Fang, Lujun, and LeFevre, Kristen. Privacy wizards for social networking sites. In *WWW* (2010), pp. 351–360.
- [42] Freeman, Linton C. A set of measures of centrality based on betweenness. *Sociometry* (1977).

- [43] Friedkin, N. Horizons of observability and limits of informal control in organizations. *Social Forces* 62, 1 (1983), 54–77.
- [44] Frikken, Keith, and Golle, Philippe. Private social network analysis: How to assemble pieces of a graph privately. In *WPES* (2006).
- [45] Ganta, S, Kasiviswanathan, S, and Smith, Adam. Composition attacks and auxiliary information in data privacy. In *KDD* (2008).
- [46] Ghinita, Gabriel, Karras, Panagiotis, Kalnis, Panos, and Mamoulis, Nikos. Fast data anonymization with low information loss. In *VLDB* (2007), pp. 758–769.
- [47] Gilbert, E. N. Random graphs. *Annals of Mathematical Statistics* (1959).
- [48] Goldenberg, Anna, Zheng, Alice X., Fienberg, Stephen E., and Airoldi, Edoardo M. A survey of statistical network models. *Foundations and Trends in Machine Learning* (2010).
- [49] Gross, Ralph, and Acquisti, Alessandro. Information revelation and privacy in online social networks. In *WPES* (2005), pp. 71–80.
- [50] Handcock, Mark, Robins, Garry, Snijders, Tom, Wang, Peng, and Pattison, Philippa. Recent developments in exponential random graph ( $p^*$ ) models for social networks. *Social Networks* (2006).
- [51] Hay, M., Miklau, G., Jensen, D., Weis, P., and Srivastava, S. Anonymizing social networks. Tech. rep., University of Massachusetts Amherst, 2007.
- [52] Hay, Michael, Li, Chao, Miklau, Gerome, and Jensen, David. Accurate estimation of the degree distribution of private networks. In *ICDM* (2009), pp. 169–178.
- [53] Hay, Michael, Miklau, Gerome, and Jensen, David. *To appear in Privacy-Aware Knowledge Discovery: Novel Applications and New Techniques*. Chapman & Hall/CRC Press, 2010, ch. Enabling Accurate Analysis of Private Network Data.
- [54] Hay, Michael, Miklau, Gerome, Jensen, David, Towsley, Don, and Weis, Philipp. Resisting structural re-identification in anonymized social networks. *VLDB* (2008).
- [55] Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and Suci, Dan. Boosting the accuracy of differentially-private histograms through consistency. In *VLDB, to appear* (2010).
- [56] Holland, Paul, and Leinhardt, Samuel. An exponential family of probability distributions for directed graphs. *JASA* (1981).
- [57] Holme, P., and Kim, B. J. Growing scale-free networks with tunable clustering. *Phys. Rev. E* (2002).

- [58] Hwang, J. T. Gene, and Peddada, Shyamal Das. Confidence interval estimation subject to order restrictions. *Annals of Statistics* (1994).
- [59] Karrer, Brian, Levina, Elizaveta, and Newman, M. E. J. Robustness of community structure in networks. *Phys. Rev. E* 77, 4 (Apr 2008), 046119.
- [60] Kenthapadi, Krishnaram, Mishra, Nina, and Nissim, Kobbi. Simulatable auditing. In *PODS* (2005), pp. 118–127.
- [61] Kifer, Daniel. Attacks on privacy and de Finetti’s theorem. In *SIGMOD* (2009).
- [62] Kleinberg, J. Navigation in a small world. *Nature* (2000).
- [63] Kleinberg, J. Cascading behavior in networks: Algorithmic and economic issues. *Algorithmic Game Theory* (2007).
- [64] Kleinberg, Jon, Papadimitriou, Christos, and Raghavan, Prabhakar. Auditing boolean attributes. In *PODS* (2000), pp. 86–91.
- [65] Klovdahl, A, Potterat, J, Woodhouse, D, Muth, J, Muth, S, and Darrow, W. Social networks and infectious disease: the Colorado Springs study. *Soc Sci Med* (1994).
- [66] Korolova, A, Motwani, Rajeev, Nabar, S, and Xu, Y. Link privacy in social networks. In *ICDE* (2008).
- [67] Kossinets, Gueorgi, and Watts, Duncan. Empirical analysis of an evolving social network. *Science* 311, 5757 (2006), 88–90.
- [68] Lazer, David, Pentland, Alex, Adamic, Lada, Aral, Sinan, Barabasi, Albert-Laszlo, Brewer, Devon, Christakis, Nicholas, Contractor, Noshir, Fowler, James, Gutmann, Myron, Jebara, Tony, King, Gary, Macy, Michael, Roy, Deb, and Van Alstyne, Marshall. Computational social science. *Science* 323, 5915 (2009), 721–723.
- [69] LeFevre, Kristen, DeWitt, David J., and Ramakrishnan, Raghu. Incognito: Efficient full-domain  $k$ -anonymity. In *SIGMOD* (2005).
- [70] LeFevre, Kristen, DeWitt, David J., and Ramakrishnan, Raghu. Mondrian multidimensional  $k$ -anonymity. In *ICDE* (2006).
- [71] LeFevre, Kristen, DeWitt, David J., and Ramakrishnan, Raghu. Workload-aware anonymization. In *KDD* (2006), pp. 277–286.
- [72] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *KDD* (2005), p. 187.
- [73] Leskovec, Jure, Backstrom, Lars, Kumar, Ravi, and Tomkins, Andrew. Microscopic evolution of social networks. In *KDD* (2008).

- [74] Leskovec, Jure, and Faloutsos, Christos. Scalable modeling of real graphs using Kronecker multiplication. In *ICML (2007)*.
- [75] Levina, Elizaveta, and Bickel, Peter. The Earth Mover’s Distance is the Mallows distance: Some insights from statistics. In *ICCV (2001)*.
- [76] Li, Chao, Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and McGregor, Andrew. Optimizing histogram queries under differential privacy. In *PODS (2010)*.
- [77] Li, Lun, Alderson, D, Doyle, John, and Willinger, Walter. A first-principles approach to understanding the internet’s router-level topology. In *SIGCOMM (2004)*.
- [78] Li, Lun, Alderson, David, Doyle, John, and Willinger, Walter. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics (2005)*.
- [79] Li, N, and Li, T.  $t$ -closeness: privacy beyond  $k$ -anonymity and  $\ell$ -diversity. In *ICDE (2007)*.
- [80] Liben-Nowell, D., and Kleinberg, J. The link prediction problem for social networks. In *CIKM (2003)*.
- [81] Liu, Kun, Das, Kamalika, Grandison, Tyrone, and Kargupta, Hillol. Privacy-preserving data analysis on graphs and social networks. In *Next Generation of Data Mining*. Chapman & Hall/CRC Press, 2008.
- [82] Liu, Kun, and Terzi, Evimaria. Towards identity anonymization on graphs. In *SIGMOD (2008)*, pp. 93–106.
- [83] Liu, Kun, and Terzi, Evimaria. A framework for computing the privacy scores of users in online social networks. In *ICDM (2009)*, pp. 288–297.
- [84] Machanavajjhala, Ashwin, Gehrke, Johannes, and Goetz, Michaela. Data publishing against realistic adversaries. In *VLDB (2009)*.
- [85] Machanavajjhala, Ashwin, Kifer, Daniel, Gehrke, Johannes, and Venkatasubramanian, Muthuramakrishnan.  $\ell$ -diversity: Privacy beyond  $k$ -anonymity. *ICDE (2006)*.
- [86] Machanavajjhala, Ashwin, Kifer, Daniel, Gehrke, Johannes, and Venkatasubramanian, Muthuramakrishnan.  $\ell$ -diversity: Privacy beyond  $k$ -anonymity. *ACM Trans. Knowl. Discov. Data* 1, 1 (2007), 3.
- [87] Mahadevan, Priya, Krioukov, Dmitri, Fall, Kevin, and Vahdat, Amin. Systematic topology analysis and generation using degree correlations. In *SIGCOMM (2006)*.

- [88] Martin, D.J., Kifer, D., Machanavajjhala, A., Gehrke, J., and Halpern, J.Y. Worst-case background knowledge for privacy-preserving data publishing. In *ICDE* (2007), pp. 126–135.
- [89] McPherson, Miller, Smith-Lovin, Lynn, and Cook, James M. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* (2001).
- [90] McSherry, Frank, and Mahajan, Ratul. Differentially-private network trace analysis. In *SIGCOMM* (2010).
- [91] McSherry, Frank, and Talwar, Kunal. Mechanism design via differential privacy. In *FOCS* (2007).
- [92] McSherry, Frank D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD* (2009), pp. 19–30.
- [93] Meyerson, Adam, and Williams, Ryan. On the complexity of optimal  $k$ -anonymity. In *PODS* (2004), pp. 223–228.
- [94] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: Simple building blocks of complex networks. *Science* (2002).
- [95] Mislove, Alan, Marcon, Massimiliano, Gummadi, Krishna P., Druschel, Peter, and Bhattacharjee, Bobby. Measurement and analysis of online social networks. In *IMC* (2007).
- [96] Mitzenmacher, M., and Upfal, E. *Probability and Computing*. Cambridge University Press, 2005.
- [97] Narayanan, A., and Shmatikov, V. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy* (2009).
- [98] Narayanan, Arvind, and Shmatikov, Vitaly. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy* (2008).
- [99] Newman, M. E. J. The structure and function of complex networks. *SIAM Review* 45, 2 (2003), 167–256.
- [100] Newman, Mark, and Girvan, Michelle. Finding and evaluating community structure in networks. *Physical Review E* (2004).
- [101] Newman, M.E.J., Strogatz, S.H., and Watts, D.J. Random graphs with arbitrary degree distributions and their applications. *Physical Review E* 64, 2 (2001).
- [102] Nissim, Kobbi, Raskhodnikova, Sofya, and Smith, Adam. Smooth sensitivity and sampling in private data analysis. In *STOC* (New York, NY, USA, 2007), ACM, pp. 75–84.
- [103] Owen, Guillermo. *Game Theory*. Academic Press Ltd, 1982.

- [104] Park, Juyong, and Barabasi, Albert-Laszlo. Distribution of node characteristics in complex networks. *Proceedings of the National Academy of Sciences* (2007).
- [105] Rastogi, Vibhor, Hay, Michael, Miklau, Gerome, and Suciu, Dan. Relationship privacy: Output perturbation for queries with joins. In *PODS* (2009), pp. 107–116.
- [106] Rastogi, Vibhor, Hong, Sungho, and Suciu, Dan. The boundary between privacy and utility in data publishing. In *VLDB* (2007).
- [107] Reed, Jason, Aviv, Adam J., Wagner, Daniel, Haerberlen, Andreas, Pierce, Benjamin C., and Smith, Jonathan M. Differential privacy for collaborative security. In *EUROSEC* (2010).
- [108] Robins, Garry, Pattison, Pip, Kalish, Yuval, and Lusher, Dean. An introduction to exponential random graph ( $p^*$ ) models for social networks. *Social Networks* (2007).
- [109] Russell, Stuart, and Norvig, Peter. *AI: A Modern Approach*. Prentice Hall, 2003.
- [110] Samarati, P. Protecting respondent’s privacy in microdata release. *Transactions on Knowledge and Data Engineering* (2001).
- [111] Samarati, P., and Sweeney, Latanya. Protecting privacy when disclosing information:  $k$ -anonymity and its enforcement through generalization and suppression. Tech. rep., SRI International, 1998.
- [112] Simsek, Ozgur, and Jensen, David. Navigating networks by using homophily and degree. *PNAS* (2008).
- [113] Singh, Lisa, and Zhan, Justin. Measuring topological anonymity in social networks. In *Intl. Conf. on Granular Computing* (2007).
- [114] Smith, Adam. Personal communication, 2010.
- [115] Sweeney, L. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine and Ethics* (1997).
- [116] Sweeney, Latanya.  $k$ -anonymity: a model for protecting privacy. *Journ. of Uncertainty, Fuzziness, and KB Systems* (2002).
- [117] Tangmunarunkit, H, Govindan, R, Jamin, S, Shenker, S, and Willinger, W. Network topology generators: Degree-based vs. structural. In *SIGCOMM* (2002).
- [118] Thompson, Brian, and Yao, Danfeng. The union-split algorithm and cluster-based anonymization of social networks. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security* (New York, NY, USA, 2009), ACM, pp. 218–227.

- [119] Wang, Da-Wei, Liao, Churn-Jung, and Hsu, Tsan-Sheng. Privacy protection in social network data disclosure based on granular computing. In *Intl. Conf. on Fuzzy Systems* (2006).
- [120] Warner, Stanley L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association* 60, 309 (1965), 63–69.
- [121] Watts, Duncan, Dodds, Peter, and Newman, Mark. Identity and search in social networks. *Science* (2002).
- [122] Watts, Duncan, and Strogatz, Steve. Collective dynamics of ‘small-world’ networks. *Nature* (1998).
- [123] Wong, Raymond Chi-Wing, Fu, Ada Wai-Chee, Wang, Ke, and Pei, Jian. Minimality attack in privacy preserving data publishing. In *VLDB* (2007).
- [124] Wu, Xintao, Ying, Xiaowei, Liu, Kun, and Chen, Lei. A survey of algorithms for privacy- preservation of graphs and social networks. In *Managing and Mining Graph Data*. Kluwer Academic Publishers, 2010.
- [125] Xiao, Xiaokui, and Tao, Yufei.  $m$ -invariance: Towards privacy preserving re-publication of dynamic datasets. In *SIGMOD* (2007).
- [126] Xiao, Xiaokui, Tao, Yufei, and Koudas, Nick. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM Trans. Database Syst.* 35, 2 (2010).
- [127] Xiao, Xiaokui, Yi, Ke, and Tao, Yufei. The hardness and approximation algorithms for  $\ell$ -diversity. In *EDBT* (2010).
- [128] Ying, X, and Wu, X. Randomizing social networks: a spectrum preserving approach. In *SIAM Conf. on Data Mining* (2007).
- [129] Ying, X., and Wu, X. Graph generation with prescribed feature constraints. In *SIAM Conf. on Data Mining* (2009).
- [130] Ying, Xiaowei, Pan, Kai, Wu, Xintao, and Guo, Ling. Comparisons of randomization and  $k$ -degree anonymization schemes for privacy preserving social network publishing. In *SNA-KDD '09: Proceedings of the 3rd Workshop on Social Network Mining and Analysis* (New York, NY, USA, 2009), ACM, pp. 1–10.
- [131] Ying, Xiaowei, and Wu, Xintao. On link privacy in randomizing social networks. In *PAKDD* (2009).
- [132] Zheleva, Elena, and Getoor, Lise. Preserving the privacy of sensitive relationships in graph data. In *PinKDD Workshop* (2007).

- [133] Zheleva, Elena, and Getoor, Lise. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW* (2009), pp. 531–540.
- [134] Zhou, Bin, and Pei, Jian. Preserving privacy in social networks against neighborhood attacks. In *ICDE* (2008).
- [135] Zhou, Bin, Pei, Jian, and Luk, Wo-Shun. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations* (2008).
- [136] Zou, Lei, Chen, Lei, and Ozsü, Tamer. *K*-Automorphism: A general framework for privacy preserving network publication. In *VLDB* (2009).