Graph Anonymization Through Edge and Vertex Addition

by

Gautam Srivastava

B.Sc., Briar Cliff University, 2004

M.Sc., University of Victoria, 2006

A Dissertation Submitted in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Graph Anonymization Through Edge and Vertex Addition

by Gautam Srivastava

B.Sc., Briar Cliff University, 2004

M.Sc., University of Victoria, 2006

Supervisory Committee

Dr. Venkatesh Srinivasan, Supervisor

(Department of Computer Science)

Dr. Bruce Kapron, Supervisor

(Department of Computer Science)

Dr. Alex Thomo, Departmental Member

(Department of Computer Science)

Dr. Valerie King, Departmental Member

(Department of Computer Science)

Dr. Issa Traore, Outside Member

(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Venkatesh Srinivasan, Supervisor

(Department of Computer Science)

Dr. Bruce Kapron, Supervisor

(Department of Computer Science)

Dr. Alex Thomo, Departmental Member

(Department of Computer Science)

Dr. Valerie King, Departmental Member

(Department of Computer Science)

Dr. Issa Traore, Outside Member

(Department of Electrical and Computer Engineering)

ABSTRACT

With an abundance of social network data being released, the need to protect sensitive information within these networks has become an important concern of data publishers. In this thesis we focus on the popular notion of k-anonymization as applied to social network

graphs. Given such a network N, the problem we study is to transform N to N', such that some property P of each node in N' is attained by at least $k-1$ other nodes in N'. We study edge-labeled, vertex-labeled and unlabeled graphs, since instances of each occur in real-world social networks.

Our main contributions are as follows

- When looking at edge additions, we show that k-label sequence anonymity of arbitrary edge-labeled graphs is NP-complete, and use this fact to prove hardness results for many other recently introduced notions of anonymity. We also present interesting hardness results and algorithms for labeled and unlabeled bipartite graphs.

- When looking at node additions, we show that on vertex-labeled graphs, the problem is NP-complete. For unlabeled graphs, we give an efficient (near-linear) algorithm and show that it gives solutions that are optimal modulo k, a guarantee that is novel in the literature.

We examine anonymization both from its theoretical foundations and empirically, showing that our proposed algorithms for anonymization maintain structural properties shown to be necessary for graph analysis.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

My mom, dad, and sister for supporting me throughout my education.

Venkatesh Srinivasan and Bruce Kapron for mentoring, support, encouragement, and patience.

Uvic, for funding me for all these years.

I believe I know the only cure, which is to make one's centre of life inside of one's self, not selfishly or excludingly, but with a kind of unassailable serenity-to decorate one's inner house so richly that one is content there, glad to welcome any one who wants to come and stay, but happy all the same in the hours when one is inevitably alone.

Edith Wharton

DEDICATION

I dedicate this Thesis to my all my teachers, coaches and professors I have ever had, you have all contributed to my life in one way or another.

Chapter 1

Introduction

Social networks are a natural phenomenon that have been studied for a long time by sociologists, anthropologists and biologists. Since the explosion of web based communities, social networks that contain social links that are either implicit (e.g. Amazon and IMDB) or explicit (e.g. Twitter and Facebook) have increased in their popularity. These social networks can be represented easily using graph like structures connecting people to one another with edges. This was described in detail by Kleinberg in [25].

If we view these online communities as digital frameworks that mirror the real world,

Figure 1.1: Social Network Represented as a Graph

there is a lot of intrinsic value in the analysis of the data stored in these social networks. In the real world, companies and researchers use surveys, questionnaires, grocery bills, buying habits, etc to help identify trends that people have by demographics, cultures, and seasons to name a few. Think about how Walmart decides to stock their shelves in the Winter as opposed to in the summer, and also price points on common items. Similarly, the amount of pertinent information that can be gained from the analysis of social networks is plentiful. Data miners can use the analysis of social networks to infer trends, study personal habits, even help cure disease.

The data that makes up most of these social networks is personal data. If we take Facebook as an example, we can create a social network graph by having people represented by nodes and edges denote friendships between people shown in Figure 1.1. While significant amounts of useful information may be extracted from this kind of network data, there are many privacy concerns that need to be addressed before the data is released. Particularly, the data may contain sensitive information about individuals that should not be disclosed without compromising their privacy. Examples include Facebook, Twitter, LinkedIn, and many other online social networks that have become the social lifeline of many individuals.

Consider the example of the online social media site PatientsLikeMe. Members of this online community get a chance to connect and share information with other patients suffering from life-changing diseases. This information could be vital in the study of Disease Research. However, can we ensure that patients' sensitive information, in this case a disease, can be protected while allowing the study of such vital information? It has already been shown that naive attempts to hide this sensitive information, such as removing names of people or identifying ID numbers, does not work [3, 21] and is open to a collection of

attacks. They showed, among other attacks, attacks that could check for the existence of edges between targeted nodes in the anonymized version of the network, and from this deduce the identity of anonymized nodes. These results demonstrate the need for a rigorous approach to graph anonymization.

Social networking sites have become increasingly popular among Canadian Internet users. According to a poll by TNS Canadian Facts, a Canadian marketing and social research firm, online teens and young adults are the heaviest users of social networking sites, with 83 percent of 13-17 year olds and 74 percent of 18-29 year olds having visited at least one such site. Six in 10 people in their 30s have visited at least one social networking site and 45 percent of those in their 40s have done so (see *http://www.priv.gc.ca*).

Many online users do not take the time to really read and understand the user agreements required by all social networks. As online media consumers, we are used to clicking a box and ignoring the text inside. It is becoming obvious that a lot of Canadians - and others - are signing over their privacy rights to these companies in exchange for access to increasingly popular social networks. These companies are then giving/selling this information over for data mining.

In a complaint filed by Canadian Internet Policy and Public Interest Clinic (CIPPIC) against Facebook Inc., we see some of the problems in the privacy of such online social networks. In summary, the complaint against Facebook by the (CIPPIC) comprised 24 allegations ranging over 12 distinct subjects. These included default privacy settings, collection and use of users personal information for advertising purposes, disclosure of users personal information to third-party application developers, and collection and use of non-users personal information. This clearly shows the necessity for anonymization techniques

of personal data that is embedded in these networks. Similar breaches in privacy can occur from any online social network, even a popular community such as Facebook.

If that is not enough motivation, let us revisit the earlier example of the online Social Network called PatientsLikeMe. A social network like this could be vital in the study of disease prevention. However, in an article by Jim Edwards entitled “PatientsLikeMe Is More Villain Than Victim in Patient Data Scraping Scandal”, Edwards describes the social network as a villain whose whole business model is based on selling private patient information to the highest bidder. Clearly, its clients are drug companies who want to know what patients say about their drugs when they are not around.

From a high level view, there are two general families of methods for achieving network data privacy. The first family encompasses “data anonymization” methods. These methods first transform the data and then release it. The data is thus made available for unconstrained analysis. The second family encompasses “privacy-aware computation” methods, which do not release data, but, rather, only the output of an analysis computation. The released output is such that it is very difficult to infer from it any information about an individual input datum. The relatively recent differentially-private methods (cf. [12, 13, 14, 31]) all belong in this family. Both families of methods have natural pros and cons. Methods in the first family give complete freedom to the analysts to perform any analysis they wish on the released data. However, they can be more vulnerable to attack. On the other hand, methods in the second family can protect the data better, but in the end do not release data, only carefully computed private outputs of specific computations. This obviously limits further analysis. Our approach belongs in the first family. Our goal is to anonymize social networks without significantly distorting them so that they can be released.

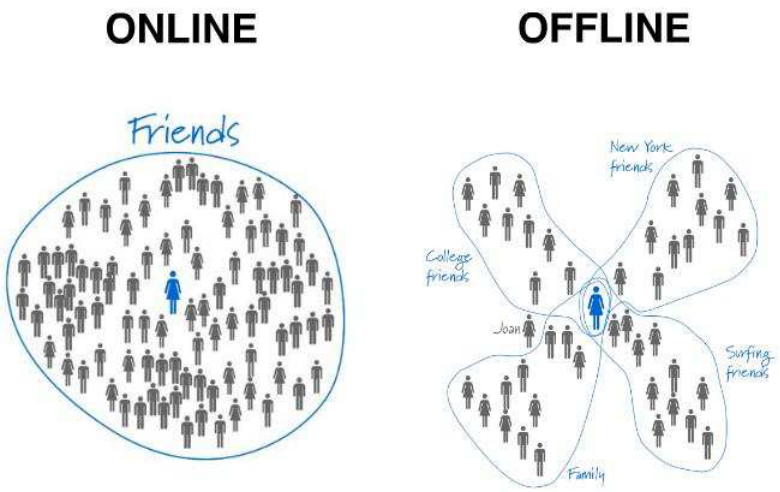


Figure 1.2: Online versus Offline Social Networks

Social networks fall into two main categories; online and offline social networks (See Figure 1.2) [10]. Offline social networks are networks created in the real world, through connections we make in our everyday lives. Through work, school, hobbies we can create a social networks of relationships to our friends. Online social networks are just that, connection we have made through social network sites like Facebook, Twitter, LinkedIn, etc. Both categories of networks would be attractive for data analysis as they may show different trends about a persons interactions online and offline. Our methods of anonymization are pertinent to both categories. More specifically, if a social network can be represented in the form of a graph (unlabeled or labeled), our anonymization techniques and results can apply to it.

We tackle the problem of Graph Anonymization 3-fold in our work; (1) anonymization through edge addition, (2) anonymization through node addition, and finally (3) looking beyond these methods of anonymization.

Given a social network graph G, can we anonymize G using modifications to the structure of G? That is the basic premise of anonymization. We look to k-anonymize a Social Network G, by making sure that each node $v \in G$ is identical to $k-1$ other nodes in G in some property P. This property P has many choices. In [29], the property P was the degree of the vertex. In this case, given a graph $G = (V, E)$, for every vertex $v \in V$, we count the number of edges adjacent to that vertex. We call this number the degree of the vertex. For k-degree anonymization, we require every vertex $v \in V$ to have at least $k-1$ other vertices in G of the same degree. Another property was to focus on the subgraph induced on the neighbors of a vertex [42]. In this version, we require this neighborhood graph to be isomorphic to $k-1$ other neighborhood graphs in G. Other anonymization notions grew off of these two main properties.

1.0.1 Anonymization Through Edge Addition

For anonymization through edge addition, we study natural generalizations of k-degree anonymity introduced in the previous section. Firstly, when social network data is represented as a graph, it is likely that we would like to anonymize only a subset of the nodes. For example, in a social network, some users may agree to have their information released, while others wish to remain anonymous, a form of personal privacy. In such a situation, it may be desirable to anonymize only a subset of the entire vertex set. Therefore, we introduce and focus on the problem of *subset anonymization*.

Secondly, the graph representing a social network can have labeled edges. As the need to represent social networks as graphs grows, so will the amount of information that needs to be stored in these graphs. The label often gives auxiliary information associated with

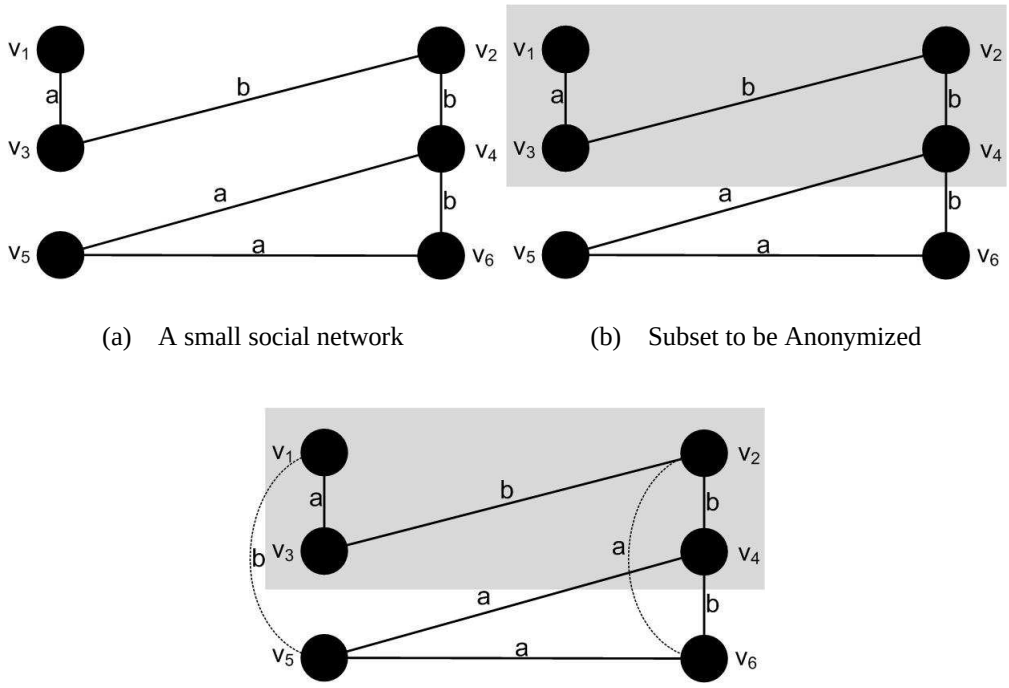
a relationship. In a purchasing example, an edge may represent the fact that a shopper has bought a certain product. Associated with such a relationship could be data such as dates of purchase, quantities, ratings, etc. In order for our graph model to support this way of associating auxiliary data with relationships, we will consider graphs whose edges are labeled by elements of some label set. For such graph, the degree of a vertex is replaced by its *label sequence* containing all the labels of the edges incident on it.

These considerations lead to the problem of k-label sequence subset anonymity in which we are given an edge labeled graph and we would like to ensure that a given subset of vertices of G is k-label sequence anonymous by adding a minimum number of edges. We will also study this problem for bipartite graphs, where the vertices to be anonymized are from one side of the bipartition. The bipartite model is useful in cases where vertices represent two types of entities, and edges exist only between entities of different types.

Example 1. *Here we see an example of k-label sequence subset anonymity. If we look at the label sequences of the vertices $\{v_1, v_2, ...\}$ in order we have $\{(a), (b, b), (a, b), (a, b, b), (a, a), (a, b)\}$. If we look at Figure 1.3(b), we choose a subset of v_1, v_2, v_3, v_4 to anonymize shown in the shaded area. Looking at this subset of vertices, if we add the dotted edges as in Figure 1.3(c), we get a new label sequence of $\{(a, b), (a, b, b), (a, b), (a, b, b)\}$ for the subset, giving us a 2-label sequence subset anonymous graph.*

1.0.2 Anonymization through Node Addition

For anonymization through node addition, we take a new approach to anonymization and look to modify a graph's node set to achieve anonymity. In previous studies, networks



(a) A small social network

(b) Subset to be Anonymized

(c) 2-label sequence subset anonymous graph

Figure 1.3: Example 1 of k-label sequence anonymity

are typically anonymized by exclusively introducing new edges into the network, keeping the vertex set unchanged. This approach is justified under the assumption that one does not wish to add new entities into the network, but we challenge this assumption. Even for microdata, the analysis that one wishes to do with the network data is at the aggregate level, so the introduction of new nodes does not necessarily have an adverse affect. To the contrary, adding new nodes with similar properties could better preserve aggregate measures than will distorting the existing nodes.

In fact, one ought to consider the intended use of the anonymized network prior to conducting the anonymization, because this affects which characteristics should be preserved. To facilitate this choice, it is important to develop alternate approaches with respective advantages. We introduce the natural complement to current k-anonymization, an approach of augmenting the network with new vertices which are connected to themselves and to the original graph. In this manner, one guarantees, for example, not to increase the size of any clique by more than one, and is very unlikely to do anything but introduce new 2- and 3-cliques. If large cliques are of particular interest to an analyst, this is clearly preferable, because adding edges among original vertices can produce false positives. Alternatively, for analysis tasks that involve monotone properties such as independent sets, the distortion is more controlled in that vertex addition can only introduce false positives, whereas edge addition can introduce false negatives, too.

This work is inspired by results of König, Erdős, and Kelly [15, 26]. König showed that, given a graph G with maximum degree d, it is always possible to to make a d-regular graph H by adding vertices and adding edges whose endpoints must include at least one new vertex. In a subsequent paper, Erdős and Kelly strengthened the result of König by giving an efficient algorithm to determine the smallest possible number of new vertices to be added to G to obtain such a graph H.

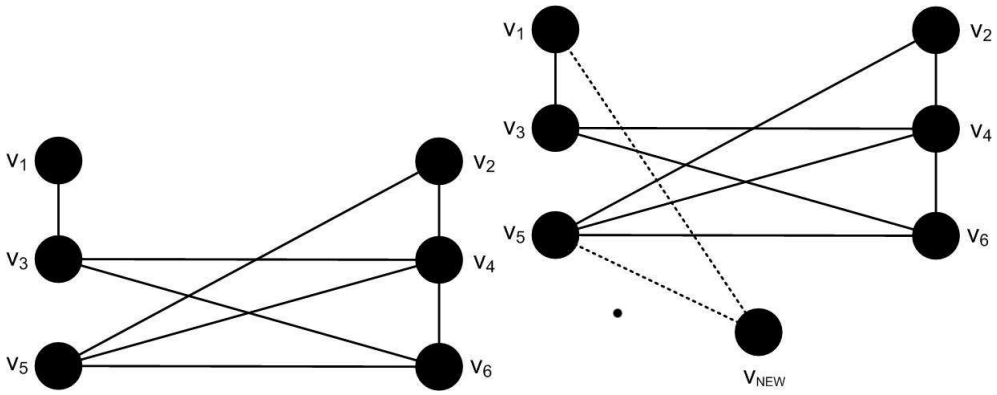
Interpreting these results in the context of graph anonymization, these papers showed that the following decision problem is in P: Given an integer m and any arbitrary graph G on n vertices, is it possible to add at most m new vertices and edges only between new vertices or an old vertex and a new vertex so that the resulting graph H on $m + n$ vertices is $(m + n)$-degree-anonymous? Requiring every vertex to be indistinguishable from every

other vertex is far too drastic for graph anonymization, however. But, two natural ways to relax this problem specification come to mind. First, we may not require that all the nodes of a graph be anonymized. For example, in the Netflix database, there are two types of vertices–customers and movies–and the database owner could be interested only in anonymizing the vertices representing the customers. Second, as in many other graph anonymization problems described above, we might only be required to k-anonymize the set of vertices for some reasonably small value of $k << d$.

Motivated by both these considerations, we study a natural generalization of the problem described above: Given a graph $G = (V, E)$, $X \subseteq V$, and an integer k, add the fewest possible new vertices and add edges satisfying the condition above to get a graph H in which X is k-degree-anonymous. We also study the analog of this problem in the vertex-labeled setting. The techniques of Erdős and Kelly do not generalize as nicely as the problem specification, so we develop here some different ideas.

We believe that there are many advantages to our problem specification. Firstly, the original graph is an induced subgraph of the new anonymized graph. This is not the case with previous approaches to graph anonymization. Due to this property, interesting graph parameters (such as those we evaluate in §3.4) will not be altered substantially so that the graph is still useful for analysis. Secondly, our approach is amenable to theoretical analysis. As described below, we are able to study the complexity of this problem for labeled and unlabeled graphs. The exact complexity of many previously defined notions of anonymization are not fully understood yet.

Example 2. *Here we see an example of anonymization through node addition. If we examine the graph in Figure 1.4(a), we see a 6 vertex graph with the degree of the 6 nodes*



(a) A small social network (b) 2-anonymous Graph

Figure 1.4: Example 2 of Anonymization through Node Addition

being $\{1, 2, 3, 3, 3, 4\}$ *respectively. Added a 7th vertex* v_{new} *to the graph and adding the dotted edges in Figure 1.4(b), we get 7 vertices of degree* $\{2, 2, 2, 3, 3, 4, 4\}$ *which is a* 2*-anonymous graph.*

1.0.3 Privacy Beyond k-Anonymity

Previous attempts at anonymization mostly focused on protecting the identity of the users. The next step is to try and protect the user information that can become available when we "'friend"' or "'link"' to other users. Ideally we not only want to protect our information, but also that of our friends. Take for example Photo Albums. When we connect to other users on Facebook, they often are given access to our pictures. If we take pictures of our friends, then their photos are linked to our accounts as well. In doing so, a simple picture of a friends house, linked to a user's account may allow an adversary information about

friends even though there is no direct link between adversary and friends.

To date there has been copious research on protecting the identity of users from adversaries who exploit structural properties of social network graphs. However, another very important class of attacks—that of identifying just the sensitive information rather than the identity of users—has been mostly ignored. We look to answer the following question: Given a social network, can we protect against an adversary who uses certain targeted nodes within the network to gain sensitive information about the friends of those nodes? Consider the example of Facebook, where this attack model can be quite effective. User u may often get many unknown *friend requests*, which, if accepted, create friendship links to his account. By establishing these links, an adversary then gains access to the network of u and to any sensitive information that a friend of u has made accessible to *friends of friends*, which now includes the adversary.

We introduce a new measure of anonymization called α-proximity to capture the susceptibility of a network to this type of attribute disclosure attack. A danger exists when particular neighbourhoods have a vastly different distribution of a given sensitive attribute than does the network as a whole, because an adversary can then conclude with greater confidence the values for that attribute of the neighbours of a targeted node. To protect against this attack on a given vertex-labeled social network G, we require that the distribution of labels within the neighbourhood of each node in the graph be within α of the overall probability distribution of the labels in the graph.

Consider the following example:

Example 3. *Figure 1.5(a) depicts a labeled graph with 6 nodes. The probability distribution of label* ***a*** *in the graph is* $p_a = 0.5$*. The label sequences of the 3 nodes labeled with* ***b***

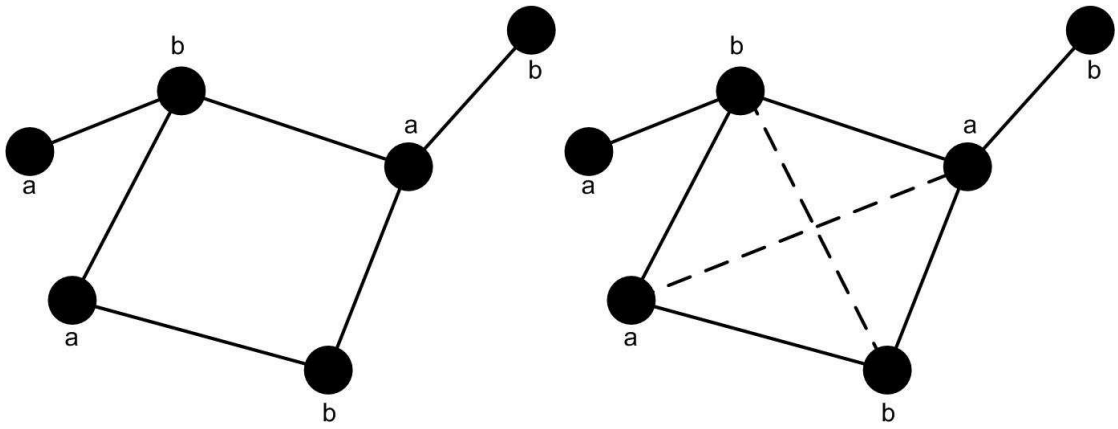
are $\{(b,a,a,a),(b,a,a),(b,a)\}$*. If an adversary were to connect to one of these 3 nodes, in such exposing their neighbours to him, he could conclude that the neighbours have label* ***a*** *with* $p_a = 0.75, 0.67$*, and* 0.5*, respectively. Only the third node's neighbourhood has the same distribution of* ***a*** *labels as does the overall graph.*

In Figure 1.5(b), 2 dotted edges have been added to the graph, which changes the label distributions of each neighbourhood. Now, instead, the label sequences are $\{(b,b,a,a,a),(b,b,a,a),(b,a)\}$*, and label distribution in each neighbourhood of label* ***a*** *is* $p_a = 0.6, 0.5$*, and* 0.5*, very closely matching the overall distribution of the labels in the graph. No matter which of the neighbourhoods the adversary can identify, he cannot refine his prediction of the label of any node in that neighbourhood by more than* 0.1.

These labels can correspond to very sensitive information. Consider the example of the online social media site PatientsLikeMe. Members of this online community get a chance to connect and share information with other patients suffering life-changing diseases. This information could be vital in the study of Disease Research. However, can we ensure that patients' sensitive information, in this case a disease, can be protected while allowing the study of such vital information?

1.1 Related Work

With its roots in databases and the contained tables, early work in privacy dealt with the privacy of statistical tables (databases) using inference control [11, 20, 35]. The current work in data privacy took form with anonymization schemes centered around the notion of k-anonymity, wherein each row in a table must be identical to and therefore indistin-



(a) A small social network

(b) The network transformed to be $(.1)$-proximal

Figure 1.5: Example of α-proximity

guishable from at least $k-1$ other rows. Work by Meyerson and Williams [32] and by Agarwal et al. [1] set the foundations of k-anonymity for tables showing the problem was NP-hard even for a reduced alphabet size. Table privacy beyond k-anonymity followed with a new notion called l-diversity [30], wherein each k-anonymous equivalence class required l different values for each sensitive attribute. In this way, l-diversity looked to not only protect identity disclosure, but was also the first attempt to protect against attribute disclosure. There are two types of privacy attack for data [28], namely identity disclosure and attribute disclosure. Identity disclosure often leads to attribute disclosure. Identity disclosure occurs when an individual is identified within a dataset, whereas attribute disclosure occurs when information that an individual wished to keep private is identified.

To address the shortcomings of l-diversity, Li et al. [28] introduced t-closeness, which

requires that the distribution of attribute values within each k-anonymous equivalence class needs to be close to that of the attribute's distribution throughout the entire table.

For graph data, anonymization has followed a similar path, starting with naive anonymization techniques in [21], and moving towards a simple k-anonymity version [29], leading to more sophisticated versions of anonymity [6, 9, 36, 37, 39]. We will describe the results of these papers below. We would like to remark that the focus of the work in our thesis is on the understanding of the complexity of the various notions of anonymization listed above. While the goal of previous research was to obtain algorithms that worked well in practice, our aim is to systematically understand the complexity of the underlying problems.

From the work in [29] focusing on the simplest form of anonymity known as degree anonymity, the work evolved to neighbourhood anonymity by Zhou and Pei [42], which was expanded by Tripathy and Panda [37]. In their model, an adversary uses information about a node's neighbours to target them. To prevent such attacks, they define a notion of k-anonymity on graphs so that nodes in an anonymized group will have isomorphic neighbourhoods. They show that anonymizing a graph under their definition using a minimal number of edge additions is NP-hard and they develop a method that well works in practice. Other landmark papers in the field [21, 41, 36, 6, 39] have introduced models of protecting from progressively stronger adversarial knowledge. These are summarized in Table 1.1.

For all of these adversarial models, it is important to understand the challenges in producing networks anonymized with respect to those models. Deepening the understanding of k-anonymity here is an important step in that direction.

Zheleva and Getoor [41] study the problem of protecting certain sensitive edges in

an edge-labeled graph under link re-identification attacks. They propose anonymization techniques using edge-removal and node-merging to prevent such attacks.

Hay *et al.* [21] model the information available to the adversary using two types of queries–vertex refinement queries and subgraph knowledge queries–and study the vulnerability of various datasets under such an attack. They propose an anonymization technique based on random perturbations against such adversaries.

Thompson and Yao [36] study i-hop degree-based attacks. In their model an adversary's prior knowledge includes the degree of the target and the degree of its neighbours within i hops. They develop a inter-cluster matching method for anonymizing graphs against 1-hop attacks through edge addition and deletion. Thomson and Yao use bipartite graphs, namely the Netflix Prize Data, to help motivate their work.

Wu et al. [39] recently proposed the k-symmetry model. They state for any vertex v in the network, there exists at least $k-1$ structurally equivalent counterparts. The authors also proposed sampling methods to extract approximate versions of the original network from the anonymized network so that statistical properties of the original network could be evaluated.

Cormode et al [9] consider a new family of anonymizations, for bipartite graph data, called (k,l)-groupings. These groupings were used to preserve the underlying graph structure perfectly, and instead anonymize the mapping from entities to nodes of the graph. They created "'safe"' groupings that were able to withstand a set of known attacks.

The other landmark work that is very closely related to our research on anonymization by node addition is of König, Erdős, and Kelly [15, 26], since our work extends their graph theoretic results. König showed that, given a graph G with maximum degree d, it is

always possible to make a d-regular graph H by adding vertices and adding edges whose endpoints must include at least one new vertex. In a subsequent paper, Erdős and Kelly gave an efficient algorithm to determine how many vertices must be added to obtain such a graph H. We generalize this problem with two relaxations. First, we may not require that all nodes of a graph be anonymized. For example, in the Amazon database, there are two types of vertices, customers and products, and the database owner could be interested only in anonymizing the customer vertices. Second, we k-anonymize the graph for an arbitrary k (which we typically assume to be some reasonably small value $<< d$).

Recently, in [40], a new angle was looked at as far as privacy protection. Users would have the ability to choose their 'level' of necessary privacy, and from these choices anonymization procedures would follow. Low level anonymization would use something simple such as naive anonymization whereas high-level anonymization would use techniques similar to what are described in this work. This type of personal privacy helps to motivate our sections on subset anonymity, where only a subset of the vertex set would require anonymization.

Table 1.1 summarizes the previous work in the field and highlights the main contributions of their research.

Table 1.1: Summarization of Related Work

Authors	**Adversarial Model**	**Graph Type**	**Permitted Operations**
Erdős & Kelly [15, 26]	n-Anonymization	Unlabeled	Vertex/Edge Addition
Liu and Terzi [29]	k-Anonymization	Unlabeled	Edge Addition/Removal
Zhou & Pei [42]	k-N'hood Anon.	Vertex-Labeled	Edge Addition
Cheng et al. [6]	k-Isomorphism	Unlabeled	Edge Addition/Removal
Hay et al. [21]	Automorphic Equiv.	Unlabeled	Label Modifications
Wu et al. [39]	k-Symmetry	Unlabeled	Vertex/Edge Addition
Tripathy and Panda [37]	k-N'hood Anon	Vertex-Labeled	Edge Addition
Kapron et al. [24]	k-Label Seq. Anon.	Vertex/Edge Labeled	Edge Addition
Chester et al. [8]	k-Anonymization	Vertex Labeled	Vertex Addition
Chester et al. [7]	α-proximity	Vertex Labeled	Vertex Addition
This Thesis	k-Anonymization	Vertex-, Edge- and unlabeled **and** Unlabeled	Vertex and Edge Addition

1.2 Our Results

In this thesis, we make the following contributions:

1.2.1 Edge Additions

- For edge-labeled graphs, we consider k-anonymization with respect to the collection of labels of incident edges. We deal with k-anonymization of subsets in arbitrary labeled graphs. We prove an NP-completeness result for this problem in a class of labeled graphs we call *table graphs*. We then use this result to prove the hardness of many seemingly different notions of graph anonymization that have recently appeared in the literature, thus providing a uniform approach to studying the complexity of graph anonymization problems.

- We consider subset k-anonymization of labeled bipartite graphs. For $k = 2$, we present a polynomial time algorithm, based on a recent algorithm of Anshelevich and Karagiozava [2] for finding minimum weight perfect matching in hypergraphs with edges of size two or three. When $k \geq 3$ we show that the problem is NP-complete.

- In the unlabeled case, we consider k-anonymization with respect to the degree of vertices. We present an algorithm for subset k-degree anonymization of unlabeled bipartite graphs that runs in time $O(n(k + d_{max}))$, where n is the number of vertices in the graph and d_{max} is the maximum degree of a vertex in the graph. We use a dynamic programming approach to achieve this bound.

1.2.2 Node Additions

- We prove that on vertex-labeled graphs, k-anonymization with a minimum number of vertex additions is NP-complete by giving a reduction to a hard table anonymization problem.

- For unlabeled graphs, we introduce an efficient (i.e., $\mathcal{O}(nk)$) k-anonymization algorithm based on dynamic programming and prove that it produces a solution that is optimal modulo k.

- We perform experiments with several well-known network datasets to demonstrate empirically that our vertex-addition approach to k-anonymization quite minimally distorts the original graph with respect to standard parameters like clustering coefficient, average path length and connectivity, even as k approaches d .

1.2.3 Privacy Beyond k Anonymity

- We propose a novel and advanced notion of data anonymization called α-proximity that protects against attribute disclosure attacks.

- We provide an algorithm that modifies a vertex-labeled graph G, so as to ensure it is α-proximal. We use a greedy approach which is common with these types of algorithms and show strong termination conditions.

- We illustrate empirically that the algorithm can transform a graph into an α-proximal graph with a quite conservative number of additional edges.

Chapter 2

Anonymization Through Edge Addition

In this chapter, we show our results pertaining to anonymization through edge addition for edge-labeled graphs. In §2.1 we first introduce the definitions for anonymization of tables, unlabeled and edge-labeled graphs. We then prove our main NP-hardness result in §2.2. We then use this result in §2.3 to show NP-hardness of many different notions of graph anonymization introduced recently. We end the chapter with §2.4 and §2.5 dealing with our results for unlabeled and labeled bipartite graphs.

2.1 Preliminaries

2.1.1 Tables and k-anonymity

Table Anonymization has been extensively studied [1, 4, 16, 18, 32]. Suppose we want to publish a table of data containing potentially sensitive information. To help protect the data, we have the ability to suppress the data entries in the table with *'s. To achieve k-

anonymization by suppressing the entries, we require that after suppression, for any given row in the table, there are $k-1$ other rows that look identical.

Table 2.1: Table Data before Anonymization

First Name	**Last Name**	**Age**	**Grad Year**
Harry	Potter	30	2012
John	Connor	45	2013
Harry	Houdini	30	2010
Sarah	Connor	32	2013

If we want to 2-anonymize the data in Table 2.1, then using the fewest suppressions to achieve 2-anonymity would result in Table 2.2 as shown.

Table 2.2: 2-Anonymous Table

First Name	**Last Name**	**Age**	**Grad Year**
Harry	*	30	*
*	Connor	*	2013
Harry	*	30	*
*	Connor	*	2013

The work of k-anonymization of tables was refined slightly in later attempts separating the attributes themselves into identifiers, quasi-identifiers, and sensitive attributes. Identifiers were attributes that directly identified a person. Examples would include, Name, Social Security Number, etc. Quasi-identifiers, were attributes that when combined with other quasi-identifiers could identify a person. Combinations such as Address with Age could help uniquely identify a person. Sensitive attributes would not be able to uniquely identify a person on their own, however this would be information that people would want to keep private. In a medical table, Disease would be an example of a sensitive attribute.

Taking a look at the table below

Table 2.3: Table Data before Anonymization

Address	Age	Zip Code	Disease
Cooper Street	34	51104	Heart Disease
Lex Avenue	20-30	90210	Cancer
Cooper Street	44	51104	Cancer
Michigan Place	20-30	23134	Cancer

We anonymize only quasi-identifiers, stripping identifiers (left out of table) and leaving sensitive attributes untouched. We get

Table 2.4: Table Data After Anonymization

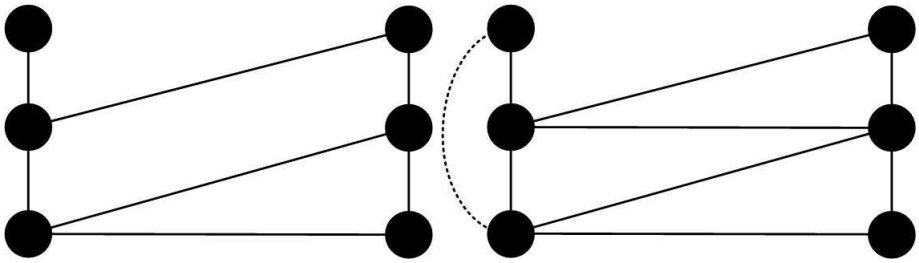
Address	Age	Zip Code	Disease
Cooper Street	*	51104	Heart Disease
*	20-30	*	Cancer
Cooper Street	*	51104	Cancer
*	20-30	*	Cancer

We now define the anonymization of tables more formally:

Definition 2.1.1. *A table consisting of a multiset V of* rows*, that is sequences of length m over a set Σ of* entry values*. Let $t : V \longrightarrow (\Sigma \bigcup \{*\})^m$. If for all $v \in V$ and $j = 1, \ldots, m$ it is the case that $t(v)_j \in \{v_j, *\}$, we call t a* suppressor*. The table $t(V)$ resulting from a suppressor t is defined to be k-anonymous if and only if for all $v \in V$ there exist at least $k-1$ distinct rows $v_1, \ldots, v_{k-1}$ such that $t(v) = t(v_1) = \ldots = t(v_{k-1})$. In other words, after applying t, each row is identical to at least $k-1$ other rows.*

Anonymizing entries is hard

In [32], the problem of finding the minimum number of suppressions to anonymize a table was proven NP-hard for $k \geq 3$ and $|\Sigma| \geq n$, where n is the number of rows in the table



(a) Small Graph (b) 3-anonymous Graph

Figure 2.1: Example 4 of k-anonymity

needed to be anonymized. After this result, [1] lowered the alphabet size to $|\Sigma| = 3$. Finally, it was shown in [4] that the problem remains hard for $|\Sigma| = 2$ and $k \geq 3$.

2.1.2 Unlabeled Graphs and k-Anonymity

Let $G = (V, E)$ be a simple graph where V, $|V| = n$, denotes the set of vertices and E denotes the set of edges. We denote the degree of a vertex v by $d(v)$.

Definition 2.1.2 (Degree Sequence)**.** *Let $X = \{x_1, x_2, \ldots, x_t\}$, $X \subseteq V$, be a subset of vertices of G. The* degree sequence *of X is $(d_1, d_2, \ldots, d_t)$ where $d_i = d(x_i)$ is the degree of the vertex x_i.*

Definition 2.1.3 (k-Anonymity)**.** *A sequence of values $S = (s_1, s_2, \ldots, s_t)$ is said to be k-anonymous if every distinct value in S occurs at least k times. A subset of vertices X in a graph G is k-anonymous if its degree sequence is k-anonymous.*

Example 4. *In Figure 2.1(a), we see a graph G on 6 nodes. From the definition for degree sequence, we get a sequence of $(1, 2, 2, 3, 4, 4)$ for the graph. After anonymization, the degree sequence in Figure 2.1(b) is $(2, 2, 2, 4, 4, 4)$, giving us a 3-anonymous graph. Note that the definition for degree sequence states $X \subseteq V$, whereas in this example $X = V$.*

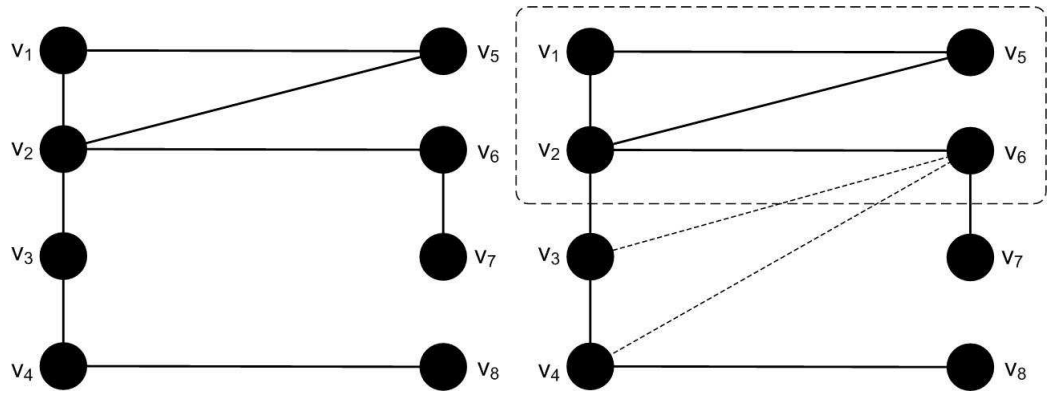
k-Degree-Based Subset Anonymization Problem (k-D-SAP):
Given a graph $G = (V, E)$, and $X \subseteq V$, find a graph $G' = (V, E \cup E')$ such that X is k-anonymous in G' and the number of new edges added, $|E'|$, is minimized.

Note: We state our anonymization problems in the optimization version of [1, 4, 32], and indeed the algorithms we give are naturally viewed in this way. On the other hand, for hardness we in fact deal with the decision version of these problems. That is, we have another input $t \in \mathbf{N}$, and we ask whether there is a set E' of edges such that G' is k-anonymous and $|E'| \leq t$.

Example 5. *Here we present a small example of k-**D-SAP**. Consider the graph G in Figure 2.2(a) Suppose we want 2-anonymity for the subset of vertices $\{v_1, v_2, v_5, v_6\}$, which has degree sequence $(2, 4, 2, 2)$. Adding the dotted edges of Figure 2.2(b) will result in the degree sequence $(2, 4, 2, 4)$, which is 2-anonymous. Since, for 2-anonymity, we require at least 2 vertices of degree 4 in the sequence, the number of edges added is the minimum.*

2.1.3 Labeled Graphs and k-Anonymity

Edge-labeled graphs are a natural model for the representation of social networks and related forms of data. The Netflix movie database [36], can be represented with nodes for movies and users and labeled edges to represent how users rank these movies.



(a) Graph G (b) 2-anonymized subset Anonymity

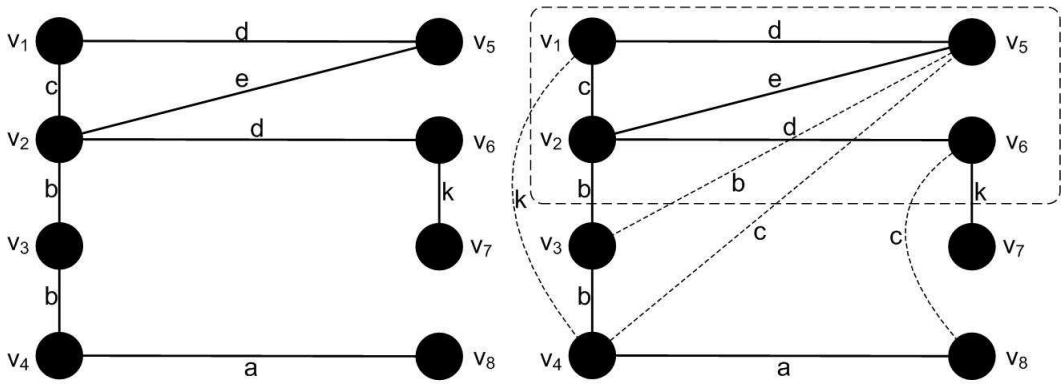
Figure 2.2: Example 1: k-D-SAP

Definition 2.1.4 (Edge-labeled Graph)**.** *An* edge-labeled graph *is a tuple* $G = (V, E, \Sigma)$ *where* V *is the set of* vertices, Σ *is the* label set *and* $E \subseteq \mathcal{P}_2(V) \times \Sigma$*, is the set of* (labeled) edges. *Here* $\mathcal{P}_2(V)$ *denotes the 2-element subsets of* V*.* E *must satisfy the property that there is at most one* $\ell \in \Sigma$ *such that* $(\{u, v\}, \ell) \in E$*. If* $(\{u, v\}, \ell) \in E$ *is a labeled edge, we say that* ℓ is the label of edge $\{u, v\}$.

Definition 2.1.5 (Label Sequence)**.** *For* $v \in V$*, we say that* $S_v = (\ell_1, \ell_2, \ldots, \ell_m)$ *is a* label sequence *of* v *if it corresponds to some ordering of the labels of the edges incident on* v*. We consider label sequences to be equivalent up to permutations.*[1]

Definition 2.1.6 (Label Sequence Anonymity)**.** *Given an edge-labeled graph* $G = (V, E, \Sigma)$*, a subset* $X \subseteq V$ *of vertices is* k-anonymous in G *if for every vertex* v *in* X*, there are at least* $k - 1$ *vertices in* X *whose label sequence is equivalent to the label sequence of* v*. If*

[1] *We use permutation-invariant sequences rather than multisets to avoid the need to deal explicitly with multiplicities.*



(a) Graph H

(b) 2-label sequence Anonymization of X in H

Figure 2.3: Example 2: Subset Label Sequence Anonymization

v and v' are vertices with equivalent label sequences we say that they are similar *and write* $v \equiv v'$.

Clearly $\equiv$ is an equivalence relation and so induces a partition $X/\equiv$ of X. We now define the anonymization problem for subsets of labeled graphs.

k-Label Sequence-Based Subset Anonymization Problem (k-LS-SAP):

Given an edge-labeled graph $G = (V, E, \Sigma)$, and $X \subseteq V$, find an edge-labeled graph $G' = (V, E \cup E', \Sigma \cup \Sigma')$ such that X is k-anonymous in G' and the number edges added, $|E'|$, is minimized.

In other words, we would like to k-anonymize X by adding the minimum number of new labeled edges to G. Note that the added edges may have labels from an expanded set $\Sigma \cup \Sigma'$.

Note: we call E' an *anonymizing set of edges* for X.

Example 6. *Here we present a small example of subset label sequence anonymization. Consider graph H in Figure 2.3(a). Here, if we have $X = \{v_1, v_2, v_5, v_6\}$, with $k = 2$ similar to Example 5, adding the dotted edges in Figure 2.3(b) with the given edge labels gives us 2-anonymity. In this case it is not sufficient just to have a 2-anonymous degree sequence; we must also consider the labels of incident edges for each vertex.*

2.1.4 Tables as Bipartite Graphs

As mentioned in §2.1, table data is often easily represented using graphs, particularly bipartite graphs.

Definition 2.1.7. *A* simple bipartite graph *is a triple (V, W, E) where V and W are* vertex sets*, and $E \subseteq V \times W$ is the set of* edges*. The pair (V, W) is called a* bipartition*, and V and W are respectively called the* left *and* right sides *of the bipartition.*

Example 7. *Consider a relational database consisting of a table for patients, a table for prescription drugs, and a table for the treatment of patients with the drugs. In Figure 2.4, we see an example instance of this database, and also its representation using a bipartite graph. Here patients are represented by vertices in V, drugs by vertices in W, and the viewing relation is represented by edges between V and W in Figure 2.4(d).*

We can add weighted edges to the graph in Figure 2.4(d) to encompass more information such as treatment time for each patient with the given drug. In general, we can easily incorporate more information from the tables in the graph this way.

Patient	Ailment
A	Flu
B	Fever
C	ADD
D	Cancer
E	Heart Disease

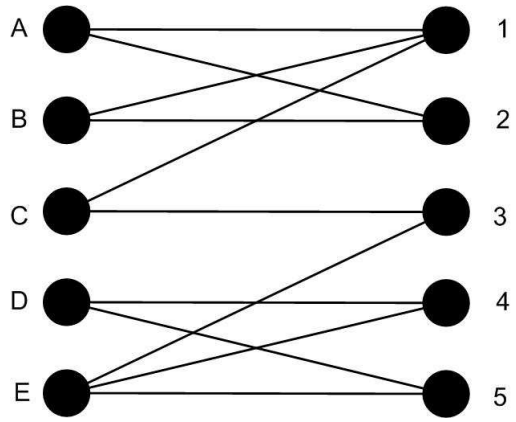
(a) Patient Table

Drug	Name
1	Ritalin
2	Aspirin
3	Tylenol
4	Penicillin
5	Diuretic

(b) Drug Table

Pat	Drug	Pat	Drug
A	1	D	4
A	2	D	5
B	1	E	3
B	2	E	4
C	1	E	5
C	3		

(c) Patient-Drug Table



(d) Patient-Drug Table Graph

Figure 2.4: Table Graph Example

2.2 Labeled Sequence Anonymization

Let $k \geq 3$ be any fixed integer, we will show k-**LS-SAP** is NP-hard by building a reduction from a Table Anonymization NP-hard anonymization problem introduced earlier in §2.1, to the decision version of k-**LS-SAP**.

k-ENTRY-ANONYMITY

Input: a Table T with n rows and l columns (also called attributes) with entries over $\{0, 1\}$ and integer t.

Question: Can the rows of T be k-anonymized by suppressing at most t entries of T? Here, an entry (0 or 1) is said to be suppressed if it is replaced by *.

Reduction: Our reduction is described as follows: Given a Table T, let $T_{(m,j)} \in \{0, 1\}$ denote the value of attribute j in row m. Then, the edge-labeled graph G_T corresponding to T is constructed as follows:

- $V_T = \{r_1, r_2, \ldots, r_n\} \cup \{c_j^i | 1 \leq j \leq l, i \in \{0, 1\}\}$.
- $E_T = \{(r_m, c_j^i, 2(j-1)+i) | T_{(m,j)} = i, 1 \leq m \leq n, 1 \leq j \leq l, i = 0, 1\} \cup \{(r_i, r_j, 2l) | 1 \leq i, j \leq n\}$.
- $\Sigma_T = \{0, 1, \ldots, 2l\}$.
- Finally, remove all isolated vertices from G_T.

Entry	A_1	A_2	A_3
1	0	1	0
2	0	0	0
3	1	1	0
4	1	1	1

Table 2.5: An Example Table T

In other words, we encode a binary table as an edge-labeled graph in which a row vertex $r_m \in V_T$ is connected to a column vertex $c_j^0 \in V_T$ (c_j^1) $\in V_T$ with label $2(j-1)$ ($2(j-1)+1$) if the (m, j)th entry of the table is 0 (1).

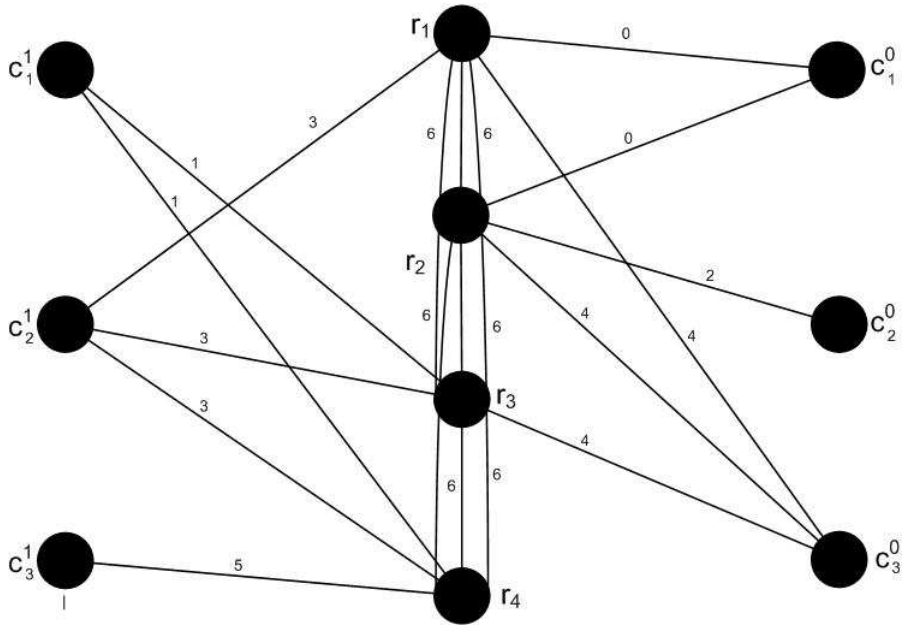


Figure 2.5: Transformation of Table T into Graph G_T

Let $X = \{r_1, \ldots, r_n\}$ denote the set of row vertices of G_T. Since there are already edges between every pair of vertices in X, no anonymizing edges will be added between these vertices. We will show that T can be k-anonymized by suppressing at most t entries if and only if we can k-anonymize X by adding at most t new labeled edges.

Let G'_T be any graph obtained from G_T such that X is k-anonymous in G'_T and it has the minimum number of new edges added. Suppose that E'_T is an anonymizing set of edges for X. Letting $\equiv$ denote vertex similarity in the anonymized graph, let $Y = \{y_1, \ldots, y_m\}$ be an equivalence class of $X/\equiv$, where $m \geq k$. We begin by establishing properties that any anonymizing set E'_T of minimum size must satisfy.

Let $Y = \{y_1, \ldots, y_m\}$ be an equivalence class of $X/\equiv$, where $m \geq k$.

Lemma 2.2.1 shows that the anonymization procedure only introduces edges with labels

already in Σ_T.

Lemma 2.2.1. *If there is an edge in E'_T labeled ℓ that is incident to Y then there is an edge in E_T labeled ℓ that is incident to Y.*

Proof. Suppose ℓ is the label of an edge in $E_T \cup E'_T$ that is incident to a vertex $y \in Y$. Then there must be an edge in E_T with label ℓ incident to some vertex $y' \in Y$. If this were not the case, then we may remove all edges labeled ℓ from E'_T which are incident to vertices in Y, and maintain the similarity of all vertices in Y with a smaller anonymizing set of edges. □

The following shows that at most one edge with label ℓ is incident to a row vertex of V_T.

Lemma 2.2.2. *For every $i \in \{0, 1\}$, and every $j \in \{1, 2, \ldots, l\}$, the label $2(j-1)+i$ appears at most once in the label sequence of a vertex $y \in Y$.*

Proof. We first show that if there is an edge in E_T labeled ℓ that is incident to $y \in Y$ then there is no edge in E'_T labeled ℓ that is incident to y. The proof proceeds by contradiction. Suppose there is such an edge labeled ℓ in $E_T \cup E'_T$ that appears more than once. So the label ℓ occurs more than once in the label sequence of y, and hence of every node in Y. By construction only **1** of these occurrences is due to an edge in E_T. We may remove the edges in E'_T corresponding to the other occurrences and maintain the similarity of all vertices in Y with a smaller anonymizing set E'_T. On the other hand, suppose that there is an edge labeled ℓ in E'_T that appears more than once and is incident to y but there is no edge labeled ℓ in E_T. Then, we note again we may remove all edges labeled ℓ from E'_T

which are incident to vertices in Y, and maintain the similarity of all vertices in Y with a smaller anonymizing set E'_T. □

Lemma 2.2.3. *There is no edge labeled $2l$ in E'_T that is incident to Y.*

Proof. Suppose that there is an edge labeled 2ℓ in E'_T incident to $y \in Y$. Since the number of edges labeled 2ℓ is the same for every vertex of Y in G_T, there must be an edge labeled 2ℓ in E'_T incident to every vertex of Y. We may remove all edges labeled 2ℓ from E'_T which are incident to vertices in Y, and maintain the similarity of all vertices in Y with a smaller anonymizing set of edges. □

We now give a proof of correctness of our reduction.

Lemma 2.2.4. *Given a Table T, the rows of T can be made k-anonymous by suppressing at most t entries if and only if X can be made k-label sequence anonymous by adding at most t edges.*

Proof. (If:) By Lemma 2.2.1 and 2.2.2, it is clear that for each $y \in Y$ and each j, $1 \leq j \leq l$, y will either

1. Have exactly one incident edge labeled $2(j-1)$ but no incident edge labeled $2(j-1)+1$.
2. Have exactly one incident edge labeled $2(j-1)+1$ but no incident edge labeled $2(j-1)$.
3. Have exactly one incident edge labeled $2(j-1)$ and exactly one incident edge labeled $2(j-1)+1$

This gives us an anonymization of the rows in T corresponding to Y. Namely, in cases (1) or (2) we leave the corresponding table entry unchanged. In case (3) we put a $*$ in the corresponding entry. Note that the number of times that (3) occurs is exactly the number of edges in E'_T incident to Y. We repeat this for each equivalence class in $X/\equiv$, and so conclude that if G can be k-anonymized by adding edges E'_T, then T can be k-anonymized by the suppression (i.e. replacement by a $*$) of $|E'_T|$ entries.

(Only if:) Going from an anonymized table to an anonymized graph is quite simple. If the anonymization procedure puts a * in place of value i in entry (m, j) of table T, the graph anonymization procedure we will add an edge from r_m to $c_j^{(1-i)}$ with weight $2(j-1)+(1-i)$. If T is properly anonymized, each row m will have $k-1$ rows that are identical to it. But then in G'_T, vertex r_m will be similar to the vertices corresponding to those $k-1$ rows. Intuitively, we may view the suppression of an entry as putting both a 0 and 1 value in that entry, and adding the corresponding edges to the graph. □

Clearly the decision version of this problem is in NP, since we can verify in polynomial time that the solution is k-label sequence subset anonymous and the edges added are less than the given value t.

Therefore, k-**LS-SAP** is NP-complete. We finish this section with an observation that will be useful for us later. It uses the idea that for any edge between a row vertex and an attribute vertex, we can always change the attribute vertex endpoint as it does not affect the label sequence of the row vertex.

Lemma 2.2.5. *We can assume, without loss of generality, that in G'_T all edges with label $2(j-1)+i$ are only of the form (r_k, c_j^i) for some k, $1 \leq k \leq n$.*

Proof. Using previous lemmas, what we say here is that given an anonymized graph G'_T, we can move edges to their proper location in G'_T and not effect the anonymity. What should be noticed is the anonymization is based of the labels of the edges, not their endpoint vertices, so moving the edges so that they follow the structure of the graph G_T, makes no change to the anonymous label sequences or anonymous groups. □

2.3 Implications towards other notions of Anonymity

In this section, we observe that the proof in the previous section showing that k-**LS-SAP** is NP-hard in fact gives us a much stronger corollary on a special class of graphs called *table graphs*. Our new notion of *table graphs* presented here can be viewed as a unifying framework to prove hardness results for graph anonymization. Many of these papers showed schemes that worked well in practice. However, the complexity of the various notions of graph anonymization are poorly understood (with the exception of [42] who showed the hardness of neighbourhood anonymity for vertex-labeled graphs). Our results here initiate a systematic study of such hardness questions in the field of social network anonymization.

Definition 2.3.1 (Table Graphs)**.** *An edge-labeled graph $G = ((U, V, W), E, \Sigma)$ is an $n \times l$ table graph if:*

- $|U| = n$ *and* $|V|, |W| = l$ *for some* n *and* l
- $E \subseteq (U \times V \times \Sigma) \cup (U \times W \times \Sigma)$
- *All edges incident to* $v_i \in V$*,* $1 \leq i \leq l$*, are labeled* $2(i-1)$
- *All edges incident to* $w_i \in W$*,* $1 \leq i \leq l$*, are labeled* $2(i-1)+1$.

k-Table Graph Anonymization: Given an $n \times l$ table graph $G = ((U, V, W), E, \Sigma)$, $X \subseteq U$, construct an $n \times l$ table graph $G' = ((U, V, W), E \cup E')$ such that X is k-label sequence anonymous in G' and $|E'|$ is minimized.

k-Table Graph Anonymization is NP-complete.

Proof. This result is easily verified from the proof in the previous section. An instance of k-LS-SAP reduces to an instance of k-Table Graph Anonymization quite easily. □

We will now use this result to show hardness results for other measures of graph anonymization. Although omitted from the next sections, all three problems (Neighbourhood, k-symmetry and i-hop) anonymization are all in NP. This can be easily verified, therefore the problems are NP-complete.

2.3.1 Neighbourhood Anonymization

In neighbourhood anonymization, we focus on the induced graph of the immediate neighbours of a vertex v instead of the degree. [42] studied neighbourhood attacks in which the adversary uses prior knowledge of the connectivity of the neighbours of a target node in a social network for identity disclosure. While [42] studied this notion for vertex labeled graphs and proved NP-hardness, we use edge-labeled graphs. Neighbourhood anonymization is defined as follows:

Definition 2.3.2 (Neighbourhood Anonymity)**.** *In an edge-labeled graph G, the neighborhood of $u \in V(G)$ is the induced subgraph on u and the vertices adjacent to u. Such a graph G is said to be* k-neighbourhood anonymous *if for a given vertex $v \in V(G)$, there are $k-1$ other vertices in G with isomorphic neighborhood.*

Let $k \geq 3$ be any fixed integer, as in the case of k-**LS-SAP**, we are given an edge labeled graph G, a subset of vertices X. We need to add the smallest number of edges to G to make X k-neighbourhood anonymous. We can now reduce k-**Table Graph Anonymization** to k-neighbourhood anonymity.

Lemma 2.3.3. *Given a Table graph G_T, X can be made k-label sequence anonymous by adding at most j edges if and only if it can be made k-neighbourhood anonymous by adding at most j edges.*

Proof. (If:) This is clear since k-neighbourhood anonymity implies k-label sequence anonymity.
(Only if:) By Lemma 2.2.5, the optimal anonymization procedure for k-label sequence anonymizing X will result in the same set of neighbours for every y in Y where Y is an equivalence class of X. Since the set of neighbours is the same, the induced subgraphs on the neighbours are also the same. Hence, for table graphs, k-label sequence anonymity implies k-neighbourhood anonymity. Therefore, k-neighbourhood anonymity is NP-hard. □

2.3.2 1-Hop Anonymization

This notion was introduced by Thompson and Yao [36]. i-hop anonymity focuses on the degrees of the immediate neighbours of a node. The assumption is that information about a node may be inferred from information about its immediate neighbors. Similar to [42], if information about a node and its immediate neighbors is known to an adversary, he can then use this information to attack the identity of a given node. We will show here that

1-hop labeled subset anonymity is NP-hard. We define i-hop anonymity for edge-labeled graphs as follows.

Definition 2.3.4 (i-hop Anonymity)**.** *The* i-hop fingerprint *of a vertex* $v \in V$*, denoted* $f_i(v)$*, is the sequence*

$$\left(\{S_u | u \in N(v,0)\}, \cdots, \{S_u | u \in N(v,i)\}\right)$$

where $N(v,j)$ *denotes the set of vertices whose minimum distance to* v *is* j *(the* jth-hop neighbours *of* v*.) We say a labeled graph* $G(V,E)$ *is* i-hop k-anonymous *if for each node* $v \in V$*, there exist* $k-1$ *other nodes with the same* i*-hop fingerprint as* v*.*

Let $k \geq 3$ be any fixed integer, as in the case of k-**LS-SAP**, we are given an edge labeled graph G, a subset of vertices X. We need to add the smallest number of edges to G to make X 1-hop k-anonymous. Again, we can reduce k-**Table Graph Anonymization** to 1-hop k-anonymity.

Lemma 2.3.5. *Given a Table graph* G_T*,* X *can be made* k*-label sequence anonymous by adding at most* j *edges if and only if it can be made* 1*-hop* k*-anonymous by adding at most* j *edges.*

Proof. (If:) This direction of the proof is straightforward, as 1-hop anonymity implies label sequence anonymity. (Only if:) By Lemma 2.2.5, k-label sequence anonymizing X optimally will result in the same set of adjacent vertices for every $y \in Y$ where Y is an equivalence class of X. Since the set of adjacent vertices is the same, the 1-hop fingerprint of every vertex $y \in Y$ is also the same. Therefore, 1-hop k-anonymity is NP-hard. □

2.3.3 Symmetry Anonymization

In [39], k-symmetry was introduced. Under this notion of anonymity, for each vertex v in the network, there exist at least $k-1$ other vertices, each of which can act as an image of v under some automorphism of the modified network. To define the concept formally, we need the following definition.

Definition 2.3.6 (Automorphism Equivalence)**.** *Two vertices u, v of G are said to be* automorphically equivalent *if there is an automorphism of $G = (V, E)$ that maps u to v. Automorphism equivalence is an equivalence relation on V and the partition of V induced by this equivalence relation is called the automorphism partition of G, denoted by* Orb(G).

k-symmetry anonymity requires that all orbits have size at most k. Formally we have

Definition 2.3.7 (k-Symmetry Anonymity)**.** *A graph G is* k-symmetry anonymous *if* $\forall \Delta \in Orb(G)$, $|\Delta| \leq k$.

Let $k \geq 3$ be any fixed integer, as in the case of k-**LS-SAP**, we are given an edge labeled graph G, a subset of vertices X and an integer k. We need to add the smallest number of edges to G to make X k-symmetry anonymous. Again, we can reduce k-**Table Graph Anonymization** to k-symmetry anonymity.

Lemma 2.3.8. *Given a Table graph G_T, X can be made k-label sequence anonymous by adding at most j edges if and only if it can be made k-symmetry anonymous by adding at most j edges.*

Proof. (If:) It is easy to see that k-symmetry anonymity implies k-label sequence anonymity. (Only if:) For k-symmetry anonymity, it is required that if $Y = \{y_1, y_2, \ldots, y_m\}$ is an

equivalence class of X, then there is an automorphism of the anonymized graph that takes y_i to y_j for $1 \leq i, j \leq m$. This is the case for a table graph that is made k-label sequence anonymous in the optimal manner. Since, by Lemma 2.2.5, two vertices in Y are adjacent to the same set of neighbours, the mapping that maps y_i to y_j and vice versa and is the identity mapping on the rest of the vertices is an automorphism. Therefore, k-symmetry anonymity is NP-hard. □

2.4 Label Sequence Anonymization for Bipartite Graphs

In this section, we consider the label-sequence-based subset anonymization problem for edge-labeled bipartite graphs. We start by restating the problem for the bipartite setting.

k-Label-sequence-Based Bipartite Subset
Anonymization Problem (k-LS-BSAP):

Given a labeled bipartite graph $G = ((U, V), E, \Sigma)$, and $X \subseteq U$, find a bipartite graph $G' = ((U, V), E \cup E', \Sigma \cup \Sigma')$ such that X is k-anonymous in G' and $|E'|$ is minimized.

2.4.1 Algorithm for $k = 2$

We first show that the problem of finding an optimal 2-anonymization can be reduced to a problem of finding a min-cost perfect matching in a hypergraph containing edges of size 2 and 3. We then use a result shown by Anshelevich and Karagiozava [2] to conclude that there is a polynomial time algorithm for finding an optimal 2-anonymization. For simplicity, we will assume that $X = U$. The algorithm we present below can be easily modified to work for any $X \subseteq U$.

As shown earlier, we can assume that, in any 2-anonymization of U, every anonymous group is of size two or three. We construct a hypergraph $H = (U, E)$ where E contains every possible subset of U of size 2 and 3. We associate a cost $c(e)$ with each edge e in H. For any edge e, $c(e)$ will be the number of new edges that need to be added to make the vertices in e have the same label sequence so that they form an anonymous group. Let S_u denote the label sequence of a vertex u in U. Then,

- If $\max(|S_u - S_v| + |S_v|, |S_v - S_u| + |S_u|) \leq |V|$,
  $c(\{u, v\}) = |S_u - S_v| + |S_v - S_u|$.
  else
  $c(\{u, v\}) = \infty$.

- If $\max(|(S_v \cup S_w) - S_u| + |S_u|, |(S_u \cup S_w) - S_v| + |S_v|, |(S_u \cup S_v) - S_w| + |S_w|) \leq |V|$,
  $c(\{u, v, w\}) = |(S_v \cup S_w) - S_u| + |(S_u \cup S_w) - S_v| + |(S_u \cup S_v) - S_w|$
  else
  $c\{u, v, w\} = \infty$.

The cost of creating an anonymous group of size two is the symmetric difference of the two label sequences provided the cost at each vertex is realizable. In other words, at a vertex u, the $|S_v - S_u| + |S_u|$ must be less than $|V|$. We remark that we treat the label sequences as multisets for the different set operations above. For example, if a label l occurs twice in a set S_u and once in another set S_v, then one of the two occurrences of the label l will be in $S_u - S_v$. The cost of anonymizing three vertices u, v and w into one group is to add all the edges present in the union of two label sequences but not in the third.

Clearly, finding the optimal 2-anonymization now reduces to finding a minimum-cost

perfect matching in the edge weighted hypergraph H constructed above. A perfect matching in H is a set of edges such that every vertex in U is present in exactly one of the edges.

We recall the result of [2].

Theorem 1. *Given any hypergraph H with edges of size two and three with an associated cost function d on the edges of H, there is a polynomial time algorithm for finding a minimum-weight perfect matching in H provided d satisfies the following simplex condition: For any edge $e = \{u, v, w\}$ in H the edges $\{u, v\}, \{v, w\}, \{u, w\}$ are also in H and*

$$d\{u, v\} + d\{v, w\} + d\{u, w\} \leq 2d\{u, v, w\}.$$

In order to use the Theorem, we need the following Lemma.

Lemma 2.4.1. *The cost function c satisfies the simplex condition. That is,*

$$c\{u, v\} + c\{v, w\} + c\{u, w\} \leq 2c\{u, v, w\}.$$

Proof. To show this, we will consider edge labels in three types of regions in the Venn diagram (See Figure 2.4.1 for the three (multi)sets S_u, S_v and S_w and show that their contribution to the LHS of the equation is at most their contribution to the RHS.

Regions of type 1 contain edge labels present in one of the three sets and not the other two. These labels contribute 2 to the LHS and 4 to the RHS. For example, suppose a label l is in S_u but not in S_v and S_w. Then, it contributes a cost of 1 to $c\{u, v\}$ and $c\{u, w\}$. On the other hand, it contributes 2 to $c\{u, v, w\}$ and hence 4 to $2c\{u, v, w\}$.

Regions of type 2 contain edges labels that are in two sets and not in the third. These labels contribute 2 to the LHS and 2 to the RHS. For example, suppose a label l is in S_u

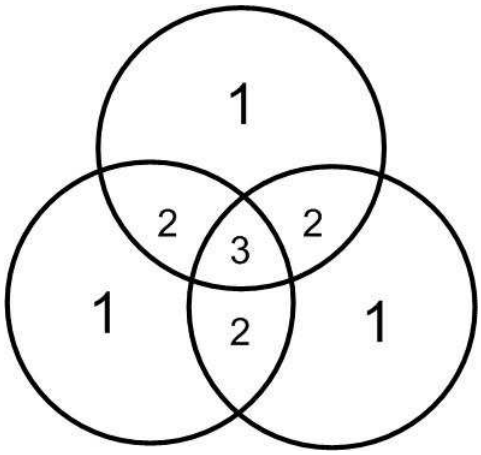


Figure 2.6: Edge Label Regions

and S_v but not in S_w. Then, it contributes a cost of 1 to $c\{u, w\}$ and $c\{v, w\}$. On the other hand, it contributes 1 to $c\{u, v, w\}$ and hence 2 to $2c\{u, v, w\}$.

Regions of type 3 contain labels present in all the three sets. These labels do not contribute to either side. In this proof, we have assumed that all the cost values are finite. The cases where at least one of them is not is straight forward to handle as the inequality is easily satisfied in these cases. □

Therefore, we have shown that label sequence-based subset anonymity problem for edge-labeled bipartite graphs is in P for $k = 2$.

Unlabeled Case: We note here that the technique used above also gives a simple algorithm for degree-based subset anonymization for the unlabeled bipartite graphs for $k = 2$. We first define it here:

k-Degree-Based Bipartite Subset Anonymization Problem (k-D-BSAP):

Given a bipartite graph $G = ((U, V), E)$, and $X \subseteq U$, find a graph $G' = ((U, V), E \cup E')$ such that, $E' \subseteq U \times V$, X is k-anonymous in G' and the number of new edges added, $|E'|$, is minimized.

In this scenario, the new cost function is the difference in the degrees. Let $d(u)$ denote the degree of a vertex u.

- $c'(\{u, v\}) = |d(u) - d(v)|$.
- $c'(\{u, v, w\}) = [max(d(u), d(v), d(w)) - d(u)]$
  $+ [max(d(u), d(v), d(w)) - d(v)]$
  $+ [max(d(u), d(v), d(w)) - d(w)]$.

Lemma 2.4.2. *c' satisfies the simplex condition.*

Proof. Without loss of generality, let $d(u) > d(v) > d(w)$. Then, the LHS evaluates to $d(u) - d(v) + d(v) - d(w) + d(u) - d(w) = 2(d(u) - d(w))$. Furthermore,

$$\begin{aligned} RHS &= 2[(d(u) - d(u) + d(u) - d(v) + d(u) - d(w)] \\ &= 2[2d(u) - d(v) - d(w)] \\ &> 2[2d(u) - d(u) - d(w)] \\ &= 2[d(u) - d(w)] \end{aligned}$$

□

Now, using the Theorem of Anshelevich and Karagiozava, we get that 2-anonymity for unlabeled bipartite graphs is in P. Later in this chapter, we will show that the problem k-**D-BSAP** is in P for all values of k.

2.4.2 Hardness Result for $k \geq 3$

We now turn from algorithms to hardness results. The proof given here is similar to the proof given earlier for the general case with some small modifications. However we give it here for completeness.

We begin with k-anonymization for edge-labeled bipartite graphs when $k \geq 3$.

Theorem 2. k**-LS-BSAP** *is* NP*-complete for* $k \geq 3$.

Proof. Let $k \geq 3$ be any fixed integer. We can build a reduction from the NP-hard table

anonymization problem introduced earlier in this chapter to the decision version of k-**LS-BSAP**, and use similar techniques as seen earlier.

k-**ENTRY-ANONYMITY**

Input: a Table T with n rows and l columns (also called attributes) with entries over $\{0, 1\}$ and integer t.

Question: Can the rows of T be k-anonymized by suppressing at most t entries of T? Here, an entry (0 or 1) is said to be suppressed if it is replaced by *.

Reduction: Our reduction is described as follows:

Given a Table T, let $T_{(m,j)} \in \{0, 1\}$ denote the value of attribute j in row m. Then, the edge-labeled bipartite graph G_T corresponding to T is constructed as follows:

- $U_T = \{r_1, r_2, \ldots, r_n\}$.
- $V_T = \{c_j^i | 1 \leq j \leq l, i \in \{0, 1\}\}$.
- Let $E_T = \{(r_m, c_j^i, 2(j-1)+i) | T_{(m,j)} = i\}$ where $1 \leq m \leq n$, $1 \leq j \leq l$ and $i \in \{0, 1\}$.
- $\Sigma_T = \{0, 1, \ldots, 2l-1\}$.
- Finally, remove all isolated vertices from G_T.

In other words, we encode a binary table as a bipartite graph in which a row vertex $r_m \in U_T$ is connected to a column vertex $c_j^0 \in V_T$ (c_j^1) $\in V_T$ with label $2(j-1)$ or $(2(j-1)+1)$ if the (m, j)th entry of the table is 0 (1).

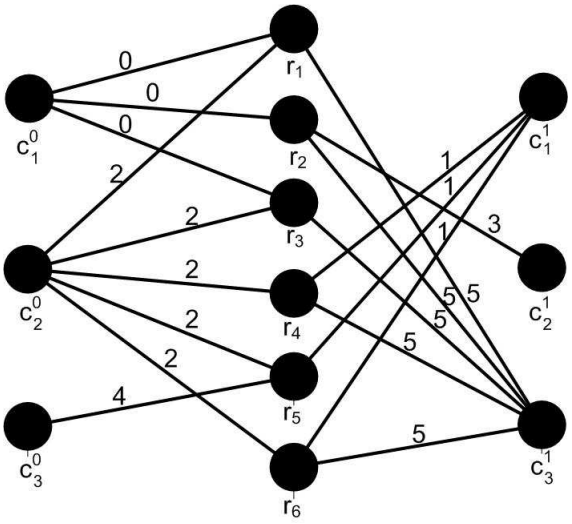


Figure 2.7: Transformation of Table T into Graph G_T

Table 2.6: Table T

Entry	A_1	A_2	A_3
1	0	0	1
2	0	1	1
3	0	0	1
4	1	0	1
5	1	0	0
6	1	0	1

Let $X = \{r_1, \ldots, r_n\}$ denote the set of row vertices of G_T. We will show that T can be k-anonymized by suppressing at most t entries if and only if we can k-anonymize X by adding at most t new labeled edges.

Let G'_T be any graph obtained from G_T such that X is k-anonymous in G'_T and it has the minimum number of new edges added. Suppose that E'_T is an anonymizing set of edges for X. Letting $\equiv$ denote vertex similarity in the anonymized graph, let $Y = \{y_1, \ldots, y_m\}$ be an equivalence class of $X/\equiv$, where $m \geq k$. We begin by establishing properties that any anonymizing set E'_T of minimum size must satisfy.

Our first lemma shows that the anonymization procedure only introduces edges with labels already in Σ_T.

Lemma 2.4.3. *If there is an edge in E'_T labeled ℓ that is incident to Y then there is an edge in E_T labeled ℓ that is incident to Y.*

Proof. Suppose that ℓ is the label of an edge in $E_T \cup E'_T$ that is incident to a vertex $y \in Y$. Then in fact there must be an edge in E_T with label ℓ that is incident to some vertex $y' \in Y$. If this were not the case, then we may remove all edges labeled ℓ from E'_T which are incident to vertices in Y, and maintain the similarity of all vertices in Y with a smaller

anonymizing set of edges. □

The following shows that at most one edge with label ℓ is incident to a row vertex of U_T.

Lemma 2.4.4. *For every $i \in \{0,1\}$, and every $j \in \{1,2,\ldots,l\}$, the label $2(j-1)+i$ appears at most once in the label sequence of a vertex $y \in Y$.*

Proof. We first show that if there is an edge in E_T labeled ℓ that is incident to $y \in Y$ then there is no edge in E'_T labeled ℓ that is incident to y. The proof proceeds by contradiction. Suppose that there is such an edge labeled ℓ in $E_T \cup E'_T$ that appears more than once. So the label ℓ occurs more than once in the label sequence of y, and hence of every node in Y. By construction only one of these occurrences is due to an edge in E_T. We may remove the edges in E'_T corresponding to the other occurrences and maintain the similarity of all vertices in Y with a smaller anonymizing set of edges. On the other hand, suppose that there is an edge labeled ℓ in E'_T that appears more than once and is incident to y but there is no edge labeled ℓ in E_T. Then, we note again that we may remove all edges labeled ℓ from E'_T which are incident to vertices in Y, and maintain the similarity of all vertices in Y with a smaller anonymizing set of edges. □

We now give a proof of correctness of our reduction.

Lemma 2.4.5. *T can be k-anonymized by suppressing t entries if and only if the subset X of vertices in G_T can be k-anonymized by adding t new labeled edges.*

Proof. (If:) By Lemma 3.2.1 and 3.2.2, it is clear that for each $y \in Y$ and each j, $1 \leq j \leq l$, y will either have

1. Exactly one incident edge labeled $2(j-1)$ but no incident edge labeled $2(j-1)+1$.
2. Exactly one incident edge labeled $2(j-1)+1$ but no incident edge labeled $2(j-1)$.
3. Exactly one incident edge labeled $2(j-1)$ and exactly one incident edge labeled $2(j-1)+1$

This gives us an anonymization of the rows in T corresponding to Y. Namely, in cases (1) or (2) we leave the corresponding table entry unchanged. In case (3) we put a $*$ in the corresponding entry. Note that the number of times that (3) occurs is exactly the number of edges in E'_T incident to Y. We repeat this for each equivalence class in $X/\equiv$, and so conclude that if G can be k-anonymized by adding edges E'_T, then T can be k-anonymized by the suppression (i.e. replacement by a $*$) of $|E'_T|$ entries.

(Only if:) Going from an anonymized table to an anonymized graph is quite simple. If the anonymization procedure puts a * in place of value i in entry (m, j) of table T, the graph anonymization procedure we will add an edge from x_m to $c_j^{(1-i)}$ with weight $2(j-1)+(1-i)$. If T is properly anonymized, each row m will have $k-1$ rows that are identical to it. But then in G'_T, vertex x_m will be similar to the vertices corresponding to those $k-1$ rows. Intuitively, we may view the suppression of an entry as putting both a 0 and 1 value in that entry, and adding the corresponding edges to the graph. □

This completes the proof of the Theorem.

We do remark that the decision version of k-**LS-BSAP** is in NP. To show this, we note that the collection of the new edges to be added along with the resulting partition of X into k-anonymized subsets is a polynomial-size certificate of membership. Therefore, k-**LS-BSAP** is NP-complete. □

2.5 k-Degree-Based Anonymization for Bipartite Graphs

In this section, we give an algorithm for the degree-based subset anonymization problem for unlabeled bipartite graphs, k-**D-BSAP**. This problem was originally defined in §2.4.1

We note that in this setting X is only allowed to be a subset of U. This is based on the understanding that for social networks represented as bipartite graphs, each side of the bipartition represents one type of entity and we are interested in anonymizing only one type. For example, in the Netflix database, we are only interesting in anonymizing subsets of customers.

We show that there is an efficient algorithm for this problem using the techniques of Liu and Terzi. For simplicity, we will assume that $X = U$. Our algorithm can be easily modified to work for any $X \subseteq U$. Let $|U| = n$ and let $d = (d_1, d_2, \ldots, d_n)$ be the degree sequence of the vertices of U. We start with the preprocessing step of sorting the degree sequence d in $O(n \log n)$ time.

The algorithm proceeds in two main steps:

- **Degree Anonymization:** Given d, the algorithm outputs the sequence $d' = (d'_1, d'_2, \ldots, d'_n)$ satisfying $d'_i \geq d_i$ such that $\Sigma_i(d'_i - d_i)$ is minimized and d' is k-anonymous.

- **Graph Construction:** The algorithm constructs the graph G' in which the degree of the vertex u_i is d'_i. In other words, U is k-anonymous in G'.

2.5.1 Degree Anonymization

We start with the following proposition which can be easily verified.

Proposition 2.5.1. *Without loss of generality, every anonymous group in d' is of size less than $2k$.*

Proof. If there is a anonymous group of size greater than $2k$ we can split this group further into two groups of size at least k. An anonymization of d with the new set of groups has cost at most the cost of the original anonymization. □

Given a sorted degree sequence d, let $DA(d[1,i])$ denote the cost of k-anonymizing the subsequence $d[1,i]$. Also, let $C(d[i,j])$ be the cost of including the vertices $\{u_i, u_{i+1}, \ldots, u_j\}$ in one anonymous group. Clearly,

$$C(d[i,j]) = \Sigma_{l=i}^{j}(d(i) - d(j)).$$

Using the proposition above, we get the following dynamic programming equations to compute d'. In particular, for $i < 2k$

$$DA(d[1,i]) = C(d[1,i])$$

while for $i \geq 2k$,

$$DA(d[1,i]) = \min_{j \leq t \leq i-k}(DA(d[1,t]) + C(d[t+1,i])),$$

where $j = \max\{k, i-2k+1\}$.

The first equation uses the fact that if $i < 2k$, it is not possible to have more than one anonymous group. Therefore, the optimal cost of creating a single group involves making all the degrees equal to $d(1)$. The second equation says that if $i > 2k$, the degree

anonymization cost consists of the degree anonymization cost of the subsequence $d[1,t]$ and the optimal cost of putting the vertices $t+1,\ldots,i$ into a single group. Moreover, this group has to be size less than $2k$. The running time of this dynamic programming step is $O(nk)$.

2.5.2 Graph Construction

We observe the following property of d'.

Proposition 2.5.2. *Let d_{max} be the maximum degree of a vertex in U in the graph G. Then $d'(i) < d_{max}$ for all i.*

Proof. Suppose not. Then, there is an anonymous group of d' in which all values are equal and greater than d_{max}. Replacing every entry by d_{max} will help produce a k-anonymous sequence with a lower degree anonymization cost. This is a contradiction. □

Therefore, once we obtain the k-anonymous degree sequence d' from the previous step, the algorithm adds for each vertex $u_i \in U$, $d'(u_i) - d(u_i)$ new edges from u_i to (arbitrary) vertices in V.

The running time of this step is $O(nd_{max})$ where d_{max} is the maximum degree of a vertex in U.

In summary, we get the following theorem:

Theorem 3. *k-**D-BSAP** $\in P$. In particular, there is an algorithm with running time $O(n(k+d_{max})+n\log n)$ that solves this problem.*

2.6 Future Work

The next natural step of our research is to see if any of the problems described have good approximation algorithm for the cases where the problem is NP-hard.

It is important to note that research is ongoing to find applicable relaxed versions of these theoretical based problems that adhere to certain applications. Taking such theoretical problems as the base may lead to efficient solutions to anonymity constraints on such large scale networks. There may be other approaches to anonymizing graphs that may prove just as effective and with better efficiency. It has been shown that certain types of anonymization are not effective (naive anonymization for example), but there may still be effective approaches that have not been looked at. The whole field of graph anonymization is fairly new, there are many avenues within it still to be developed. Newer notions of l-diversity and α-proximity (See Chapter 4) may have some reasonable solution spaces that could be shared with relaxed versions of the anonymity constraints.

The delicate balance between user privacy and requirements for analysis is something that needs to be considered when data of a private nature is released to third parties. This is what motivates our results in this paper. With a strong grasp now on the underlying complexity of many graph anonymity models, we can now move forward and examine efficient solutions to the anonymity problems at hand.

Chapter 3

Anonymization Through Node Additions

In this chapter we show our results for anonymization through node addition for unlabeled and vertex-labeled graphs. In §3.1 we introduce the definitions and notions of anonymity pertaining to this chapter. We then present a hardness result for vertex-labeled graphs in §3.2. Moving onto our algorithmic results, we show an efficient algorithm in §3.3 for anonymizing unlabeled graphs. We conclude the chapter in §3.4 with experimental results from testing the performance of the algorithm in §3.3 through a collection of common social network measures.

3.1 Graphs and k-Anonymity

Vertex-labeled graphs correspond to social networks in which the nodes have identifiers, or attributes associated with the node.

3.1.1 Labeled Graphs

We start with the definition of a vertex-labeled graph and a label sequence.

Definition 3.1.1. *A vertex-labeled graph is a simple, undirected graph $G = (V, E, L, \mathcal{L})$ where V is a set of vertices, $E \subseteq V \times V$ is the set of edges, L is a set of labels and $\mathcal{L}$ is a labeling function, $\mathcal{L} : V \to L$, that assigns a label to every vertex in V.*

Definition 3.1.2. *For $v \in V$, we say that $S_v = (l_1, l_2, \ldots, l_m)$ is a label sequence for v if it corresponds to some ordering of the labels of v and the vertices that are adjacent to v. We will consider label sequences of vertices to be equivalent up to reordering.*

Then, k-anonymity for labeled graphs relates to the uniqueness of label sequences:

Definition 3.1.3. *Given a vertex-labeled graph $G = (V, E)$, a subset $X \subseteq V$ of vertices is k-anonymous in G if for every vertex $v \in X$, there are at least $k - 1$ other vertices in X whose label sequence is the same as the label sequence of v.*

We denote by $u \equiv v$ that vertices u and v have the same label sequence. Clearly, $\equiv$ is an equivalence relation and hence induces a partition $X/\equiv$ of X into equivalence classes. If X is k-anonymous in G, every equivalence class in X is of size at least k.

3.2 A Hardness Result for Labeled Graphs

Let $k \geq 3$ be any fixed integer, we will show that anonymizing a subgraph of a given vertex-labeled graph through vertex addition is NP-hard. The anonymization problem we study in this paper is defined as follows:

LABELED SUBGRAPH ANONYMIZATION

Input: A vertex-labeled graph $G = (V, E, L, \mathcal{L})$, a set $X \subseteq V$ of vertices, and integer t.

Question: Is there a vertex-labeled graph $G' = (V \cup V', E \cup E', L', \mathcal{L}')$ such that $|V'| \leq t$, $E' \subseteq (V \times V') \cup (V' \times V')$, $L'_{|V} = L$, $\mathcal{L}'_{|\mathcal{V}} = \mathcal{L}$ and X is k-anonymous in G'?

That is, can we k-anonymize X by adding at most t new labeled vertices? New edges are allowed only between an old vertex and a new vertex or between new vertices.

Theorem 4. *Labeled Subgraph Anonymization is NP-complete.*

Proof. To show that this problem is NP-hard, we build a reduction from the following anonymization problem on tables that was shown to be NP-hard by Meyerson and Williams [32].

k-ATTRIBUTE-ANONYMITY

Input: a table T with n rows and l columns (also called attributes) with entries over $\{0, 1\}$ and integer t.

Question: Can the rows of T be k-anonymized by suppressing at most t attributes of T? Here, an attribute is said to be suppressed if all its entries (0 and 1) are replaced by *.

Reduction: Our reduction is described as follows:

Given a Table T, let $T_{(m,j)} \in \{0, 1\}$ denote the value of attribute j in row m. Then, the vertex-labeled graph G_T corresponding to T is constructed as follows:

- $V_T = \{r_1, r_2, \ldots, r_n\} \cup \{c_j^i | 1 \leq j \leq l, i \in \{0, 1\}\}$.
- Let $E_T = \{(r_m, c_j^i) | T_{(m,j)} = i\}$ where $1 \leq m \leq n$, $1 \leq j \leq l$ and $i \in \{0, 1\}$.
- $L = \{R\} \cup \{A_j^i | 1 \leq j \leq l, i \in \{0, 1\}\}$.

Entry	A_1	A_2	A_3
1	0	0	1
2	0	1	1
3	0	0	1
4	1	0	1
5	1	0	0
6	1	0	1

Table 3.1: An Example Table T

- $\mathcal{L}(r_i) = R$ and $\mathcal{L}(c_j^i) = A_j^i$.
- Finally, remove all isolated vertices from G_T.

In other words, we encode a binary table as a graph in which a row vertex r_m with label R is connected to a column vertex c_j^0 (c_j^1) with label A_j^0 (A_j^1) if the (m, j)th entry is 0 (1).

Let $X = \{r_1, \ldots, r_n\}$ denote the set of row vertices of G_T. We will show that T can be k-anonymized by suppressing at most t attributes if and only if we can k-anonymize X by adding at most $2t$ new labeled vertices.

Let G'_T be any graph obtained from G_T such that X is k-anonymous in G'_T. We will now show that we may assume without loss of generality, G'_T satisfies a set of properties. If it does not, we convert it into one that satisfies the properties without increasing the number of new labeled vertices added in G'_T.

Our first lemma shows that the anonymization procedure does not introduce vertices with new labels not in L_T. Furthermore, none of the new vertices added to G_T will be labeled by R.

Lemma 3.2.1. *$L'_T = L_T$ and $\mathcal{L}'_T(v) \neq R$ for any $v \in V'_T$.*

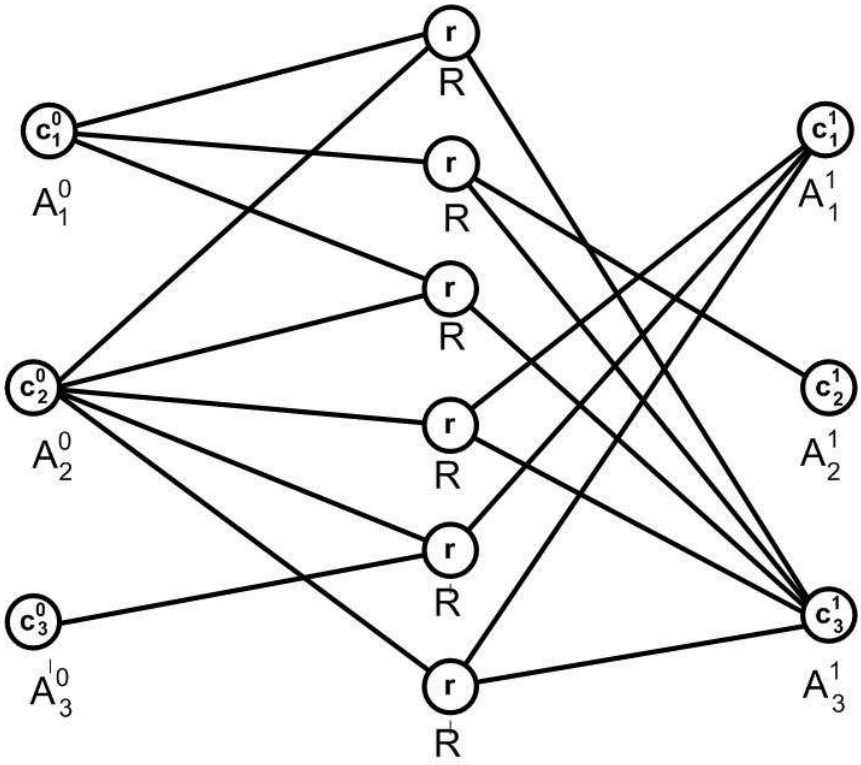


Figure 3.1: Transformation of Table T into Graph G_T

Proof. Suppose that there are one or more vertices with a new label l, $l \notin L_T$ in G'_T. Let Y be any equivalence class of $X/\equiv$. Every vertex $y \in Y$ has the same number of labels l in its label sequence in G'_T. Therefore, the label sequences of all vertices in Y will remain the same after all the new vertices with label l are removed from the graph G'_T. Since this is true of any equivalence class Y, we can assume that $L'_T = L_T$. To show that $\mathcal{L}'_T(v) \neq R$ for any $v \in V'_T$, note that the label R does not appear in the label sequence of any vertex of Y in G_T. In addition, R must appear the same number of times in the label sequence of any vertex of Y in G'_T. Therefore, by the same reasoning as above, all new nodes with the label R in G'_T can be removed and Y will remain k-anonymous. □

Lemma 3.2.2. *For every i, $i \in \{0,1\}$, and every j, $j \in \{1,2,\ldots,l\}$, the label A_j^i appears at most once in the label sequence of a row vertex from X in the graph G'_T.*

Proof. Let Y be any equivalence class of X. All vertices in Y have the same label sequence in G'_T. If a label A_j^i appears in the label sequence of a vertex y of Y in G_T, we can assume that no edge between y and a new vertex labeled A_j^i is added in G'_T. If not, this label will appear more than once in the label sequence of y and hence in the label sequence of every other vertex of Y. Also, the number of occurrences of this label in every label sequence will the same. Since at most one of these will be due to an edge of G_T, for each vertex v in Y, we can remove all but one edge in $E_T \cup E'_T$ between v and a node labeled A_j^i in G'_T and still preserve anonymity of vertices in Y. □

Lemma 3.2.3. *For every i, $i \in \{0,1\}$, and every j, $j \in \{1,2,\ldots,l\}$, at most one new vertex with the label A_j^i is added in the graph G'_T.*

Proof. Suppose not. We merge all the new nodes with the label A_j^i into a single node. This only reduces the number of new vertices added with label A_j^i. In Lemma 3.2.2, we have shown that every label A_j^i appears at most once in the label sequence of every vertex in X. Therefore, the merge operation will not create multiple edges and all the label sequences of the vertices of X will remain the same as before after this modification. □

Lemma 3.2.4. *All new vertices appear in pairs. That is, if $A_j^0 \in \mathcal{L}(V'_T)$, then $A_j^1 \in \mathcal{L}(V'_T)$ and vice versa.*

Proof. Suppose a new vertex with the label A_j^0 is added. Let us consider all the vertices in X to which this vertex is adjacent in G'_T. Then one of those vertices v must be in an

equivalence class Y that had a vertex v' with an edge to vertex with label A_j^0 in G_T. If there is no such Y, it implies that all the vertices in X adjacent to this new vertex had the label A_j^1 in their label sequence before anonymization. Therefore, the new vertex with the label A_j^0 can be removed and all equivalence classes of X will remain k-anonymous. Now note that v has both A_j^0 and A_j^1 in its label sequence in G'_T. Hence, the anonymization procedure must create a new vertex labeled A_j^1 and add an edge between v' and this new vertex to preserve anonymity. □

We now give a proof of correctness of our reduction.

Lemma 3.2.5. *T can be k-anonymized by suppressing t attributes if and only if the subset X of vertices in G_T can be k-anonymized by adding $2t$ new vertices.*

Proof. (only if:) Suppose T can be k-anonymized by suppressing at most t attributes. For every column A_j that is suppressed, we add two new vertices A_j^0 and A_j^1 and add an edge from every row vertex labeled R to one of these new vertices that was not in its label sequence originally. At the end of this procedure, if a set of rows of T formed a k-anonymous group, the corresponding set of row vertices in X form a k-anonymous group. To do this, we added $2t$ new vertices.

(if:) As shown in Lemma 3.2.4, we can assume that the anonymization procedure adds new vertices in pairs. If the procedure adds a new pair (A_j^0, A_j^1), then suppress the column A_j in T. Note that if $2t$ new vertices are added in G'_T, t attributes are suppressed in T. At the end of this procedure, it is easy to see that if a set of row vertices in X form a k-anonymous group in G'_T, then the corresponding set of rows of T are k-anonymous. □

We now remark that this problem is also in NP. At the first glance, this is not clear as

there is no natural bound on the integer t in terms of the size of the graph G. However, based on Lemmas 3.2.1 and 3.2.2, it is clear that we can assume without loss of generality $t \leq |L|$. Since we can combine vertices of equivalent label and still do not create multiple edges between vertices. Therefore, a membership in this language can be shown by a list of at most t new vertices and a list of new edges. Therefore, this problem is NP-complete. This completes the proof of the theorem. □

3.3 An Efficient Algorithm for Unlabeled Graphs

In this section we show, that for the special case of unlabelled graphs, k-degree-anonymization can be solved very near-optimally in linear time. We produce a k-degree-anonymous graph $\mathcal{G}' = (\mathrm{V} \cup \mathrm{V}', \mathrm{E} \cup \mathrm{E}')$ from an original, unlabelled social network graph $\mathcal{G} = (\mathrm{V}, \mathrm{E})$.[1] In $\mathcal{G}'$ we require that all the original vertices, V, are k-degree-anonymous–and we can relax this constraint to an input subset $X \subseteq \mathrm{V}$. We also require that the new vertices are concealed as well so that they cannot be readily identified and removed from the graph in order to recover $\mathcal{G}$ (i.e., $\mathrm{V} \cup \mathrm{V}'$ is k-degree-anonymous in $\mathcal{G}'$). As mentioned in the introduction, we seek to minimise $|\mathrm{V}'|$, while maintaining the constraint that $\mathrm{E} \subseteq \mathrm{V}' \times (\mathrm{V} \cup \mathrm{V}')$. We will prove the following theorem at the end of this section:

Theorem 5. *Our algorithm produces a k-anonymous graph $\mathcal{G}'$ containing the input graph $\mathcal{G}$ as an induced subgraph, using $\mathcal{O}(nk)$ time and $\mathcal{O}(n)$ space. The number of new vertices added is optimal up to an additive factor of k.*

[1] For simplicity in this section, we regard a graph as a 2-tuple. We note that equivalently, for consistency, we could express an unlabelled graph as $\mathcal{G} = (\mathrm{V}, \mathrm{E}, \mathcal{L}, \ell)$ where $\exists \sigma \in \Sigma : \forall v \in \mathrm{V}, \ell(v) = \sigma$. However, the simpler notation simplifies the exposition.

At a high level, the algorithm proceeds in three stages. At first, we design a precise recursion to group the vertices of V by target degree (the degree they will have in V'). The recursion establishes a grouping such that the $\max$ deficiency, a parameter in determining with how many nodes V must be augmented, is minimized. We evaluate the recursion using dynamic programming with $\mathcal{O}(nk)$ execution cost.

The second stage is to determine precisely how many vertices with which we wish to augment V in order to guarantee that we can k-anonymise all of V'. This number is a function of k and $\max$ deficiency, the parameter arising out of and minimized by the recursion evaluated in Stage 1.

Finally, we introduce a particular means of adding new edges, each of which has at least one endpoint in $\mathrm{V}' \setminus \mathrm{V}$, with the objective of satisfying all the target degrees established during the recursion of Stage 1 and k-anonymizing the new vertices added during Stage 2. A critical property of our specific approach is that the edges are added in such a manner as to establish tractability of the problem of k-anonymizing the new vertices, a problem that has been shown to be NP-hard in the general case by [24].

As we describe the three stages of the algorithm in the following subsections, we illustrate their execution by 3-anonymizing the graphs in Figure 3.2.

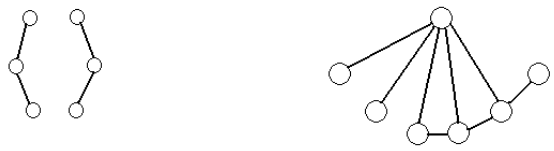


Figure 3.2: The Two Small Example Graphs Used to Illustrate our Anonymization Procedure

3.3.1 Stage 1: Determining Each Node's Target Degree

Our algorithm first proceeds by identifying which nodes should have the same degree and what degree that should be. Similar to [29], we construct a recursion on the degree sequence of $\mathcal{G}$ to compute these groups and degrees. Contrary to their work, however, we need to minimize $\max \text{deficiency}$ rather than total deficiency, because $\max \text{deficiency}$ most significantly influences the optimality of our solution. Before describing the recursion, however, we formally introduce some key concepts:

Definition 3.3.1. *The* degree *of a vertex v in a graph $\mathcal{G} = (\mathrm{V}, \mathrm{E})$ is the number of neighbours it has,* $|\{u \in \mathrm{V} : (u, v) \in \mathrm{E}\}|$.

For example, the uppermost node in the second example graph of Figure 3.2 has a degree of 5, because it is connected to 5 other vertices (neighbours). The rightmost node in the same graph has degree 1 because it has only one incident edge.

Definition 3.3.2. *A* degree sequence *of a graph $\mathcal{G}$ is the sequence $(d_1, \ldots, d_n)$ composed by sorting in descending order the degrees of every node in* V.

Again referring to the second example graph of Figure 3.2, the degree sequence is $(5, 3, 3, 2, 1, 1, 1)$, the degrees of each of the seven vertices sorted in descending order.

Definition 3.3.3. *A* k-partitioning *of a degree sequence is a partitioning of the degree sequence into disjoint partitions such that every degree appears in exactly one partition and every partition contains at least k elements.*

Two possible 3-partitionings of the degree sequence in our running example are $((5, 3, 3), (2, 1, 1, 1))$ and $((5, 3, 3, 2), (1, 1, 1))$. There are other possible 3-partitionings of

this degree sequence that have non-contiguous partitions, such as $((5,3,2,1),(3,1,1))$, but we will show through Proposition 3.3.6 that these are never better choices for our algorithm.

Definition 3.3.4. *The* $\max$ deficiency *of a* k*-partitioning of a degree sequence is the largest difference between highest and smallest degree within any partition.*

For example, taking the two 3-partitionings in our running example, the $\max$ deficiency of $((5,3,3),(2,1,1,1))$ is $\max(5-3,2-1)=2$ and of $((5,3,3,2),(1,1,1))$ is $\max(5-2,1-1)=3$. Finally,

Definition 3.3.5. *The* total deficiency *of a* k*-partitioning of a degree sequence is the sum over every* d_i *in the sequence of the difference between* d_i *and the largest degree in the same partition.*

Using the same example we find that the total deficiency of $((5,3,3),(2,1,1,1))$ is:

$$((5-5)+(5-3)+(5-3)+(2-2)+(2-1)+(2-1)+(2-1))=7$$

and of $((5,3,3,2),(1,1,1))$ is:

$$((5-5)+(5-3)+(5-3)+(5-2)+(1-1)+(1-1)+(1-1))=7.$$

So, precisely stated, we want to produce an optimal k-partitioning of the degree sequence of $\mathcal{G}$, one which has the minimum $\max$ deficiency.

To produce an optimal k-partitioning of the degree sequence of $\mathcal{G}$, we offer an incremental algorithm which operates from left (position 1) to right (position n) on the degree

sequence, maintaining the optimal (i.e., with minimum $\max \text{deficiency}$) k-partitioning of those values in the degree sequence seen so far. The ability to do this with $\mathcal{O}(nk)$ cost is a consequence of the following propositions:

Proposition 3.3.6. *The* $\max \text{deficiency}$ *of a partition containing a highest degree of* d_i *and a smallest degree of* d_j *will be less or equal to the* $\max \text{deficiency}$ *of any partition containing* d_i *and any* d_{j+c} *or containing* d_j *and any* d_{i-c}*,* $\forall c \in \mathbb{N}$.

Proposition 3.3.7. *For any partition* $(d_i, \ldots, d_l)$*, its* $\max \text{deficiency}$ *is greater or equal to that of the partitions* $(d_i, \ldots, d_j)$, $(d_{j+1}, \ldots, d_l)$*, for* $i < j < l$*. That is to say, it never produces a higher* $\max \text{deficiency}$ *when one splits a partition.*

Both propositions follow because the degree sequences are sorted and $\max \text{deficiency}$ (i.e., difference) is transitive. Importantly, they allow us to construct a recursion, and an incremental, dynamic programming algorithm to evaluate that recursion, because they imply that there are only k ways to produce an optimal k-partitioning of the first x elements if the best possible k-partitionings are known for the first i elements, $\forall i < x$. From Proposition 3.3.6, it is clear that the x'th element should be added to the right of the first $(x-1)$ elements. From Proposition 3.3.7, it is clear that if there is an optimal split point for the rightmost partition that is farther than $2k-1$ positions left of x, then it can be split into sub-partitions such that the rightmost partition begins at some other position at least as far right as $x-2k+1$. The rightmost position must also have at least k elements; so, the rightmost partition of the optimal k-partitioning on the first x elements must begin between $x-2k+1$ and $x-k$, inclusive.

Our algorithm evaluates the recursion "bottom-up", constructing an optimal k-partitioning

of the degree sequence by incrementally adding the next x'th rightmost degree and determining the best possible k-partitioning. If there are fewer than $2k$ degrees in the sequence, there is not any choice but to group them all together, because at least $2k$ elements are required to make two partitions of size $\geq k$. When, on the other hand, there are at least $2k$ degrees, then we evaluate the cost of splitting off a rightmost partition at any of the k positions between $x-2k+1$ and $x-k$, inclusive, and choose the rightmost of the cheapest among them. We invoke the recursion by recognizing that the $\max \text{ deficiency}$ incurred by creating a rightmost partition that starts at some position i is exactly the larger of the best possible k-partitioning up to $i-1$ and the degree differences between the i'th and x'th degrees in the sequence.

The following recursion evaluates the cost of splitting and thus constructs an optimal k-partitioning. In the statement of the recursion below, the Δ function computes the $\max \text{ deficiency}$ of a particular partition; the Start function keeps track of where x's partition starts, should x be the rightmost degree in it; and the Cost function computes the overall cost ($\max \text{ deficiency}$) of the best possible partitioning up to the $x'th$ element. The Start function allows us to retrace the best possible k-partitioning up to any x'th position: the rightmost partition is given by $[\text{Start}(x), x]$; the next rightmost position is given by $[\text{Start}(\text{Start}(x)-1), \text{Start}(x)-1]$; &c.

Pos	1	2	3	4	5	6	7
Deg Seq	5	3	3	2	1	1	1
$\mathrm{Cost}(1,x)$	0	2	2	3	4	2	2
$\mathrm{Start}(x)$	1	1	1	1	1	4	4

Table 3.2: The Values of the Recursion for the Anonymization of the Second Example Graph

Let

$$\begin{aligned}\mathrm{Cost\_Split} &= \min_{i\in[\max(k,x-2k+1),x-k]}(\max(\mathrm{Cost}(1,i-1),\Delta(i,x))),\\ \mathrm{Pos\_Split} &= \mathrm{argmin}_{i\in[\max(k,x-2k+1),x-k]}(\max(\mathrm{Cost}(1,i-1),\Delta(i,x))).\end{aligned}$$

Then,

$$\begin{aligned}\Delta(x,y) &= d_x - d_y,\\ \mathrm{Cost}(1,x) &= \Delta(1,x),\ \text{if } x < 2k,\\ \mathrm{Cost}(1,x) &= \mathrm{Cost\_Split},\ \text{if } x \geq 2k,\\ \mathrm{Start}(x) &= 1,\ \text{if } x < 2k,\\ \mathrm{Start}(x) &= \mathrm{Pos\_Split},\ \text{if } x \geq 2k.\end{aligned}$$

For our second example graph of Figure 3.2, the degree sequence is $(5,3,3,2,1,1,1)$. The optimal k-partitioning of this degree sequence is $((5,3,3),(2,1,1,1))$, which we arrive at by evaluating the recursion, as tabulated in Table 3.2.

Two parameters important for the next stages of our algorithm arise out of the k-partitioning of the degree sequence, namely $\max\ \text{deficiency}$ and total deficiency. As we show later in Lemma 3.3.9, the $\max\ \text{deficiency}$ is a lower bound on $|\mathrm{V}' \setminus \mathrm{V}|$ in an optimal solution. We necessarily must bound the value of total deficiency, because this allows to upper bound the number of edges added by–and thus execution cost of–our entire algorithm.

Lemma 3.3.8. *The* total deficiency *of an optimal* k*-partitioning of a sorted degree sequence is upper-bounded by* $(n-1)(2k-1)$.

Proof. First, note that from Proposition 3.3.7, any optimal partitioning with a partition sized $2k$ or greater can be split into two partitions such that the $\max\ \text{deficiency}$ is not increased. Furthermore, doing so will decrease the total deficiency unless the contribution of the partition is already zero. So, no optimal partition should contain $2k$ or more elements.

If $[\text{front}(d_i), \text{end}(d_i)]$ represents the partition containing degree d_i, $[\text{front}(p_j), \text{end}(p_j)]$ denotes the partition p_j, and $|\mathcal{P}|$ denotes the number of partitions, then the total deficiency

Figure 3.3: An Example Graph for which the Additional Vertex Has the Same Degree as a 3-Anonymous Group of Original Vertices

of an optimal k-partitioning is given by:

$$\begin{aligned}
& \sum_{d_i} \mathrm{front}(d_i) - d_i \\
\leq\ & \sum_{d_i} \mathrm{front}(d_i) - \mathrm{end}(d_i) \\
\leq\ & (2k-1) \sum_{p_j} \mathrm{front}(p_j) - \mathrm{end}(p_j) \\
\leq\ & (2k-1) \left(\mathrm{front}(p_1) - \mathrm{end}(p_{|\mathcal{P}|})\right) \\
\leq\ & (2k-1)(n-1),
\end{aligned}$$

where the second inequality follows because the deficiency contributed by a particular degree is certainly no more than the $\max \mathrm{deficiency}$ contributed by its entire partition, the third inequality follows because $\mathrm{front}(p_{j+1}) \leq \mathrm{end}(p_j)$, and the fourth inequality follows because the maximum degree in a simple graph is $n-1$. □

3.3.2 Stage 2: Determining m, the Number of New Nodes

The next step in anonymizing a graph is to determine precisely how many vertices should be added. Ideally, we would add exactly $\max \text{deficiency}$ vertices, because this is a lower bound on how many *must* be added, as shown in Lemma 3.3.9:

Lemma 3.3.9. *To k-degree-anonymize a set of vertices $X \subseteq \mathrm{V}$ by means of vertex addition, one must add at least as many new vertices as the $\max \text{deficiency}$ of an optimal k-partitioning of the degree sequence of X.*

Proof. First note that any graph $\mathcal{G}'$ in which X is k-degree-anonymous corresponds to some k-partitioning of the original degree sequence of X. In order to satisfy every target degree arising from the k-partitioning, some vertex will require as many new edges as the $\max \text{deficiency}$ of that k-partitioning. Also, because edges can only be added from each $v \in X$ to new vertices and because we investigate only simple graphs (i.e., those which do not contain multiple edges between the same source and destination nodes), clearly these new edges must connect to $\max \text{deficiency}$ new vertices in order to satisfy that particular vertex's requirement.

From among all k-partitionings of the degree sequence of X, corresponding to all graphs $\mathcal{G}'$ in which X is k-degree-anonymous, the optimal choice minimises $\max \text{deficiency}$, so also minimises how many new vertices are required in $\mathcal{G}'$. □

However, we need to anonymize the new vertices as well, and this may require adding more than $\max \text{deficiency}$ nodes, as illustrated by the example in Figure 3.5. Our approach to adding edges we describe in more detail in the next subsection, but there are a couple important things to note here because they influence this stage. For conciseness, let $md =$

$\max$ deficiency and let $td = \text{total deficiency}$. Then, once every target degree is achieved through edge additions in Stage 3, $td (\mod md)$ of the new vertices will have some degree, call it d, and the other $md - td (\mod md)$ will have a degree of $d - 1$. If these both appear in the target degrees of the k-partitioning, the entire graph $\mathcal{G}'$ is k-anonymous; on the other hand, if one or the other does not, the new vertices need be explicitly anonymized as well. To accomplish this, we create a k-anonymous group of the new vertices by introducing intra-new-vertex edges to establish that they all have the same degree. This requires that: 1) there are k new vertices with which to form a k-anonymous group; and 2) there are an odd number of new vertices (we explain why in the next subsection).

Both these conditions are satisfied when we add exactly

$$m = (1 + \max(md, k))(\mod 2) + \max(md, k)$$

new vertices to the graph to create a new vertex set V' of size $n + m$.

3.3.3 Stage 3: The Edge Insertion

The final stage of our anonymization algorithm is to add new edges to the graph (restricting the addition to those with an endpoint in $\mathrm{V}' \setminus \mathrm{V}$, as per the problem definition), in order to meet the target degrees established in Stage 1. This must be done carefully because not any arbitrary approach is guaranteed to succeed: the task of anonymizing the m new vertices added in Stage 2 is, in fact, NP-hard.

Thus the motivation behind our *cycling* approach: it is designed such that it will always produces a scenario in which the anonymization of the new vertices is tractable.

First, we order the m additional vertices (arbitrarily). Let $def(v_i)$ be the discrepancy between the $i'th$ degree in the degree sequence and the largest degree within the same partition (the *deficiency* of the i'th vertex). We then connect the first $def(v_1)$ of the m additional vertices to the vertex corresponding to v_1, the next $def(v_2)$ additional vertices to the vertex corresponding to v_2, and so on until all m additional vertices have an edge. This process ends at some v_j.

We continue with another iteration, this time starting at v_j and with subsequent iterations until we have satisfied the deficiency of every node in the original graph. This is illustrated for the second example graph in the first five steps of Figure 3.4.

Because of the nature of this *cycling* procedure, always adding an edge to an additional vertex that has not yet been visited on that particular iteration, we can guarantee that the degrees of the additional vertices are all within one. In fact, if exactly d iterations are required in order to anonymize the original graph, then (as we hinted in Stage 2) $td(\mathrm{mod}\ m)$ of the m additional vertices will have degree d and the remaining $m - td(\mathrm{mod}\ m)$ will have degree $d-1$. Because we serviced each original vertex in turn, we can be certain to not accidentally introduce the same edge twice. So, we have successfully k-anonymized every vertex that was in the original graph. Quite importantly, we have done so in a manner that guarantees that the remaining edge-addition-based problem is tractable because there is an odd number of vertices and the degrees are all within one.

The last detail is to ensure that all the new vertices are themselves k-anonymous. As a first recourse, if d and $d-1$ are both present as target degrees from the recursion in Stage 1, then the additional vertices will already belong to some k-anonymous group. Figure 3.3 demonstrates that 3-anonymizing the first example graph is such a case, where the

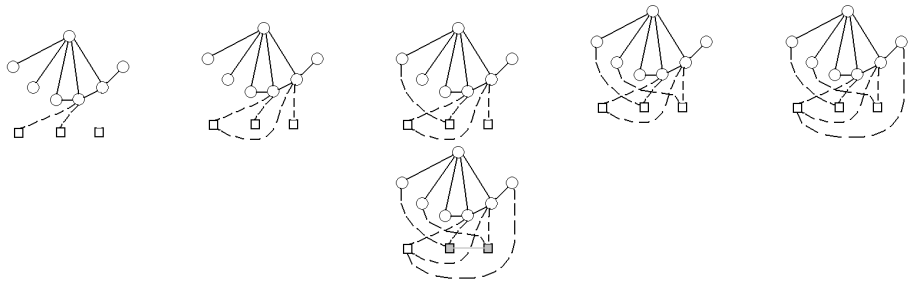


Figure 3.4: 3-Anonymizing the Second Example Graph with Three Additional Vertices

additional vertex fits nicely into the 3-anonymous group of degree 1 vertices.

In the event that either d or $d-1$ is not present in the anonymized degree sequence, we explicitly anonymize the new vertices. For the $m - td \pmod m$ vertices with degree $d-1$, we randomly pair them and add an edge between each pair. If $m - td \pmod m$ is even and $m \geq k$, we know this is sufficient to guarantee all new vertices have the same degree (namely d) as at least $k-1$ other vertices.

If, instead, $m - td \pmod m$ is odd, then this pairing will leave out one last vertex, call it r. Because of our diligence in selecting m, we know that $m-1$ is even (and at least k), so we can add an edge from r to each of two other additional vertices so that all three have degree $d+1$. The remaining $m-3$ vertices with degree d can then all be paired off again (since $m-3$ is even) and all additional vertices will be anonymized with degree $d+1$. Some care must be taken to not accidentally re-add an edge between additional nodes, but this is really quite trivial because of how we proceeded with the preceding edge addition. Figure 3.5 illustrates this scenario. The degrees (4 and 3) of the two additional nodes are irreconcilable and $md-1$ is odd, so we instead added the extra vertex as in Figure 3.4.

3.3.4 Algorithmic Analysis

From the algorithm described in this section, we prove the following theorem:

Theorem 5. *Our algorithm produces a k-anonymous graph $\mathcal{G}'$ containing the input graph $\mathcal{G}$ as an induced subgraph, using $\mathcal{O}(nk)$ time and $\mathcal{O}(n)$ space. The number of new vertices added is optimal up to an additive factor of k.*

Proof. First, we note that the degree sequence partitioning requires $\mathcal{O}(n)$ space and $\mathcal{O}(nk)$ time. If this recursion is evaluated bottom-up (i.e., from $x = 1$ to $x = n$) and the results are memoised after each iteration, then the running time is linear in nk because the degree sequence has length n and for each element of the degree sequence there is an iteration that will cost at most a comparison to $2k$, lookups of Cost, and computations of Δ for each $i \in [k, 2k)$. The memory cost of the memorization is $3n$ units of memory: an array of size n each for storing the results of the Cost and Start calculations and an additional array of length n in which the original degree sequence is kept.

Second, the number of edges added is bounded by $td + m$, since td edges are added between original and additional vertices; the subsequent anonymization of additional vertices never changes an additional vertex's degree by more than $(d + 1) - (d - 1) = 2$; and at least one additional vertex already has a degree of d. We introduce at most m additional vertices, at most either $md+1$ or $k+1$. A lower bound on the optimal number of additional vertices is $mdabbr \geq 1$, since any graph with a md of 0 is already n-anonymous. So, we add at most $md + 1 - md = 1$ more vertex than optimal or we add at most $k + 1 - 1 = k$ more vertices than optimal.

Since m is bounded by n (because neither md nor k can be larger than n) and the td is

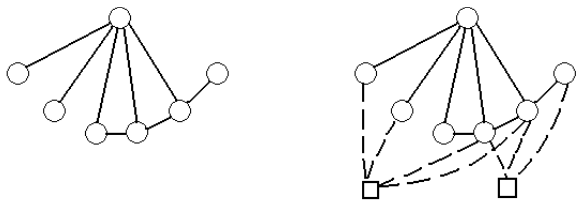


Figure 3.5: A Failed Attempt at 3-Anonymizing a Graph by Using Only Two Additional Vertices

bounded by $(n-1)(2k-1)$, the addition of these edges requires $\mathcal{O}(nk)$ time. Since this requires constant memory, the overall space usage is $\mathcal{O}(n)$. □

Note that we assume in this proof that the sorted degree sequence can be provided based on the representation of the graph. If this is not the case, an $\mathcal{O}(n \log n)$ preprocessing step to compute the sorted degree sequence subsumes our algorithm.

3.3.5 On the Security of $\mathcal{G}'$

Perhaps an attacker's best chance at de-constructing our anonymization comes from cycling through all possible choices of the new equivalence classes (k-anonymous groups), removes them from the graph and then runs our algorithm to check if the output matches the original anonymized graph. We note here, that the last step where the adversary has to check if the output graph is the "same" as the initial anonymized graph requires an algorithm for checking graph isomorphism. No efficient (polynomial time) algorithm is known for this problem. In addition, there are examples in which an equivalence class of additional vertices is merged with an equivalence class of original vertices because they have the same degree at the end of the anonymization procedure. For such graphs, this attack will not be able to extract the additional vertices only.

3.4 Experimental Results

We present a systematic set of experiments to evaluate the efficiency of the algorithm developed in Section 3.3. Recall that our main motivation for developing the algorithm is to preserve the privacy (through k-anonymity) while at the same time preserving the properties of the original graph as much as possible.

3.4.1 Metrics and Setup

DATASETS: We choose five datasets from diverse domains for our empirical study. The domains range from power grid network modeling generators and substations in a power network to an email communication network in a company (Enron). The properties of each of these datasets are shown in Table 3.4. Note that Prefuse [22] and Football [19] are small datasets containing over a 100 nodes, Power Grid [38] and Net Science [34] are midsized with a few thousand nodes, while Enron [27] is a large dataset with over $36K$ nodes and hundreds of thousands of edges.

Graph	Nodes	Edges	APL	CC
Enron	36692	183831	3.39	0.09
Power Grid	4941	6594	18.99	0.10
Net Science	1589	2742	0.35	0.69
Prefuse	129	159	3.16	0.07
Football	115	613	2.51	0.61

Table 3.3: Structural Properties of Datasets

An example of the co-authorship social network graph for the Net Science dataset is shown in Figure 3.6.

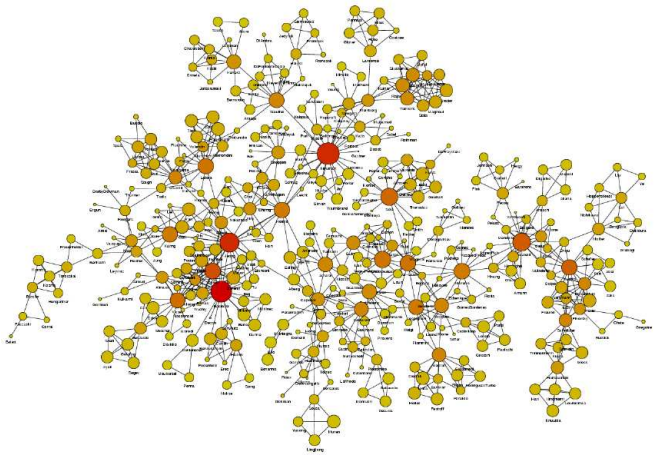


Figure 3.6: Net Science Network

METRICS: We measure the distortion introduced by the algorithm via metrics which are commonly studied properties in the social network literature [5]. The three metrics we chose to study are defined below.

1. Clustering Coefficient (CC): Informally, clustering coefficient measures the percentage of paths of length 2 which are also triangles. This metric in some sense measures *triadic closure* of graphs–social networks are known to have significant triadic closure (friends of a person are also likely to know each other). More formally,

$$CC = \frac{||\{u, v, w \in V : (u, v) \in E \wedge (u, w) \in E \wedge (v, w) \in E\}||}{||\{(u, v, w \in V : (u, v) \in E \wedge (u, w) \in E\}||}$$

,

for all ordered triples $u, v, w \in V$.

2. Average Path Length (APL): This metric is a measure of the expected path length in the graph between any two randomly chosen connected vertices. This metric is highly relevant

as it is directly related to the *six degrees of separation* that is known to exist between randomly chosen people in social networks [33]. Define a predicate $C(u, v)$ to be *true* if u and v are connected in the graph and $false$ if they are not connected. Define $CP = \{(u, v) : C(u, v) = true\}$ to be the set of all the pairs of vertices that are connected. We define the average path length to be:

$$APL = \frac{\sum_{(u,v)\in CP} PathLength(u, v)}{\mid CP \mid}$$

. We assume that $PathLength(u, u) = 0$ for all $u \in V$.

3. Hop Plot: The connectivity of a graph is (typically graphically) modeled using the *hop plot*. The hop plot studies reachability for each path length k. For a given value of k, the hop plot displays, summed over all the vertices, the number of nodes reachable from that vertex using paths of length at most k. The maximum value for any value of k is n^2 where n is the number of vertices in the graph. The smallest value of k for which the maximum value of n^2 is reached is the *diameter* of the social network, the path length using which any two nodes in the graph can reach each other. Changing or distorting the connectivity of a graph drastically would change the shape of their hop plots. This is the main motivation behind studying these plots.

SETUP: A java implementation of the algorithm[2] was used to measure the distortion based on the metrics for the five chosen

datasets defined earlier in this section. The resulting graphs were manually verified to be k-anonymous. All experiments were performed on a quad-core Intel Xeon 5140 $2.33GHz$ processor with $4MB$ of L2 cache and $6GB$ of RAM.

[2] Available at `http://webhome.csc.uvic.ca/~schester/NodeAnon/`

3.4.2 Results

Figures 3.7, 3.8, and 3.9 show how the metrics of the resulting graphs over varying k compare to those of the original graphs for the five datasets in the experimental study. Experimental studies in literature typically study small values of k for graphs and social networks (close to 3 and rarely even 100). We vary k as a fraction of n for our experiments, while still maintaining that $k << d$. For large and midsize datasets, we vary k from $k = 0.25$ up to 2% of the number of nodes in the graph. For the two small datasets (with over 100 nodes), this is too refined, so we vary k from $k = 1$ up to 5% of the number of nodes in the original graphs. For the large and midsize datasets, 2% of the number of nodes would still translate to k values of 80 to 720 which is a fairly large number with respect to degree anonymization.

Distortion Performance: Figure 3.7 corresponds to the plots measuring distortion for the large Enron dataset and Figures 3.8 and 3.9 show plots measuring distortion for the mid-size and small sized datasets respectively. From the plots, it can be observed that the values of Clustering Coefficient and Average Path Length in the distorted graphs are *close* to the corresponding values before distortion. For the biggest dataset (Enron), even for a significantly large value of k (at 2%), these values are very close to the ones before distortion, leading us to believe that for real social communication networks, the approach we have presented is reasonably good. Note that for the Net Science dataset, the distortion lowers both the CC and the APL values compared to the original graph. This can be explained by looking at the structure of this network (Figure 3.6). The densest nodes in the network are not connected to each other. By necessity, an optimal partitioning of the graph into groups each containing at least k vertices will group together these dense and disconnected

vertices and create much shorter paths through the graph. Additionally, the addition of dummy nodes would create paths of length 2 between the original nodes but would leave them disconnected (as edges are not added between existing nodes). The grouping would answer the decrease in CC while the addition of bridges connecting originally disconnected vertices would account for the reduction in APL.

By observing the hop plots for the datasets, it can be seen that the shapes of these plots are pretty much the same for all values of k. Of course, due to the addition of new nodes, the diameter and the maximum y-axis value necessarily increase. However, the shape of the plots remain intact subsequent to distortion.

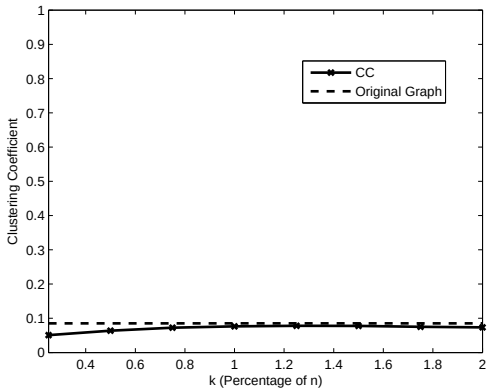
Running Time: For the largest dataset (Enron), the running time for the actual anonymization took a little over one minute (70 seconds) for all the values of k in the plots. The running time reported is the average of 5 independent runs of the algorithm for each value of k. The evaluation time was dominated by the computation of the metrics on the original and distorted graphs. These took about 20 to 30 minutes each. Given that privacy is of paramount importance, the times spent on evaluating the metrics on distorted graphs are still reasonable for large graphs. The times on smaller graphs were much lower and had the same trend where the computation of the metrics dominated the running time. We also evaluated a sampling procedure to compute APL and CC metrics. For APL, we sampled 1 million pairs of random vertices of the graphs and measured the average path length between the two vertices. This method produced APL values consistently within 0.001 of the exact APL value for all the datasets. For CC, we averaged the computation over randomly sampled triplets and observed that the sampling procedure produced values consistently within 0.1 of the original values. The sampling procedure took a few minutes as opposed

to 20 or 30 minutes and provide a viable alternative in case time is of essence.

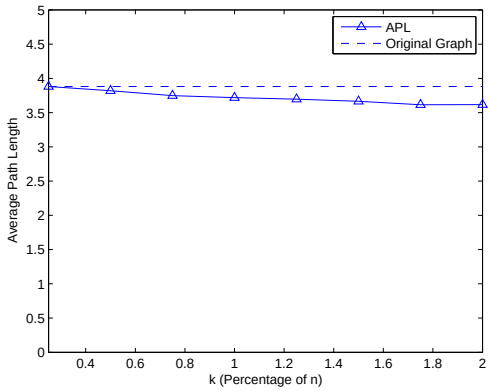
3.5 Future Work

To extend the work done in this chapter, we intend to look at approximation algorithms for the labeled case, now that its complexity is known. Additionally, we intend to investigate slightly stronger adversarial models to better elucidate the jump in complexity incurred by the corresponding jumps in adversarial knowledge.

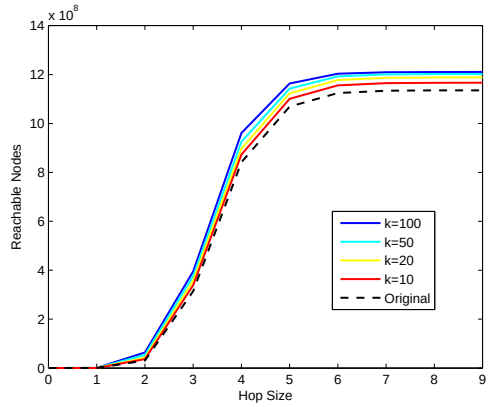
The concept of node additions to achieve k-anonymity is new. The implications of adding new nodes to a social network to achieve privacy is not fully understood. It would be beneficial to pursue and in-depth study of how the concept of node additions effects graph properties in general away from the k-anonymity setting.



(a) Enron - CC vs k

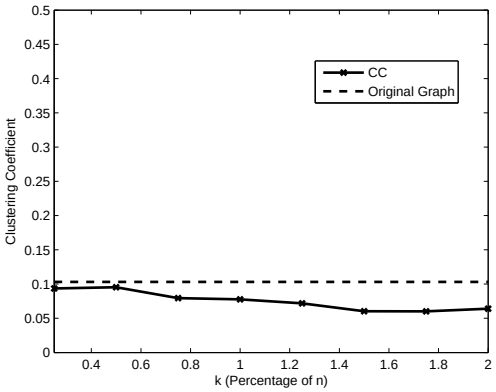


(b) Enron - APL vs k

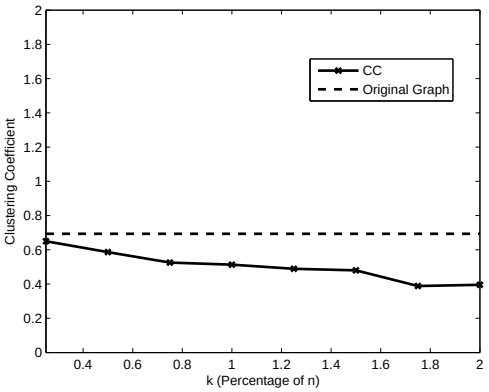


(c) Enron Hop Plot

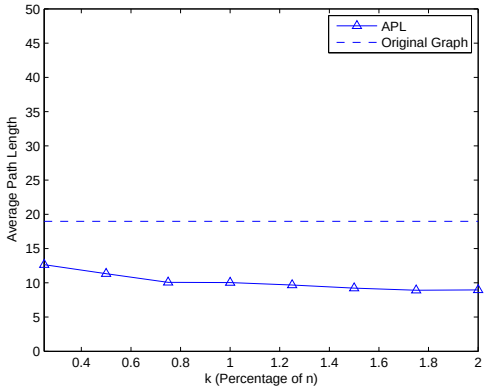
Figure 3.7: Results - Large Dataset (Enron)



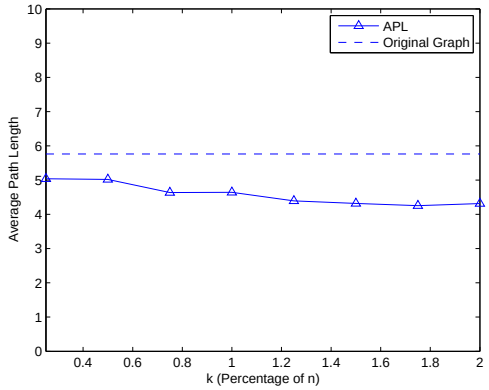
(a) Power Grid - CC vs k



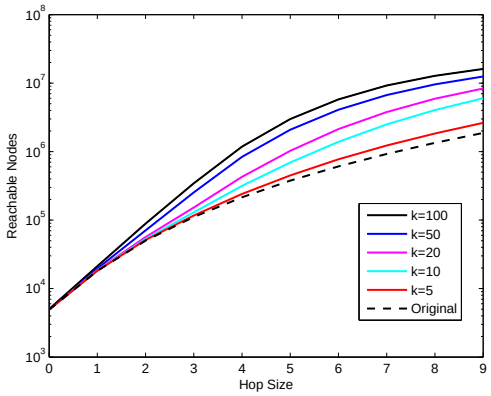
(b) Net Science - CC vs k



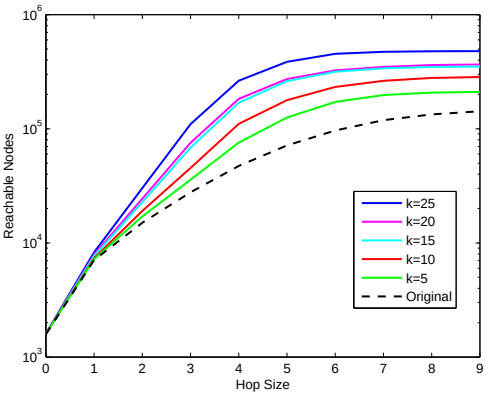
(c) Power Grid - APL vs k



(d) Net Science - APL vs k

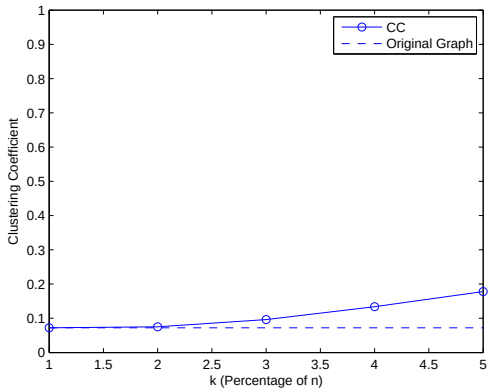


(e) Power Grid Hop Plot

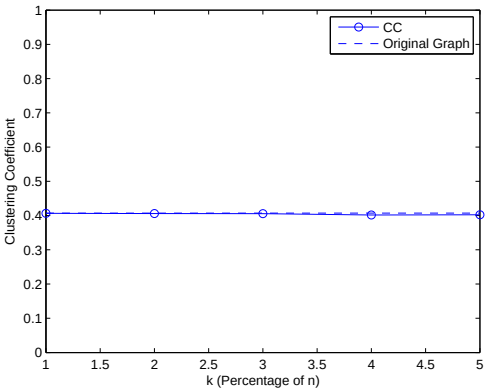


(f) Net Science Hop Plot

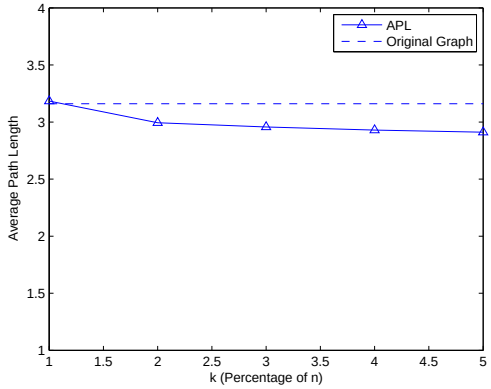
Figure 3.8: Results - Medium Datasets (Power Grid and Net Science)



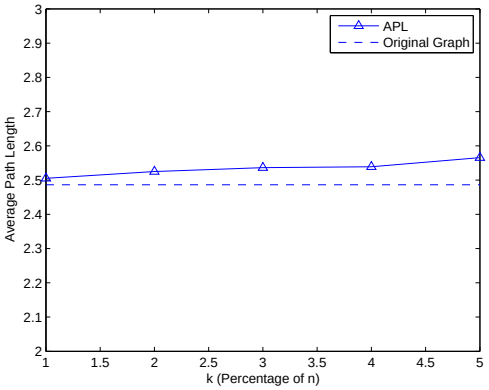
(a) Prefuse - CC vs k



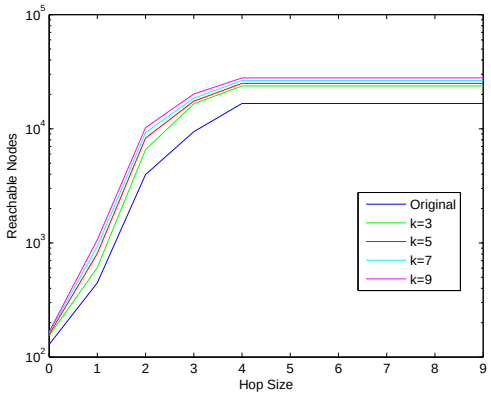
(b) Football - CC vs k



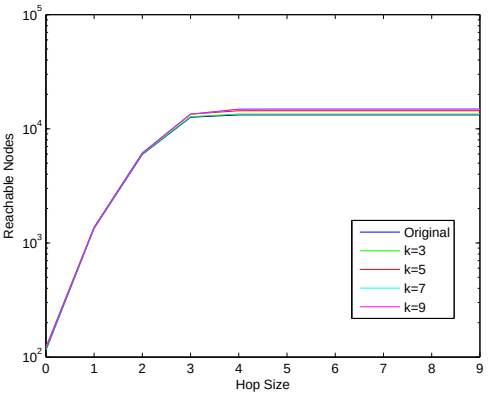
(c) Prefuse - APL vs k



(d) Football - APL vs k



(e) Prefuse Hop Plot



(f) Football Hop Plot

Figure 3.9: Results - Small Datasets (Prefuse and Football)

Chapter 4

Privacy Beyond k-anonymization

In this chapter, we present our results for α-proximity. The original motivation for this body of work was the progression of data privacy that developed for table data, which is outlined in §4.1. In §4.2, we outline all preliminary contributions necessary for this chapter. We present our main result in §4.4 giving an efficient algorithm for α-proximity. Finally, in §4.5, we provide experimental results showing the effectiveness of the algorithm in §4.4 in protecting the edge occupancy of the α-proximized graph.

4.1 Motivating α-proximity

Starting with k-anonymization, the table privacy problem progressed to more involved definitions known as l-diversity and t-closeness. Protection that k-anonymity provides is easy to understand for tables. A table that satisfies k-anonymity for some value k, will allow a record to be protected from identification with confidence of at least $1/k$. k-anonymity

protects against identity disclosure, as mentioned in Chapter 1, but provides little to no protection against attribute disclosure. This was recognized in depth in [30]. A new measure of privacy, called l-diversity was proposed, which required that for each equivalence class in a given table, there were l distinct values of the sensitive attribute inside the equivalence class and each equivalence is k-anonymous for the identifiable attributes. We state the definition here:

Definition 4.1.1. *The l-diversity principle. An equivalence class is said to have l-diversity if there are at least l well-represented values for the sensitive attribute. A table is said to have l-diversity if every equivalence class of the table has l-diversity.*

Example 8. *We give an example of the importance of l-diversity here. Take a look at Table 4.1, where the identifying attributes are Postal Code and Age . We see the original table (Table 4.1(a)) and its 3-anonymous version as well (Table 4.1(b)) with respect to the identifying attributes. Suppose Alice knows Bob is in his 20s and lives somewhere in the vicinity of the V8N postal area and is also in this table. She can easily conclude looking at the 3-anonymous table that Bob must have Cancer. This is the homogeneity attack. If we were to take the first anonymous group of entries, and adjust the values to Heart Disease to make it 2-diverse, as in Table 4.1(c), we protect against this type of attack.*

Some limitations of l-diversity were pointed out in [28], namely

- l-diversity might be difficult and unnecessary to achieve: If we take a binary attribute, say 0 and 1, thats distribution is heavily skewed in one direction, creating l-diversity may in fact misrepresent the data substantially.

Postal Code	Age	Disease
V8N6N3	29	Cancer
V8N4B7	22	Cancer
V8N4J8	27	Cancer
V8P4N8	40	ADD
V8N4N4	40	Heart Disease
V8T4B9	40	Flu

(a) Patient Table

Postal Code	Age	Disease
V8N***	2*	Cancer
V8N***	2*	Cancer
V8N***	2*	Cancer
V8****	40	ADD
V8****	40	Heart Disease
V8****	40	Flu

(b) 3-Anonymous Table

Postal Code	Age	Disease
V8N***	20-40	Heart Disease
V8N***	20-40	Cancer
V8N***	20-40	Cancer
V8****	*	Cancer
V8****	*	ADD
V8****	*	Flu

(c) 2-Diverse Table

Figure 4.1: 2-diversity Example

- l-diversity is insufficient to prevent attribute disclosure: Take an equivalence class with 50 records, 49 of which are one value for a given attribute and 1 record has a different value. We have 2-diversity here, however with 98% probability we can deduce the value of this attribute for this equivalence class.

Seeing its limitations, a new data privacy definition was proposed called t-closeness in [28]. t-closeness required that the distribution within each equivalence class of the sensitive attributes be t-close to overall distribution of the attribute.

Definition 4.1.2. *An equivalence class is said to have t-closeness if the distance between*

the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

In this chapter, we try and build on the original principal of k-anonymization for graphs, and take a similar path for graph data privacy as was taken for table data privacy. l-diversity, has been defined for graphs in [42]. For completeness we give this definition here:

Definition 4.1.3. *Let G be a social network and G‘ be an anonymization of G. G‘ is said to be l-diverse if in every equivalence class of vertices, at most $1/l$ of the vertices are associated with the most frequent sensitive label.*

The next natural step was to look for an analogous definition for t-closeness that preserved distributions within graph data. Through much research, we were able to come up with the privacy measure of α-proximity. It is explained in the next section.

4.2 Attribute Disclosure

Before detailing our notion of anonymity to protect against attribute disclosure, we formalize the attack. Specifically, we are assuming that the social network is an undirected, simple, vertex-labeled graph. We recall the definition here for convenience:

Definition 4.2.1 (labeled social network)**.** *A* labeled social network *is a graph* $\mathrm{G} = (\mathrm{V}, \mathrm{E}, \mathcal{L}, \ell)$*, where* V *is a vertex set,* $\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ *is the edge set,* $\mathcal{L}$ *is an alphabet of labels, and* $\ell = \mathrm{V} \mapsto \mathcal{L}$ *is a labeling function that assigns a label* $l \in \mathcal{L}$ *to each vertex in* V *For simplicity,* $(u, v) \in \mathrm{E} \rightarrow (v, u) \in \mathrm{E}$*. We assume for convenience that the elements of* $\mathcal{L}$ *are ordered.*

For a particular node, its label sequence is the collection of labels of itself and all its friends:

Definition 4.2.2 (label sequence neighbourhood)**.** *The* label sequence neighbourhood *of a vertex* $v \in \mathrm{V}$*, denoted* $\eta(v)$*, is the set of vertices used to make up the* label sequence *of* v*. That is to say,* $\eta(v) = \{v\} \cup \{u \in \mathrm{V} : (u, v) \in \mathrm{E}\}$.[1]

In terms of attribute sensitivity and information disclosure, it is important to consider the distribution of labels over a set of vertices:

Definition 4.2.3 (label distribution)**.** *For* $\mathrm{W} \subseteq \mathrm{V}$*, let* $\mathrm{count}(l_i, \mathrm{W})$ *denote the number of vertices in the set* W *which have label* l_i*. Then, the* label distribution *over* W*, denoted* $\mathrm{distr}(\mathrm{W})$*, is the vector* $\frac{\langle \mathrm{count}(l_1,\mathrm{W}),\ldots,\mathrm{count}(l_{|\mathcal{L}|},\mathrm{W})\rangle}{|\mathrm{W}|}$*. Also, the distribution of a particular label,* l_i *is one element of the vector,* $\mathrm{count}(l_i, \mathrm{W})/|\mathrm{W}|$*.*

Then, given two distributions $\mathrm{distr}(\mathrm{W}_i)$, $\mathrm{distr}(\mathrm{W}_j)$, we define a distance measure:

Definition 4.2.4 (distance between two distributions)**.** *The distance between two distributions,* $\mathrm{distr}(\mathrm{W}_i)$, $\mathrm{distr}(\mathrm{W}_j)$*, denoted* $\delta(\mathrm{distr}(\mathrm{W}_i), \mathrm{distr}(\mathrm{W}_j))$*, is simply the sum of the differences between all but their last elements. For example, the distributions* $\langle .7, .2, .1\rangle$ *and* $\langle .2, .4, .4\rangle$ *have a distance of .5+.2=.7.*

Given these definitions, we can now formally describe the adversary's attack. By identifying the label sequence of a node v in a labeled social network, an adversary gains knowledge about the labels of all the nodes in the neighbourhood of v. Whereas beforehand, he could only surmise the probability that their label is l_i to be $\mathrm{count}(l_i, \mathrm{V})/|\mathrm{V}|$, he now knows that the probability is closer to $\mathrm{count}(l_i, \eta(v))/|\eta(v)|$. That is:

[1] *Note that this definition is equivalent to a traditional neighbourhood if and only if every vertex has a self-edge, an alternative formulation that we exploit for simplicity in our implementation.*

Definition 4.2.5 (NAD attack)**.** *A* neighbourhood attribute disclosure attack *against node* v *is one in which the adversary discovers a more refined estimate of* $\mathrm{distr}(\eta(v))$ *than he had when he only knew* $\mathrm{distr}(\mathrm{V})$*. His knowledge gain is* $\delta(\mathrm{distr}(\mathrm{V}), \mathrm{distr}(\eta(v)))$**.**

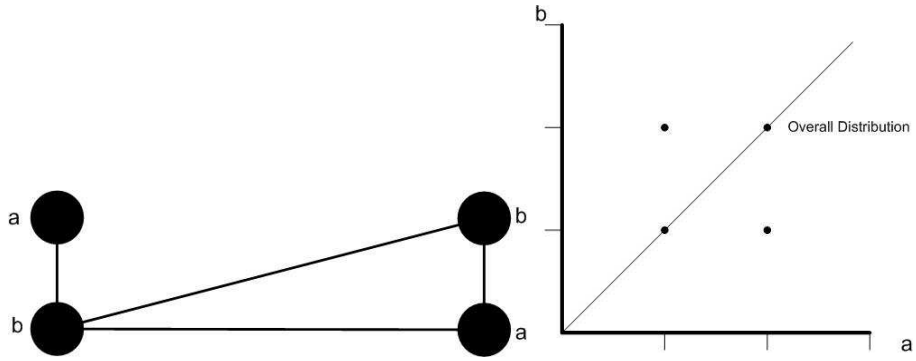
To combat this type of attack, we define a new measure of anonymity:

Definition 4.2.6 (α-proximity)**.** *The neighbourhood of a vertex* v *is said to be* α-proximal *if* $\delta(\mathrm{distr}(\eta(v)), \mathrm{distr}(\mathrm{V})) \leq \alpha$*. A labeled social network* G *is* α-proximal *if the neighbourhood of every vertex in* V *is* α-proximal*.*

4.3 An Interpretation for α-Proximity

Our ongoing research on α-proximity led to a way to represent the distribution of the vertex labels in a graph. Overall, we can describe α-proximity problem as the desire that every vertex's immediate neighbours have a similar label distribution as that of the complete graph. When looking at this graphically, we can easily create a line in an n-dimensional space, pertaining to this desired distribution. For example, for a graph G with a binary label set, we can easily represent this on a Cartesian graph with the x-axis representing one label and the y-axis the other.

Example 9. *Here we see a graph* G *on 4 vertices, with a label set* $\{a, a, b, b\}$*. We have 2 vertices with label* ***a*** *and 2 vertices with label* ***b****. Therefore, on a 2-dimensional graph, we create a line from (0,0) to (2,2) which represents the distribution of the labels in the entire graph* G*. We can also add in points for each vertex in the graph* G*, and see their* α*-proximity to the overall distribution.*



(a) A small social network (b) Interpretation of Network

Figure 4.2: Interpretation of α-proximity

From this simple example, we see the ease with which we can not only represent the notion of α-proximity graphically, but also measuring the distance of each vertex from the α-proximal line required.

4.4 An Algorithm to Produce α-proximal Graphs

Here we describe a greedy algorithm to compute an α-near graph G' from an input graph G by augmenting the edge set with elements of $(\mathrm{V} \times \mathrm{V}) \setminus \mathrm{E}$. For simplicity of discussion, we assume the labels are binary (i.e., $|\mathcal{L}| = 2$), but the ideas generalize quite readily.

The algorithm relies on two intuitions. First, $\mathrm{K}_{|\mathrm{V}|}$ is α-proximal. So, given that the algorithm adds an edge on every iteration, it always progresses towards a solution in at most $(\mathrm{V} \times \mathrm{V}) \setminus \mathrm{E}$ steps. Second, by partitioning the vertex set based on the label of a vertex and the label to which it needs be connected to be α-proximal, we construct partitions of

Algorithm 1 Greedy α-proximity

1: **while** graph is not α-near **do**
2: Partition V into $\mathrm{V}_{a->a}$, $\mathrm{V}_{a->b}$, $\mathrm{V}_{b->a}$, $\mathrm{V}_{b->b}$, and $\mathrm{V}_{\alpha-\mathrm{proximal}}$
3: Greedily add edges among vertices in $\mathrm{V}_{a->a}$ and among vertices in $\mathrm{V}_{b->b}$
4: Greedily add edges from vertices of $\mathrm{V}_{a->b}$ to vertices of $\mathrm{V}_{b->a}$
5: **if** graph is not α-near **then**
6: Add a random new edge from $(\mathrm{V} \times \mathrm{V}) \setminus \mathrm{E}$
7: **end if**
8: **end while**

mutually beneficial vertices. That is to say, all vertices in $\mathrm{V}_{a->a}$ have label a and require more edges to other vertices of label a. Consequently, from a greedy perspective, it does not make sense to add edges from this partition to other partitions. On the other hand, vertices of label a which require more edges to vertices of label b (i.e., elements of $\mathrm{V}_{a->b}$) pair perfectly with elements of $\mathrm{V}_{b->a}$.

The addition of the random edge allows the algorithm to continue in the event that adding edges within compatible partitions is itself insufficient.

4.5 Experimental Evaluation

Despite the simplicity of the algorithm in §4.4, it is very effective. The predominant concern that it evokes is whether the quality of the solutions it produces degrades towards the trivial solution $\mathrm{K}_{|\mathrm{V}|}$ for every input graph. To assess the reality of this concern, we conducted an experimental investigation.

Experimental Setup

We implemented[2] the algorithm in C and ran it against first the toy example from §1.0.3, for which it produces the same solution we present, and then against a series of randomly generated graphs.

The dependent variable we evaluate is *occupancy change*: out of the $\binom{n}{2}$ possible edges in an undirected graph on n vertices, what percentage do we add to the graph G by running the algorithm. We generate fifteen synthetic, random graphs by selecting uniformly from pairs of vertices until we have the desired number of edges. We vary the number of nodes through $\{10, 20, 30, 40, 50\}$ and the initial occupancy through $\{25\%, 50\%, 75\%\}$. We fix the label set to be binary, since this is the most common type of label, and distribute the two labels evenly among the vertices.

The results of the experiments are illustrated in Figure 4.3. In Figure 4.3(a), we give the occupancy change in terms of varying α with the initial occupancy held fixed at 25%, and in Figure 4.3(b), we present the occupancy change in terms of varying initial occupancy with $\alpha = .1$. Note that the cases when the solution degrades to $\mathrm{K}_{|\mathrm{V}|}$ are those when the sum of the initial occupancy and the occupancy change is 1.

Discussion of Results

The results of the experiment in §4.5 are quite promising. Out of the sixty trials, only two instances, both with very small α, resulted in $\mathrm{K}_{|\mathrm{V}|}$ and the majority of trials produced graphs with $\leq 60\%$ occupancy. This is very important, because if the output is, indeed,

[2]The implementation is available on the author's webspace, `http://webhome.csc.uvic.ca/~schester/`.

$K_{|V|}$, then the analytical value of the network is lost. Rather, the closer that G' is to G, the better the analytical value is retained.

In terms of the variables that we investigated, no clear pattern emerges with respect to the number of vertices. As is clear from its definition, increases in α require adding fewer edges. More interestingly, the drop off is quite abrupt at very small values of α and relatively minimal thereafter. This implies that requiring better anonymization than $\alpha = .1$ is perhaps unrealistic. The results with respect to initial occupancy reveal that for fixed n and α, one can expect similar total occupancy in the resultant graph, independent of initial occupancy.

It is especially encouraging that there are a number of trials that required occupancy changes of $\leq 10\%$. Not only does this imply that the algorithm did well on those instances, but it also implies that our proposed metric, α-proximity, can be quite readily achieved on certain input graphs.

4.6 Future Work

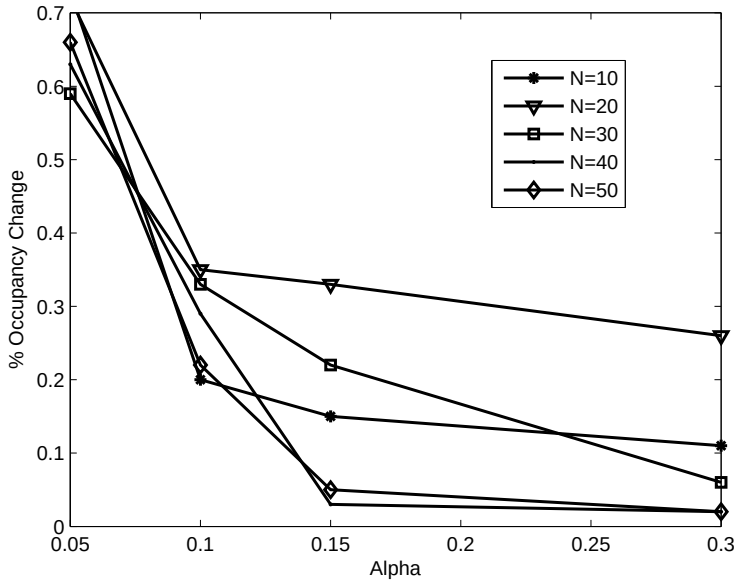
There is substantial opportunity to extend this research. A first direction is in determining whether we can produce guarantees on the optimality of the solutions produced by this algorithm, as a function of the variables we investigated in this experimental study and of the skew in the overall distribution of labels.

We are also investigating new algorithmic ideas based on dynamic programming and on modifying our greedy algorithm to eliminate the non-determinism in a sensible (as opposed to arbitrary) manner. Our approach with respect to the latter involves exhaustively explor-

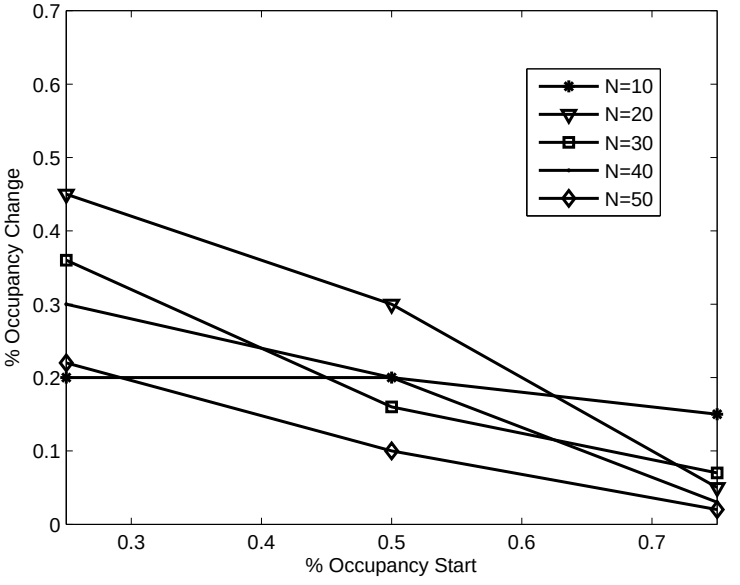
ing the solution spaces on small graphs to see whether there is a systematically reliable route towards good solutions and whether those conclusions generalize to larger graphs.

In a slightly different direction, our experimental study here motivates researching which characteristics of social networks make them more amenable to easy transformation into α-proximity graphs, given that several trials resulted in small occupancy changes.

Given the success of these first ideas, we plan to conduct more rigorous experimental evaluation, using larger datasets, real-life networks, and power law graphs. As part of this, we would analyze the effect on structural properties like clustering coefficient of transforming a graph G into an α-proximity graph G'.



(a) Change in edge occupancy when starting at 25%, as α varies.



(b) Change in edge occupancy with $\alpha = .1$, as starting occupancy varies.

Figure 4.3: Experimental Results

Chapter 5

Conclusions

The challenge that this thesis tries to address is to design a scheme that prevents learning private information in a social network, while still allowing useful data mining. We addressed this challenge through k-anonymization, going through a systematic study of the underlying problem itself, and many of its connected notions and problems. We presented complexity results and algorithms for protecting the privacy of individuals in social networks while allowing users (miners) to mine useful trends and patterns. Our focus was on three main types of social network graphs vertex-labeled, edge-labeled and unlabeled. We show complexity results under all three of these types of graphs, and were able to give algorithms to solve the anonymization conditions as well.

In the first part of the thesis, we considered only using edge-additions to achieve k-anonymity. After introducing some problems in that setting, we were able to show some key complexity results as well as algorithms for some related problems. We were also able to breakdown the complexity of some recently proposed anonymization problems through

a unifying framework called table graphs.

The second part of the thesis dealt with using node additions (vertex additions) to achieve k-anonymity. This method of anonymization was introduced in [7]. Although the problem was shown to be hard for the vertex-labeled case, an efficient algorithm was introduced where the resulting k-anonymous graph is optimal.

In the final part of this work, we explored privacy beyond k-anonymity in the form α-proximity. Following the progression of privacy for table data, the next natural step for social network graph data led to α-proximity. We were able to show hardness for the problem, and in doing so pursuing a greedy algorithm to achieve α-proximity was logical.

The collective work done in this thesis provided a systematic study of k-anonymization from a theoretical standpoint. This is something that was lacking in all the published work to date. We felt that understanding the complexity of the route problems for anonymization would not only help other researchers justify their algorithmic work in the field, but also help the understanding that privacy for graph data is difficult to obtain even for the relaxed of settings. Moreover, looking forward to the future of graph data, we realize the implications of allowing more data to be stored in social networks through edge-label and vertex-labels. Therefore, We believe that it is important to pursue the privacy needs of such social networks.

Future work to further this thesis is plentiful. This work falls into the category of data anonymization. Another path of social network security and privacy is through differential privacy. Recent initial work in differential privacy for graph data shows a promising new avenue to protect the information of social networks. A few recent papers on top conferences has created a new concrete path in this direction [17, 23].

There may also be opportunity to look at different adversarial models with respect to data anonymization. To date, very particular adversary models have been examined with specific information about a node or set of nodes in a graph. It may be interesting to examine adversarial models with more global knowledge of a graph, or even an adversarial model that looks to combine information he can gain from multiple anonymizations of similar data or networks.

Bibliography

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *International Conference on Database Theory, ICDT 2005*, pages 246–258, 2005.

[2] E. Anshelevich and A. Karagiozova. Terminal backup, 3d matching, and covering cubic graphs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC 2007*, pages 391–400, 2007.

[3] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Conference on World Wide Web, WWW 2007*, pages 181–190, 2007.

[4] P. Bonizzoni, G. D. Vedova, and R. Dondi. The *k*-anonymity problem is hard. In *Fundamentals of Computation Theory, FCT 2009*, pages 26–37, 2009.

[5] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2, 2006.

[6] J. Cheng, A. W.-C. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *Special Interest Group on Management of Data, SIGMOD 2010*, pages 459–470, 2010.

[7] S. Chester, B. Kapron, G. Ramesh, G. Srivastava, and S. Venkatesh. k-anonymization of social networks by vertex addition. In *Advances in Databases and Information Systems, ADBIS*, 2011.

[8] S. Chester and G. Srivastava. Social network privacy for attribute disclosure attacks. In *Advances in Social Networks Analysis and Mining, ASONAM*, 2011.

[9] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *Very Large Databases, VLDB J.*, 19(1):115–139, 2010.

[10] C. B. Cornell. Online vs offline social networks, June 2009.

[11] J. Domingo-Ferrer, editor. *Inference Control in Statistical Databases, From Theory to Practice*, volume 2316 of *Lecture Notes in Computer Science*. Springer, 2002.

[12] C. Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006*, pages 1–12, 2006.

[13] C. Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation, 5th International Conference, TAMC*, pages 1–19, 2008.

[14] C. Dwork. Differential privacy in new settings. In *SIAM Symposium on Discrete Algorithms, SODA*, pages 174–183, 2010.

[15] P. Erdős, P. and Kelly. The minimal regular graph containing a given graph. In *American Mathematics Monthly v70*, pages 1074–1075, 1967.

[16] B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Anonymity for continuous data publishing. In *International Conference on Extending Database Technology, EDBT 2008*, pages 264–275, 2008.

[17] J. Gehrke, E. Lui, and R. Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *TCC*, pages 432–449, 2011.

[18] A. Gionis and T. Tassa. k-anonymization with minimal loss of information. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 439–450, 2007.

[19] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proc. Natl. Acad. Sci.*, volume 99, pages 7821–7826, 2002.

[20] J. J. S. González. Extending cell suppression to protect tabular data against several attackers. In *Inference Control in Statistical Databases*, pages 34–58, 2002.

[21] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of Very Large Databases*, 1(1):102–114, 2008.

[22] J. Heer. Prefuse: a toolkit for interactive information visualization. In *In CHI 05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430. ACM Press, 2005.

[23] U. Kang, D. H. Chau, and C. Faloutsos. Mining large graphs: Algorithms, inference, and discoveries. In *ICDE*, pages 243–254, 2011.

[24] B. Kapron, G. Srivastava, and S. Venkatesh. Social network anonymization via edge addition. In *Advances in Social Networks Analysis and Mining, ASONAM*, 2011.

[25] J. Kleinberg and D. Easley. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, first edition, 2010.

[26] D. König. Akademische verlagsgesellschaft. In *Leipzig*, 1936.

[27] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[28] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *International Conference on Data Engineering, ICDE*, pages 106–115, 2007.

[29] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Special Interest Group on Management of Data, SIGMOD 2008*, pages 93–106, 2008.

[30] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. *l*-diversity: Privacy beyond *k*-anonymity. *ACM Transactions on Knowledge Discovery from Data, TKDD*, 1(1), 2007.

[31] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *International Conference on Knowledge Discovery and Data Mining, KDD*, pages 627–636, 2009.

[32] A. Meyerson and R. Williams. General k-anonymization is hard. In *Principles of Database Systems*, 2004.

[33] S. Milgram. The small world problem. In *Psychology Today*, volume 2, pages 60–67, 1967.

[34] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. In *Physical Review E*, volume 74, 2006.

[35] D. A. Robertson and R. Ethier. Cell suppression: Experience and theory. In *Inference Control in Statistical Databases*, pages 8–20, 2002.

[36] B. Thompson and D. Yao. The union-split algorithm and cluster-based anonymization of social networks. In *ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009*, pages 218–227, 2009.

[37] B. K. Tripathy and G. K. Panda. A new approach to manage security against neighborhood attacks in social networks. In *Advances in Social Networks Analysis and Mining, ASONAM 2010*, pages 264–269, 2010.

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

[39] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. k-symmetry model for identity anonymization in social networks. In *International Conference on Extending Database Technology, EDBT 2010*, pages 111–122, 2010.

[40] M. Yuan, L. Chen, and P. S. Yu. Personalized privacy protection in social networks. *Proceedings of Very Large Databases, PVLDB*, 4(2):141–150, 2010.

[41] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, Security, and Trust in KDD, PinKDD 2007*, pages 153–171, 2007.

[42] B. Zhou and J. Pei. The k -anonymity and l -diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl. Inf. Syst.*, 28(1):47–77, 2011.

Appendix A

APPENDIX

A.1 Full Size Graphs

Here we present Full Size Graphs for those presented in Chapter 3 and 4.

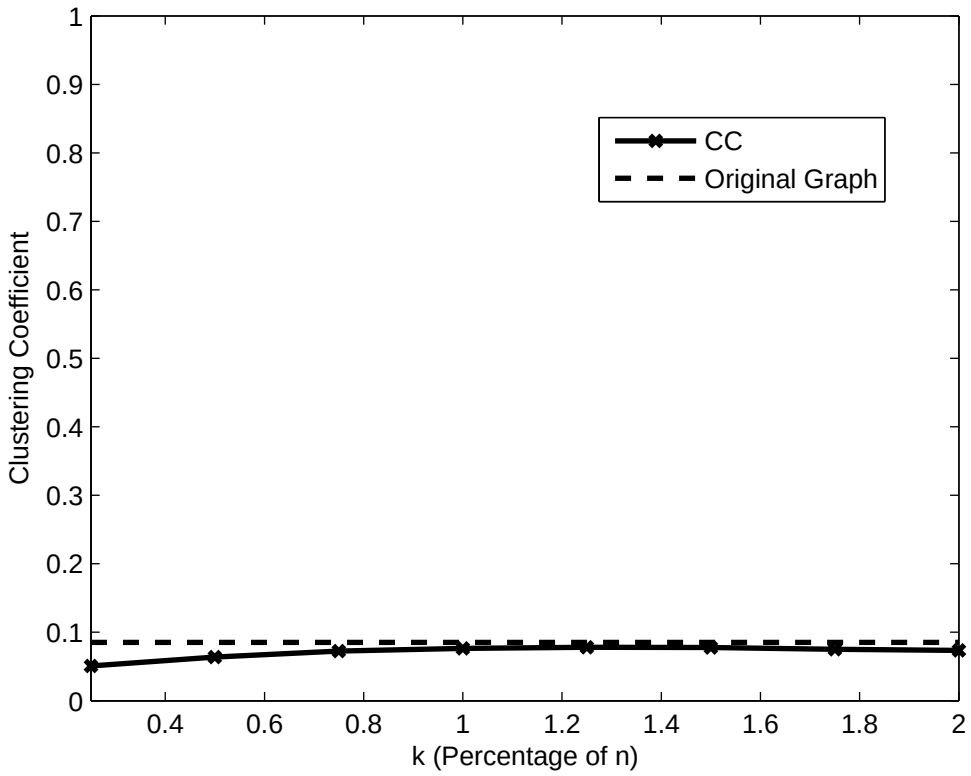


Figure A.1: Enron - CC vs k

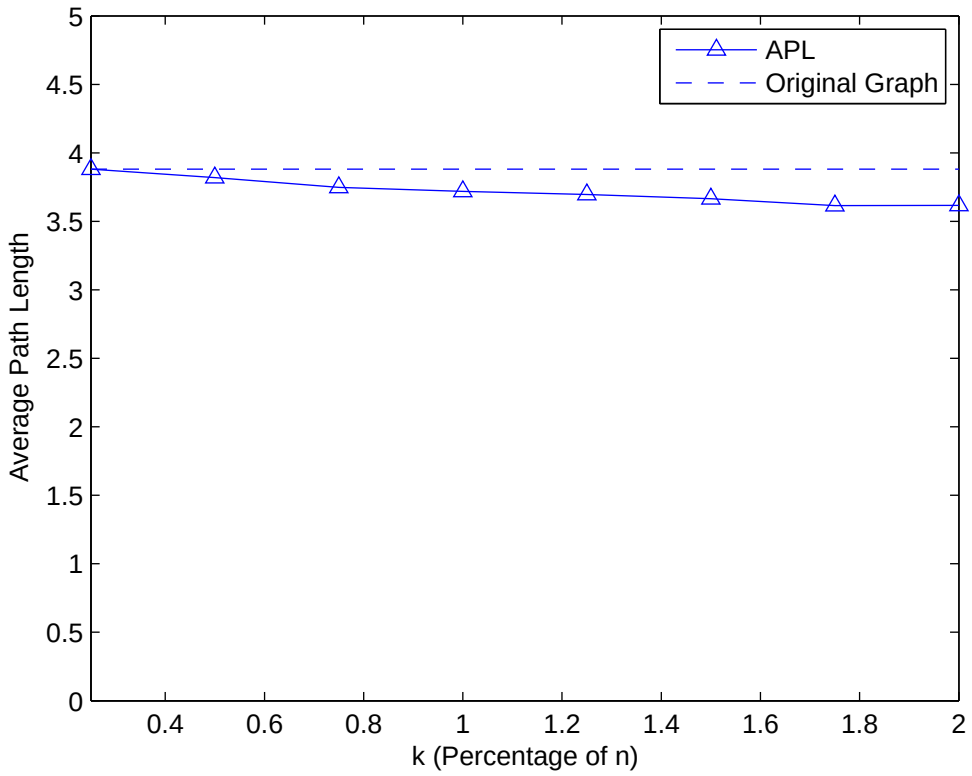


Figure A.2: Enron - APL vs k

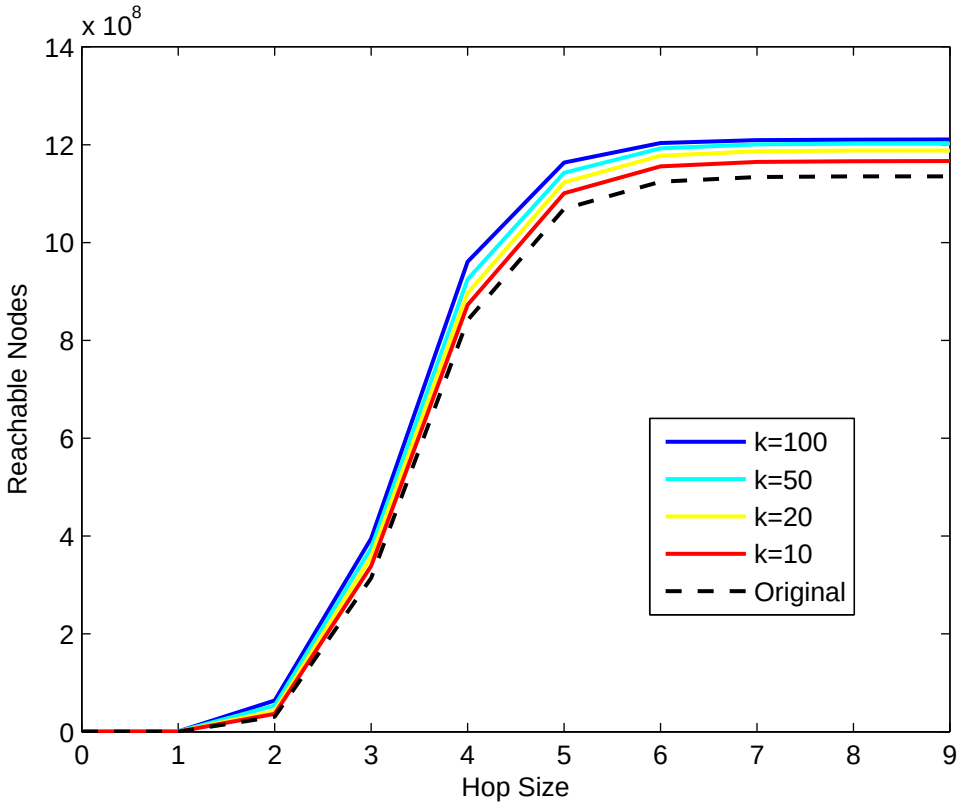


Figure A.3: Enron Hop Plot

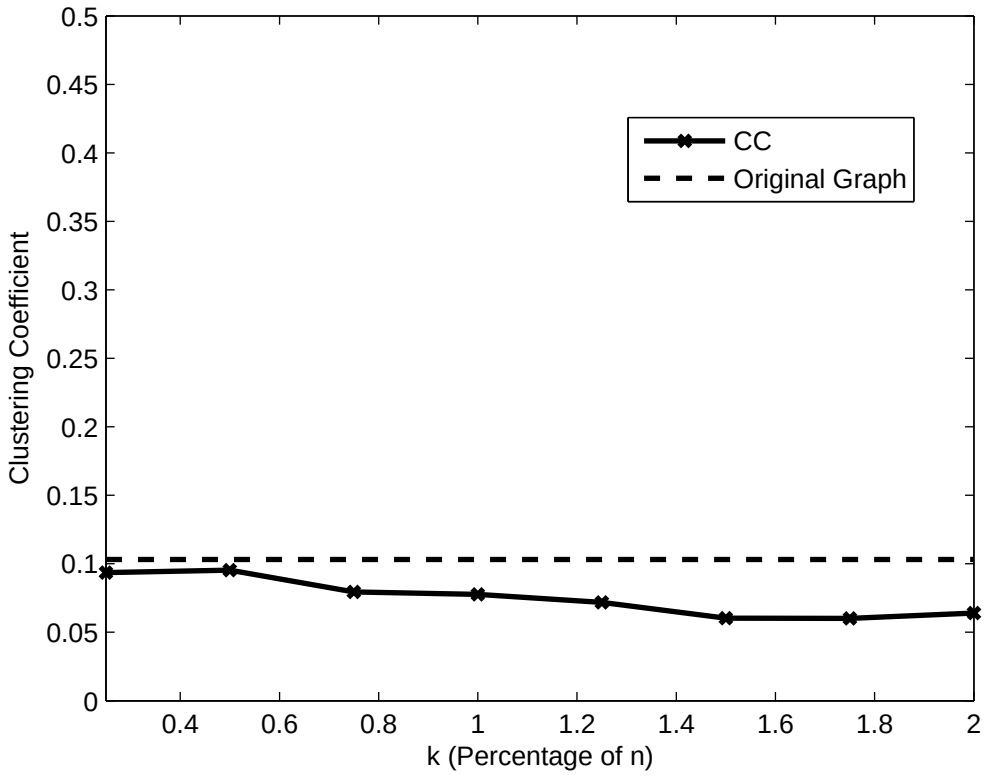


Figure A.4: Power Grid - CC vs k

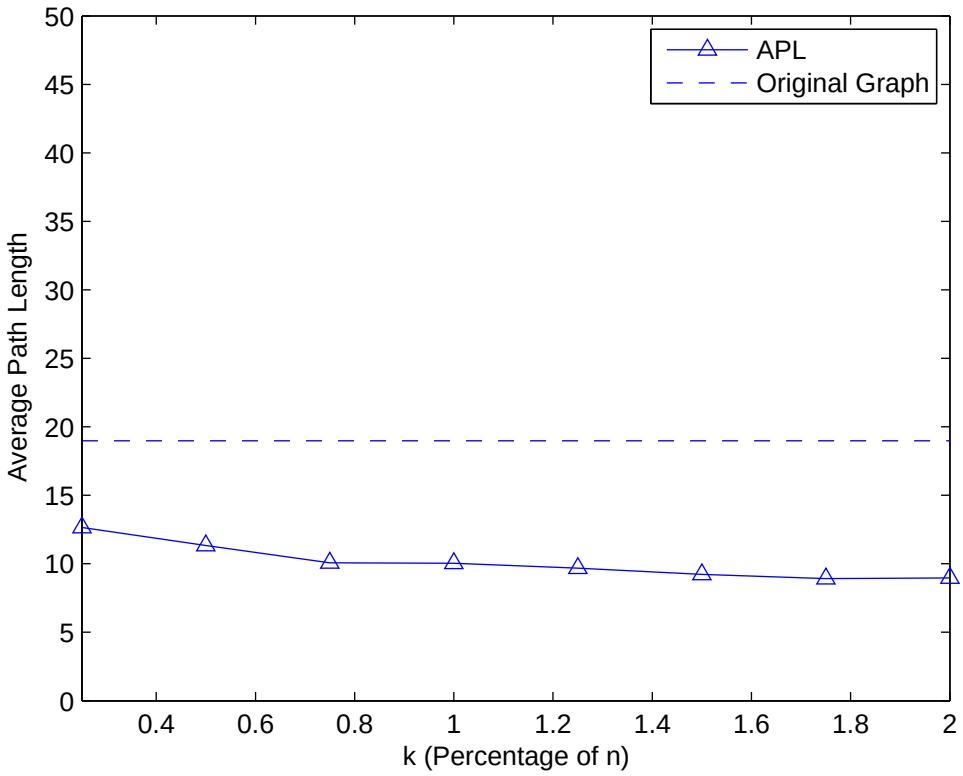


Figure A.5: Power Grid - APL vs k

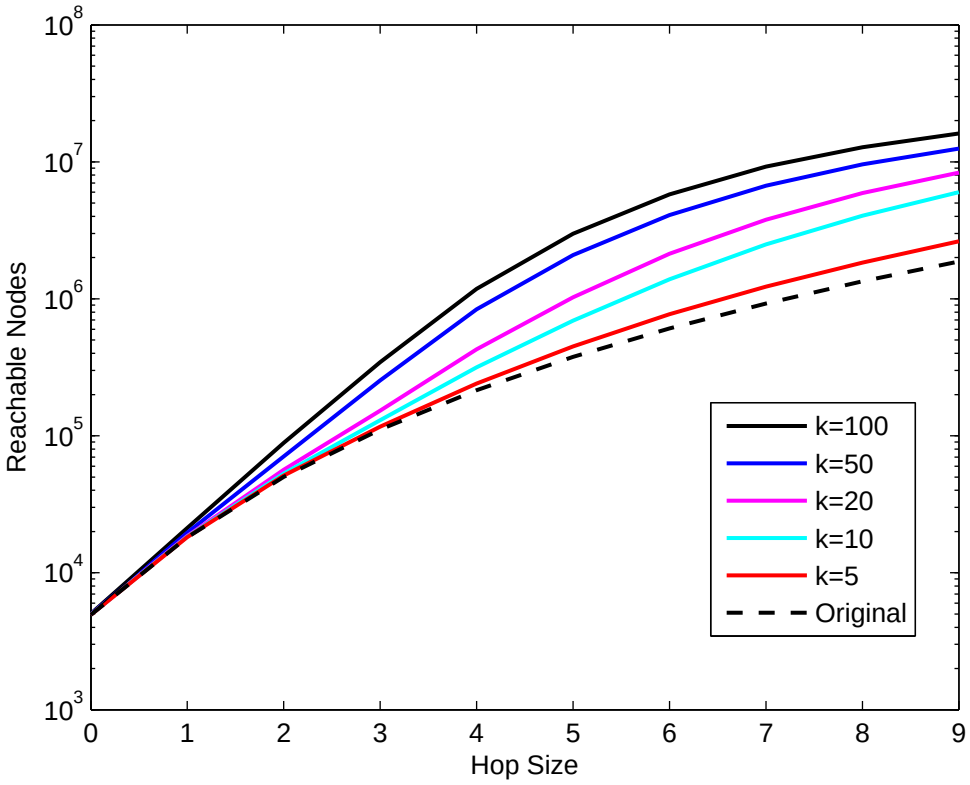


Figure A.6: Power Grid Hop Plot

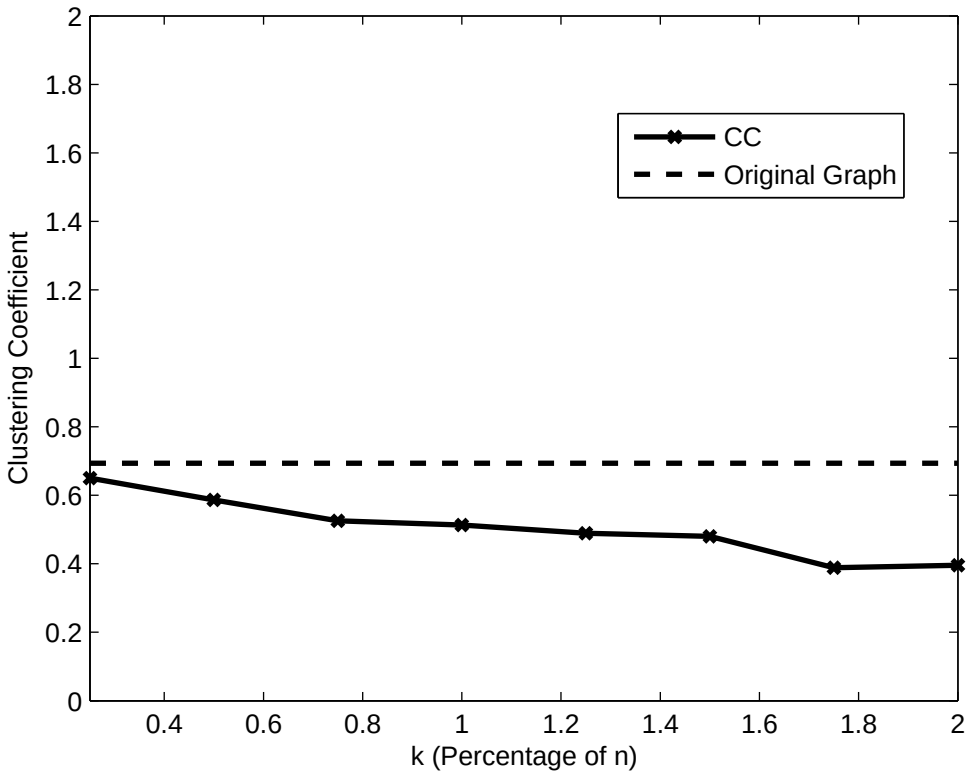


Figure A.7: Net Science - CC vs k

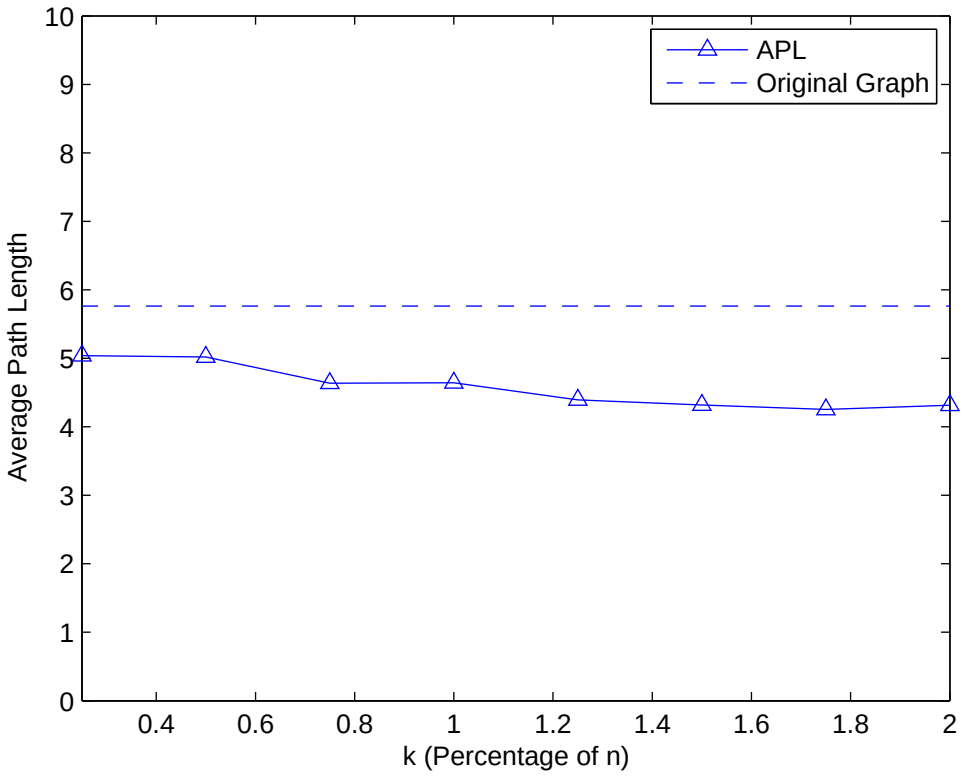


Figure A.8: Net Science - APL vs k

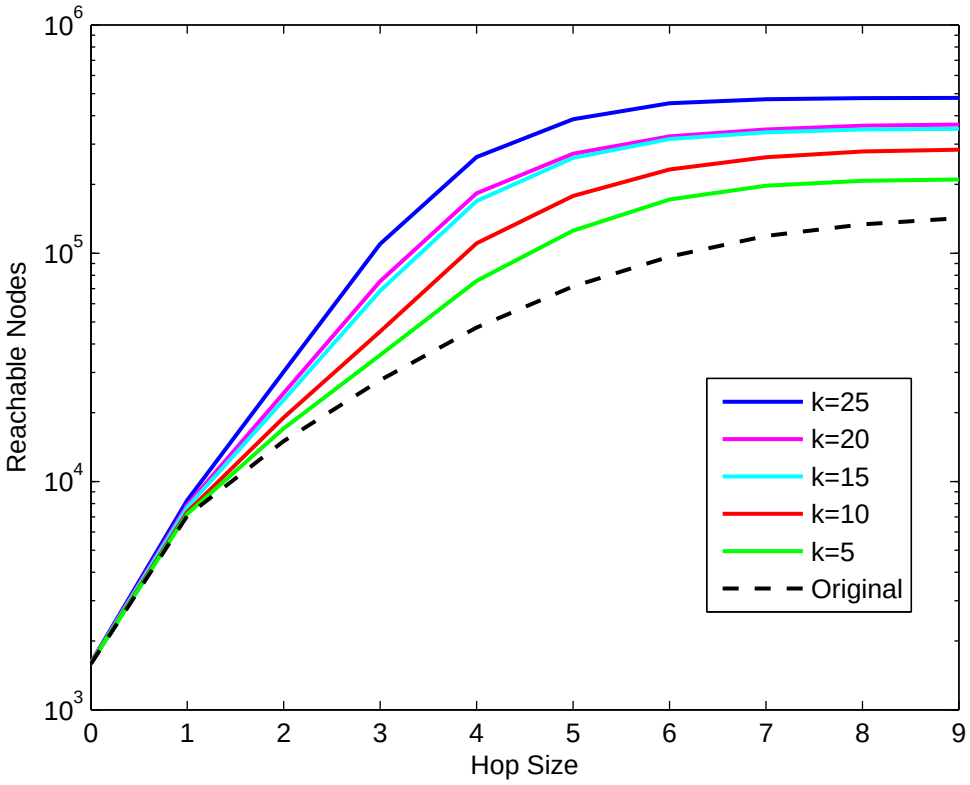


Figure A.9: Net Science Hop Plot

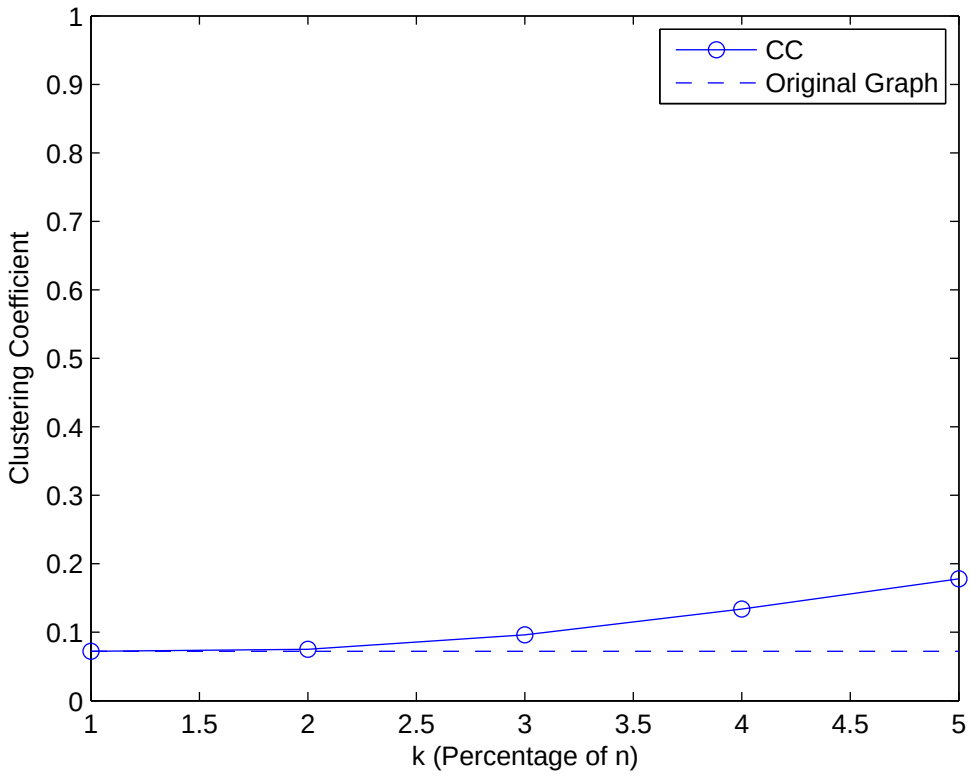


Figure A.10: Prefuse - CC vs k

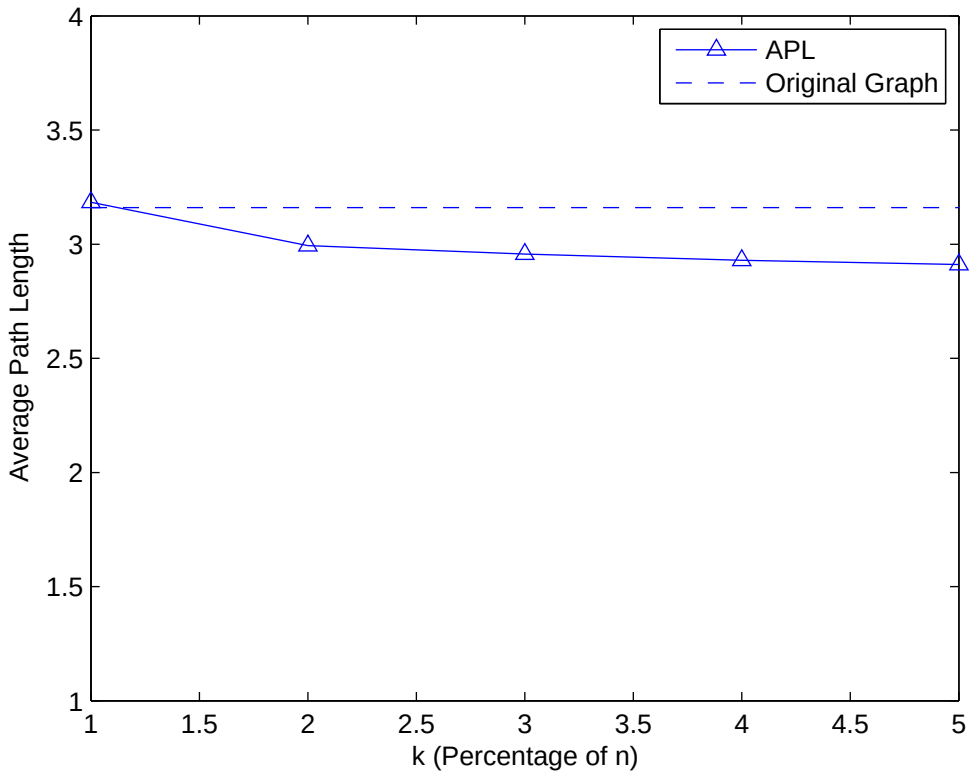


Figure A.11: Prefuse - APL vs k

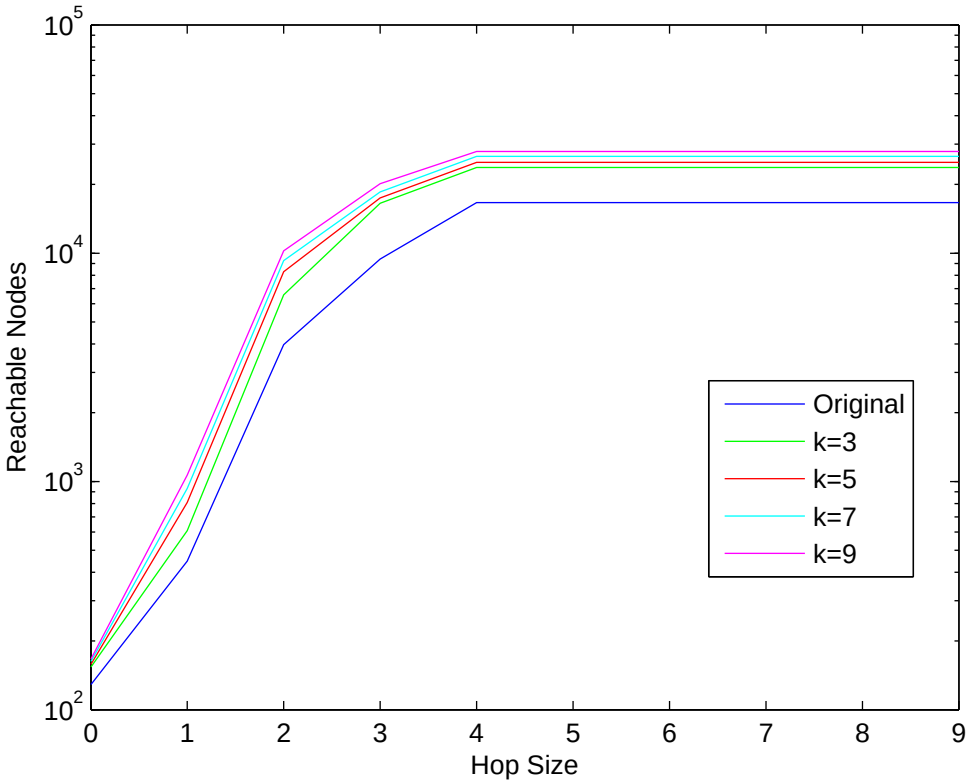


Figure A.12: Prefuse Hop Plot

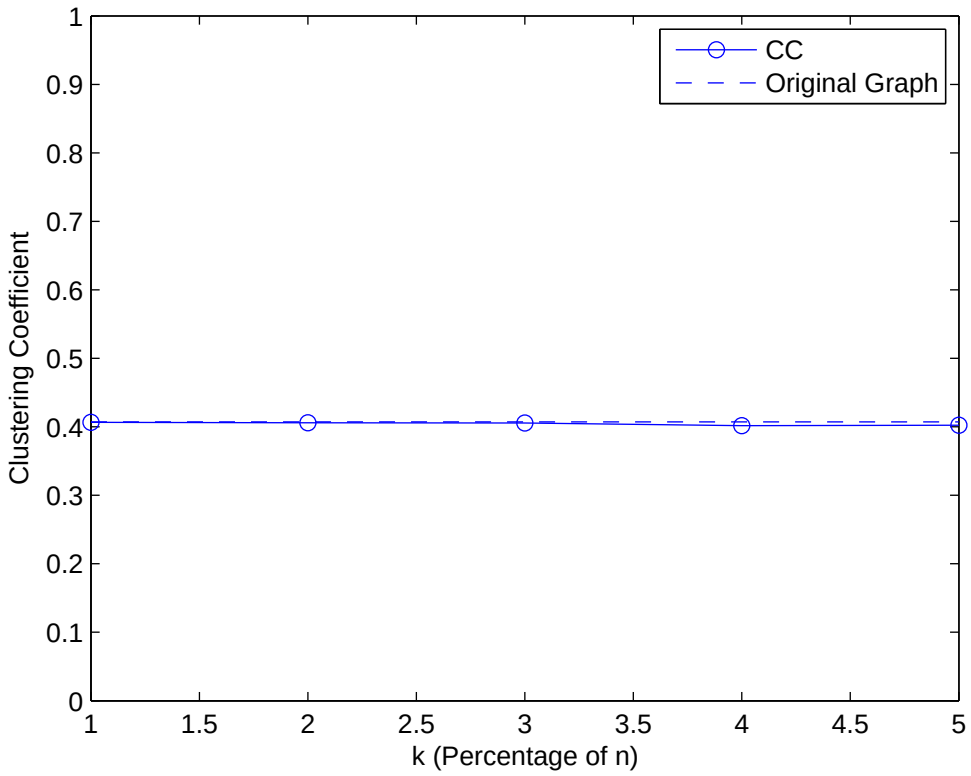


Figure A.13: Football - CC vs k

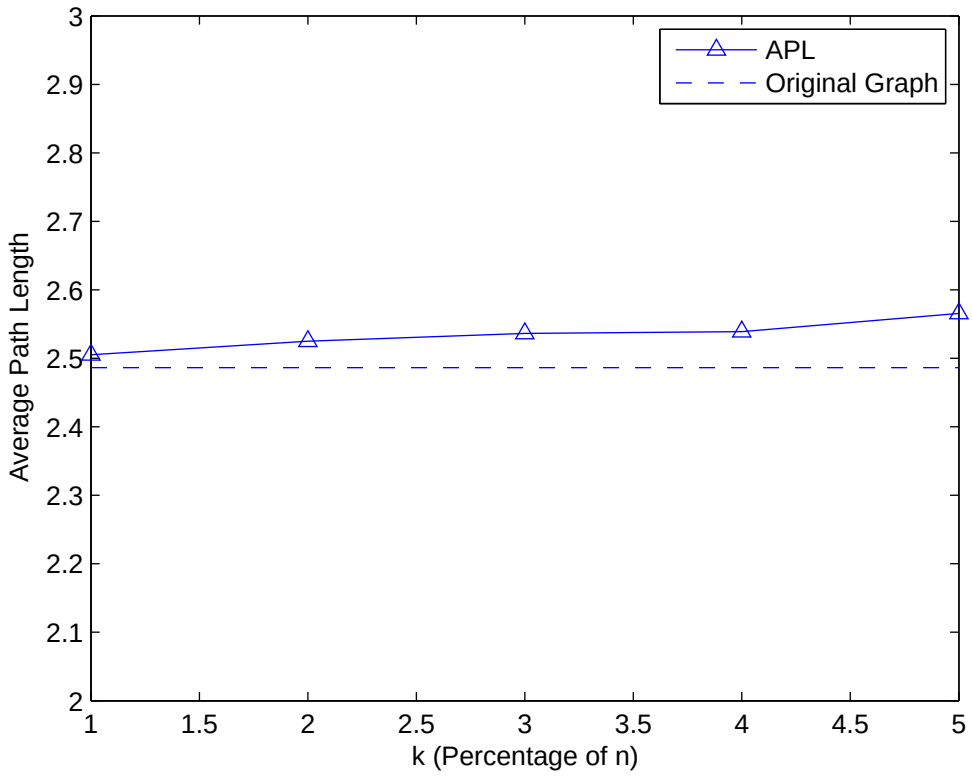


Figure A.14: Football - APL vs k

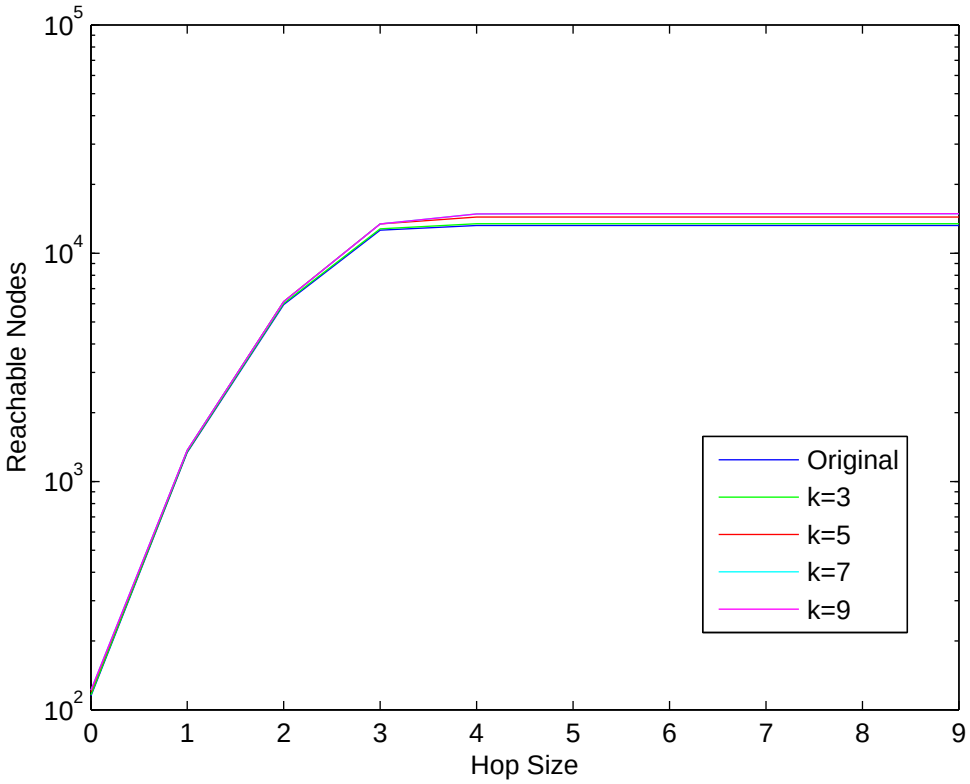


Figure A.15: Football Hop Plot

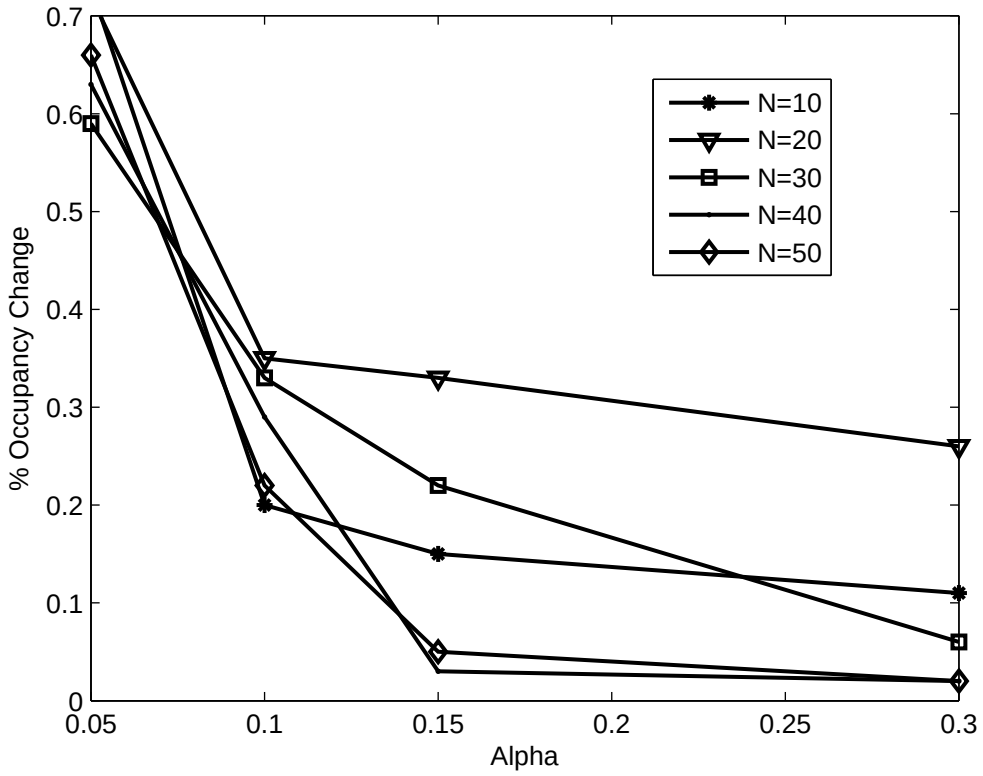


Figure A.16: Change in edge occupancy when starting at 25%, as α varies.

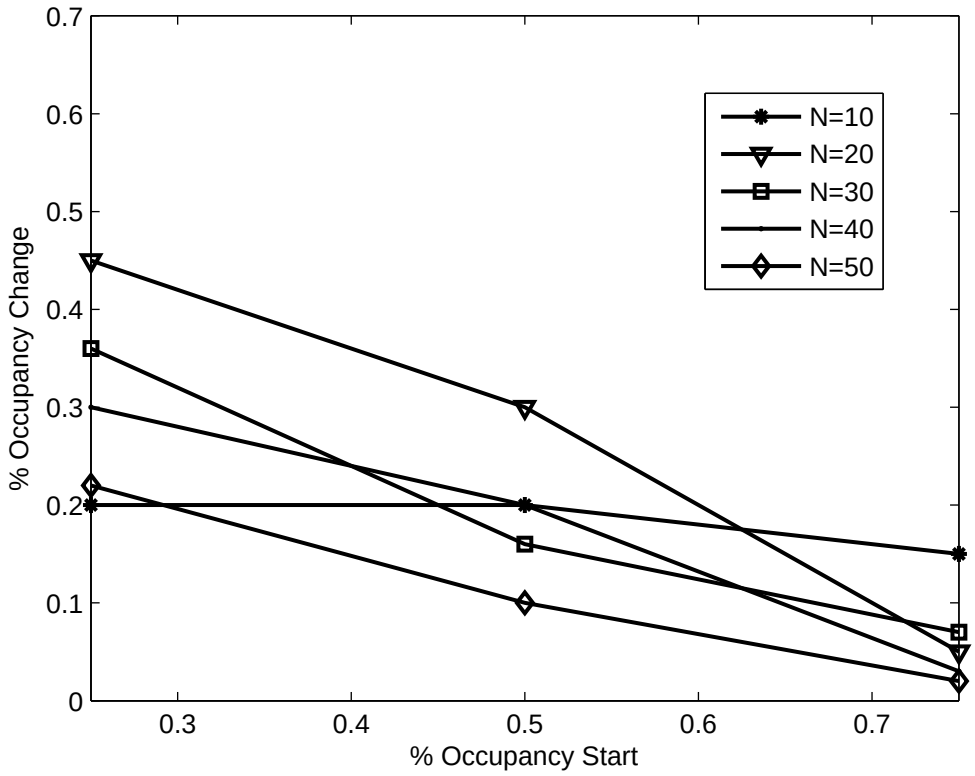


Figure A.17: Change in edge occupancy with $\alpha = .1$, as starting occupancy varies.