Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

PRIVACY-PRESERVING AND DATA UTILITY IN GRAPH MINING

A DISSERTATION SUBMITTED TO UNIVERSITAT AUTÒNOMA DE BARCELONA IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

Author:

Jordi Casas Roma

Advisers:

Dr. Jordi Herrera Joancomartí
Dr. Vicenç Torra i Reventós

Barcelona

September 15, 2014

Acknowledgements

En primer lloc voldria donar les gràcies a les dues persones que han fet possible tot aquest procés; els meus dos directors de tesi, en Jordi Herrera i Vicenç Torra. Sense la seva paciència, els seus consells i la seva direcció no hagués estat possible finalitzar aquesta part en el camí de l'aprenentatge. Moltes gràcies a tots dos!

Però aquest treball tampoc hagués estat possible sense el suport incondicional de familiars i amics. Gràcies a la Glòria, pel seu suport i ànims en tot moment, sobretot quan el camí feia pujada. Una menció molt especial als meus pares; a la meva mare, que sempre m'ha recolzat, escoltat i animat. I al meu pare, que em va transmetre la passió que m'ha permès arribar fins aquí; m'agradaria que avui hi puguessis ser present... La llista seria massa llarga per anomenar a tots amics que m'han acompanyat durant aquest camí, però m'agradaria pensar que ells ja saben qui són.

I cannot forget thanking Professor Michalis Vazirgiannis for hosting me in LIX at École Polytechnique and sharing with me hours of work and very interesting conversations. I would also like to thank all people from DaSciM group, specially François Rousseau and Fragkiskos Malliaros to share amazing discussions. I have learnt a lot from them, and part of this thesis is accomplished thanks to them.

Finalment, una darrera dedicatòria pels meus companys de la Universitat Oberta de Catalunya i del grup de recerca KISON, especialment per en Josep Prieto i en David Megias, que han confiat en mi des del inici i m'han ajudat en tot moment a aconseguir aquest objectiu.

Moltes gràcies a tots!

Jordi Casas Roma

Abstract

In recent years, an explosive increase of graph-formatted data has been made publicly available. Embedded within this data there is private information about users who appear in it. Therefore, data owners must respect the privacy of users before releasing datasets to third parties. In this scenario, anonymization processes become an important concern.

However, anonymization processes usually introduce some kind of noise in the anonymous data, altering the data and also their results on graph mining processes. Generally, the higher the privacy, the larger the noise. Thus, data utility is an important factor to consider in anonymization processes. The necessary trade-off between data privacy and data utility can be reached by using measures and metrics to lead the anonymization process to minimize the information loss, and therefore, to maximize the data utility.

In this thesis we will focus on graph-based privacy-preserving methods, targeting on medium and large networks. Our work will be focused on two main topics: firstly, privacy-preserving methods will be developed for graph-formatted data. We will consider different approaches, from obfuscation to k-anonymity methods. Secondly, the anonymous data utility will be the other key-point in our work, since it is critical to release useful anonymous data. Roughly, we want to achieve results on the anonymous data as close as possible to the ones performed on the original data. These two topics are closely related, and a trade-off between them is necessary to obtain useful anonymous data.

Keywords: Privacy, k-Anonymity, Randomization, Graphs, Social networks, Data utility, Information loss.

Contents

Acknowledgements i

Abstract iii

Index v

List of Figures ix

List of Tables xi

1 Introduction 2

1.1 Motivation . . . 2

1.2 Objectives . . . 3

1.3 Contributions . . . 4

1.4 Document layout . . . 5

I Preliminary concepts 8

2 Graph theory 10

2.1 Graph's basic and notation . . . 10

2.2 Measures and metrics . . . 12

2.3 Graph degeneracy . . . 15

2.4 Edge modification or perturbation methods . . . 18

2.5 Datasets . . . 19

3 Privacy and risk assessment 22

3.1 Problem definition . . . 22

3.2 Privacy breaches . . . 24

3.3 Privacy-preserving methods . . . 24
3.4 Re-identification and risk assessment . . . 33

II Data utility 38

4 Data utility 40
4.1 Related work . . . 40
4.2 Generic information loss measures . . . 42
4.3 Specific information loss measures . . . 43

5 Correlating generic and specific information loss measures 48
5.1 Introduction . . . 48
5.2 Experimental framework . . . 49
5.3 Experimental results . . . 51
5.4 Conclusions . . . 58

6 Improving data utility on edge modification processes 60
6.1 Introduction . . . 60
6.2 Edge Neighbourhood Centrality . . . 61
6.3 Core Number Sequence . . . 68

III Privacy-preserving approaches 80

7 Random-based methods 82
7.1 Randomization using Edge Neighbourhood Centrality . . . 82
7.2 Rand-NC algorithm . . . 83
7.3 Empirical Results . . . 84
7.4 Re-identification and risk assessment . . . 87
7.5 Conclusions . . . 88

8 *k*-Anonymity methods 90
8.1 *k*-degree anonymity model . . . 90
8.2 Evolutionary Algorithm for Graph Anonymization . . . 91
8.3 Univariate Micro-aggregation for Graph Anonymization . . . 97
8.4 Comparing *k*-degree anonymous algorithms . . . 108

8.5 Conclusion 113

IV Conclusions 116

9 Conclusions and further research 118

9.1 Conclusions 118

9.2 Further research 120

Our contributions 122

Bibliography 123

List of Figures

2.1 Illustration of a graph G and its decomposition in k-shells. 17

2.2 Basic operations for edge modification. 18

3.1 Naïve anonymization of a toy network, where G is the original graph, $\widetilde{G}$ is the naïve anonymous version and $\widetilde{G}_{Dan}$ is Dan's 1-neighbourhood. 23

3.2 Vertex refinement queries example (taken from [48]). 36

4.1 Framework for evaluating the clustering-specific information loss measure. . 44

5.1 Experimental framework for testing the correlation between generic and specific information loss measures. 50

6.1 Pairs of edge relevance metrics and network characteristics metrics on Karate network. 65

6.2 Illustration of coreness-preserving edge deletion. Green dashed edges can be safely removed while red solid ones cannot without altering the coreness. 70

6.3 Illustration of coreness-preserving edge addition. Green solid edges can be safely added, while red dashed edges cannot without altering the coreness. 72

6.4 Average distance ($Jazz$) and precision index ($Karate$) values for an anonymization varying from 0% to 25%. 76

7.1 Empirical results of NC-score for each edge and its probability computed by Equation 7.1 on Polbooks network. 84

7.2 Examples of the error evolution computed on our experimental framework. 86

7.3 Candidate set size ($Cand_{\mathcal{H}_1}$) evaluation on our three tested networks. . . . 87

8.1 EAGA experimental results for Zachary's karate club, American football college and Jazz musicians networks. 95

8.2 Basic operations for network modification with vertex invariability. Dashed lines represent deleted edges while solid lines are the added ones. 100
8.3 Neighbourhood centrality values of tested networks. 103
8.4 Degree histogram for tested networks (grey horizontal lines indicate the k values used in our experiments). 106
8.5 Valid switch operation among vertices v_i, v_j, v_k and v_p (dashed lines represent deleted edges while solid lines are the added ones). 110

List of Tables

2.1 Summary of the basic notation. 11

2.2 Networks used in our experiments and their basic properties. 20

5.1 Parameters for information loss measures correlation framework. 51

5.2 Pearson self-correlation value (r) and its associated ρ-value of generic information loss (GIL) measures. 52

5.3 Pearson self-correlation value (r) and its associated ρ-value of precision index. 53

5.4 The Pearson correlation values (r) between clustering precision value and generic information loss measures and its average values μ. An asterisk indicates ρ-values ≥ 0.05, i.e, results not statistically significant. 55

5.5 Results of multivariate regression analysis between clustering precision value and some generic information loss measures. 56

5.6 Pearson correlation averaged values (μ) and standard deviation (σ) for each perturbation method. 57

5.7 Pearson correlation averaged values (μ) and standard deviation (σ) for each dataset. 58

6.1 Pearson correlation values between EB and network characteristic metrics. 64

6.2 Pearson correlation values between LS and network characteristic metrics. . 66

6.3 Pearson correlation values between NC and network characteristic metrics. 66

6.4 Average error for *Crnss Add/Del*, *Rand Add/Del* and *Rand Switch* edge modification processes on 9 generic information loss measures. 75

6.5 Average error for *Crnss Add/Del*, *Rand Add/Del* and *Rand Switch* edge modification processes on 4 clustering-specific information loss measures. . 77

7.1 Results for *Rand-NC* (NC), *Random Perturbation* (RP) and *Random Switch* (RS) algorithms. 85

8.1 Data utility and information loss results for UMGA-R (R) and UMGA-NC (NC) algorithms. 105
8.2 Candidate set size of $\mathcal{H}_1$ ($cand_{\mathcal{H}_1}$) for original tested networks. 107
8.3 UMGA's parameters and execution time results for large tested networks. . 107
8.4 Results for EAGA, UMGA, Liu and Terzi (L&T) and Chester et al. (Chester) algorithms. 112

Chapter 1

Introduction

This first chapter introduces the motivation of our work in Section 1.1, followed by our objectives in Section 1.2. Next, our contributions are presented in Section 1.3, and finally Section 1.4 sketchs the structure of this thesis.

1.1 Motivation

In recent years, an explosive increase of network data has been made publicly available. Embedded within this data there is private information about users who appear in it. Therefore, data owners must respect the privacy of users before releasing datasets to third parties. In this scenario, anonymization processes become an important concern. Among others, the study of Ferri et al. [34] reveals that though some user groups are less concerned by data owners sharing data about them, up to 90% of members in others groups disagree with this principle. Backstrom et al. [4] point out that the simple technique of anonymizing networks by removing the identities of the vertices before publishing the actual network does not always guarantee privacy. They show that there exist adversaries that can infer the identity of the vertices by solving a set of restricted graph isomorphism problems. Some approaches and methods have been imported from anonymization on structured or relational data, but the peculiarities of anonymizing network data avoid these methods to work directly on graph-formatted data. In addition, divide-and-conquer methods do not apply to anonymization of network data due to the fact that registers are not separable, since removing or adding vertices and edges may affect other vertices and edges as well as the properties of the network [94].

In order to overcome this issue, methods that add noise to the original data have been developed to hinder the re-identification processes. But the noise introduced by the anonymization steps may also affect the data, reducing its utility for subsequent data mining processes. Usually, the larger the data modification, the harder the re-identification but also the less the data utility. Therefore, it is necessary to preserve the integrity of the data (in the sense of retaining the information that we want to mine) to ensure that the data mining step is not altered by the anonymization step. Among different measures, for the anonymization step to be considered any useful and thus valid, the analysis performed on the perturbed data should produce results as close as possible to the ones the original data would have led to.

Information loss measures check the quantity of harm inflected to the original data by the anonymization process, that is, it measures the amount of original information that has been lost during the masking process. Information loss measures can be considered for general or specific purposes. Considering an example of a social network, generic information loss would mean that graph properties such as centrality or spectral measures are preserved in the anonymous data. Specific information loss measures refer to a particular data analysis. For instance, clusters of nodes (i.e. users) are considered in case of clustering-specific information loss measures. On the other hand, re-identification (also known as disclosure risk) evaluates the privacy of the respondents against possible malicious uses that attackers could do with the released information. Normally, these measures are computed in several scenarios where the attacker has partial knowledge of the original data. A trade-off between data privacy and data utility must be reached, and this will be our main purpose of this thesis.

1.2 Objectives

In this work we will focus in two main topics. Firstly, we will focus our work on data utility and information loss. It is an important concern to provide high quality data in privacy-preserving processes. Secondly, we will also aim on privacy-preserving methods. Considering different methodologies and strategies, we want to provide strong privacy methods for medium and large networks. We detail our objectives as the following ones:

- Analyse the information loss measures in order to provide a framework to quantify data utility and information loss occurred during the anonymization process.

- Provide a clear set of measures and metrics to estimate the data utility on real graph-mining processes (for instance, on clustering and community detection processes).

- Develop new methods and algorithms to achieve privacy on medium and large networks. These methods have to provide the necessary privacy level while keeping the data utility as close as possible to the original data. The necessary trade-off between data privacy and data utility is one of our main scopes.

1.3 Contributions

This work contributes in two independent but related lines of research. As we have mentioned previously, one refers to the data utility and information loss. Our contributions on this area are the following ones:

- We study different generic information loss measures for graphs comparing such measures to the clustering-specific ones. We want to evaluate whether the generic information loss measures are indicative of the usefulness of the data for subsequent data mining processes. The study and comparison is performed in Chapter 5 and it was presented in [8a].

- We provide two metrics designed to reduce the information loss and improve the data utility on anonymization processes in Chapter 6.

  - The first one, shown in Section 6.2, quantifies the edge's relevance and can help us to preserve the most important edges in the graph. This metric leads an edge modification process to remove the less important edges, keeping the basic network structural and spectral properties. The results were presented in [5a].

  - The second one is based on the idea of *coreness-preserving* edge modification. Instead of modifying any edge at random, we propose only do it when it does not change the core numbers of both its endpoints and by extension the whole core number sequence of a graph. By doing so, we preserve better the underlying graph structure, retaining more data utility. This work is shown in Section 6.3 and it was presented in [10a].

The other line of research concerns methods and algorithms for privacy-preserving on graph-formatted data. Our contributions here are the following ones:

- We introduce a random-based algorithm, which uses the concept of *neighbourhood centrality* to obfuscate the graph structure and lead the process to a lower information loss and better data utility. This is described in Chapter 7 and the results of this work were presented in [6a].

- Different methods for privacy-preserving based on k-anonymity are proposed in Chapter 8.

  - The first one is an evolutionary algorithm which creates a k-degree anonymous version of the original graph. Section 8.2 provides the details of this algorithm, which was presented in [1a] and [3a].
  - The second algorithm is presented in Section 8.3. It uses the univariate micro-aggregation concept to achieve k-degree anonymity on large, undirected and unlabelled graphs. It was presented in [4a]. An extension of the algorithm was presented in [9a]. In this work we use the concept of *neighbourhood centrality* to reduce the information loss and improve the data utility on anonymous data.
  - Finally, in Section 8.4 we introduce a comparison among some k-degree anonymous algorithms. This work was presented in [7a].

1.4 Document layout

This thesis is organized in three parts. The first one covers the preliminary concepts. Chapter 2 describes some basic graph theory, measures and metrics for graphs, concepts related to graph degeneracy, edge modification or perturbation, and the summary of the datasets used in this work. An state of the art is summarized in Chapter 3, where we review the most important privacy breaches, the methods developed to assure the privacy in different scenarios, and some basic attacks as well as models to quantify the disclosure risk on anonymous data.

The second part focuses on data utility. In Chapter 4 we define the framework for computing the information loss on anonymization processes. We consider the generic information loss measures but also the clustering-specific ones. Next, in Chapter 5 we study the correlation between the generic and the clustering-specific information loss measures. Lastly, two measures for reducing the information loss and improving the data utility are proposed in Chapter 6.

Then, the third part of this thesis stresses on methods and algorithms for privacy-preserving on networks. A random-based algorithm is introduced in Chapter 7. Nonetheless, an important part of our work is related to k-anonymity methods. In Chapter 8 we present two different approaches, one based on evolutionary algorithms and the other based on univariate micro-aggregation, to achieve the k-anonymity model on graph-formatted data. We also compare and summarize our methods with other well-know methods, demonstrating that our methods improve the others in terms of data utility and information loss.

Finally, we present the conclusions and our future research in Chapter 9.

Part I

Preliminary concepts

Chapter 2

Graph theory

This chapter introduces the basic concepts from graph theory used in the rest of the work. In Section 2.1 we define the basic types of graphs and their main features, as well as the notation used in the thesis. Next, in Section 2.2 we introduce the structural and spectral graph metrics used in our experiments. Concepts related to graph degeneracy are discussed in Section 2.3 and edge modification or perturbation methods are detailed in Section 2.4. Lastly, Section 2.5 presents the datasets used in our tests and their main properties.

2.1 Graph's basic and notation

A graph is a pair $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices (also called nodes) and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of links (also called edges or arcs), where $e_i = (v_i, v_j) : v_i, v_j \in V$. An undirected link $\{v_i, v_j\} \in E$ connects v_i to v_j and vice versa. They are called *edges* and the graph is *undirected*. On the contrary, a directed link $(v_i, v_j) \in A$ connects v_i to v_j but not vice versa. They are called *arcs* and the graph $G = (V, A)$ is *directed* (also called *digraph*). We define $n = |V|$ to denote the number of vertices and $m = |E|$ to denote the number of edges or $m = |A|$ to denote the number of arcs. Finally, an edge or arc connecting the same vertex, i.e, $\{v_i, v_i\}$ or (v_i, v_i), is called *loop*.

The set of 1-neighbourhood of vertex v_i is denoted as $\Gamma(v_i) = \{v_j : \{v_i, v_j\} \in E\}$ for undirected networks. On directed networks we use $\Gamma(v_i)^{-1} = \{v_j : (v_j, v_i) \in A\}$ to define the set of predecessor vertices and $\Gamma(v_i) = \{v_j : (v_i, v_j) \in A\}$ for the set of successor vertices. In the context of undirected networks, we can define the *degree* of a vertex v_i

Symbol	Meaning
$G = (V, E)$	Undirected graph
$V = \{v_1, v_2, \dots, v_n\}$	Set of vertices
$n = \lvert V \rvert$	The number of vertices
$E = \{e_1, e_2, \dots, v_m\}$	The set of edges (undirected links)
$\{v_i, v_j\}$	An edge (undirected link connecting v_i to v_j and vice versa)
$G = (V, A)$	Directed graph (digraph)
$A = \{a_1, a_2, \dots, a_m\}$	The set of arcs (directed links)
(v_i, v_j)	An arc (directed link connecting v_i to v_j, but not vice versa)
$m = \lvert E \rvert$ or $m = \lvert A \rvert$	The number of edges or arcs
$deg(v_i)$	Degree value of node v_i
$deg_{in}(v_i)$ and $deg_{out}(v_i)$	In-degree and out-degree values of vertex v_i
$\overline{deg}(G)$	Average degree of graph G
$deg^{max}(G)$	Maximum degree of graph G
$\Gamma(v_i)$ and $\Gamma^{-1}(v_i)$	Set of successor and predecessor vertices to vertex v_i
$d(v_i, v_j)$	Distance from vertex v_i to v_j
$\widetilde{G} = (\widetilde{V}, \widetilde{E})$ or $\widetilde{G} = (\widetilde{V}, \widetilde{A})$	Perturbed undirected or directed graph

Table 2.1: Summary of the basic notation.

in a graph G as the number of adjacent nodes (or edges), denoted by $deg_G(v_i)$ or simply $deg(v_i)$ in the case of the whole graph (and not a subgraph). Note that $deg(v_i) = |\Gamma(v_i)|$ when loops are not allowed. On directed networks, the *in-degree* of vertex v_i is the number of its predecessors and it is denoted by $deg_{in}(v_i) = |\Gamma(v_i)^{-1}|$, while the *out-degree* is the number of direct successors and it is denoted by $deg_{out}(v_i) = |\Gamma(v_i)|$. Moreover, we denote the average degree of the network as $\overline{deg}(G) = \frac{2m}{n}$ and the maximum degree as $deg^{max}(G) = max(deg(v_i)) : v_i \in V$. We also use $d(v_i, v_j)$ to denote the length of the shortest path from v_i to v_j, meaning the number of edges (or arcs) along the path. A briefly survey of notation can be found in Table 2.1.

A bipartite (or k-partite) network is a special graph where there are two types (or k types) of entities, and an association only exists between two entities of different types. They are defined as $G = (V, W, E)$ where V and W are the sets of two types of vertices and E is the set of edges $E \subseteq V \times W$. A *multi-entity* network (also called *hyper-graph*) allows relations between more than two vertices. It is denoted by $G(V, I, E)$, where I is the entity set, each entity $i \in I$ is an interaction between/among a subset of entities in V, and E is the set of hyper-edges: for $v \in V$ and $i \in I$, an edge $\{v, i\} \in E$ represents vertex v participates in interaction i. A *multi-graph* is a graph where repeated edges or arcs are allowed. Finally, a *pseudo-graph* is a multi-graph with loops. Furthermore, all types of graph can contain labels or weights in the edges or in the vertices. They are called *edge-labelled* or *edge-weighted* graphs and *vertex-labelled* or *vertex-weighted* graphs.

We define a *simple graph* as an undirected and unlabelled graph without loops. We use $G = (V, E)$ and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ to indicate the original and a perturbed graph, respectively.

2.2 Measures and metrics

In our experiments we use several graph measures based on structural and spectral properties. In the rest of this section we review these measures. Vertex metrics evaluate each vertex independently, giving a score for each vertex, while network metrics evaluate the graph in its whole through a global score.

2.2.1 Vertex metrics

Vertex *betweenness centrality* (C_B) measures the fraction of shortest paths that go through each vertex. This measure indicates the centrality of a vertex based on the flow between other vertices in the network. A vertex with a high value indicates that this vertex is part of many shortest paths in the network, which will be a key vertex in the network structure. Formally, we define the betweenness centrality of v_i as:

$$C_B(v_i) = \frac{1}{n^2} \sum_{\substack{s,t=1 \\ s \neq t}}^{n} \frac{g_{st}^i}{g_{st}} \tag{2.1}$$

where g_{st}^i is the number of shortest paths from v_s to v_t that pass through v_i, and g_{st} is the total number of shortest paths from v_s to v_t.

The second centrality measure is *closeness centrality* (C_C), which is described as the inverse of the average distance to all accessible vertices. Closeness is an inverse measure of centrality in which a larger value indicates a less central vertex, while a smaller value indicates a more central vertex. We define the closeness centrality of v_i as:

$$C_C(v_i) = \frac{n}{\sum_{j=1}^{n} d(v_i, v_j)} \tag{2.2}$$

And the last centrality measure is *degree centrality* (C_D), which evaluates the centrality of each vertex associated with its degree, i.e, the fraction of vertices connected to it. A higher value indicates greater centrality in the graph. The degree centrality of node v_i is defined in Equation 2.3.

$$C_D(v_i) = \frac{deg(v_i)}{m} \tag{2.3}$$

Clustering coefficient (C) is a measure widely used in the literature. The clustering of each vertex is the fraction of possible triangles that exist. For each node the clustering coefficient is defined by:

$$C(v_i) = \frac{2tri(v_i)}{deg(v_i)(deg(v_i) - 1)} \tag{2.4}$$

where $tri(v_i)$ is the number of triangles surrounding vertex v_i.

Sub-graph centrality (SC) is used to quantify the centrality of vertex v_i based on the sub-graphs. Formally:

$$SC(v_i) = \sum_{k=0}^{\infty} \frac{P_i^k}{k!} \tag{2.5}$$

where P_i^k is the number of paths from v_i to v_i with length k.

2.2.2 Network metrics

Average distance ($\overline{dist}$) (also known as *average path length*) is defined as the average of the distances between each pair of vertices in the graph. It measures the minimum average number of edges between any pair of vertices. Formally, it is defined as:

$$\overline{dist}(G) = \frac{\sum_{i,j=1}^{n} d(v_i, v_j)}{\binom{n}{2}} \tag{2.6}$$

Diameter (D) is defined as the largest minimum distance between two vertices in the graph, as Equation 2.7 shows.

$$D(G) = max(d(v_i, v_j)) \ \forall i \neq j \tag{2.7}$$

Clustering coefficient (C) of a graph is the average clustering coefficient computed for all of its vertices, as defined in Equation 2.8.

$$C(G) = \frac{1}{n} \sum_{i=1}^{n} C(v_i) \tag{2.8}$$

where $C(v_i)$ is the clustering coefficient for v_i.

Harmonic mean of the shortest distance (h) is an evaluation of connectivity, similar to the average distance or average path length. The inverse of the harmonic mean of the shortest distance is also known as the global efficiency, and it is computed by Equation 2.9.

$$h(G) = \left(\frac{1}{n(n-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{1}{d(v_i, v_j)} \right)^{-1} \tag{2.9}$$

Modularity (Q) indicates the goodness of the community structure. It is defined as the fraction of all edges that lie within communities minus the expected value of the same quantity in a network in which the vertices have the same degree, but edges are placed at random without regard for the communities.

Transitivity (T) is one type of clustering coefficient, which measures and characterizes the presence of local loops near a vertex. It measures the percentage of paths of length 2 which are also triangles. Possible triangles are identified by the number of triads (two edges with a shared node), as we can see in Equation 2.10.

$$T(G) = \frac{3 \times (\text{number of triangles})}{(\text{number of triads})} \tag{2.10}$$

Sub-graph centrality (SC) is computed as the average score of sub-graph centrality in each vertex of the graph. Formally:

$$SC(G) = \frac{1}{n} \sum_{i=1}^{n} SC(v_i) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{\infty} \frac{P_i^k}{k!} \tag{2.11}$$

2.2.3 Spectral measures

We also use two important eigenvalues of the graph spectrum, which are closely related to many network characteristics [88]. The first one is the *largest eigenvalue* (λ_1) *of the adjacency matrix* A, where λ_i are the eigenvalues of A and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. The maximum degree, chromatic number, clique number, and extend of branching in a connected graph are all related to λ_1. The spectral decomposition of A is:

$$A = \sum_{i=1}^{n} \lambda_i e_i e_i^T \tag{2.12}$$

where e_i is the eigenvector corresponding to λ_i eigenvalue.

The second one is the *second smallest eigenvalue (μ_2) of the Laplacian matrix L*, where μ_i are the eigenvalues of L and $0 = \mu_1 \leq \mu_2 \leq \ldots \leq \mu_m \leq m$. The eigenvalues of L encode information about the tree structure of G. μ_2 is an important eigenvalue of the Laplacian matrix and can be used to show how good the communities separate, with smaller values corresponding to better community structures. The Laplacian matrix is defined as:

$$L = D - A \tag{2.13}$$

where $D_{n \times n}$ is a diagonal matrix with row-sums of A along the diagonal, and 0's elsewhere.

2.2.4 Other metrics

The number of nodes, edges and average degree are not considered as parameters to assess anonymization process, since most of the anonymization methods analysed in this work keep these values constant.

2.3 Graph degeneracy

The concept of *degeneracy* for a graph was first introduced by Seidman in [75] along with a description of its use as a *graph decomposition* technique.

Let k be an integer. A subgraph $\mathcal{H}_k = (V', E')$, induced by the subset of vertices $V' \subseteq V$ (and a fortiori by the subset of edges $E' \subseteq E$), is called a *k-core* if and only if $\forall v_i \in V'$, $deg_{\mathcal{H}_k}(v_i) \geq k$ and $\mathcal{H}_k$ is the maximal subgraph with this property, i.e, it cannot be augmented without losing this property. In other words, the k-core of a graph corresponds to the maximal connected subgraph whose vertices are at least of degree k within the subgraph.

From the k-core, we can then define the notion of *k-shell*, which corresponds to the subgraph induced by the set of vertices that belong to the k-core but not the $(k+1)$-core [17], denoted by $\mathcal{S}_k$ such that $\mathcal{S}_k = \{v_i \in G,\ v_i \in \mathcal{H}_k \wedge v_i \notin \mathcal{H}_{k+1}\}$.

The *core number* of a vertex v_i is the highest order of a core that contains this vertex, denoted by $core(v_i)$. It is also referred as the *shell index* since the k-shell is exactly the part of the k-core that will not survive in the $(k+1)$-core. We claim that it represents the *true degree* of a node as opposed to its apparent degree. Basically, its value corresponds to how cohesive one's neighbourhood is and is a measure of user engagement in a network

[63]. Indeed, to belong to a k-core, a node needs at least k neighbours also meeting the same requirements, thus forming a community of “close” nodes. Again, in the case of a social network, the core number of a node would correspond to the number of close friends the user has, his inner circle that would collapse if he were to leave (through the *cascading effect* implied by the k-core condition – see the impact of the removal of node D in the 3-shell of Figure 2.1 for instance).

Goltsev *et al.* in [39] defined the *k-corona* as the subgraph induced by the set of vertices from the k-shell with exactly k neighbours, denoted by $\mathcal{C}_k$ hereinafter. Nodes from the k-shell with more than k neighbours belong to $\mathcal{S}_k \setminus \mathcal{C}_k$. We introduce the notion of *effective degree* as the degree of a node v_i in the last core it belongs to, denoted by $deg^{eff}(v_i)$ such that $\forall v_i \in \mathcal{S}_k,\ deg^{eff}(v_i) = deg_{\mathcal{H}_k}(v_i)$. It follows that $\mathcal{C}_k = \{v_i \in \mathcal{S}_k,\ core(v_i) = deg^{eff}(v_i)\}$.

We define the *coreness* (also known as *core number sequence* or *shell index sequence*) as the sequence of each vertex core number by analogy with the degree sequence – this is the terminology also adopted by the `igraph` library[1].

The core of maximum order is called the *main core* and its core number is denoted by $core(G)$, which is also referred as the *degeneracy* of the graph. The set of all the k-cores of a graph (from the 0-core to the main core) forms the so-called *k-core decomposition* of the graph. Similarly, we can define the *k-shell decomposition* as the set of all k-shells.

Figure 2.1 illustrates the decomposition of a given graph G of 34 vertices and 36 edges into disjoints shells and nested cores of order 0, 1, 2 and 3. Colour indicates the shell a vertex belongs to: white for the 0-shell, light gray for the 1-shell, dark gray for the 2-shell and black for the 3-shell (main core). In this example, $core(A) = deg^{eff}(A) = deg(A) = 0$, $core(B) = 1$, $deg^{eff}(B) = deg(B) = 2$, $core(D) = deg^{eff}(D) = 3$, $deg(D) = 6$ and $core(G) = 3$.

From all the concepts defined so far, we can derive the following properties:

$$\forall v_i \in G,\ core(v_i) \leq deg^{eff}(v_i) \leq deg(v_i) \tag{2.14}$$

$$\forall k,\ \mathcal{C}_k \subseteq \mathcal{S}_k \subseteq \mathcal{H}_k \tag{2.15}$$

$$\forall i \neq j,\ \mathcal{S}_i \cap \mathcal{S}_j = \varnothing \tag{2.16}$$

[1] http://igraph.org/python/doc/igraph.GraphBase-class.html

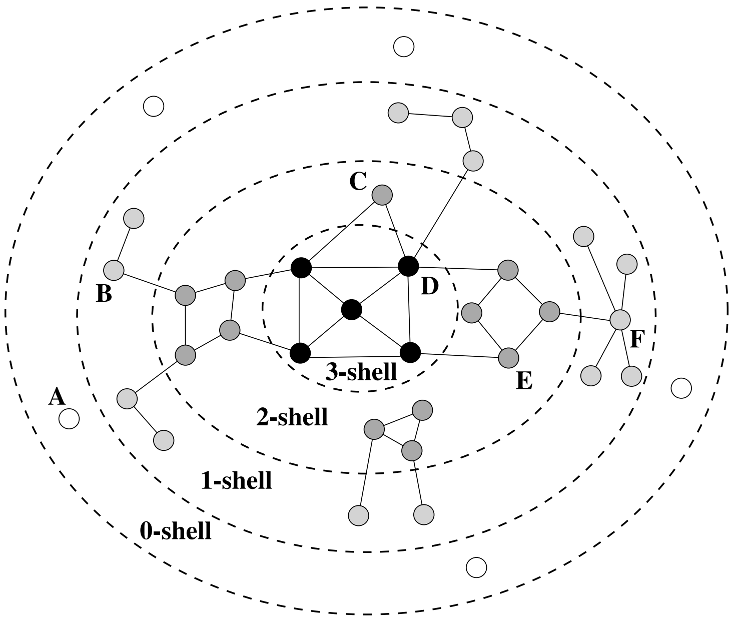


Figure 2.1: Illustration of a graph G and its decomposition in k-shells.

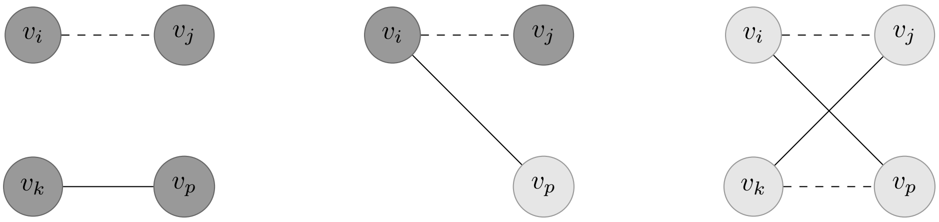
$$\forall j > i,\ \mathcal{H}_j \subseteq \mathcal{H}_i \tag{2.17}$$

Equation 2.14 relates the true, effective and apparent degrees. We also note in particular that neither the shells nor the cores need to form a single connected component (e.g., in Figure 2.1, the 2-shell in dark gray is composed of four disconnected components and the 2-core of only two).

Algorithms and complexity

The brute force approach for computing the coreness of a graph follows immediately from the procedural definition. Each k-core, from the 0-core to the k_{max}-core, can be obtained by iteratively removing all the nodes of degree less than k. Basically, for a given k, while there are nodes that can be removed because they have less than k neighbours then do so and re-check their neighbours (re-checking up to $\mathcal{O}(m)$ nodes overall). This leads to an algorithm with complexity $\mathcal{O}(k_{max}n + m)$ in time and $\mathcal{O}(n)$ space.

Thanks to Batagelj and Zaveršnik in [6], the core number sequence of an unweighted graph can be more efficiently computed in linear time ($\mathcal{O}(n + m)$) and space ($\mathcal{O}(n)$). It immediately follows that the effective degree sequence can also be computed in linear time since for each node, when computing the degree, you only need to consider edges with nodes of equal or higher core number.



(a) Edge add/del (b) Edge rotation (c) Edge switch

Figure 2.2: Basic operations for edge modification.

2.4 Edge modification or perturbation methods

As we will see in Chapter 3, several privacy-preserving methods are based on edge modifications (i.e. adding/removing edges) such as randomization and k-anonymity methods. We define three basic *edge modification* processes to change the network's structure by adding and/or removing edges. These methods are the most basic ones, and they can be combined in order to create complex combinations. We are interested in them since they allow us to model, in a general and conceptual way, most of the privacy-preserving methods based on edge-modification processes. In the following lines we will introduce these basic methods, also called *perturbation methods*, due to the fact that they can model the perturbation introduced in anonymous data during the anonymization process.

There exist three basic edge modifications illustrated in Figure 2.2. Dashed lines represent existing edges to be deleted and solid lines the edges to be added. Node color indicates whether a node changes its degree (dark grey) or not (light grey) after the edge modification has been carried out. These are:

- *Edge add/del* is the most generic edge modification. It simply consists of deleting an existing edge $\{v_i, v_j\} \in E$ and adding a new one $\{v_k, v_p\} \notin E$. Figure 2.2a illustrates this operation. It is usually referred as *Random add/del* or *Rand add/del* when edges are added and deleted at random considering the entire edge set, without restrictions or constraints. Same applies for all three edge modifications.

- *Edge rotation* occurs between three nodes $v_i, v_j, v_p \in V$ such that $\{v_i, v_j\} \in E$ and $\{v_i, v_p\} \notin E$. It is defined as deleting edge $\{v_i, v_j\}$ and creating a new edge $\{v_i, v_p\}$ as Figure 2.2b illustrates. Note that *edge switch* would have been more appropriate but it had already been defined in the relevant literature in the context of a "double switch".

- *Edge switch* occurs between four nodes $v_i, v_j, v_k, v_p \in V$ where $\{v_i, v_j\}, \{v_k, v_p\} \in E$ and $\{v_i, v_p\}, \{v_k, v_j\} \notin E$. It is defined as deleting edges $\{v_i, v_j\}$ and $\{v_k, v_p\}$ and adding new edges $\{v_i, v_p\}$ and $\{v_k, v_j\}$ as Figure 2.2c illustrates.

As aforementioned, most of the anonymization methods rely on one (or more) of these basic edge modification operations. It is true that some of them do not apply edge modification over the entire edge set but this behavior is specific and different for each anonymization method. We believe that this covers the basic behavior of edge-modification-based methods for graph anonymization, even though each method has its specific peculiarities.

For all perturbation methods, the number of nodes and edges remain the same but the degree distribution changes for Edge add/del and Edge rotation while not for Edge switch. Clearly, Edge add/del is the most general concept and all other perturbations can be modelled as a particular case of it: Edge rotation is a sub case of Edge add/del and Edge switch a sub case of Edge rotation.

All random-based anonymization methods are based on the concept of Edge add/del. For example, the Random Perturbation algorithm [48], Spctr Add/Del [88] and Rand Add/Del-B [87] use this concept to anonymize graphs. Most k-anonymity methods can be also modeled through the Edge add/del concept [47, 94, 97]. Edge rotation is a specification of Edge add/del and a generalization of Edge switch: at every edge rotation, one node keeps its degree and the others change theirs. The UMGA algorithm [18] applies this concept to anonymize the graph according to the k-degree anonymity concept. Other methods are related to Edge switch: for instance, Rand Switch and Spctr Switch [88] apply this concept to anonymize a graph. Additionally, Liu and Terzi in [59] also apply this concept to the graph's reconstruction step of their algorithm for k-degree anonymity.

2.5 Datasets

Several real data sets are used in our experiments. Although all these sets are undirected and unlabelled, we have selected them because they have different graph properties. They are the following ones:

- *Zachary's karate club* [91] is a small social graph widely used in clustering and community detection. It shows the relationship among 34 members of a karate club.

Network	$\|V\|$	$\|E\|$	$\overline{deg}$	$\overline{dist}$	D	k
Zachary's karate club	34	78	4.588	2.408	5	1
Polbooks	105	441	8.40	3.078	7	1
American college football	115	613	10.661	2.508	4	1
Jazz musicians	198	2,742	27.697	2.235	6	1
Flickr	954	9,742	20.423	2.776	4	1
URV email	1,133	5,451	9.622	3.606	8	1
Polblogs	1,222	16,714	27.31	2.737	8	1
GrQc	5,242	14,484	5.53	6.048	17	1
Caida	26,475	53,381	4.032	3.875	17	1
Amazon	403,394	2,443,408	6.057	6.427	25	1
Yahoo!	1,878,736	4,079,161	2.171	7.423	26	1

Table 2.2: Networks used in our experiments and their basic properties.

- *US politics book* data (polbooks) [52] is a network of books about US politics published around the 2004 presidential election and sold by the on-line bookseller Amazon. Edges between books represent frequent co-purchasing of books by the same buyers.

- *American college football* [37] is a graph of American football games among Division IA colleges during regular season Fall 2000.

- *Jazz musicians* [38] is a collaboration graph of jazz musicians and their relationship.

- *Flickr* is a sub-graph collected from Flickr OSN. This data has been obtained from [50], where a sampling process has been performed over original data provided by [66]. Nodes represent the users and edges the relationship among them. Although relations are directional in this network, we have eliminated the direction of the edges to get an undirected graph.

- *URV email* [41] is the email communication network at the University Rovira i Virgili in Tarragona (Spain). Nodes are users and each edge represents that at least one email has been sent.

- *Political blogosphere* data (polblogs) [1] compiles the data on the links among US political blogs.

- *GrQc* collaboration network [57] is from the e-print arXiv and covers scientific collaborations between authors papers submitted to General Relativity and Quantum

Cosmology category.

- *Caida* [56] is an undirected network of autonomous systems of the Internet connected with each other from the CAIDA project, collected in 2007.
- *Amazon* [55] is the network of items on Amazon that have been mentioned by Amazon's “People who bought X also bought Y” function. If a product v_i is frequently co-purchased with product v_j, the network contains an edge from v_i to v_j.
- *Yahoo! Instant Messenger friends connectivity graph (version 1.0)* [86] contains a non-random sample of the Yahoo! Messenger friends network from 2003. An edge between two users indicates that at least one user is a contact of the other (the direction of the contact relationship is ignored, producing an undirected graph).

Table 2.2 shows a summary of the datasets' main features, including the number of nodes ($|V|$), number of edges ($|E|$), average degree ($\overline{deg}$), average distance ($\overline{dist}$), diameter (D) and default k-degree anonymity value.

Chapter 3

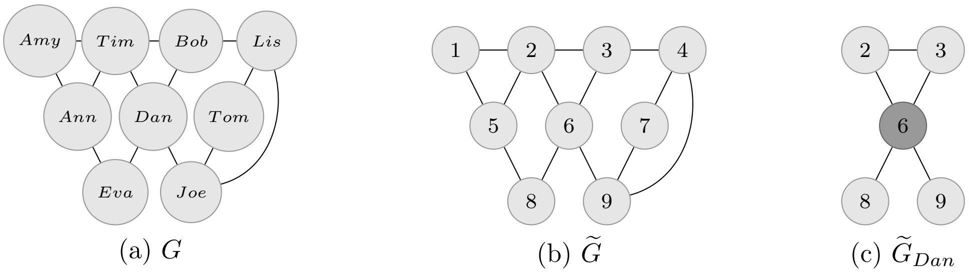
Privacy and risk assessment

This chapter introduces the privacy-preserving (or anonymization) scenario and problem definition on networks in Section 3.1. Next, in Section 3.2 we present the main privacy breaches and then we review the state of the art of privacy-preserving techniques, in Section 3.3. Lastly, we finish this chapter in Section 3.4 discussing the re-identification and risk assessment methods.

3.1 Problem definition

Currently, large amounts of data are being collected on social and other kinds of networks, which often contain personal and private information of users and individuals. Although basic processes are performed on data anonymization, such as removing names or other key identifiers, remaining information can still be sensitive, and useful for an attacker to re-identify users and individuals. To solve this problem, methods which introduce noise to the original data have been developed in order to hinder the subsequent processes of re-identification. A natural strategy for protecting sensitive information is to replace identifying attributes with synthetic identifiers. We refer to this procedure as simple or *naïve anonymization*. This common practice attempts to protect sensitive information by breaking the association between the real-world identity and the sensitive data.

Figure 3.1a shows a toy example of a social network, where each vertex represents an individual and each edge indicates the friendship relation between them. Figure 3.1b presents the same graph after a naïve anonymization process, where vertex identifiers have been removed and the graph structure remains the same. One can think users' privacy is secure, but an attacker can break the privacy and re-identify a user on the anonymous



(a) G (b) $\widetilde{G}$ (c) $\widetilde{G}_{Dan}$

Figure 3.1: Naïve anonymization of a toy network, where G is the original graph, $\widetilde{G}$ is the naïve anonymous version and $\widetilde{G}_{Dan}$ is Dan's 1-neighbourhood.

graph. For instance, if an attacker knows that Dan has four friends and two of them are friends themselves, then he can construct the 1-neighbourhood of Dan, depicted in Figure 3.1c. From this sub-graph, the attacker can uniquely re-identify user Dan on anonymous graph. Consequently, user's privacy has been broken by the attacker.

Two types of attacks have been proposed which show that identity disclosure would occur when it is possible to identify a sub-graph in the released graph in which all the vertex identities are known [4]. In the *active attack* an adversary creates k accounts and links them randomly, then he creates a particular pattern of links to a set of m other users that he is interested to monitor. The goal is to learn whether two of the monitored vertices have links between them. When the data is released, the adversary can efficiently identify the sub-graph of vertices corresponding to his k accounts with high probability. With as few a $k = \mathcal{O}(log(n))$ accounts, an adversary can recover the links between as many as $m = \mathcal{O}(log^2(n))$ vertices in an arbitrary graph of size n. The *passive attack* works in a similar manner. It assumes that the exact time point of the released data snapshot is known, and that there are k colluding users who have a record of what their links were at that time point. Other attacks on naively anonymized network data have been developed, which can re-identify vertices, disclose edges between vertices, or expose properties of vertices (e.g., vertex features). These attacks include: matching attacks, which use external knowledge of vertex features [59] [97] [94]; injection attacks, which alter the network prior to publication [4]; and auxiliary network attacks, which use publicly available networks as an external information source [69]. To solve these problems, methods which introduce noise to the original data have been developed in order to hinder the subsequent processes of re-identification.

3.2 Privacy breaches

Zhou and Pei [94] noticed that to define the problem of privacy preservation in publishing social network data, we need to formulate the following issues: firstly, we need to identify the sensitive information to be preserved. Secondly, we need to model the background knowledge that an adversary may use to attack the privacy. And thirdly, we need to specify the usage of the published social network data so that an anonymization method can try to retain the utility as much as possible while the privacy information is fully preserved.

Regarding to the privacy information to be preserved, three main categories of privacy breaches are pointed out in social networks:

1. *Identity disclosure* occurs when the identity of an individual who is associated with a vertex is revealed.

2. *Link disclosure* occurs when the sensitive relationship between two individuals is disclosed.

3. *Attribute disclosure* which seeks not necessarily to identify a vertex, but to reveal sensitive labels of the vertex. The sensitive data associated with each vertex is compromised.

Identity disclosure and link disclosure apply on all types of networks. However, attribute disclosure only applies on vertex-labelled networks. In addition, link disclosure can be considered a special type of attribute disclosure, since edges can be seen as a vertex attributes. Identity disclosure often leads to attribute disclosure. Identity disclosure occurs when an individual is identified within a dataset, whereas attribute disclosure occurs when sensitive information that an individual wished to keep private is identified. In this thesis, we will focus on identity disclosure.

3.3 Privacy-preserving methods

From a high level view, there are three general families of methods for achieving network data privacy. The first family encompasses "graph modification" approaches. These methods first transform the data by edge or vertex modifications (adding and/or deleting) and then release the perturbed data. The data is thus made available for unconstrained analysis.

The second family encompasses "generalization" or "clustering-based" approaches. These methods can be essentially regarded as grouping vertices and edges into partitions called super-vertices and super-edges. The details about individuals can be hidden properly, but the graph may be shrunk considerably after anonymization, which may not be desirable for analysing local structures. The generalized graph, which contains the link structures among partitions as well as the aggregate description of each partition, can still be used to study macro-properties of the original graph. Even if it holds the properties of the original graph, it does not have the same granularity. More so than with other anonymization algorithms, the generalization method decreases the utility of the anonymous graph in many cases, while increasing anonymity. Hay et al. [47] applied structural generalization approaches using the size of a partition to ensure node anonymity. Zheleva and Getoor [93] focus on the problem of preserving the privacy of sensitive relationships in graph data, i.e, link disclosure. Nerggiz and Clifton [70] presented a methodical approach to evaluate clustering-based k-anonymity algorithms using different metrics and attempting to improve precision by ignoring restrictions on generalisation approaches. Campan and Truta [15] [16] worked on undirected networks with labelled-vertices and unlabelled-edges. The authors developed a new method, called SaNGreeA, designed to anonymize structural information. It clusters vertices into multiple groups and then, a label for each partition is assigned with summary information (for instance, the number of nodes in the partition). Then, Ford et al. [35] presented a new algorithm, based on SaNGreeA, to enforce p-sensitive k-anonymity on social network data based on a greedy clustering approach. He et al. [49] utilized a similar anonymization method that partitions the network in a manner that preserves as much of the structure of the original social network as possible. Cormode et al. [25] studied the anonymization problem on bipartite networks, focusing on the pharmacy example (customers buy products). The association between two nodes (e.g., who bought what products) is considered to be private and needs to be protected while properties of some entities (e.g., product information or customer information) are public. Other interesting works can be found in [49, 8, 77, 76].

Finally, the third family encompasses "privacy-aware computation" methods, which do not release data, but 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. For instance, differential privacy [28] is a well-known privacy-aware computation approach. Differential privacy imposes a guarantee on the data release mechanism rather than on the data itself. It emphasizes that the structure of allowable queries on a statistical database must be designed such that a malicious attacker with the ability to query the database, but without direct access to the full database, cannot determine the unique characteristics of a specific individual. Hence, the goal is to provide statistical information about the data while preserving the privacy of users. Dwork in [29] uses differential privacy to prevent the de-anonymization of social network users while allowing for useful information via queries to the database. McSherry and Mironov [64] apply differential privacy methods to network recommendation systems. Mironov et al. [65] show that the computational power of an adversary should be considered when designing differential privacy systems since this allows for a bounding mechanism that results in systems more appropriate to real-world attacker resources. Other interesting works can be found in [32, 45, 44] and interesting differential privacy summaries in [46, 30, 31].

Some interesting privacy-preserving surveys, including concepts, methods and algorithms from all these three families we have commented above, can be found in [67, 46, 96, 27].

In this thesis we will focus on graph modification approaches, since they allow us to release the entire network for unconstrained analysis, providing the widest range of applications for data mining and knowledge extraction. Graph modification approaches anonymize a graph by modifying (adding and/or deleting) edges or vertices in the graph. These modifications can be made at randomly, and we will refer to them as *randomization*, *random perturbation* or *obfuscation* methods, or in order to fulfil some desired constraints, and we will call them *constrained perturbation* methods.

3.3.1 Random perturbation

These methods are based on adding random noise in original data. They have been well investigated for structured or relational data. Naturally, edge randomization can also be considered as an additive-noise perturbation. Notice that the randomization approaches protect against re-identification in a probabilistic manner.

Naturally, graph randomization techniques can be applied removing some true edges and/or adding some false edges. Two natural edge-based graph perturbation strategies are: firstly, *Rand add/del* method randomly adds one edge followed by deleting another edge. This strategy preserves the number of edges in the original graph. Secondly, *Rand switch* method randomly switches a pair of existing edges $\{v_i, v_j\}$ and $\{v_k, v_p\}$ to $\{v_i, v_p\}$ and $\{v_k, v_j\}$, where $\{v_i, v_p\}$ and $\{v_k, v_j\}$ do not exist in the original graph. This strategy preserves the degree of each vertex and the number of edges.

Hay et al. [48] proposed a method, called *Random perturbation*, to anonymize unlabelled graphs based on randomly removing p edges and then randomly adding p fake edges. The set of vertices does not change and the number of edges is preserved in the anonymous graph. Ying and Wu [88] studied how different randomization methods (based on *Rand add/del* and *Rand switch*) affect the privacy of the relationship between vertices. The authors also developed two algorithms specifically designed to preserve spectral characteristics of the original graph, called *Spctr Add/Del* and *Spctr Switch*. An interesting comparison between a randomization and a constrained-based method, in terms of identity and link disclosure, is presented by Ying et al. [87]. In addition, the authors developed a variation of Random perturbation method, called *Blockwise Random Add/Delete* strategy (or simply *Rand Add/Del-B*). This method divides the graph into blocks according to the degree sequence and implements modifications (by adding and removing edges) on the vertices at high risk of re-identification, not at random over the entire set of vertices.

More recently, Bonchi et al. [11, 12] offered a new information-theoretic perspective on the level of anonymity obtained by random methods. The authors make an essential distinction between image and pre-image anonymity and propose a more accurate quantification, based on entropy, of the anonymity level that is provided by the perturbed graph. They stated that the anonymity level quantified by means of entropy is always greater or equal than the one based on a-posteriori belief probabilities. In addition, the authors introduced a new random-based method, called *Sparsification*, which randomly removes edges, without adding new ones. Their method is compared on three large datasets, showing that randomization techniques for identity obfuscation may achieve meaningful levels of anonymity while still preserving features of the original graph. The extended version of the work in [12] also studied the resilience of obfuscation by random sparsification to adversarial attacks that are based on link prediction. Finally, the authors showed how the randomization method may be applied in a distributed setting, where the network data is distributed among several non-trusting sites, and explain why randomization is far more suitable for such settings than other existing approaches.

A new interesting anonymization approach is presented by Boldi et al. [10] and it is based on injecting uncertainty in social graphs and publishing the resulting uncertain graphs. While existing approaches obfuscate graph data by adding or removing edges entirely, they proposed to use a perturbation that adds or removes edges partially. From a probabilistic perspective, adding a non-existing edge $\{v_i, v_j\}$ corresponds to changing its probability $p(\{v_i, v_j\})$ from 0 to 1, while removing an existing edge corresponds to changing its probability from 1 to 0. In their method, instead of considering only binary edge probabilities, they allow probabilities to take any value in range [0,1]. Therefore, each edge is associated to an specific probability in the uncertain graph.

Other approaches are based on generating new random graphs that share some desired properties with the original ones, and releasing one of this new synthetic graphs. For instance, these methods consider the degree sequence of the vertices or other structural graph characteristics like transitivity or average distance between pairs of vertices as important features which the anonymization process must keep as equal as possible on anonymous graphs. Usually, these methods define $\mathcal{G}_{d,S}$ as the space of networks which: (1) keep the degree sequence d and (2) preserve some properties S within a limited range. Therefore, $\mathcal{G}_{d,S}$ contains all graphs which satisfy both properties. For example, an algorithm is proposed for generating synthetic graphs in $\mathcal{G}_{d,S}$ with equal probability in [89] and a method that generates a graph with high probability to keep properties close to the original ones in [42].

3.3.2 Constrained perturbation

Another widely adopted strategy of graph modification approaches consists on edge addition and deletion to meet some desired constraints. Probably, the k-anonymity is the most well-known model in this group. Even though, other models and extensions have been developed.

k-anonymity

The k-anonymity model was introduced in [74, 81] for the privacy preservation on structured or relational data. Formally, the k-anonymity model is defined as: let $RT(A_1, \dots, A_n)$ be a table and QI_{RT} be the quasi-identifier associated with it. RT is said to satisfy k-anonymity if and only if each sequence of values in $RT\,[QI_{RT}]$ appears with at least k occurrences in $RT\,[QI_{RT}]$. The k-anonymity model indicates that an attacker can not distinguish between different k records although he manages to find a group of quasi-identifiers. Therefore, the attacker can not re-identify an individual with a probability greater than $\frac{1}{k}$.

Some concepts can be used as quasi-identifiers to apply k-anonymity on graph formatted data. A widely option is to use the vertex degree as a quasi-identifier. Accordingly, we assume that the attacker knows the degree of some target vertices. If the attacker identifies a single vertex with the same degree in the anonymous graph, then he has re-identified this vertex. That is, $deg(v_i) \neq deg(v_j)\ \forall j \neq i$. These methods are called *k-degree anonymity*, and they are based on modifying the graph structure (by adding and removing edges) to ensure that all vertices satisfy k-anonymity for their degree. In other words, the main objective is that all vertices have at least $k-1$ other vertices sharing the same degree. Liu and Terzi [59] developed a method that adds and removes edges from the original graph $G=(V,E)$ in order to construct a new anonymous graph $\widetilde{G}=(\widetilde{V},\widetilde{E})$ which is k-degree anonymous, $V=\widetilde{V}$ and $E\cap\widetilde{E}\approx E$.

Instead of using a vertex degree, Zhou and Pei [94] consider the 1-neighbourhood sub-graph of the objective vertices as a quasi-identifier. For a vertex $v_i\in V$, v_i is k-anonymous in G if there are at least $k-1$ other vertices $v_1,\dots,v_{k-1}\in V$ such that $\Gamma(v_i),\Gamma(v_1),\dots,\Gamma(v_{k-1})$ are isomorphic. Then, G is called *k-neighbourhood anonymous* if every vertex is k-anonymous considering the 1-neighbourhood. They proposed a greedy method to generalize vertices labels and add fake edges to achieve k-neighbourhood anonymity. The authors consider the network as a vertex-labelled graph $G=(V,E,L,\mathcal{L})$, where V is the vertex set, $E\subseteq V\times V$ is the edge set, L is the label set and $\mathcal{L}$ is the labelling function $\mathcal{L}:V\rightarrow L$ which assigns labels to vertices. The main objective is to create an anonymous network $\widetilde{G}$ which is k-anonymous, $V=\widetilde{V}$, $E=E\cup\widetilde{E}$, and $\widetilde{G}$ can be used to accurately answer aggregate network queries. More recently, an extended and reviewed version of the paper was presented in [95], demonstrating that the neighbourhood anonymity for vertex-labelled graphs is NP-hard. However, Tripathy and Panda [82] noted that their algorithm could not handle the situations in which an adversary has knowledge about vertices in the second or higher hops of a vertex, in addition to its immediate neighbours. To handle this problem, they proposed a modification to the algorithm to handle such situations.

Other authors modelled more complex adversary knowledge and used them as quasi-identifiers. For instance, Hay et al. [47] proposed a method named *k-candidate anonymity*. In this method, a vertex v_i is k-candidate anonymous with respect to question Q if there are at least $k-1$ other vertices in the graph with the same answer. Formally, $|cand_Q(v_i)|\geq k$ where $cand_Q(v_i)=\{v_j\in V: Q(v_i)=Q(v_j)\}$. A graph is k-candidate anonymous with respect to question Q if all of its vertices are k-candidate with respect to question Q. Zhou et al. [97] and Zhou and Pei [95] consider all structural information about a target vertex as quasi-identifier and propose a new model called *k-automorphism* to anonymize a network and ensure privacy against this attack. They define a k-automorphic graph as follows: (a) if there exist $k-1$ automorphic functions $F_a(a=1,\dots,k-1)$ in G, and (b) for each vertex v_i in G, $F_{a_1}(v_i)\neq F_{a_2}(1\leq a_1\neq a_2\leq k-1)$, then G is called a k-automorphic graph. The key point is determining the automorphic functions. In their work, the authors proposed three methods to develop these functions: graph partitioning, block alignment and edge copy. K-Match algorithm (KM) was developed from these three methods and allows us to generate k-automorphic graphs from the original network.

Chester et al. [20, 21] permit modifications to the vertex set, rather than only to the edge set, and this offers some differences with respect to the utility of the released anonymous graph. The authors only created new edges between fake and real vertices or between fakes vertices. They studied both vertex-labelled and unlabelled graphs. Under the constraint of minimum vertex additions, they show that on vertex-labelled graphs, the problem is NP-complete. For unlabelled graphs, they give a near-linear $\mathcal{O}(nk)$ algorithm.

Stokes and Torra [79] developed methods based on matrix decomposition using the first p eigenvalues of the adjacency matrix or the first p values of the diagonal of the singular value decomposition of the adjacency matrix. A second set of methods use concepts related to graph partitioning. The authors presented two methods for k-anonymity using the Manhattan distance and the 2-path similarity for computing the clusters which group vertices into partitions of k or more elements.

Singh and Zhan presented in [78] a measure, called *topological anonymity*, to quantify the level of obscurity in the structure of a connected network based on vertex degree and clustering coefficient, and they proved that scale-free networks are more resilient to privacy breaches. Related to this work, Nagle et al. in [68] proposed a local anonymization algorithm based on k-degree anonymity that focuses on obscuring structurally important vertices that are not well anonymized, thereby reducing the cost of the overall anonymization procedure.

Lu et al. in [60] proposed a greedy algorithm, called *Fast k-degree anonymization* (FKDA), that anonymizes the original graph by simultaneously adding edges to the original graph and anonymizing its degree sequence. Their algorithm is based on Liu and Terzi's work in [59] and it tries to avoid testing the realizability of the degree sequence, which is a time consuming operation. The authors tested their algorithm and compared to Liu and Terzi's one.

The methods we have presented above works with simple and undirected graphs, but other types of graph are also considered in the literature. Bipartite graphs allows us to represent rich interactions between users on a social network. A rich-interaction graph is defined as $G = (V, I, E)$ where V is the set of users, I is the set of interactions and $E \subseteq V \times I$. All vertices adjacent to specific $i_s \in I$ shares an interaction, i.e, for $v_j \in V : (v_j, i_s) \in E$ all vertices interact on the same i_s. Lan et al. [53] presented an algorithm to meet k-anonymity through automorphism on bipartite networks, called BKM (*Bigraph k-automorphism match*). They discuss information loss, of both descriptive and structural data, through quasi-identifier generalisations using two measures for both data, namely Normalised Generalised Information Loss (NGIL) and Normalised Structure Information Loss (NSIL) respectively. Wu et al. [85] considered the privacy preserving publication against sensitive edges identification attacks in social networks, which are expressed using bipartite graphs. Three principles against sensitive edge identification based on security-grouping theory [76] were presented: positive one-way (c_1,c_2)-security algorithm, negative one-way (c_1,c_2)-security algorithm and two-way (c_1,c_2)-security algorithm. Based on these principles, a clustering bipartite algorithm divides the simple anonymous bipartite graph into n blocks, and then clusters the blocks into m groups which includes at least k blocks, and form the morphic graph with an objective function of the minimum anonymous cost (computed by the difference between original an anonymous vertices and edges).

Edge-labelled networks present specific challenges in terms of privacy and risk disclosure. Kapron et al. [51] used edge addition to achieve anonymization on social networks modelled as an edge-labelled graph, where the aim is to make a pre-specified subset of vertices *k-label sequence anonymous* with the minimum number of edge additions. Here, the label sequence of a vertex is the sequence of labels of edges incident to it. Additionally, Das et al. [26] considered edge weight anonymization in social graphs. Their approach builds a linear programming (LP) model which preserves properties of the graph that are expressible as linear functions of the edge weights. Such properties are related to many graph-theoretic algorithms (shortest paths, k-nearest neighbours, minimum spanning tree and others). LP solvers can then be used to find solutions to the resulting model where the computed solution constitutes the weights in the anonymized network. k-anonymity model is applied to edge weight, so an adversary can not identify an edge with a probability greater than $\frac{1}{k}$ based on edge weight knowledge.

Beyond k-anonymity

Other models that extend the concept of k-anonymity were proposed. For instance, Feder et al. [33] called a graph (k, ℓ)-anonymous if for every vertex in the graph there exist at least k other vertices that share at least ℓ of its neighbours. Given k and ℓ they defined two variants of the graph-anonymization problem that ask for the minimum number of edge additions to be made so that the resulting graph is (k, ℓ)-anonymous. The authors showed that for certain values of k and ℓ the problem is polynomial-time solvable, while for others it is NP-hard. Their algorithm solves optimally the weak $(2, 1)$-anonymization problem in linear time and the strong $(2, 1)$-anonymization problem can be solved in polynomial time. The complexity of minimally obtaining weak and strong $(k, 1)$-anonymous graphs remains open for $k = 3, 4, 5, 6$ while is NP-hard for $(k, 1)$-anonymization problem when $k > 6$.

Stokes and Torra in [80] defined a generalisation of k-anonymity (for both regular and non-regular graphs) by introducing *n-confusion* as a concept to anonymise a database table. They also redefined a (k,ℓ)-anonymous graph and presented various k-anonymization algorithms.

Finally, some authors proposed to anonymize only a subset of vertices, instead of all vertex set. The method is called *k-subset anonymity.* The goal is to anonymize a given subset of nodes, while adding the fewest possible number of edges. Formally, the k-degree-subset anonymity problem is defined as given an input graph $G = (V, E)$ and an anonymizing subset $X \subseteq V$, produce an output graph $\widetilde{G} = (V, E \cup \widetilde{E})$ such that X is k-degree-anonymous and $|\widetilde{E}|$ is minimized. Chester et al. [19] introduced the concept of k-subset-degree anonymity as a generalization of the notion of k-degree-anonimity. Additionally, they presented an algorithm for k-subset-degree anonymity which is based on using the degree constrained sub-graph satisfaction problem. The output of the algorithm is an anonymous version of G where enough edges have been added to ensure that all the vertices in X have the same degree as at least $k - 1$ others.

Attribute disclosure

These aforementioned works all protect against identity disclosure. Regarding attribute disclosure, Machanavajjhala et al. [62] introduced for tabular data the notion of *ℓ-diversity*, wherein each k-anonymous equivalence class requires ℓ different values for each sensitive attribute. In this way, ℓ-diversity looks to not only protect identity disclosure, but was also the first attempt to protect against attribute disclosure. Zhou and Pei [95] adapt the work of Machanavajjhala et al. by defining ℓ-diversity for graphs. They proposed a method to achieve the ℓ-diversity on edge-labelled networks.

However, even l-diversity can experience privacy breaches under the *skewness attack* or *similarity attack* [58]. To address the shortcomings of ℓ-diversity, Li et al. [58] 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. Chester et al. [22] demonstrated that for general, vertex-labelled graphs, the vertex label sequence-based anonymization, and consequently t-closeness, is NP-complete.

More recently, Chester and Srivastava [23] argued that t-closeness cannot be clearly applied to social networks. They proposed a notion of data anonymization called *α-proximity* that protects against attribute disclosure attacks, and provide an algorithm that modifies a vertex-labelled graph by adding new fake edges, so as to ensure it is α-proximal.

Complexity

Kapron et al. [51] analysed privacy issues for arbitrary and bipartite graphs. For arbitrary graphs, they show NP-hardness and use this result to prove NP-hardness for neighbourhood anonymity, i-hop anonymity, and k-symmetry anonymity. For bipartite graphs, they show that k-degree anonymity of unlabelled bipartite graphs is in P for all $k \geq 2$. Moreover, the authors show that *k-label sequence anonymity*, a label sequence containing all the labels of the edges adjacent to it, is in P for $k = 2$ but it is NP-hard for $k \geq 3$ for labelled bipartite graphs.

Chester et al. [22] studied the complexity of anonymizing different kind of networks (labelled, unlabelled and bipartite). For general, edge-labelled graphs, label sequence subset anonymization (and thus table graph anonymization, k-neighbourhood anonymity, i-hop anonymity, and k-symmetry) are NP-complete for $k \geq 3$. For bipartite, edge-labelled graphs, label sequence subset anonymization is in P for $k = 2$ and is NP-complete for $k \geq 3$. For bipartite, unlabelled graphs, degree-based subset anonymization is in P for all values of k. And for general, vertex-labelled graphs, the authors show that vertex label sequence-based anonymization, and consequently t-closeness, is NP-complete.

3.4 Re-identification and risk assessment

Re-identification and risk assessment are important tasks to evaluate the quality of the anonymous data. Determining the knowledge of the adversary is the main problem. Thus, considering a variety of knowledges of the adversary, different methods for assessing the re-identification and risk assessment have been developed.

3.4.1 Re-identification

In cryptanalysis, the authors distinguish between two basic types of attacks, and also it may be an interesting basic classification for network social attacks, although it is also valid for other types of networks:

- **Active attacks**: an adversary actively tries to affect the data to make it easier to decipher or decode. In the case of anonymized social networks, an adversary in an active attack tries to compromise privacy by strategically creating new user accounts and links before the anonymized network is released, so that these new vertices and edges will then be present in the anonymized version.

- **Passive attacks**: an adversary simply observes data as it is presented. In the case of anonymized social networks, passive attacks are carried out by individuals who try to learn the identities of vertices only after the anonymized network has been released.

Backstrom et al. [4] presented both active and passive attacks to anonymized social networks. The first one is *walk-based attack*, in which an adversary chooses an arbitrary set of users whose privacy it wishes to violate, creates a small number of new user accounts with edges to these targeted users, and creates a pattern of links among the new accounts with the goal of making it stand out in the anonymized graph structure. The adversary then efficiently finds these new accounts together with the targeted users in the anonymized network that is released. At a theoretical level, the creation of $\mathcal{O}(\sqrt{log(n)})$ nodes by the attacker in a network with n vertices can begin compromising the privacy of arbitrary targeted vertices, with high probability for any network. The recovery algorithm uses a search over short walks in anonymized network. They find that on a network with 4.4 million vertices, the creation of 7 nodes by an attacker (with degrees comparable to those of typical nodes in the network) can compromise the privacy of roughly 2,400 edge relations on average. The second one is *cut-based attack*, in which users of the system do not create any new vertices or edges, they simply try to find themselves in the released network, and from this to discover the existence of edges among users to whom they are linked. In the same network with 4.4 million of vertices, the authors find that for the vast majority of users, it is possible for them to exchange structural information with a small coalition of their friends, and subsequently uniquely identify the sub-graph on this coalition in the ambient network. Using this, the coalition can then compromise the privacy of edges among pairs of neighbouring nodes.

Ying and Wu [90] designed an attack based on the probability of an edge exists and the similitude between pairs of vertices on anonymous graph. The attack is modelled using matrix operations: $\widetilde{A} = A + E$ where $\widetilde{A}$ and A are the adjacency matrix of anonymous and original graphs, and E is the perturbation matrix, where each position represents:

$$e_{ij} = \begin{cases} 1 & \text{edge } (i,j) \text{ is created} \\ -1 & \text{edge } (i,j) \text{ is deleted} \\ 0 & \text{others} \end{cases}$$

In structured or relation data, some methods allow an attacker to reconstruct the original matrix (A) from the anonymized matrix ($\widetilde{A}$) and some *a priori* knowledge about the perturbation method applied. Some authors are working to apply these methods to network data, but up to now, the results have not been good enough.

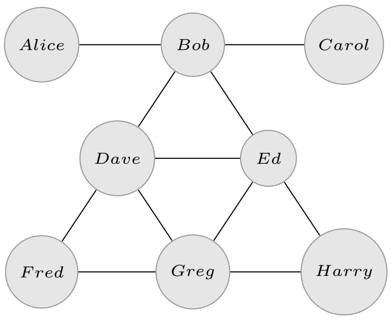
Ying and Wu [90] investigated how well the edge randomization approach via addition/deletion can protect privacy of sensitive links. The authors have conducted theoretical analysis and empirical evaluations to show that vertex proximity measures can be exploited by attackers to enhance the posterior belief and prediction accuracy of the existence of sensitive links among vertices with high similarity values.

Even fulfilling some privacy models, an attacker can succeed on acquiring private information. For instance, a privacy leakage can occur on a k-degree anonymous network and user's privacy information can be revealed to an attacker. For example, we suppose an adversary who wants to know if there is a relation (edge) between users (vertices) v_1 and v_2. The k-degree anonymity model does not allow an attacker to uniquely re-identify each vertex. Instead, he will obtain two sets V_{G1} where $v_i \in V_{G1} \Leftrightarrow deg(v_i) = deg(v_1)$ and V_{G2} where $v_i \in V_{G2} \Leftrightarrow deg(v_i) = deg(v_2)$. If there are edges between each vertex on V_{G1} and each vertex on V_{G2}, an adversary can infer, with absolutely confidence, that a relation exists between vertices v_1 and v_2, although he is not able to re-identify each user in group V_{G1} and V_{G2}. So, even fulfilling the k-degree anonymity model a link disclosure can occur.

3.4.2 Risk assessment: vertex refinement queries

Hay et al. [48, 47] proposed three models of adversary knowledge: *vertex refinement queries*, *sub-graph knowledge queries* and *hub fingerprint centrality*. In this work we will focus on vertex refinement queries.

Vertex refinement queries are used to model the knowledge of the adversary and also to analyse the network in terms of k-anonymity. This class of queries, with increasing attack power, models the local neighbourhood structure of a vertex in the network. The weakest knowledge query, $\mathcal{H}_0(v_j)$, simply returns the label of the vertex v_j. The queries



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

Figure 3.2: Vertex refinement queries example (taken from [48]).

are successively more descriptive: $\mathcal{H}_1(v_j)$ returns the degree of v_j, $\mathcal{H}_2(v_j)$ returns the list of each neighbours' degree, and so on. The queries can be defined iteratively, where $\mathcal{H}_i(v_j)$ returns the multi-set of values which are the result of evaluating $\mathcal{H}_{i-1}$ on the set of vertices adjacent to v_j, as shown in Equation 3.1.

$$\mathcal{H}_i(v_j) = \{\mathcal{H}_{i-1}(v_1), \mathcal{H}_{i-1}(v_2), \ldots, \mathcal{H}_{i-1}(v_m)\} \tag{3.1}$$

where $v_1, v_2, \ldots, v_m$ are the vertices adjacent to v_j.

An equivalence relation on vertices is defined for each query $\mathcal{H}_i$ in the natural way. Formally, two vertices v_i and v_j in a graph are equivalent relative to $\mathcal{H}_i$, denoted $v_i \equiv_{\mathcal{H}_i} v_j$, if and only if $\mathcal{H}_i(v_i) = \mathcal{H}_i(v_j)$. Then, for an attacker with knowledge that can be modelled as a query $\mathcal{H}_i$, all equivalent vertices to $\mathcal{H}_i$ are indistinguishable. Formally if $v_i, v_j \in V$ and $v_i \equiv_{\mathcal{H}_i} v_j$, then $cand_{\mathcal{H}_i}(v_i) = cand_{\mathcal{H}_i}(v_j)$.

A candidate set $cand_{\mathcal{H}_i}$ for a query $\mathcal{H}_i$ is a set of all vertices with the same value of $\mathcal{H}_i$. Thereby, the cardinality of a candidate set for $\mathcal{H}_i$ is the number of indistinguishable vertices under $\mathcal{H}_i$. Note that if the cardinality of the smallest candidate set under $\mathcal{H}_1$ is k, the probability of re-identification is $\frac{1}{k}$. Hence, the k-degree anonymity value is k.

$$cand_{\mathcal{H}_1} = \{v_j \in V : \mathcal{H}_1(v_i) = \mathcal{H}_1(v_j)\} \tag{3.2}$$

Example 3.1 *Figure 3.2 presents a toy example network and its values of $\mathcal{H}_0$, $\mathcal{H}_1$ and $\mathcal{H}_2$. As we can see, query $\mathcal{H}_0$ answers with label ϵ to any vertex, indicating that the network has no vertex labels. Query $\mathcal{H}_1$ indicates the number of adjacent vertices, i.e., the degree, and finally query $\mathcal{H}_2$ answers with the set of adjacent vertices. In the example, Bob, Dave, Ed and Greg are equivalent relative to $\equiv_{\mathcal{H}_1}$, while only Dave $\equiv_{\mathcal{H}_2}$ Ed. Therefore, if an* *adversary has knowledge which can be modelled as $\mathcal{H}_i$, all vertices equivalent relative to $\mathcal{H}_i$ are indistinguishable for him.*

However, the main problem of this approach is that it can not consider adversary's partial information. That is, using this approach an adversary with partial knowledge of the adjacent vertices to a target vertex can not be modelled. Sub-graph knowledge queries has been developed to overcome this limitation.

Part II

Data utility

Chapter 4

Data utility

This chapter introduces data utility and information loss related concepts. After an introduction in Section 4.1, we will proceed to discuss the generic information loss measures in Section 4.2. And lastly, Section 4.3 proposes the specific information loss measures, specifically the clustering-based information loss measures.

4.1 Related work

Usually, authors use some measures from graph theory and compare the values obtained by the original and the anonymous data in order to quantify the noise introduced by the anonymization process. When we quantify the information loss as described above, we talk about generic information loss measures. It is important to emphasize that these generic information loss measures only evaluate structural and spectral changes between original and anonymous data. That is, these measures do not evaluate the data mining processes on anonymous data, and as such, they are general or application-independent. The analysis of specific and application-dependent quality measures is an open problem.

Several authors have been defining and using information loss measures to evaluate the perturbation on anonymous released data, and here we will review some of them. Hay et al. [48] utilized five structural properties from graph theory for quantifying network's structure. For each node, the authors evaluate closeness centrality, betweenness centrality and path length distribution (computed from the shortest path between each pair of nodes). For the graph as a whole, they evaluate the degree distribution and the diameter. The objective is to keep these five measures as close as possible to their original values, assuming that it involves little distortion in the anonymous data. Then, Ying and Wu

[88] and Ying et al. [87] used both real space and spectrum based characteristics to study how the graph is affected by randomization methods. The authors focused on four real space characteristics of the graph and on two important eigenvalues of the graph spectrum. The real space characteristics are: the harmonic mean of the shortest distance, the modularity, the transitivity, and the sub-graph centrality. Since graph spectrum has close relations with many graph characteristics and can provide global measures for some network properties, the authors also considered the following two spectral characteristics: the largest eigenvalue of the adjacency matrix and the second smallest eigenvalue of the Laplacian matrix.

Alternatively, Zou et al. [97] defined a simple method for evaluating information loss on graph privacy-preserving data publishing, which computes the difference between the original and the anonymous edge set, $Cost(G, \widetilde{G}) = (E \cup \widetilde{E}) - (E \cap \widetilde{E})$. Liu and Terzi [59] used the clustering coefficient and average path length for the same purpose. Hay et al. [47] examined five properties commonly measured and reported on network data: degree (distribution of the degrees of all nodes in the graph), path length (distribution of the lengths of the shortest paths between randomly sampled pairs of nodes), clustering coefficient, network resilience (the number of nodes in the largest connected component of the graph when nodes are removed in degree decreasing order) and infectiousness (measured by calculating the proportion of nodes infected by a hypothetical disease, which is simulated by first infecting a randomly chosen node and then transmitting the disease to each neighbour with the specified infection rate).

Truta et al. [83] studied an existing anonymization approach with respect to how it preserves the structural content of the initial social network; specifically, they studied how various graph metrics (centrality measures, radius, diameter, etc.) change between the initial and the anonymous social network, using SaNGreeA, a clustering-based algorithm. Later, Alufaisan and Campan in [2] investigated how well two anonymization methods preserve the importance of nodes in the network, where node importance is expressed by centrality measures. The authors chose a graph-modification approach and a clustering-based approach in their comparison. The clustering-based algorithm is de-anonymized using uniform and scale-free models.

The above measures are all generic and application-independent. In Section 4.2 we describe the generic information loss measures used in this thesis, and we also describe our framework for computing these measures.

There are existing studies that work on graphs trying to maximize some specific task-oriented utility. Budi et al. [13] defined the kb-anonymity model, which combines privacy-preserving using the k-anonymity model and specific task of behaviour-preserving test and debugging data. Lucia et al. [61] improved the model to avoid the probing attack for evolving programs. Both papers considered the anonymization of paths in a program code, which can be represented as a graph, and use a specific utility measurement, which is *test coverage*. The aim is to ensure that the replaced data exhibits the same kind of program behaviour shown by the original data so that the replaced data may still be useful for the purposes of testing and debugging.

The analysis of specific and application-dependent information loss measures is still an open problem. In Section 4.3 we define our proposal for evaluating the clustering-specific information loss measures.

4.2 Generic information loss measures

We will use different generic measures to quantify network's structure in the rest of this thesis. These generic measures are used to compare both the original and the anonymous data to assess the noise introduced in the perturbed data by the anonymization process. These generic measures evaluate some key graph's properties. They evaluate the graph structure, so they are general or, in other words, application-independent. Information loss was defined by the discrepancy between the results obtained between the original and the anonymous data.

The *network measures* evaluate the entire network as a unique score. We compute the error on these metrics as follows:

$$\epsilon_m(G, \widetilde{G}) = |m(G) - m(\widetilde{G})| \tag{4.1}$$

where m is one of the network characteristic metrics, G is the original graph and $\widetilde{G}$ is the perturbed one.

On the contrary, the *vertex measures* evaluate all vertices in the graph, and give us a score value for each vertex. To assess the perturbation introduced in the graph by the anonymization process, we compute the vector of differences for each vertex between the original and the anonymous graph. Then, we compute the root mean square (RMS) to obtain a single value for the whole graph. We calculate the difference of these measures as follows:

$$\epsilon_m(G, \widetilde{G}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(m(v_i) - m(\widetilde{v}_i)\right)^2} \tag{4.2}$$

where $m(v_i)$ is the value of the metric m for the vertex v_i of G and $m(\widetilde{v}_i)$ the value of m for the vertex v_i of $\widetilde{G}$. In our experiments we use Equation 4.2 to compute a value representing the error induced in the whole graph by the anonymization process in the vertex-based measures.

Moreover, we define the divergence on *core number sequence* (or *coreness*) between two graphs using Equation 4.3.

$$Cor(G, \widetilde{G}) = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{4.3}$$

where $C_i = 1$ if $core(v_i) = core(\widetilde{v}_i)$ and 0 otherwise.

And finally, we also propose the *edge intersection* (EI) as a measure for anonymization cost, which is defined as the percentage of original edges which are also in the anonymous graph. Formally:

$$EI(G, \widetilde{G}) = \frac{|E \cap \widetilde{E}|}{max(|E|, |\widetilde{E}|)} \tag{4.4}$$

Although it is a very simple measure, it is useful to quantify the information loss during privacy-preserving data publishing processes.

4.3 Specific information loss measures

We define the specific information loss measures as a task-specific measure for quantifying the data utility and the information loss associated to a data publishing process. In this thesis, we focus on clustering-specific processes, since it is an important application for social and healthcare networks. We want to analyse the utility of the perturbed data by evaluating it on different clustering processes. Like generic graph measures, we compare the results obtained both by the original and the perturbed data in order to quantify the level of noise introduced in the perturbed data. This measure is specific and application-dependent, but it is necessary to test the perturbed data in real graph-mining processes.

We consider the following approach to measure the clustering assessment for a par-

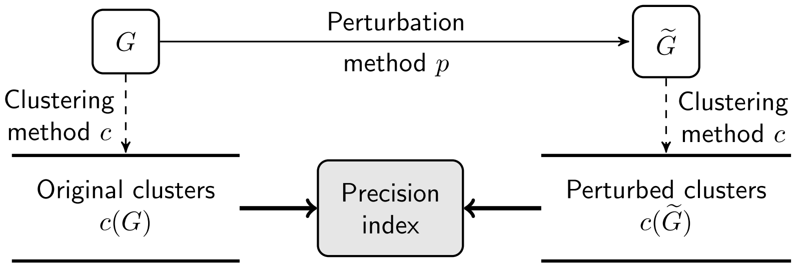


Figure 4.1: Framework for evaluating the clustering-specific information loss measure.

ticular perturbation and clustering method: (1) apply the perturbation method p to the original data G and obtain $\widetilde{G}$; (2) apply a particular clustering method c to G and obtain clusters $c(G)$ and apply the same method to $\widetilde{G}$ to obtain $c(\widetilde{G})$; (3) compare the clusters $c(G)$ to $c(\widetilde{G})$, as shown in Figure 4.1. In relation to information loss, it is clear that the more similar $c(\widetilde{G})$ is to $c(G)$, the less information loss. Thus, clustering specific information loss measures should evaluate the divergence between both sets of clusters $c(G)$ and $c(\widetilde{G})$.

Ideally, results should be the same. That is, the same number of sets with the same elements in each set. In this case, we can say that the anonymization process has not affected the clustering process. When the sets do not match, we should be able to calculate a measure of divergence.

For this purpose, we use the *precision index* [14]. Assuming we know the true communities of a graph, the precision index can be directly used to evaluate the similarity between two cluster assignments. Given a graph of n nodes and q true communities, we assign to nodes the same labels $l_{tc}(\cdot)$ as the community they belong to. In our case, the true communities are the ones assigned on the original dataset (i.e. $c(G)$) since we want to obtain communities as close as the ones we would get on non-anonymized data – we are not interested in the ground truth communities. Assuming the perturbed graph has been divided into clusters (i.e. $c(\widetilde{G})$), then for every cluster, we examine all the nodes within it and assign to them as predicted label $l_{pc}(\cdot)$ the most frequent true label in that cluster (basically the mode). Then, the precision index can be defined as follows:

$$precision\_index(G,\widetilde{G}) = \frac{1}{n}\sum_{v\in G} \nVdash_{l_{tc}(v)=l_{pc}(v)} \tag{4.5}$$

where $\nVdash$ is the indicator function such that $\nVdash_{x=y}$ equals 1 if $x=y$ and 0 otherwise. Note that the precision index is a value in the range [0,1], which takes the value 0 when there is no overlap between the sets and the value 1 when the overlap between the sets is complete.

4.3.1 Clustering methods

Six clustering algorithms are used in our experiments to evaluate the perturbation methods. They have been referred as "particular clustering methods c" in the previous section. Thus, they are used in this work to evaluate the create the clusters on original and anonymous data in order to quantify the clustering-specific information loss. All of them are unsupervised algorithms for graph formatted data based on different concepts and developed for different applications and scopes. An extended revision and comparison of them, among others, can be found in [54, 92]. The selected clustering algorithms are:

- Markov Cluster Algorithm (*MCL*) was developed by S. Van Dongen [84]. The algorithm is based on the simulation of flow in graphs and it is widely used in bioinformatics. It starts by computing an integer power of the diffusion matrix (usually the square), which yields the probability matrix of a random walk after a specific number of steps. This step is called *expansion*. Next, it computes the probability of the walker to be trapped within a community. This step is called *inflation*. The expansion and inflation steps are iterated until it obtains a disconnected tree, where its components are the communities. The inflation parameter controls the granularity of the result sets, and its value is adjusted according to the data of each graph. Its complexity can be lowered to $\mathcal{O}(nk^2)$ if, after each inflation steps, only the k largest elements of the resulting matrix are kept, whereas the others are set to zero.

- Algorithm of Girvan and Newman (*Girvan-Newman* or *GN*) [71] is an important community detection algorithm in graphs. It is a hierarchical divisive algorithm, in which edges are iteratively removed based on the value of their betweenness centrality. The algorithm has a complexity $\mathcal{O}(n^3)$ on a sparse graph.

- Fast greedy modularity optimization (*Fastgreedy*) by Clauset, Newman and Moore [24] is a hierarchical agglomeration algorithm for detecting community structure. Starting from a set of isolated nodes, the edges of the original graph are iteratively added to produce the largest possible increase of the modularity at each step. Its running time on a sparse graph is $\mathcal{O}(n \log^2 n)$.

- *Walktrap* [72] by Pons and Latapy tries to find densely connected sub-graphs, also called communities in a graph via random walks. The idea is that short random walks tend to stay in the same community. They proposed a measure of similarities between nodes based on random walks to capture the community structure in a graph. It runs in time $\mathcal{O}(mn^2)$ and space $\mathcal{O}(n^2)$ in the worst case.

- *Infomap* by Rosvall and Bergstrom [73] use the problem of optimally compressing the information on the structure of the graph to find the best cluster structure. This is achieved by compressing the information of a dynamic process taking place on the graph, namely a random walk. The optimal compression is achieved by optimizing a quality function. Such optimization can be carried out rather quickly with a combination of greedy search and simulated annealing.

- *Multilevel* by Blondel et al. [9] is a multi-step technique based on a local optimization of Newman-Girvan modularity in the neighbourhood of each node. After a partition is identified in this way, communities are replaced by super-nodes, yielding a smaller weighted network. The procedure is then iterated, until modularity does not increase any further. The computational complexity is essentially linear in the number of edges of the graph.

MCL, Walktrap and Infomap are based on the random walk concept, while Girvan-Newman and Fastgreedy are based on hierarchical edge betweenness, and Multilevel is based on modularity concept. Although some algorithms permit overlapping among different clusters, we have not allowed such overlapping in our experiments by setting the parameter to zero. This is because setting overlapping to zero simplifies the method which evaluates the similarity between results.

Chapter 5

Correlating generic and specific information loss measures

Information loss measures have been defined to evaluate the noise introduced in the anonymized data. Generic information loss measures ignore the intended anonymized data use. When data has to be released to third-parties, and there is no control on what kind of analyses users could do, these measures are the standard ones. In this chapter we study different generic information loss measures for graphs, comparing such measures to the clustering-specific ones. After a short introduction in Section 5.1, our experimental framework is presented in Section 5.2. Then, in Section 5.3 we discuss the results to evaluate whether the generic information loss measures are indicative of the usefulness of the data for subsequent data mining processes, and lastly, in Section 5.4 we go into the conclusions.

5.1 Introduction

The anonymization processes should allow the analysis performed into the anonymous data lead to results as equal as possible to the ones obtained when applying the same analysis to the original data. Nevertheless, data modification is in contradiction with data utility. The larger data modification, the less data utility. Thus, a good anonymization method hinders the re-identification process while causing minimal distortion in the data.

Several measures have been designed to evaluate the goodness of the anonymization methods. Generic information loss measures evaluate in what extent the analysis on anonymous data differs from the original data. Each measure focuses on a particular property of the data. We assume that if these metrics show little variation between original and anonymous data, then the subsequent data mining processes will also show little variation between original and anonymous data. However, the behaviour of anonymous data in the subsequent data mining processes may not coincide with the expected results. Since evaluating the distortion introduced in the graph is not enough, it is necessary to assess the noise introduced in the subsequent data mining processes. No analysis has been made to evaluate whether these measures are suitable to accommodate the information loss when data are used to specific purposes. In our work we consider the case of clustering-specific processes.

In this chapter we study different generic information loss measures for graphs, comparing such measures to the clustering-specific ones. We want to evaluate whether the generic information loss measures are indicative of the usefulness of the data for subsequent data mining processes.

5.2 Experimental framework

As we have stated before, some authors evaluate their anonymization methods comparing the results of generic measures between original and anonymous data. They assume that small distortion on these measures involves little distortion on anonymous data utility. Our objective is to evaluate the correlation between generic information loss (GIL) measures and specific information loss (SIL) measures based on clustering processes. Hence, if the GIL indicates that there is little perturbation on anonymous data, then the clustering results on the anonymous data must be close to the results on the original data. Otherwise, the GIL measures used in graph assessment are not representative of real data utility.

To conduct this experiment, we have to test as much anonymization methods as we can. Nevertheless, as we have seen in Section 3.3, there are several anonymization methods and it is hard to analyse all of them. Even so, all graph modification approaches are based on edge modification and can be modelled as an additive-noise perturbation. So, we can model the generic behaviour of these methods through basic edge modification or perturbation. We have defined the basic edge modification or perturbation methods on Section 2.4.

Our experimental framework is shown in Figure 5.1. We have chosen five graph formatted datasets (Zachary's karate club, Jazz musicians, Flickr and URV email), three

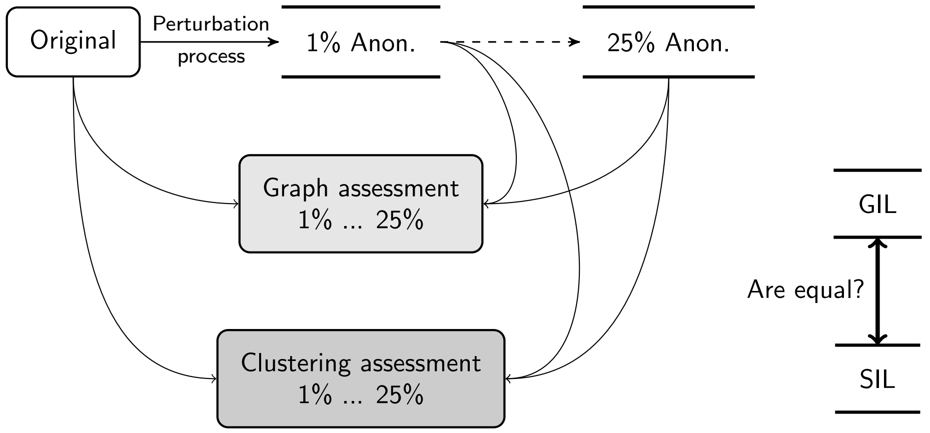


Figure 5.1: Experimental framework for testing the correlation between generic and specific information loss measures.

edge perturbation methods, several generic information loss measures and six graph clustering algorithms. The process is as follows: (1) we apply random noise to graph datasets (network's details are shown in Section 2.5) using edge perturbation methods (Section 2.4). Then, (2) we evaluate original and perturbed data using GIL measures for quantifying network structure (Section 4.2). Next, (3) we apply the clustering processes (Section 4.3.1) both on original and on perturbed data and we use clustering-based specific measures (Section 4.3) to evaluate the results. Lastly, (4) we compare the GIL and SIL results to analyse whether GIL measures are useful to predict the clustering real data utility. If the degree of similarity between them are close, the GIL measures provide correct information about data utility. Otherwise, these measures do not provide correct information about the utility of the anonymous data for clustering.

For each method and dataset, we considered 10 independent executions with a perturbation percentage ranging from 1% to 25% of the total number of edges, increasing the perturbation percentage by 1% at each step. We compute perturbation percentage using the edge difference (ED), which is defined as the percentage of original edges that are not present in the perturbed graph, as shown in Equation 5.1.

$$ED(G,\widetilde{G}) = 1 - \frac{|E \cap \widetilde{E}|}{max(|E|, |\widetilde{E}|)} \tag{5.1}$$

Parameter	Value
Perturbation range	From 1% to 25%
Execs	20
MCL Inflation	1.8

Table 5.1: Parameters for information loss measures correlation framework.

5.3 Experimental results

To compare the generic and the clustering-specific measures, we have computed these measures for pairs of graphs (G, $\widetilde{G}$) using some particular perturbation method p. That is $\widetilde{G} = p(G)$.

In this section, we show the results of our experiments. For each dataset (G) we apply the three perturbation methods, and then, we assess graph measures on the perturbed data. Next, we apply clustering algorithms and compare the original and the perturbed results using the precision index, our cluster-based measure (details can be found in Section 4.3). We refer to specific information loss measure as a result of precision index applied to specific clustering algorithm on original and anonymous data. In other words, we refer to specific information loss measure as a value of $precision\_index(c(G), c(\widetilde{G}))$, where c is one of our clustering methods and $\widetilde{G} = p(G)$ for a particular perturbation method p. From now on, we will use the name of the clustering algorithm to refer to its precision index computed as we have mentioned above.

Perturbation methods have been applied with a percentage of noise. It has been added iteratively with a 1% of the number of edges at each step. The goal is that we can see how the structural properties of the graph evolve. Note that the 0% perturbation values are the original graph results. The values showed by all measures and metrics are averaged values computed from independent executions. In our framework, we have used the Pearson correlation to compute the linear dependence between all measures and metrics, where the ρ-values refer to the observed significance level of a hypothesis test with the *null hypothesis* that correlation is equal to 0.

Parameters used in our experiments are detailed in Table 5.1. “Perturbation range” specifies the percentage of noise introduced, “execs” is the number of independent executions for every experiment and “MCL inflation” parameter controls the granularity of the resulting clusters on Markov Cluster Algorithm (MCL). Others clustering methods are used with default values.

In our experiments we want to address the following questions:

Pearson	$\overline{dist}$	D	C_B	C_C	C_D	EI	C	T	λ_1	μ_2
r	0.846	0.150	0.957	0.902	0.992	0.999	0.974	0.948	0.247	0.097
ρ-value	0	0.007	0	0	0	0	0	0	0	0.006

Table 5.2: Pearson self-correlation value (r) and its associated ρ-value of generic information loss (GIL) measures.

- Do the generic information loss measures and precision index behave in similar way independently of the dataset? In Section 5.3.1 we analyse whether the GIL and SIL measures present similar behaviour over different datasets, i.e, they behave in similar way independently of the specific characteristics of the dataset.

- Are the generic information loss measures correlated with clustering-specific measures? In Section 5.3.2 we compare the GIL versus SIL measures in order to describe the correlation among them.

- What are the effects of various data perturbation strategies? In Section 5.3.3 we comment the correlation's results based on the three perturbation methods presented on Section 2.4.

- What are the differences between measures when various graph datasets are considered? In Section 5.3.4 we discuss the correlation's results for each specific dataset in order to analyse the differences among them.

5.3.1 Analysing measures

Before comparing the generic information loss measures with the precision index, we analyse all measures in order to evaluate whether each measure behaves in similar way to itself on different datasets. If a measure presents a high self-correlation, then it will conduct in a similar way independently of the data where it is applied. Therefore, we analyse the self-correlation of all generic and clustering-specific measures. Self-correlation is measured by comparing the results of specific measure over each dataset to all others. Then, we calculated the Pearson correlation on the resulting set.

Most of the GIL measures show strong self-correlation over all datasets used in our experiments. As we can see in Table 5.2, average distance ($\overline{dist}$), betweenness centrality (C_B), closeness centrality (C_C), degree centrality (C_D), edge intersection (EI), clustering (C), and transitivity (T) present self-correlation values higher than 0.84 with ρ-values equal to 0, which implies statistical significance. These results confirm that the measures

Pearson	*MCL*	*Infomap*	*Multilevel*	*Girvan-Newman*	*Fastgreedy*	*Walktrap*
r	0.287	0.626	0.777	0.828	0.782	0.656
ρ-value	0	0	0	0	0	0

Table 5.3: Pearson self-correlation value (r) and its associated ρ-value of precision index.

evolve in a similar way over all datasets, i.e., the behaviour of the measures is similar independently of the dataset in which they are applied. Diameter (D), the largest eigenvalue of the adjacency matrix (λ_1) and the second smallest eigenvalue of the Laplacian matrix (μ_2) present weak and very weak self-correlation values. It denotes that their behaviour is clearly subordinate to the dataset. It is interesting to underline that diameter cannot be computed on Flickr and URV Email datasets because the perturbation methods generate some isolated nodes, so the number of connected components is greater than one and the diameter of the dataset cannot be computed. Furthermore, on Football dataset the diameter keeps the same value on all perturbed data. That is because Football dataset does not follow the power-law on degree distribution. All nodes have a degree value between 7 and 12, and it is improbable to increase or decrease the diameter value by random edge perturbation. Therefore, the behaviour of the diameter on a perturbation process is very dependent of the dataset.

The precision index values are presented in Table 5.3. As we have mentioned, we use the name of the clustering algorithm to refer to its specific information loss measure computed as $precision\_index(c(G), c(\widetilde{G}))$, where c is one of our clustering methods and $\widetilde{G} = p(G)$ for a particular perturbation method p. Most of them show strong self-correlation values and all of them are statistically significant. Multilevel, Girvan-Newman and Fastgreedy achieve self-correlation values greater than 0.77, while Infomap and Walktrap achieve values close to 0.6, and MCL presents weak self-correlation with a value of 0.28.

5.3.2 Comparing generic and clustering-specific measures

Pearson correlation values for all generic information loss and clustering-specific measures are shown in Table 5.4. These values are computed using all datasets and all perturbation methods. The ρ-values for each correlation are showed within brackets and the last column and row show the average values for each row and column (μ).

The first and the second generic information loss measures which we have analysed are average distance ($\overline{dist}$) and diameter (D). Both measures are related to paths and cannot be calculated for graphs with two or more connected components. Thus, both measures cannot be computed on few perturbed graphs because isolated nodes have appeared during perturbation process. The Pearson correlation index for average distance and the precision for all clustering methods is 0.732. As we can see in Table 5.4, average distance achieves strong correlation values with all clustering methods, except with MCL, where the achieved value is quite lower than others and indicates a moderate correlation between average distance and the precision values of MCL clustering. As we can see, ρ-values are 0 for all experiments, demonstrating that results are statistically significant. On the contrary, diameter does not present correlation since its ρ-values are greater than 0.05 on three experiments, and no statistical significance can be assigned to its correlation value. In addition, the correlation value is 0.128, which is very low.

Two of the three centrality measures show similar behaviour and achieve strong correlation values. The correlation index is 0.753 for betweenness centrality (C_B), 0.848 for closeness centrality (C_C) and 0.422 for degree centrality (C_D). Therefore, betweenness and closeness centrality are strong correlated to clustering-based measures. Betweenness and closeness centrality achieve correlation values higher than 0.831 for Multilevel, Girvan-Newman and Fastgreedy clustering algorithm. Clearly, the Girvan-Newman algorithm are related to betweenness centrality since it uses the edge betweenness values to discover the clusters, and the edge betweenness and the node betweenness are related. Multilevel and Fastgreedy use the concept of modularity, which is also related to node centrality. The other centrality measure, degree centrality (C_D), presents different behaviour and a moderate correlation value of 0.422. Our perturbation methods introduce different noise on degree centrality measure: Edge add/del modify the degree of four nodes, Edge rotation modify the degree of only two nodes and Edge switch does not modify the degree of any node. Thus, Edge switch keeps the same values for this measure but introduce noise on anonymous data and perturb the precision values with all clustering methods. Similar situation is presented on Edge rotation. Clearly, degree centrality is not a suitable measure to analyse the noise introduced by perturbation methods using both Edge rotation or Edge switch.

Edge intersection (EI) is a simple measure which is strong correlated to clustering-based measures. Additionally, clustering coefficient (C) and transitivity (T) are also strong correlated to clustering-based measures. These measures show strong correlation to the precision results of Infomap, Multilevel, Girvan-Newman, Fastgreedy and Walktrap. As the measures we have seen in the previous paragraph, these ones show lower correlation to the precision results of MCL. The mean correlation indexes are 0.785, 0.814 and 0.743

Pearson	*MCL*	*Infomap*	*Multilevel*	*Girvan-Newman*	*Fastgreedy*	*Walktrap*	μ
$\overline{dist}$	0.580	0.716	0.807	0.785	0.747	0.755	0.732
D	0.201	0.101 *	0.098 *	0.134	0.218	0.014 *	0.128
C_B	0.559	0.687	0.854	0.865	0.831	0.724	0.753
C_C	0.667	0.833	0.903	0.909	0.874	0.899	0.848
C_D	0.296	0.380	0.416	0.504	0.481	0.457	0.422
EI	0.581	0.820	0.861	0.887	0.814	0.748	0.785
C	0.614	0.833	0.889	0.909	0.836	0.802	0.814
T	0.557	0.763	0.840	0.840	0.770	0.690	0.743
λ_1	0.191	0.482	0.509	0.546	0.529	0.397	0.442
μ_2	0.086 *	0.152	0.131	0.154	0.135	0.040 *	0.116
μ	0.433	0.577	0.631	0.653	0.624	0.553	NA

Table 5.4: The Pearson correlation values (r) between clustering precision value and generic information loss measures and its average values μ. An asterisk indicates ρ-values ≥ 0.05, i.e, results not statistically significant.

for edge intersection, clustering coefficient and transitivity.

Next, we will analyse the spectrum-based generic information loss measures. These are the largest eigenvalue of the adjacency matrix (λ_1) and the second smallest eigenvalue of the Laplacian matrix (μ_2). The largest eigenvalue of the adjacency matrix presents moderate correlation, achieving a value of 0.442. The second eigenvalue of the Laplacian matrix does not get good results on any clustering-based measures, achieving an averaged value of 0.116. Furthermore, the ρ-values demonstrate the results are not statistically significant.

Finally, it is important to underline that the precision index on clustering measures achieves moderate correlation, with values from 0.624 to 0.653, on Multilevel, Girvan-Newman and Fastgreedy algorithms. Infomap and Walktrap achieve lower correlation values, but still moderate correlation. MCL presents the worst results of all clustering algorithms with a correlation value of 0.433.

Aggregating some generic measures

In the previous paragraphs we have compared the correlation between each individual generic information loss measure and the precision index. Here, we consider an overall assessment between a group of some GIL measures and the precision index. This experiment tries to illustrate which group of one or more GIL measures are the best ones to explain the clustering-based measures.

Num.	*GIL measures*	*r-square*	σ
1	C_C	0.725	0.146
2	$C_B + C_C$	0.742	0.150
3	$C_B + C_C + EI$	0.765	0.155
4	$D + C_B + C_C + EI$	0.777	0.127
5	$\overline{dist} + D + C_B + C_C + EI$	0.787	0.117

Table 5.5: Results of multivariate regression analysis between clustering precision value and some generic information loss measures.

Let us consider the r-square from a multivariate regression analysis, where the dependent variable is the precision index and the independent variable is a set of one or more GIL measures. The r-square value is indicative of the aggregate correlation between a set of the GIL measures and the precision values. We compute the regression analysis between all combinations from 1 to 5 GIL measures and the precision index. The result for only one measure is perfectly consistent with Pearson correlation analysis, which has been explained previously, where closeness centrality achieves the best result, as we can see in Table 5.5. The first column indicates the number of GIL measures considered, the second one the GIL measures set which achieves the best result, the third column the r-square value, and the last one the standard deviation (σ). The best combination of two GIL measures is betweenness and closeness centrality, with a r-square value of 0.742 and standard deviation of 0.15. For three GIL measures, the best combination is the betweenness centrality, closeness centrality and edge intersection. All GIL measures have obtained high individual correlation values, therefore these results are predictable. Nevertheless, when we consider the best combination of four GIL measures, the diameter appears, which has obtained very low individual correlation values. It is interesting because diameter can help us to predict the clustering-specific perturbation in combination with other GIL measures, but it can be useless when we consider only the diameter. Finally, average distance is added to the group when considering a combination of five measures. The r-square value is 0.787. It is relevant to note that if more measures are considered, higher r-square values are obtained. Thus, the r-square difference between 2 and 5 measures is close to 0.04, which implies a little gain and also a considerable increment of computational and time cost. The problem and its peculiarities are the key points to determine the best combination (one or more) of GIL measures to predict the clustering-specific ones.

Pearson	*Edge add/del*	*Edge rotation*	*Edge swap*
μ	0.670	0.698	0.705
σ	0.206	0.208	0.211

Table 5.6: Pearson correlation averaged values (μ) and standard deviation (σ) for each perturbation method.

5.3.3 Comparing perturbation methods

Next, we will briefly analyse the results based on perturbation methods. Edge add/del is the most general edge modification. It adds and deletes edges at random over entire nodes set. Therefore, an edge is created in every step and another is deleted, changing the degree of four nodes (two nodes decrease their degree while two others increase theirs). Hence, the degree sequence and all related measures suffer high perturbation when Edge add/del is applied. Edge rotation is a sub-set of Edge add/del. It modifies an edge keeping one node and changing the other. Thus, two nodes change their degree in every step (one node decreases its and another increases its). Therefore, the degree sequence and related measures suffer quite less perturbation than Edge add/del. Finally, Edge switch exchanges two edges between four nodes, but none of them modify their degree. Accordingly, the degree sequence and related measures do not modify their values, which means that these measures do not transmit the noise introduced on data.

The degree centrality (C_D) measure evaluates the centrality of each node associated with its degree. Edge switch does not change the node's degree, so it does not introduce perturbation on this measure. But the precision index for all clustering methods on all datasets show that Edge switch introduces perturbation on data. Therefore, this measure is not correlated with clustering results and it is not a suitable measure to evaluate the noise introduced in the perturbed data.

Table 5.6 presents average results and standard deviation for each perturbation method. For each method we compute the correlation between all generic and specific information loss measures over all datasets, and then we calculate the averaged value and the standard deviation for all values which are statistically significant (i.e, ρ-value < 0.05). It is interesting to note that Pearson correlation values obtained by Edge switch are higher, on almost all cases, than Edge rotation and Edge add/del. The average value is 0.705 for Edge switch, while Edge rotation gets a value of 0.698 and Edge add/del a value of 0.670. The standard deviation is similar in all methods.

Pearson	*Karate*	*Football*	*Jazz*	*Flickr*	*URV Email*
μ	0.716	0.796	0.717	0.780	0.729
σ	0.247	0.119	0.170	0.184	0.163

Table 5.7: Pearson correlation averaged values (μ) and standard deviation (σ) for each dataset.

5.3.4 Comparing datasets

In this section, we will summarise the results based on datasets. Table 5.7 presents the Pearson correlation averaged values and standard deviation for each dataset used in our experiments. For each dataset we compute the correlation between all generic and clustering-based information loss measures over all perturbation methods, and then we calculate the averaged value and the standard deviation for all values which are statistically significant (i.e, ρ-value < 0.05). We can see important differences between datasets. For instance, correlation values between generic information loss measures and precision index on American college football (Football) achieve the highest correlation value and the lowest standard deviation value, while on Zachary's karate club (Karate) the value keeps quite low and the standard deviation value rises. As we have commented, Football dataset is a collaboration network representing American football games among colleges during regular season. Hence, it does not follow the power-law on degree sequence. The minimum degree value is 7 and the maximum is 12. Therefore, the connectivity is homogeneous in this graph since there is no hubs and all vertices are highly connected to others. Probably, this graph is more robust to noise than other graphs and the perturbation methods do not cause abruptly disruption on perturbed data.

5.4 Conclusions

Firstly, we have analysed the behaviour of the generic information loss measures and precision index over different datasets. As we have seen, some measures behave in similar way independently of the data where they are applied. Only diameter, the largest eigenvalue of the adjacency matrix and the second smallest eigenvalue of the Laplacian matrix present weak self-correlation values, indicating that their behaviour is dependent on the data where they are applied.

Secondly, we have compared the generic information loss measures and the precision index. Our experiments are based on correlation between each generic information loss measure and precision index computed for each clustering algorithm. The tests showed strong correlation between some generic information loss measures and the clustering-based ones. Some of those discussed are the average distance, betweenness centrality, closeness centrality, edge intersection, clustering coefficient and transitivity. Degree centrality and the largest eigenvalue of the adjacency matrix present moderate correlation values. As we have mentioned, these measures are clearly subordinate to the perturbation method applied, and therefore the correlation is lower than other measures. Finally, some generic information loss measures, like diameter and the second smallest eigenvalue of the Laplacian matrix, show weak correlation values with the clustering-specific ones. In addition, some experiments related with these measures are not statistically significant.

Thirdly, we have demonstrated that considering two or more generic information loss measures helps us to get a higher correlation value. Even so, the gain is not great and the complexity rises when considering two or more generic measures.

Fourthly, we have exposed that the perturbation method affects the correlation between generic and cluster-specific information loss measures. As we have seen, datasets perturbed by Edge switch show higher correlation between generic and specific information loss measures than the datasets perturbed by Edge add/del or Edge rotation.

Finally, we have discussed the effect of the datasets on correlation between generic and clustering-specific information loss measures. We have seen substantial differences between datasets in our experimental framework. The degree distribution and the connectivity of the graph affect the robustness of the network and how the perturbation alters the graph's structure.

Certainly, the purpose of the data should be taken into account during the anonymization process. Each dataset has its own properties which should be analysed to choose the best anonymization method. If different datasets are generated according to each problem-specific environment, then it is necessary to analyse the background knowledge an attacker can infer from different anonymized datasets.

Chapter 6

Improving data utility on edge modification processes

In this chapter we focus on data utility and information loss on edge modification processes. After a brief introduction in Section 6.1, we propose two methods for improving data utility and reducing the information loss: the first one, presented in Section 6.2, is based on edge centrality score, and the second one is based on the core number sequence and it is shown is Section 6.3.

6.1 Introduction

Data mining relies on a great set of tools and techniques to mine the vast amount of data that is available today in order to extract useful and relevant patterns and gain insights on somebody's records. However, these data often contain personal and private information about users and individuals. In order to overcome this issue, methods that add noise to the original data have been developed to hinder the re-identification processes. But the noise introduced by the anonymization steps may also affect the data, reducing its utility for subsequent data mining processes. Usually, the larger the data modification, the harder the re-identification but also the less the data utility. Therefore, it is necessary to preserve the integrity of the data (in the sense of retaining the information that we want to mine) to ensure that the data mining step is not altered by the anonymization step. Among other things, for the anonymization step to be considered any useful and thus valid, the analysis performed on the obfuscated data should produce results as close as possible to the ones the original data would have led to and this is how we evaluated our experiments. Considering the example of a social network, this would mean that graph properties such as centrality measures and clusters of nodes (i.e, users) are preserved in the anonymous data. A trade-off between *data privacy* and *data utility* must be reached and it is in this spirit that we developed our proposed edge modification techniques.

Edge modification techniques are widely used in network's anonymization. Nevertheless, none of these works consider the edge's relevance. Edge's relevance can help us to decide which edges can be removed or modified and which ones must be preserved. If we want to preserve the network properties, such as average distance, diameter, node centrality and more, we have to find the most relevant edges and preserve them from removing or modifying processes. This can lead anonymization methods to a better data utility and less information loss.

6.2 Edge Neighbourhood Centrality

In this section we use different metrics for edge's relevance in order to analyse the effect of edge deletion on graph structure. We work with simple, undirected and unlabelled networks. We want to introduce a new metric for edge's relevance which can help us to preserve the most important edges on network, and leads an edge modification process to remove the less important edges, keeping the basic network structural and spectral properties.

6.2.1 Metrics for Edge's Relevance

We consider the following metrics for quantifying edge's relevance: Edge betweenness (EB), Link salience (LS) and Edge neighbourhood centrality (NC).

Edge betweenness (EB) is defined as the number of the shortest paths that go through an edge in a network [37]. An edge with a high edge betweenness score represents a bridge-like connector between two parts of a network, and the removal of which may affect the communication between many pairs of nodes through the shortest paths between them. The edge betweenness of edge $\{v_i, v_j\}$ is defined by:

$$EB_{\{v_i,v_j\}} = \frac{1}{n^2} \sum_{\substack{s,t=1 \\ s\neq t}}^{n} \frac{g_{st}^{\{i,j\}}}{g_{st}} \tag{6.1}$$

where $g_{st}^{\{i,j\}}$ is the number of shortest paths from node v_s to v_t that pass through edge

$\{v_i, v_j\}$, and g_{st} is the total number of shortest paths from node v_s to v_t.

Link salience (LS) [40] is defined as a linear superposition of all shortest path trees (SPTs). It quantifies the fraction of SPTs each link participates in. The salience of an edge $\{v_i, v_j\}$ is computed as follows:

$$LS_{\{v_i,v_j\}} = \frac{1}{n}\sum_{k=1}^{n} T_{ij}(v_k) \tag{6.2}$$

where $T(v_k)$ is the shortest path tree (SPT) rooted at node v_k. $T(v_k)$ can be represented as a matrix with elements:

$$T_{ij}(v_k) = \begin{cases} 1 & \text{if } \sum_{l=1}^{n} \sigma_{ij}(v_l, v_k) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

where $\sigma_{ij}(v_l, v_k)$ is equal to 1 if edge $\{v_i, v_j\}$ is on the shortest path from v_k to v_l, and otherwise is equal to 0. Despite the apparent similarity between edge betweenness and link salience, both quantities capture very different qualities of edges. Salience is insensitive to a position of an edge, acting as a uniform filter.

Calculating both metrics above of all the vertices on a network involves calculating the shortest paths between all pairs of vertices. This takes $\mathcal{O}(n^3)$ time with the Floyd-Warshall algorithm. On a sparse network, Johnson's algorithm may be more efficient, taking $\mathcal{O}(n^2 log(n) + nm)$ time. So, both metrics are not useful for large networks, since every time we remove an edge we must re-calculate all edges' values. Therefore, we present a new simple metric for quantifying edge relevance, called edge neighbourhood centrality.

Edge neighbourhood centrality (NC) of an edge $\{i, j\}$ is defined as the fraction of nodes which are neighbours of v_i or v_j, but not of v_i and v_j simultaneously. The edge neighbourhood centrality is computed as follows:

$$NC_{\{v_i,v_j\}} = \frac{|\Gamma(v_i) \cup \Gamma(v_j)| - |\Gamma(v_i) \cap \Gamma(v_j)|}{2 \times deg^{max}(G)} \tag{6.4}$$

where $\Gamma(v_i)$ is the 1-neighbourhood of node v_i. Note that this metric can be computed on $\mathcal{O}(m)$ using the adjacency matrix representation of the network. An edge with high score is a bridge-like for neighbourhood nodes. All measures presented above are in range [0,1].

6.2.2 Experimental Results

Our experiments use edge betweenness (EB), link salience (LS) and edge neighbourhood centrality (NC) to evaluate all edges on each network. For each edge metric, we evaluate all edges. One by one, we remove each edge from the graph and then we compute the error introduced in the graph by comparing some network characteristic metrics on original and on modified graphs. Four different data sets are used in our experiments: Zachary's Karate Club, American College Football, Jazz Musicians and C.Elegans. We analyse the following structural and spectral metrics in order to quantify the noise introduced by edge deletion. Structural metrics are related to topological characteristics whereas spectral metrics are based on spectral characteristics.

In this section we analyse the correlation based on Pearson correlation coefficient between each edge metrics and all network characteristic metrics. A high correlation points out that removing edges with high score based on edge metric produces larger noise on perturbed graphs, while removing edges with low score produces fewer noise. On the contrary, a low correlation value suggests that removing edges with high or low score does not imply introducing more or less noise on perturbed data. We discuss in the next three sections the three edge metrics: edge betweenness, link salience, and edge neighbourhood centrality. Results on the four networks described are reported, and Figure 6.1 displays the results for the Karate network, which is the one that can be better visualized because it is the smallest dataset.

6.2.2.1 Edge Betweenness

Here we use the edge betweenness (EB) as a measure to quantify edge's relevance. As we can see on Table 6.1, average distance shows very high correlation for Karate and Football networks and high correlation for Jazz and CElegans. Correlation values between EB and diameter are low because diameter is much more stable than EB. Figure 6.1a shows EB score for each edge (solid red line) and the error on diameter (dashed blue line) introduced by this each removal on Karate network. Only twice the diameter has been modified, nevertheless it was when high-scored edges were deleted from the graph.

Node betweenness presents very high correlation values, obtaining an average value of 0.97. In Figure 6.1b we can see EB and node betweenness values. Clearly, EB is closely related to node betweenness centrality. On the other hand, closeness centrality presents lower values than node betweenness, except on Football network.

Results for transitivity, λ_1 and μ_2 are quite different. Pearson correlation values are

Metric	*Karate*	*Football*	*Jazz*	*CElegans*	Average
Average Distance	0.83	0.93	0.79	0.73	0.82
Diameter	0.44	0.25	0.32	0.20	0.30
Node Betweenness	0.94	0.99	0.98	0.96	0.97
Closeness	0.44	0.95	0.37	0.25	0.50
Transitivity	-0.10	0.57	0.28	0.22	0.29
λ_1	0.26	-0.04	-0.11	0.42	0.21
μ_2	0.61	0.09	0.07	0.08	0.21
Average	0.52	0.55	0.42	0.41	0.47

Table 6.1: Pearson correlation values between EB and network characteristic metrics.

low or very low on all datasets, except for transitivity on Football and λ_1 on Karate network. For instance, correlation value between EB and transitivity is -0.10 on Karate, which means that there is not relation between EB and transitivity. That is, removing edges with low EB score may introduce higher noise on network than removing edges with high EB score. This is because important EB-based edges are local bridges, so removing them we do not destroy any triangle, while low EB-based score edges are part of a triangle, so removing them we decrease the transitivity measure of the all involved nodes on this triangle.

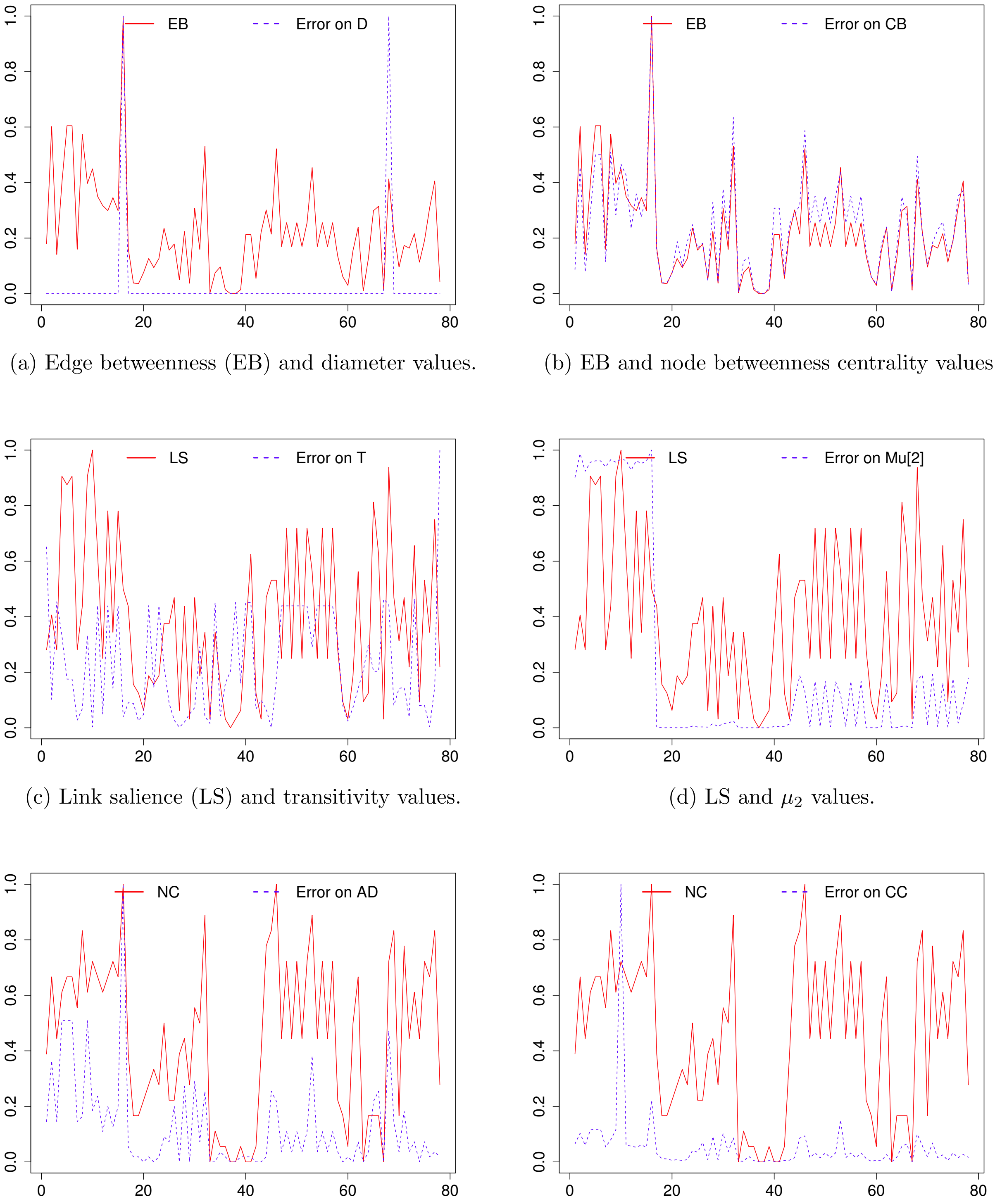
6.2.2.2 Link Salience

The second measure for edge's relevance is link salience (LS). A small set of nodes are scored with high salience value, $LS \approx 1$. These nodes form the skeleton of the network whereas the others have low salience values, configuring a bi-modal distribution [40].

Table 6.2 presents the correlation values between LS and network characteristic metrics. Average distance shows good correlation values, but lower than the ones showed above. Once again, correlation values between LS and diameter presents low score. Node betweenness achieves high score, but lower than EB correlation on all networks. However, closeness gets a good score, even better than correlation with EB.

Like for the above metric, the correlation between LS and transitivity, λ_1 and μ_2 are not clear. For instance, the correlation value between LS and transitivity is 0.17 on Karate, meaning that there is not relation between EB and transitivity, as shown in Figure 6.1c. They score very low values, suggesting that using salience as an edge's relevance does not guarantee better data utility and lower information loss. Figure 6.1d also exemplifies it.

The correlation values between salience and the structural and spectral analysed met-



(a) Edge betweenness (EB) and diameter values.

(b) EB and node betweenness centrality values.

(c) Link salience (LS) and transitivity values.

(d) LS and μ_2 values.

(e) Edge neighbourhood centrality (NC) and average distance values.

(f) NC and closeness centrality values.

Figure 6.1: Pairs of edge relevance metrics and network characteristics metrics on Karate network.

Metric	*Karate*	*Football*	*Jazz*	*CElegans*	Average
Average Distance	0.61	0.80	0.82	0.79	0.75
Diameter	0.20	0.10	0.13	0.22	0.16
Node betweenness	0.67	0.86	0.84	0.56	0.73
Closeness	0.45	0.86	0.53	0.33	0.54
Transitivity	0.17	0.53	0.36	0.32	0.34
λ_1	-0.10	-0.08	-0.17	0.01	0.09
μ_2	0.48	0.04	0.09	0.04	0.16
Average	0.38	0.47	0.42	0.32	0.40

Table 6.2: Pearson correlation values between LS and network characteristic metrics.

Metric	*Karate*	*Football*	*Jazz*	*CElegans*	Average
Average Distance	0.55	0.63	0.18	0.52	0.47
Diameter	0.24	0.07	0.03	-0.03	0.09
Node betweenness	0.75	0.78	0.36	0.46	0.59
Closeness	0.33	0.69	0.01	0.06	0.27
Transitivity	-0.06	0.51	-0.09	0.09	0.19
λ_1	0.41	-0.05	0.43	0.52	0.35
μ_2	0.48	0.21	0.35	0.19	0.31
Average	0.40	0.42	0.21	0.27	0.32

Table 6.3: Pearson correlation values between NC and network characteristic metrics.

rics is lower, in general, than the ones obtained between EB and the same metrics.

6.2.2.3 Edge neighbourhood centrality

Finally, we test our new simple metric for quantifying edge's relevance. As we can see on Table 6.3, average distance presents a moderate correlation on all networks, except Football where the correlation is not clear. Figure 6.1e illustrates values for NC and average diameter on Karate network. Neither diameter shows correlation with NC.

Once again, node betweenness keep on moderate scores on all networks, expect Football. Closeness and transitivity do not achieve good results on Karate, Jazz nor CElegans, but they achieve quite better scores on Football network. Figure 6.1f depicts values of NC and closeness. Finally, λ_1 and μ_2 present irregular values, showing weak correlation on some networks.

6.2.3 Conclusions

Average distance is a metric related to path lengths. Edge betweenness, as we have seen, is the edge metric which has the strongest correlation with average distance, scoring the highest correlation values for this metric. Hence, removing important edge-betweenness-based edges affects it in a significant way.

Also we have seen that diameter is a stable metric with respect to network perturbation. So, it is difficult to correlate any of our edge's relevance metrics with diameter, although we have seen on Figure 6.1a that diameter perturbation occurs when an important edge is removed from the network. It suggests that both EB and NC identify an important local bridge, and removing it produces and increment on several shortest paths along the graph.

Node betweenness and closeness are widely used for clustering and community detection algorithms, so edge-modification-based anonymization algorithms must consider which edges remove or modify in order to reduce information loss and preserve data utility for clustering purposes. All metrics get a good correlation values, but the best one is EB. Clearly, EB and node betweenness are closely correlated. Some edges with high EB score are local bridges. So, removing them we introduce a high noise on node betweenness and closeness metrics. On the other hand, transitivity is not affected by removing local bridges and probably this measure is affected by removing edges which participate on many triangles (that is, edges with low value of EB, LS and NC).

Finally, both spectral measures (λ_1 and μ_2) do not have high correlation with any edge's relevance metric. The best one is NC, which achieves moderate scores on all networks. But, using EB and LS the results are not clear. The noise introduced are not correlated with the score of these edge metrics. The eigenvalues of the adjacency matrix encode information about the cycles of a network as well as its diameter [88]. The maximum degree, chromatic number, clique number, and extend of branching in a connected network are all related to λ_1, while the eigenvalues of L encode information about the tree-structure of the network.

Edge betweenness has been proved as the best edge's relevance metric. The values of correlation are larger than others on several cases, showing that removing or modifying edges with high values of edge betweenness introduce large noise on important structural and spectral metric. Nevertheless, it is based on shortest path between all pair of nodes and it implies a high computational cost. Therefore, it is not a good metric for medium or large networks. Link salience, although is a good metric for visualizing and understanding network structure (skeleton), does not achieve the same results as edge betweenness. Despite it gets good results for structural metrics, it fails to achieve good results on spectral metrics. Finally, we have proposed a new simple metric for edge's relevance. Edge neighbourhood centrality shows quite good results on all structural and spectral metrics. The results are not as good as the ones achieved with edge betweenness, but the complexity much lower for this metric than others. Therefore, it is a good metric to estimate edge's relevance on medium or large networks.

Although some differences among them, these measures are able to identify the most important edges, which may not be removed or modified, and the least important ones, which can be removed or modified. Edge's relevance can help edge-modification-based anonymization processes to achieve better results, raising data utility and reducing information loss. So, incorporating edge's relevance on edge modification processes can lead us to produce a more useful anonymized networks.

6.3 Core Number Sequence

In this section, we explored the idea of *coreness-preserving* edge modification. Basically, instead of modifying any edge at random, we only do it when it does not change the core numbers of both its endpoints and by extension the whole core number sequence of a graph. By doing so, we preserve better the underlying graph structure, retaining more data utility than in the random case but we still alter the node degrees, hindering the re-identification process and thus maintaining a certain level of privacy.

6.3.1 Our approach

In this section, we present our approach that preserves the coreness of a graph while anonymizing it. The preliminary concepts upon which our work is built are: the notion of *apparent degree*, *degeneracy* (see Section 2.3) and *edge modification* (details are presented in Section 2.4).

As we have stated before, we define the *coreness* (also known as *core number sequence* or *shell index sequence*) as the sequence of each vertex core number by analogy with the degree sequence. Hence, a coreness-preserving graph modification approach means a technique that alters the graph (by adding and/or deleting edges) without changing the core number of any node.

Apparent degree

Here, we considered undirected and unweighted edges, which represent relationships between entities that are bidirectional and independent of their strength (e.g. being friend on Facebook but not following someone on Twitter). Hereinafter, we will refer to the degree of a node as the *apparent degree* because it represents the total number of neighbors a node has in a network but is usually an overestimation of the number of true relations. In the case of a social network, not all of someone's connections are truly his friends but sometimes just acquaintances.

Idea and brute force algorithm

Our idea came from the observation that there is some space left between the two extremal edge modification techniques that are Rand add/del and Rand switch. The former ensures privacy by randomly modifying the structure of the graph at the cost of rapidly increasing the information loss while the latter maintains the apparent degree of every node to preserve some data utility, allowing an attacker to quickly re-identify nodes, starting from those of unique degree (unchanged k-anonymity between the original and the perturbed networks).

Preserving the coreness of the graph seems like a good trade-off. Instead of maintaining the *apparent degree*, we allow for some slack as Equation 2.14 suggests, increasing the level of randomization (privacy), but in the meantime preserving the *true degree* of every node that supposedly holds the information to mine (utility). Indeed, graph degeneracy has been used for community detection [36] and the core number has been shown to be a more robust version of the apparent degree [7] and directly correlated with betweenness centrality [3]. Again, in the case of a social network, it seems legitimate to add/remove random relations as long as it preserves the community a node belongs to. Removing "satellite connections" while preserving "close friends" seems like an intuitive way to achieve our goal and empirical results on various networks support our claim.

The most straightforward way to apply our idea would be to randomly select an edge to add/delete, perform the operation and re-compute the coreness of the graph to check for any difference. Even though there exists an optimal linear algorithm for degeneracy, it seems too expensive to do it for each edge modification, especially since the core number sequence should only change locally around the edge if anything. We present in the next subsections the algorithms we developed to perform faster edge modifications.

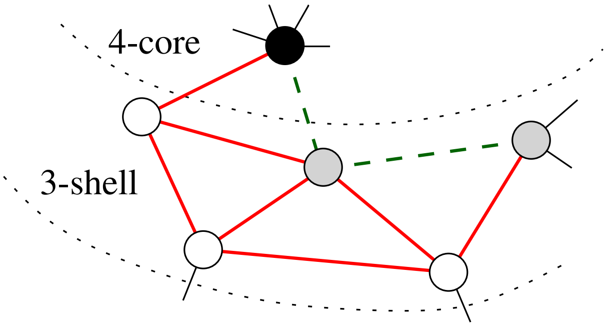


Figure 6.2: Illustration of coreness-preserving edge deletion. Green dashed edges can be safely removed while red solid ones cannot without altering the coreness.

Coreness-preserving edge modification

Let $e = \{v_i, v_j\}$ be an undirected edge between vertices v_i and v_j. In the case of edge deletion, $e \in E$ and in the case of edge addition, $e \notin E$. The goal is to check whether we can add/delete the edge to/from the graph without changing its core number sequence. Moreover, because the edge modification directly impacts v_i and v_j, it is sufficient to check whether the edge modification preserves the core numbers of v_i and v_j. Indeed, it is not possible for the coreness to change without $core(v_i)$ or $core(v_j)$ changing.

Since the graph is undirected, for any edge, we can choose v_i and v_j such that $core(v_i) \leq core(v_j)$. In the next subsections, we consider $core(v_i) = k$ and we will refer to the k-shell that v_i belongs to as the *lower shell* ($\mathcal{S}_k$). It is also the k-shell v_j belongs to if and only if $core(v_i) = core(v_j)$, otherwise we will refer to the other subgraph as the *upper core*. In both cases (add/delete), it is important to note that if $core(v_i) < core(v_j)$ then we only need to check whether the edge modification preserves $core(v_i)$. Indeed, by the time we reach the upper core (in the basic algorithm), the edge will not exist since node v_i will have been removed and thus it will have no incidence on $core(v_j)$.

Coreness-preserving edge deletion

The intuitive idea is quite simple: an edge can be removed if none of its two endpoints belong to the k-corona of the lower shell.

For the node(s) in the lower shell, deleting an edge means decreasing their effective degree by one. Thus, for a node to remain in its k-shell after losing one connection, it needs at least $k + 1$ neighbours in the first place – in other words, it cannot belong to the k-corona. In the case where $core(v_i) < core(v_j)$, we only need to check that $v_i \notin \mathcal{C}_k$ while in the case where $core(v_i) = core(v_j)$, we need to check that both v_i and v_j do not belong to the k-corona of their common k-shell. In any case, assuming the coreness and the effective degree sequence have been pre-computed, the check can be done in constant time and if the edge were to be removed, we would only need to decrease the effective degree of the node(s) from the lower shell.

Figure 6.2 illustrates this procedure. We drew a couple of points from the 3-shell ($\mathcal{S}_3$) and 4-core ($\mathcal{H}_4$) of a graph. White nodes belong to $\mathcal{C}_3$, gray nodes to $\mathcal{S}_3 \setminus \mathcal{C}_3$ and black nodes to $\mathcal{H}_4$. Solid lines indicate edges, black ones with nodes not displayed for space constraints. The green dashed edges can be safely removed (but not both of them, the check being only valid for one edge) because none of their endpoints belong to the k-corona of the lower shell (here the 3-shell). The red solid edges cannot be deleted because at least one of the two endpoints belong to the 3-corona.

Coreness-preserving edge addition

Checking whether an edge can be added to a graph without changing its core number sequence appears to be less trivial than for deleting one. Nevertheless, we came up with an efficient alternative to the brute force approach mentioned above. Instead of recomputing the core number for every node in the graph, we are trying to estimate as fast as possible if the core number of v_i will remain unchanged after adding the new edge – in other words, if v_i will move from $\mathcal{S}_k$ to $\mathcal{H}_{k+1}$ or not.

The algorithm is as follows. First, we add the new edge to the graph and we update the effective degree(s). Then, we traverse the lower shell ($\mathcal{S}_k$) starting from node v_i. For v_i to be in $\mathcal{H}_{k+1}$, it needs at least $k+1$ neighbours also in $\mathcal{H}_{k+1}$. There are four kinds of neighbours: (a) nodes with a lower core number, (b) nodes with an equal core number and in the k-corona ($\mathcal{C}_k$), (c) nodes with an equal core number and not in $\mathcal{C}_k$ and (d) nodes with a higher core number. Neighbours of type (a) and (b) cannot be by definition in $\mathcal{H}_{k+1}$. Neighbours of type (c) *could* be in $\mathcal{H}_{k+1}$ thanks to the new added edge. Neighbours of type (d) are by definition in $\mathcal{H}_{k+1}$. So, for a given node v_i, its effective degree in $\mathcal{H}_{k+1}$ is lower-bounded by the number of neighbours of type (d) and upper-bounded by the number of neighbours of type (c) and (d). The goal of the algorithm is to estimate as fast as possible this effective degree to see if it is greater than k or not.

From the bounds, it follows that if the lower bound is strictly greater than k then the node is definitely in $\mathcal{H}_{k+1}$ and thus the edge cannot be added (otherwise v_i would move up to the next core) – this happens when a node has already k connections with nodes in $\mathcal{H}_{k+1}$. Conversely, if the upper bound is strictly less than $k+1$ then the node

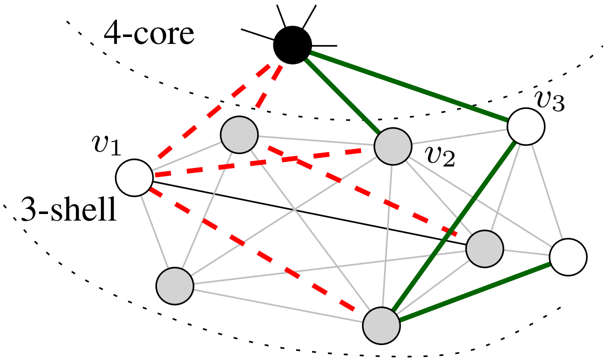


Figure 6.3: Illustration of coreness-preserving edge addition. Green solid edges can be safely added, while red dashed edges cannot without altering the coreness.

is definitely not in $\mathcal{H}_{k+1}$ and thus the edge can be safely added. In-between, we need to visit the neighbours of type (c) and apply on them the same procedure. It all comes down to the nodes from $\mathcal{S}_k \setminus \mathcal{C}_k$: even though they had enough neighbours to be in $\mathcal{H}_{k+1}$, these neighbours were not connected enough to be in $\mathcal{H}_{k+1}$ but the new added edge could make the difference. This leads to a recursive definition of the problem, i.e. *depth-first-search* (DFS) traversal of $\mathcal{S}_k \setminus \mathcal{C}_k$. Then, whenever we visit a node that cannot move to the next core, we back propagate this information to lower the estimate on the degree we have for its parent. If at any point the degree estimate for the source node v_i drops below $k + 1$ then we know the edge can be safely added. Otherwise, we keep visiting new neighbours of type (c) until we run out. At that point, we have basically found at least $k + 1$ nodes of type (c) ready to move up altogether in $\mathcal{H}_{k+1}$, thus preventing the edge addition.

Even in the case of $core(v_i) = core(v_j)$, we only need to do the traversal once, from v_i, since v_j will be visited anyway. It is not possible for v_i to move to the next core without v_j and vice-versa since the new added edge has to survive in the next core. Hence, contrary to edge deletion, we only need to run the check on v_i.

In the worst case scenario (corresponding to a very tight k-shell), we may need to visit all the nodes in $\mathcal{S}_k \setminus \mathcal{C}_k$ so we have an overall complexity of $\mathcal{O}(m)$ time and $\mathcal{O}(n)$ space. But in practice, we only visit the nodes from $\mathcal{S}_k \setminus \mathcal{C}_k$ that have at least $k+1$ neighbours in $\mathcal{S}_k \setminus \mathcal{C}_k \cup \mathcal{H}_{k+1}$, reducing very rapidly the number of candidates as observed experimentally. Moreover, we implemented a *breadth-first-search* (BFS) traversal to take advantage of the back propagation as soon as possible to prune nodes that cannot move to $\mathcal{H}_{k+1}$ anyway.

Figure 6.3 illustrates this procedure using the same set up as Figure 6.2. The green solid edges can be safely added while the red dashed edges cannot. We note that we cannot assert whether an edge can be added just by checking if its endpoint(s) belong to the k-corona (like v_1 or v_3) or not (v_2). It appears to depend on the structure of the neighbourhood, hence the proposed algorithm.

Note that at the cost of some privacy (false negatives for edge addition), it is possible to limit the depth of the BFS traversal and cap the number of visited vertices. We did not need it in our experiments but for some applications where speed matters or for some particular graph structures with tight giant cores, it might help and ensure constant time edge addition.

Coreness-preserving edge rotation

We can easily combine the two previous approaches to check for coreness-preserving edge rotation. Considering three nodes like in Figure 2.2b, we only need to check that edge (v_i, v_j) can be safely deleted and edge (v_i, v_p) added. This has as effect the modification of the apparent degrees of v_j and v_p as wanted while not changing any core number.

Additional remarks

Next, we introduce a couple of additional remarks on our proposed approach.

- Two kinds of slack: Given an edge to be added/deleted, for the endpoint in the upper core, this modification has no impact on its core number. Therefore, the bigger the gap between its apparent degree and its true degree, the more slack for perturbation it has since it possesses many "satellite connections". For the endpoint(s) in the lower shell, the apparent degree has already been reduced to its effective degree and therefore the slack for perturbation is between its effective degree and its true degree. The bigger the gap, the easier the edge deletion but the harder the edge addition.

- Tight vs. relaxed k-shell: Edge addition and edge deletion have opposite goals that translate into opposite favourable k-shells (by favourable, we mean k-shells where the likelihood that the edge modification can be safely performed is high). Indeed, the edge deletion is more likely to be possible if the k-corona is small compared to the k-shell while it is generally the opposite for edge addition. A sparse k-shell means that most effective degrees are close to the shell index so it is less likely that by adding a new edge, there are enough connected nodes to move to the next core but in the meantime, deleting one edge might be hard. Conversely, a dense k-shell is less favourable for edge addition and the algorithm would probably take longer (more neighbours of type (c)). This is the reason why we suggest alternating edge additions and deletions when performing the anonymization to help maintaining the k-shells neither too tight nor too relaxed.

- Nodes from the 0-shell: Similarly to Rand switch, our method cannot add edges to nodes from the 0-shell since they would immediately move to the 1-shell. We see two options to deal with this particularity: (a) artificially allow them to move to the 1-shell under the assumption that the impact on the data utility is limited or (b) consider that an attacker cannot learn anything from a node without neighbours (except if $\mathcal{S}_0$ is a singleton, in which case he can re-identify the node but this one only). In our experiments, we considered the second option.

6.3.2 Experiments

In this section, we present the experimental set up and the empirical results of our work. Our framework generates perturbed (i.e. anonymized) networks from the original one by applying our proposed method, coreness-preserving edge modification denoted by "Crnss Add/Del" following Ying and Wu's terminology in [88] and the two baselines "Rand Add/Del" and "Rand Switch". For each method, we considered 10 independent executions with a anonymization ranging from 0% to 25% of the total number of edges. We quantified the noise introduced on the perturbed data using some generic information loss measures and clustering-specific information loss.

We used 5 standard real networks to test our approach: *Zachary's karate club*, *Jazz*, *URV email*, *Polblogs* and *Caida* (see Section 2.5 for further details).

Generic graph properties

In order to quantify the noise introduced in the data, we used several structural and spectral graph properties. Some structural measures evaluate the graph in its whole through a global score: *average distance* ($\overline{dist}$), *diameter* (D), *harmonic mean of the shortest distance* (h), *sub-graph centrality* (SC) and *transitivity* (T). Other metrics evaluate the graph locally (at the node-level): *betweenness centrality* (C_B) and *closeness centrality* (C_C). Additionally, we used two spectral measures that are closely related to many graph properties [88]. These are *the largest eigenvalue of the adjacency matrix* A (λ_1) and *the second smallest eigenvalue of the Laplacian matrix* L (μ_2).

Variations in the generic graph properties are a good way to assess the information loss but they are just a proxy to the changes in data utility we actually want to measure. What we are truly interest in is, given a data mining task we want to perform on the graph, quantify the disparity in the results when performing the task on the anonymized dataset. Clustering is one of standard tasks and we chose the extracted clusters as the data utility researchers could be interested in.

Clustering-specific measures

We ran 4 graph clustering algorithms to evaluate the edge modifications techniques using the implementations from the `igraph` library. All of them are unsupervised algorithms based on different concepts and developed for different applications and scopes. An extended revision and comparison of them can be found in [54, 92]. The selected clustering algorithms are: *Fastgreedy*, *Walktrap*, *Infomap* and *Multilevel* (details can be found in Section 4.3.1). Walktrap (WT) and Infomap (IM) are based on the random walk concept while Fastgreedy (FG) is based on hierarchical edge betweenness and Multilevel (ML) on graph modularity. Although some algorithms permit overlapping among different clusters, we did not allow it in our experiments by setting the corresponding parameter to zero, mainly for ease of evaluation.

	Method	$\overline{dist}$	D	h	SC	T	C_B	C_C	λ_1	μ_2
Karate	Crnss Add/Del	0.069	0.681	0.024	7.238	0.042	0.039	**0.036**	0.296	0.216
	Rand Add/Del	**0.048**	**0.331**	**0.012**	6.812	0.031	0.030	0.053	0.280	**0.058**
	Rand Switch	0.120	0.596	0.047	**1.648**	**0.019**	**0.022**	0.037	**0.119**	0.184
Jazz	Crnss Add/Del	**0.100**	**0.300**	**0.053**	7.367	**0.096**	0.007	**0.028**	1.035	0.061
	Rand Add/Del	0.188	1.927	0.095	9.933	0.111	0.008	0.045	2.862	1.856
	Rand Switch	0.127	0.504	0.067	**5.441**	0.105	**0.006**	0.032	**0.654**	**0.056**
URV email	Crnss Add/Del	**0.083**	**0.073**	0.068	6.495	0.047	0.126	0.017	1.429	0.050
	Rand Add/Del	0.099	0.508	**0.042**	6.776	**0.038**	0.128	0.147	1.873	0.317
	Rand Switch	0.114	0.165	0.094	**3.549**	0.044	**0.088**	**0.012**	**0.512**	**0.003**
Polblogs	Crnss Add/Del	**0.045**	**0.750**	**0.039**	1.055	0.043	0.101	0.162	2.856	0.276
	Rand Add/Del	0.116	2.319	0.071	1.119	0.040	0.144	0.110	8.002	0.429
	Rand Switch	0.088	0.846	0.064	**0.476**	**0.033**	**0.089**	**0.038**	**0.518**	**0.062**
Caida	Crnss Add/Del	0.055	2.491	0.051	-	0.139	**0.009**	**0.184**	27.836	0.007
	Rand Add/Del	**0.022**	3.973	0.145	-	0.034	0.010	0.238	**20.139**	0.553
	Rand Switch	0.066	**0.827**	**0.040**	-	**0.006**	0.019	0.216	32.738	**0.006**

Table 6.4: Average error for *Crnss Add/Del*, *Rand Add/Del* and *Rand Switch* edge modification processes on 9 generic information loss measures.

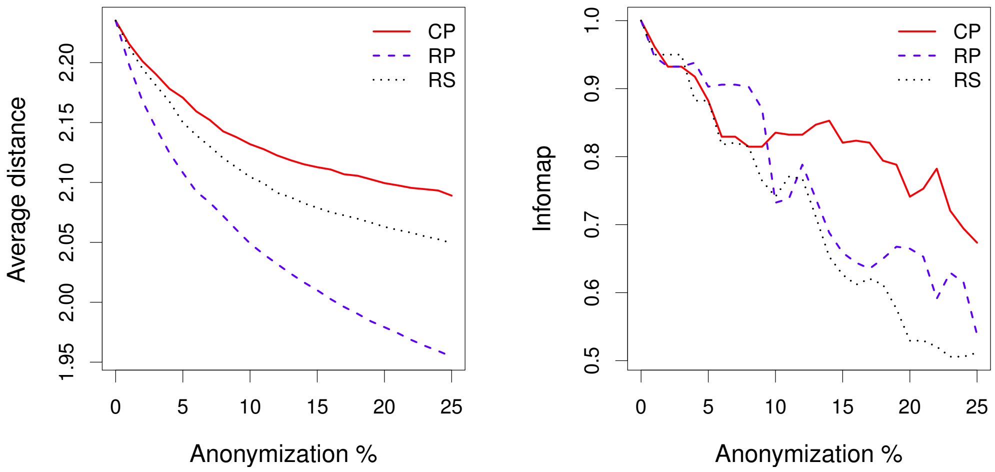


Figure 6.4: Average distance (*Jazz*) and precision index (*Karate*) values for an anonymization varying from 0% to 25%.

Results

We present in Table 6.4 the average errors over 10 independent runs and 25 levels of anonymization for *Crnss Add/Del*, *Rand Add/Del* and *Rand Switch* for all the information loss measures aforementioned. Note that for the biggest network Caida, some results (indicated by -) could not be computed in a reasonable time for reasons unrelated to our approach but due to the measure in itself. For instance, SC relies on all the paths starting and ending at each vertex, which can be very expensive to compute for large networks. Bold values indicate the method that achieves the best result (i.e. lowest information loss) on each measure and dataset (i.e. per cell).

Our edge modification method that preserves the coreness (Crnss Add/Del) performs better than the others methods on generic measures related to graphs paths, i.e. average distance ($\overline{dist}$), diameter (D), harmonic mean of the shortest distance (h) and closeness centrality (C_C). For instance, Figure 6.4a illustrates in more details the results for average distance. The 0% perturbation point in the upper left corner represents the value of this metric on the original graph. Thus, values close to this point indicate low noise on perturbed data. As we can see, Crnss Add/Del remains closer to the original value than the others methods obtaining the lowest average error value (0.1). Nevertheless, our approach does not yield the best results on betweenness centrality (C_B) but still close to Rand Switch.

Subgraph centrality (SC) and transitivity (T) are measures related to centrality and

	Method	IM	ML	FG	WT
Karate	Crnss Add/Del	**0.172**	**0.156**	**0.249**	**0.247**
	Rand Add/Del	0.236	0.186	0.287	0.327
	Rand Switch	0.284	0.213	0.267	0.323
Jazz	Crnss Add/Del	**0.086**	**0.089**	**0.042**	0.089
	Rand Add/Del	0.131	0.118	0.081	0.113
	Rand Switch	0.101	0.099	0.057	**0.074**
URV email	Crnss Add/Del	**0.210**	**0.348**	**0.347**	0.287
	Rand Add/Del	0.313	0.426	0.402	0.533
	Rand Switch	0.251	0.373	**0.347**	**0.281**
Polblogs	Crnss Add/Del	**0.070**	0.464	**0.024**	**0.023**
	Rand Add/Del	0.226	0.418	0.194	0.227
	Rand Switch	0.123	**0.401**	0.046	0.066
Caida	Crnss Add/Del	**0.115**	**0.239**	**0.276**	0.225
	Rand Add/Del	0.217	0.316	0.343	0.723
	Rand Switch	0.128	0.241	0.277	**0.221**

Table 6.5: Average error for *Crnss Add/Del*, *Rand Add/Del* and *Rand Switch* edge modification processes on 4 clustering-specific information loss measures.

triads in a network. Clearly, our method does not lead to the best results but still close to Rand Add/del in most cases. Rand Switch gets the best results on these two measures, getting a significant advantage specifically in subgraph centrality, probably because it maintains the node degrees. The last two generic measures are the spectral ones. Edge Switch performs better on the largest eigenvalue of the adjacency matrix (λ_1) and the second smallest eigenvalue of the Laplacian matrix (μ_2), achieving significant differences on λ_1.

Table 6.5 presents the error on precision index computed using the four clustering algorithms described in Section 4.2. Bold values indicate the method that achieves the best result (i.e. lowest information loss) on each measure and dataset (i.e. per cell). It is crucial to consider the results on real graph mining processes since the main goal of the anonymous data is to provide valuable information to researchers and others third-parties in order to understand the structure and the behavior of real networks. Hence, we have considered clustering as an important graph mining task that can be useful to quantify the data utility loss in our experimental framework. The coreness-preserving approach outperforms the others in all datasets and clustering methods. For instance, it is interesting to note that, although our approach does not get good results on generic measures for the Karate dataset, it achieves the best results on all clustering methods as Figure 6.4b illustrates for the Infomap method. As we can see, Crnss Add/Del outperforms the others methods and gets precision index values greater than 0.8 up to 20% of anonymization. Therefore, even if the generic measures could indicate that Rand Switch introduce less noise on perturbed data, the clustering processes show the opposite, demonstrating that our method preserve better the data utility and reduce the information loss.

6.3.3 Conclusions

In this section, we have proposed a novel edge modification technique for graph anonymization. By preserving the core number sequence of a graph, its coreness, our method achieves a better trade-off between data utility and data privacy than existing edge modification operations. In particular, it outperforms Edge add/del in terms of data utility and Edge switch in terms of data privacy as empirical results have shown, be it on generic graph properties or real applications such as clustering. Moreover, its straightforward adaptation to existing algorithms for constrained graph anonymization shows good promises of reusability, not only in privacy. For instance, future works might involve the use of the developed algorithms for updating the coreness in evolving networks through local updates.

Part III

Privacy-preserving approaches

Chapter 7

Random-based methods

In this chapter we devise a simple and efficient algorithm for randomization on networks. Our algorithm considers the edge's relevance, preserving the most important edges of the network in order to improve the data utility and reduce the information loss on anonymous graphs. After a short introduction in Section 7.1, we devise our random-based algorithm in Section 7.2. Then, Section 7.3 presents the results and the re-identification and risk assessment in Section 7.4. Lastly, the conclusions are discussed in Section 7.5.

7.1 Randomization using Edge Neighbourhood Centrality

In this section, we will present the *Rand-NC*[1] (short for Randomization using Edge Neighbourhood Centrality) algorithm, which is designed to achieve random-based privacy on undirected and unlabelled networks. The algorithm performs modifications to the original network $G = (V, E)$ only in edge set (E). Thus, the vertex set (V) remains the same during randomization process.

Our approach is a probabilistic random-based method which considers edge relevance in order to achieve less information loss, and consequently better data utility on anonymous graphs. Our probability distribution is computed according to a relevance value assigned to each edge. This relevance value is computed by a simple metric, called *edge neighbourhood centrality* (details in Section 6.2), which evaluates the 1-neighbourhood information flow through an edge.

[1] Source code available at: `http://deic.uab.cat/~jcasas/`

7.2 Rand-NC algorithm

Our randomization algorithm runs in a two-step approach: the first step removes $w < m$ (usually $w << m$, where $m = |E|$) edges from the original edge set (E). The process starts by computing the edge neighbourhood centrality (NC) score for each edge $\{i, j\} \in E$ using Equation 6.4. Then, it calculates the probability of each edge to be removed as follows:

$$p(\{i, j\}) = \frac{1}{NC^2_{\{i,j\}}} \tag{7.1}$$

According to the power-law property of scale-free real networks [5], notice that there are few edges with high NC values. Hence, we are interested on preserving these edges, since they are bridge-like connectors and are critical for the structure and the information flow on the network. Consequently, the probability of removing these edges during the anonymization process is low, in accordance with Equation 7.1.

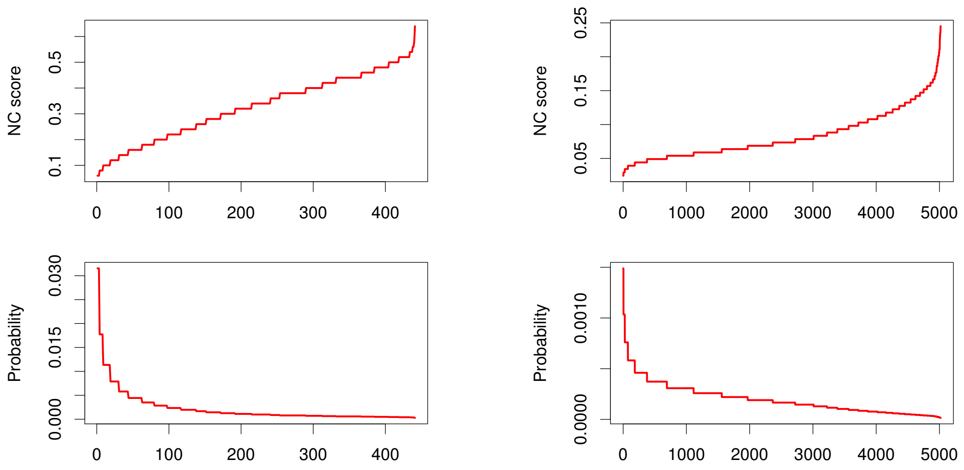
In a second step we add w fake edges $\{i, j\} \notin E$. Let the set of all possible fake edges be the complement of the original graph, which has the same vertex set but whose edge set consists of the edges not present in G.

Let $G^c = (V, E^c)$ be the complement of the graph $G = (V, E)$. We select a random set of $m = |E|$ edges using a uniform probability distribution from E^c, and create a subset $E^s \subseteq E^c$ with $|E^s| = m$. Notice that usually $|E^c| >> |E|$ because of the sparse property of real networks, and therefore we only need to consider a subset of m elements. Next, we compute the NC-score for each edge $\{i, j\} \in E^s$ by adding it to E and computing the score in G using Equation 6.4.

Note that this process considers the score of each edge independently, and possible dependencies between NC-scores of different edges will not be considered. The NC-score for an edge $\{i, j\}$ will be modified if for some $v_u \in V$ an edge $\{i, u\}$ or $\{u, j\}$ is added or deleted, since this implies that the 1-neighbourhood of the vertices v_i or v_j will be changed. Even so, we do not recompute the NC-values after an edge is added or removed due to the computational complexity. As we stated before, one of our goals is keep a low complexity to enable our approach to work with medium or large networks.

Finally, we randomly select w edges from E^s, according to the probability distribution computed by Equation 7.1, and then we add them to the edge set in order to obtain the anonymous version of edge set ($\widetilde{E}$), which fulfils $|E| = |\widetilde{E}|$ and $|E| - |E \cap \widetilde{E}| = w$.

For instance, we can see empirical data about Polbooks network (see Section 2.5 for network's details) in Figure 7.1. Firstly, the NC-score and its probability value on G



(a) Graph G

(b) Complement graph G^c

Figure 7.1: Empirical results of NC-score for each edge and its probability computed by Equation 7.1 on Polbooks network.

are shown in Figure 7.1a. As we can see, edges with high NC-score are assigned to low probability values by Equation 7.1, and vice versa. Secondly, the same values for the complement graph G^c are shown in Figure 7.1b. Note that $|E^c| >> |E|$. In a similar way, edges with low NC-score present higher probability to be added to $\widetilde{G}$ during randomization process than the ones with high NC-score.

7.3 Empirical Results

In this section we will present the results of our randomization method and, in addition, we will compare our method with two basic and relevant methods for randomization on networks. The selected methods are: *Random Perturbation* (in short, RP) and *Random Switch* (RS). In subsequent sections, we will analyse these methods and ours, and we will compare the empirical results on different real networks. Our experimental framework considers 10 independent executions of the randomization methods with a perturbation range between 0% and 25% of total number of edges, i.e, $0 \leq p \leq 25$. Note that 0% perturbation represents the original network. The parameters are the same for all methods and they achieve similar privacy levels, since they apply the same concept to preserve the graph's privacy. Therefore, the evaluation of the results allows us to compare anonymous data in terms of data utility and information loss. Three different real networks are used in our experiments: Zachary's karate club (*Karate*), US politics book data (*Polbooks*) and

Karate	D	h	SC	T	Cor	C_B	C_C	λ_1	μ_2
NC	**0.296**	**0.010**	3.091e	0.028	0.392	0.025	0.047	**0.090**	**0.050**
RP	0.331	0.012	6.812	0.031	0.349	0.030	0.053	0.280	0.058
RS	0.596	0.047	**1.648**	**0.019**	**0.069**	**0.022**	**0.037**	0.119	0.184
Polbooks	D	h	$SC(\times 10^2)$	T	Cor	C_B	C_C	λ_1	μ_2
NC	**1.550**	**0.181**	**6.887**	**0.063**	0.338	0.019	**0.055**	0.208	**0.366**
RP	1.719	0.201	12.555	0.075	0.340	0.020	0.061	0.583	0.580
RS	1.888	0.239	8.564	0.082	**0.110**	0.019	0.070	**0.109**	0.500
URV email	D	h	$SC(\times 10^5)$	T	Cor	C_B	C_C	λ_1	μ_2
NC	0.388	**0.037**	3.696	**0.033**	0.582	0.001	0.175	0.580	0.320
RP	0.508	0.042	6.776	0.038	0.578	0.001	0.147	1.873	0.317
RS	**0.165**	0.094	**3.549**	0.044	**0.211**	0.001	**0.012**	**0.512**	**0.003**

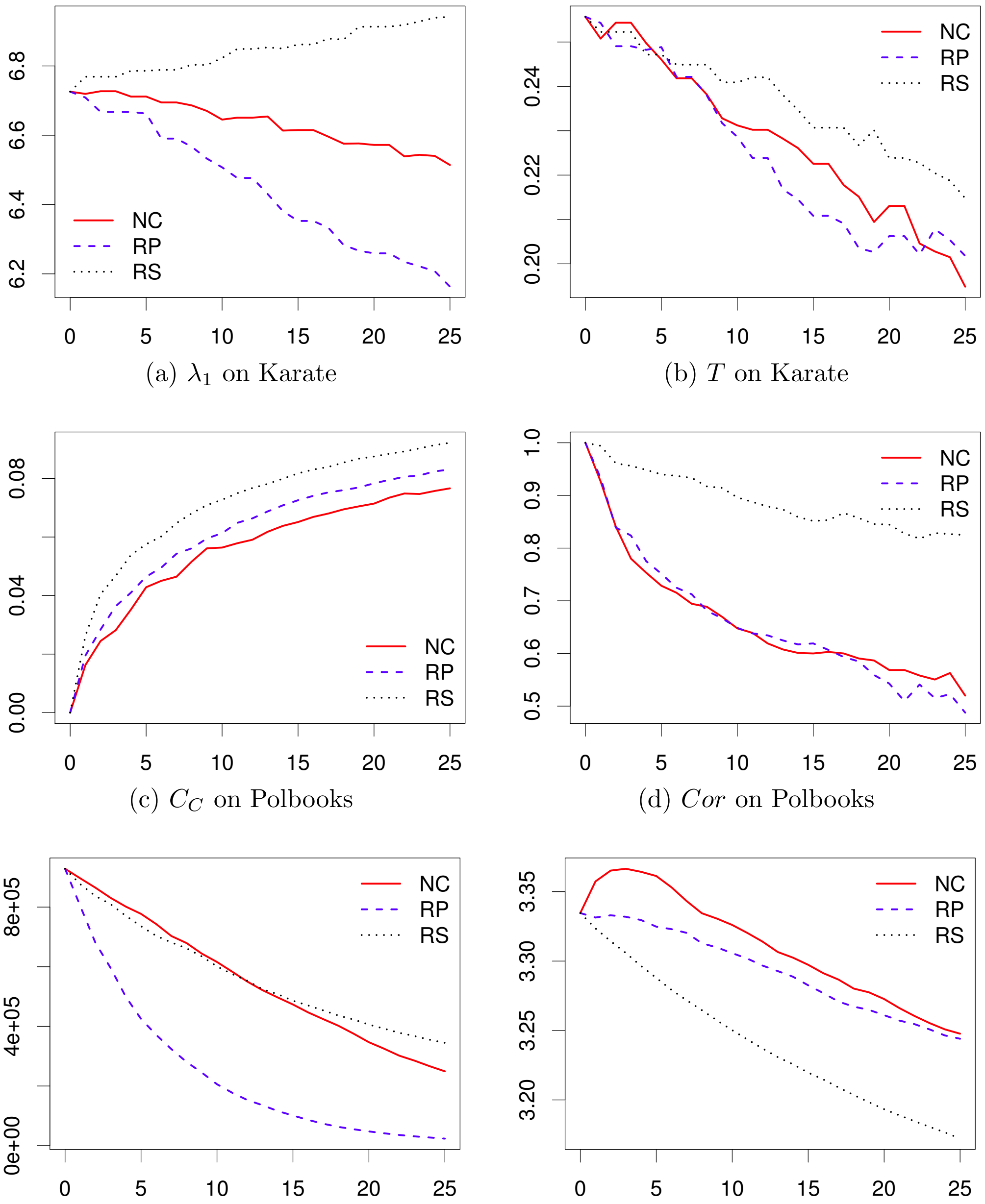
Table 7.1: Results for *Rand-NC* (NC), *Random Perturbation* (RP) and *Random Switch* (RS) algorithms.

URV email (networks' details are provided in Section 2.5).

Results are displayed in Table 7.1. Each cell indicates the error value for the corresponding measure and algorithm. For each dataset and algorithm, we compare the results obtained on D, h, SC, T, Cor, C_B, C_C, λ_1 and μ_2. Bold cells indicate the algorithm that achieves the best result (i.e, lowest information loss) for each measure and dataset. The lower the value, the better the algorithm. Although deviation is undesirable, it is inevitable due to the edge modification process.

The first tested network is Zachary's karate club. As shown in Table 7.1, our algorithm achieves the best results on D, h, λ_1 and μ_2. For instance, we can deepen on the behaviour of λ_1 error in Figure 7.2a. Perturbation parameter p varies along the horizontal axis from 0% to 25%. The $p = 0$ value represents the value of this metric on the original graph. Thus, values close to this point indicate low noise on perturbed data. As we can see, Rand-NC remains closer to the original value than the others algorithms. Random Switch achieves the best results on SC and Cor. Furthermore, it also gets the best results on T, C_B and C_C, but they are quite similar on all randomization methods, suggesting that all of them introduce similar noise on these measures, as we can see in Figure 7.2b.

The second dataset, Polbooks, is a small collaboration network. The results of the comparability are very encouraging. Rand-NC outperforms on all measures, except on Cor and λ_1, where Random Switch achieves lower error values. For instance, we can see the average error of the closeness centrality in Figure 7.2c, which behaves in a similar way for the three tested algorithms. The Cor measure, shown in Figure 7.2d, is based on the



(a) λ_1 on Karate

(b) T on Karate

(c) C_C on Polbooks

(d) Cor on Polbooks

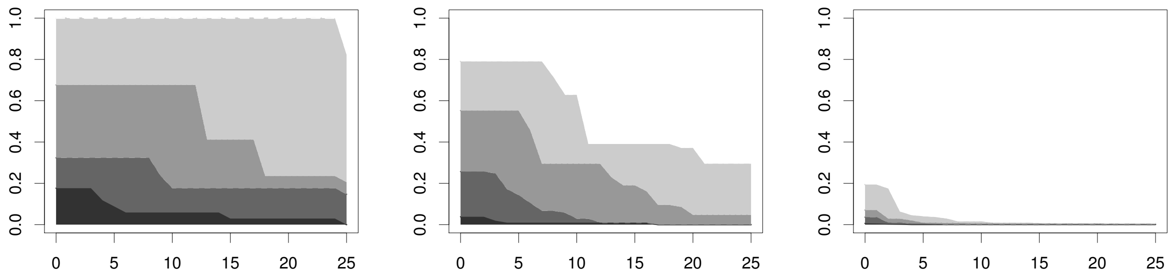
(e) SC on URV email

(f) h on URV email

Figure 7.2: Examples of the error evolution computed on our experimental framework.

k-core decomposition and is closely related to the degree of the vertices. Consequently, it is not surprising that Random Switch gets the best results, since this method does not change the vertices' degree, keeping the degree sequence equal to the original one.

Finally, the last dataset is URV email communication network. The best results on D, SC, Cor, C_C, λ_1 and μ_2 are achieved by Random Switch method. Even so, our algorithm



(a) Karate (b) Polbooks (c) URV email

Figure 7.3: Candidate set size ($Cand_{\mathcal{H}_1}$) evaluation on our three tested networks.

gets good results on λ_1 and SC, where its values are close the ones by Random Switch. Figure 7.2e presents the sub-graph centrality details. In addition, Rand-NC achieves the best results on h and T, as shown in Figure 7.2f.

7.4 Re-identification and risk assessment

Assuming that the randomization method and the number of fake edges w are public, the adversary must consider the set of possible worlds implied by $\widetilde{G}$ and w. Informally, the set of possible worlds consists of all graphs that could result in $\widetilde{G}$ under w perturbations. Using E^c to refer to all edges not present in E, the set of possible worlds of $\widetilde{G}$ under w perturbations, denoted by $\mathcal{W}_w(\widetilde{E})$ is computed by:

$$\mathcal{W}_w(\widetilde{E}) = \binom{|E|}{w}\binom{|E^c|}{w} \tag{7.2}$$

Therefore, and according to Hay et al. [48], the candidate set of a target vertex v_i includes all vertices $v_j \in \widetilde{G}$ such that v_j is a candidate in some possible world. We compute the candidate set of the target vertex v_i based on *vertex refinement queries* of level 1 ($\mathcal{H}_1$) (see Section 3.4.2 for further details) as shown in Equation 7.3.

$$Cand_{\mathcal{H}_1}(v_i) = \{v_j : deg^-(v_j) \leq deg(v_i) \leq deg^+(v_j)\} \tag{7.3}$$

where $deg^-(v_j) = round(deg(v_j)(1 - \frac{w}{m}))$ is the minimum expected degree of v_j and $deg^+ = round(deg(v_j)(1 + \frac{w}{m}))$ is the maximum expected degree after uniformly random edge deletion/addition process.

Using this re-identification model based on vertex degree, we observe that Random Switch method does not change the vertex degree during the randomization process.

Consequently, this method does not improve the privacy under an adversary's knowledge based on the degree value of some target vertices. Contrary, both Rand-NC and Random Perturbation methods modify the degree sequence and hinder the re-identification process according to this adversary knowledge model. Figure 7.3 shows the $Cand_{\mathcal{H}_1}$ evolution on Rand-NC and Random Perturbation methods. Note that both methods achieve the same candidate set, since they apply the same number of edge modifications in our experiments. Perturbation varies along the horizontal axis from 0% (original graph) to 25%. The trend lines show the percentage of vertices whose equivalent candidate set size falls into each of the following groups: [1] (black), [2,4], [5,10], [11,20], [21, ∞] (white). As we can see, candidate set sizes shrink while perturbation increases in all datasets. The number of nodes at high risk of re-identification (the black area) decreases quickly. Nevertheless, well-protected vertices present different behaviour on our tested network. For instance, Zachary's karate club (Figure 7.3a) does not achieve a significant increment on well-protected vertices due to the fact that it is a small network with only 34 vertices and, therefore, a candidate set size of [21, ∞] is probably too strict. Finally, notice that the probability computed in Equation 7.3 corresponds to an uniformly random process, which is exactly what the method Random Perturbation proposes. A slightly difference occurs on Rand-NC, but we omit the experimental results due to the very small difference between them.

7.5 Conclusions

In this chapter we have presented a new algorithm for randomization on networks, which is based on edge set modification according to edge's relevance. Instead of modifying one of the possible edges at random, this method considers the edge's relevance and preserves the most important edges in the network. Our method achieves a better trade-off between data utility and data privacy than other methods. In particular, it outperforms Random Perturbation in terms of data utility and Random Switch in terms of data privacy as empirical results have shown.

We have presented some experimental results on real networks. Even though our algorithm does not perform better than Random Switch in some specific metrics and networks, the privacy level achieved by our method is clearly higher than the one achieved by Random Switch. In addition, Rand-NC gets the best results on several metrics. Therefore, we have demonstrated that our algorithm is better, in terms of data utility and privacy, than the two well-known Random Perturbation and Random Switch algorithms.

Chapter 8

k-Anonymity methods

In this chapter we will focus on k-anonymous methods, specifically on k-degree anonymous methods for undirected networks. We will start by discussing preliminary concepts in Section 8.1. Next, we will present our two algorithms for k-degree anonymity on networks. The first one is based on evolutionary algorithms (Section 8.2), and the second one is based on univariate micro-aggregation (Section 8.3). Then, in Section 8.4 we will compare our methods with other well-known k-degree anonymous methods in terms of data utility and information loss. Finally, we will discuss the conclusions on Section 8.5.

8.1 k-degree anonymity model

We use d to define the degree sequence of G, where $d = \{d_1, d_2, \ldots, d_n\}$ is a vector of length n and d_i is the degree of vertex $v_i \in V$. We refer to the ordered degree sequence as a monotonic non-decreasing sequence of the vertex degrees, that is $d_i \leq d_j \; \forall i < j$.

The degree sequence is an interesting tool since the concept of k-degree anonymity for a network can be directly mapped to its degree sequence, as Liu and Terzi showed in [59] and we recall in the following definitions:

Definition 1 *A vector of integers V is k-anonymous if every distinct value $v_i \in V$ appears at least k times.*

Definition 2 *A network $G = (V, E)$ is k-degree anonymous if the degree sequence of G is k-anonymous.*

Regarding to the degree sequence, notice that:

- The number of elements determines the number of vertices. Thus, this value cannot be altered.
- Each $d_i \in d$ must be an integer in the range $[0, n-1]$, since each d_i is the degree of vertex v_i.
- The total number of edges is half the sum of the degree sequence, i.e, $\sum_{i=1}^{n} d_i = 2m$, since each edge is counted twice in the degree sequence. Therefore, $\sum_{i=1}^{n} d_i = 0 \bmod 2$.

We define the distance between two integer vectors as follows:

$$\Delta(\widetilde{d}, d) = \sum_{i=0}^{n} |\widetilde{d}_i - d_i| \tag{8.1}$$

Δ can be used as a measure of information loss of a protected graph $\widetilde{G}$ with degree sequence $\widetilde{d}$ with respect to the original graph with degree sequence d. In this case, the lower the value of Δ obtained, the lower the information loss of the anonymous graph $\widetilde{G}$.

8.2 Evolutionary Algorithm for Graph Anonymization

In this section we will present our first approach for graph anonymization, called *Evolutionary Algorithm for Graph Anonymization* (EAGA)[1], which is based on evolutionary algorithms and focused on creating k-degree anonymous graphs.

8.2.1 The EAGA algorithm

A high-level description of our proposal allows us to structure our anonymization algorithm in two steps, similar to the approach by Liu and Terzi [59]:

1. In the first step, from the original degree sequence of $G = (V, E)$, $d = \{d_1, \cdots, d_n\}$, we construct a new sequence $\widetilde{d}$ which is k-degree anonymous and minimize the distance Δ from the original sequence computed by Equation 8.1.
2. In the second step, we construct a graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ where $\widetilde{V} = V$, $\widetilde{E} \cap E \approx E$ and the degree sequence is equal to $\widetilde{d}$.

[1] Source code available at: `http://deic.uab.cat/~jcasas/`

The process of creating the k-anonymous degree sequence determines the anonymization level and the distance from the original degree sequence. An optimal sequence has to provide the requested k-anonymity level and has to minimize the distance from the original degree sequence. This last condition is decisive for data utility and information loss.

Step I: Obtaining the k-degree anonymous sequence

Our proposal uses evolutionary algorithms to generate the k-degree anonymous sequence. Algorithm 1 details the steps to generate an anonymous degree sequence.

Algorithm 1 Algorithm pseudo-code for generating k-degree anonymous sequence.

```
Require: Original degree sequence ($d$) and the $k$-anonymity value ($k$).
Ensure: $k$-degree anonymous sequence ($\widetilde{d}$).
  INITIALIZE $population \Leftarrow d$
  $k\_actual \Leftarrow$ GET_K $population$
  while $k\_actual < k$ do
    MUTATE $population$
    EVALUATE $new\ candidates$
    $population \Leftarrow$ SELECT $individuals$
    $k\_actual \Leftarrow$ GET_K $individuals$
  end while
  $\widetilde{d} \Leftarrow$ SELECT $best\ candidate$
  return $\widetilde{d}$
```

As we have shown in Algorithm 1, population is initialized from original degree sequence. Next, the sentences in the *while* loop are the generation step. Here we apply the basic mutation process (MUTATE function in Algorithm 1) which adds one to an element of the sequence and subtracts one to another element of the sequence. This operation represents edge rotation, which is one of the most basic edge modifications on a graph. For instance, if an edge $\{v_i, v_j\}$ is modified by replacing one node, one can obtain $\{v_i, v_p\}$. This edge modification is represented on the degree sequence as a subtraction on node v_j (because it decreases its degree) and a addition on node v_p (because it increases its degree). It is important to note that our algorithm does not use *crossover* since this operation systematically breaches the rule that preserves the number of edges of the graph, generating invalid candidates. We consider the performance of the algorithm would be affected by the inclusion of this type of evolution, and improvements would not occur in time or quality of the solution found.

When candidate generation is done, we evaluate the candidates in order to find the best one. The score of each candidate is determined by the fitness function (EVALUATE function in Algorithm 1). This function assigns different score punctuation whether each individual fulfils the desired k or not. Individuals who do not meet the desired k-anonymity value are scored in range [0,1] considering two parameters:

- The number of vertices which do not fulfil the k-anonymity.
- The dispersion level, computed as the average distance from all nodes to the mean degree value.

On the contrary, individuals who fulfil the desired privacy level are scored in range [1,2] considering only one parameter:

- The distance from the original sequence. The target is to minimize this value, as we described in Equation 8.1.

Finally, the candidate selection uses the *steady-state* model. According to it, the worst candidates of the actual generation are replaced by the best candidates of the new generation.

Step II: Modifying the original graph

The result of the first step is a k-degree anonymous sequence. Then, in the second step we apply the necessary modifications to original graph in order to obtain the k-anonymous one. The anonymized k-degree sequence informs the degree for each node on anonymized graph. Therefore, the difference between original and anonymized degree sequence points to vertices which have to increase or decrease their degree. Hence, we have to add or remove edges to/from these vertices.

As we can see in Algorithm 2, this step begins computing the difference vector, $\delta = \widetilde{d} - d$, which allows us to easily detect which vertices have to increase or decrease their degree. The algorithm removes incident edges to vertices which have to decrease their degree, while it adds new edges to vertices which have to increase their degree. We apply these modifications removing the edge $\{v_i, v_j\} \in E$, where v_j belongs to vertices which have to decrease their degree, and adding a new edge $\{v_i, v_p\}$, where v_p belongs to vertices which have to increase their degree.

Algorithm 2 Algorithm pseudo-code for modifying the original network.

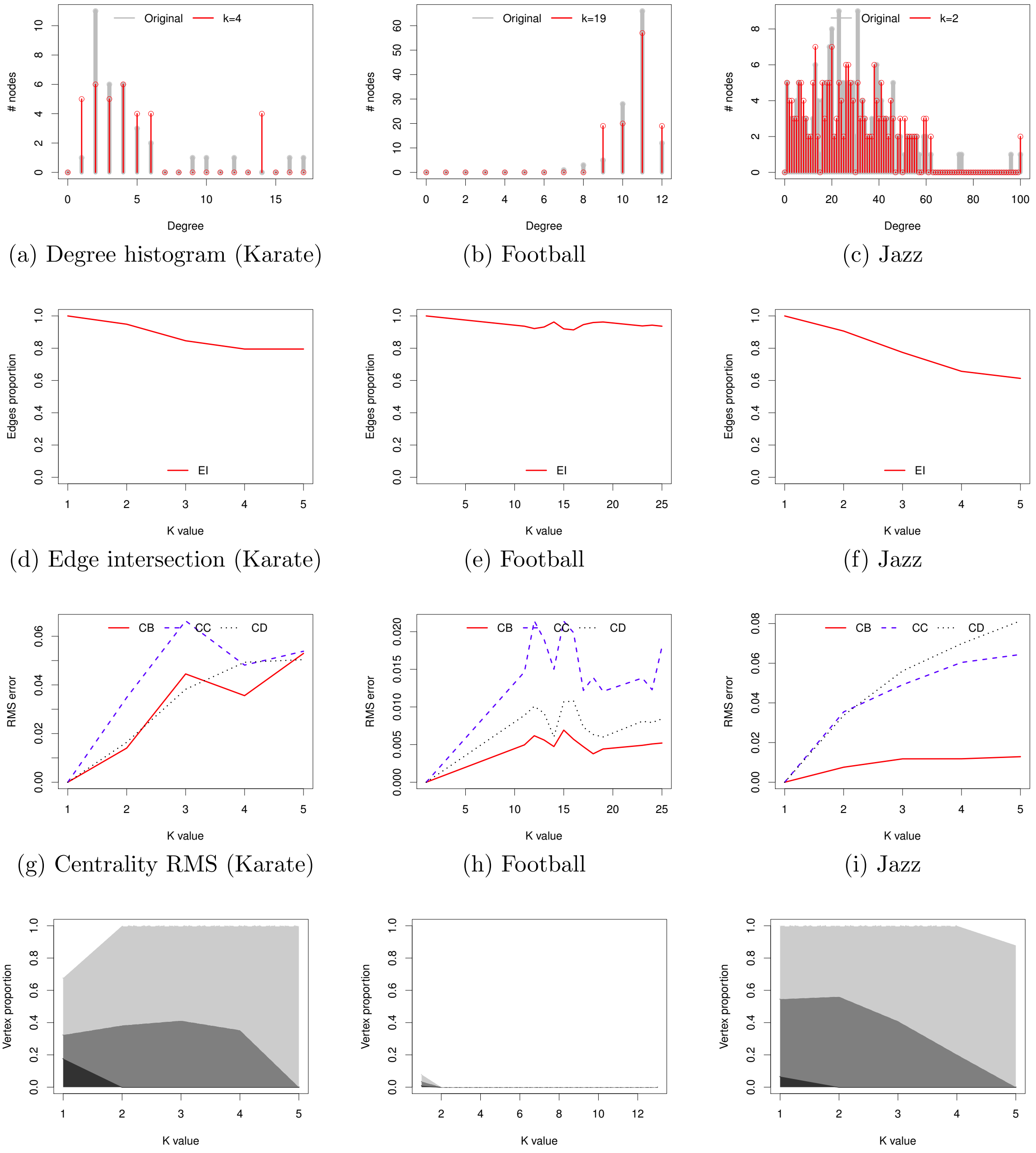
```
Require: Original graph $G = (V, E)$, original degree sequence $d$ and the $k$-degree anonymous sequence $\widetilde{d}$.
Ensure: The graph $\widetilde{G} = (V, \widetilde{E})$ where the degree sequence is $\widetilde{d}$ and $\widetilde{E} \cap E \approx E$.
  $\widetilde{G} = (V, \widetilde{E}) \Leftarrow \widetilde{G}_0 = (V, \widetilde{E})$
  $d_{dif} = d - \widetilde{d}$
  $V_{del} = \{v_i \in V : d_{dif}(i) < 0\}$
  $V_{add} = \{v_i \in V : d_{dif}(i) > 0\}$
  while $V_{del} \neq \emptyset$ and $V_{add} \neq \emptyset$ do
    $\widetilde{E} = \widetilde{E} \setminus \{v_i, v_j\}$ where $\{v_i, v_j\} \in E$ and $v_j \in V_{del}$
    $V_{del} = V_{del} \setminus v_j$
    $\widetilde{E} = \widetilde{E} \cup \{v_i, v_p\}$ where $\{v_i, v_p\} \notin E$ and $v_p \in V_{add}$
    $V_{add} = V_{add} \setminus v_p$
  end while
  return $\widetilde{G}$
```

8.2.2 Experimental results

For each dataset, we analyse the evolution of the degree histogram, comparing the histogram on anonymous and original graphs. We use edge intersection (EI) to quantify the number of edges which were on the original graph and still are on the anonymous one. Clearly, the higher the value, the less the perturbation. In addition, we use betweenness (C_B), closeness (C_C) and degree (C_D) centrality measures (See 4.2 for details) to quantify the noise introduced in the anonymous data.

Lastly, we want to evaluate how the vertex set evolves during the process of anonymization. In order to do this, we use the *vertex refinement queries* (see Section 3.4.2 for further details). We use the $cand_{\mathcal{H}_1}$ to analyse the evolution of vertices, in terms of k-degree anonymity, during the anonymization process. Results are shown in Figure 8.1. The first row contains the degree histogram of original (grey) and anonymous (red) graphs. The second one contains the edge intersection as a function of k (horizontal axis). The following row presents the root mean square (RMS) of the three used centrality measures (i.e, betweenness, closeness and degree centrality) and finally, the fourth row shows the k-degree analysis performed by $Cand_{\mathcal{H}_1}$.

The first dataset is a small social network with 34 nodes, 78 edges and a k-degree anonymity value equal to 1. EAGA algorithm anonymizes it to k values equal to 2, 3, 4 and 5. Figure 8.1a shows the original and $k = 4$ anonymous degree histogram. As we can see, the degree histogram of original graph follows the power-law (total number of



(a) Degree histogram (Karate) (b) Football (c) Jazz

(d) Edge intersection (Karate) (e) Football (f) Jazz

(g) Centrality RMS (Karate) (h) Football (i) Jazz

(j) $Cand_{\mathcal{H}_1}$ (Karate) (k) Football (l) Jazz

Figure 8.1: EAGA experimental results for Zachary's karate club, American football college and Jazz musicians networks.

nodes exponentially decrease when degree value grows). Edge intersection is shown in Figure 8.1d. An anonymous graph with $k = 2$ the algorithm achieves an edge intersection value of 94.87%, while this value descends to 79.49% when k-anonymity value is equal to 5. Figure 8.1g shows the RMS on the three centrality measures. Clearly, we want to keep these values close to zero, since the lower the values, the less noise introduced on anonymous data. Though, the error increases as the k value grows. Finally, we show the evolution of $Cand_{\mathcal{H}_1}$ as an extension to k-degree anonymity value in Figure 8.1j, since this information allows us to see how vertices evolve in terms of re-identification during the anonymization process. The black area shows the percentage of vertices which can be directly re-identified (i.e, vertices with a unique degree value), the dark grey area shows the percentage of vertices with high risk of re-identification (i.e, groups between 2 and 4 vertices with the same degree value), the light grey area shows percentage of vertices with moderate risk of re-identification (i.e, groups between 5 and 10 vertices with the same degree value) and, finally, the white area shows percentage of vertices with low or very low risk of re-identification (i.e, groups of more than 11 vertices with the same degree value). The number of vertices which can be directly re-identified (black area) descends to zero when k achieves a value of 2, while the number of vertices in high risk of re-identification (dark grey area) does not fall to zero until $k = 5$. When k is equal to 5, all vertices are in moderate risk of re-identification.

The second dataset is a collaboration network with 115 nodes, 613 edges and a value of $k = 1$. EAGA algorithm anonymizes it to a k values in range 11-19, 23, 24 and 25. The degree histogram has experienced just a few number of modifications in order to achieve a k-anonymity value of 19, as we can see in Figure 8.1b. Accordingly, this particular graph structure only needs a few edge modifications to achieve a great k-anonymity value. It is important to note that more than 93% of edges remain the same in all anonymous graphs (96.25% in graph with $k = 19$ and 93.64% in graph with $k = 25$), as Figure 8.1e shows. Centrality measures (Figure 8.1h) show irregular perturbation on anonymous graphs. Finally, $Cand_{\mathcal{H}_1}$ is shown in Figure 8.1k and it presents an important decrease on groups with direct, high and moderate re-identification risk until a zero value has been reached when the k-anonymity value is equal to 11. From this k value on, all nodes are well-protected.

The third and the last dataset is a collaboration network with 198 nodes, 2,742 edges and a k-degree anonymity value equal to 1. This graph presents an average degree quite higher than others, close to 27 edges/vertex. The degree histogram in Figure 8.1c reveals the existence of two outlier vertices (usually called *hubs*), with degree values of 96 and 100. It is important to underline that these two nodes are in danger of direct re-identification, but at the same time, and due to their important centrality role, they are key vertices for the graph structure. EAGA algorithm anonymizes the graph to a k values equal to

2, 3, 4 and 5. The perturbation on anonymous data is higher than the previous datasets, since the problem of the outlier values. It will be necessary to modify the two hubs of the graph in order to get $k \geq 3$ values and, therefore, those hubs will lose their centrality and produce high perturbation on anonymous data, reducing significantly the data utility. The anonymized graph with $k = 2$ presents a high percentage of edge intersection, but it falls to 61% for a $k = 5$, as we can see in Figure 8.1f. Figure 8.1i presents the error of the centrality measures, which indicates that small perturbations have been introduced on the anonymous graph, specifically on betweenness centrality. Finally, the $Cand_{\mathcal{H}_1}$ is presented in Figure 8.1l and shows a reduction of nodes in directly and high risk of re-identification. The $k = 5$ anonymous dataset presents the 90% of its vertices in moderate risk of re-identification, while only the 10% is well-protected.

8.3 Univariate Micro-aggregation for Graph Anonymization

In this section, we present our second approach to k-degree anonymity, which is called *Univariate Micro-aggregation for Graph Anonymization* (UMGA)[2] algorithm. It is designed to achieve k-degree anonymity on large, undirected and unlabelled networks. The algorithm performs modifications to the original network $G = (V, E)$ only on edge set (E). Hence, the vertex set (V) does not change during anonymization process.

8.3.1 The UMGA algorithm

Like in the algorithm described in Section 8.2, this one is based on a two-step approach:

1. *Degree Sequence's Anonymization.* We construct a k-degree anonymous sequence $\widetilde{d} = \{\widetilde{d}_1, \ldots, \widetilde{d}_n\}$ from the degree sequence $d = \{d_1, \ldots, d_n\}$ of the original network $G = (V, E)$ using Definition 1. In addition, we use the function Δ from Equation 8.1 to reduce the distance between the anonymous sequence and the original one.

2. *Graph modification.* We build a new graph $\widetilde{G} = (V, \widetilde{E})$ where its degree sequence is equal to $\widetilde{d}$ by using basic edge modification operations. These operations allow us to modify the network's structure according to the anonymous degree sequence $(\widetilde{d})$. By Definition 2, the anonymous graph $\widetilde{G}$ will be k-degree anonymous.

[2] Source code available at: `http://deic.uab.cat/~jcasas/`

8.3.1.1 Degree Sequence's Anonymization

The objective of this first step is to anonymize the degree sequence of the original network (d), constructing a k-anonymous degree sequence ($\widetilde{d}$). Using Definition 1 we have to modify the values of d in order to create groups of k or more elements. Our method uses the optimal univariate micro-aggregation [43] to achieve the best group distribution and then it computes the values for each group that minimize the distance Δ from the original degree sequence.

Without loss of generality, we assume d to be an ordered degree sequence of the original network. Otherwise, we apply a permutation f to the sequence to reorder the elements. Let k be an integer such that $1 \leq k < n$ which is the k-degree anonymity value. Typically, k is much smaller than n. In order to apply the optimal univariate micro-aggregation, and according to Hansen and Mukherjee [43], we construct a new directed network $H_{k,n}$ and get the optimal partition which is exactly the set of groups that corresponds to the arcs of the shortest path from vertex 0 to vertex n on this network. We denote by g the optimal partition, where g has $\frac{n}{k} \leq p \leq \frac{n}{2k-1}$ groups and each of them (g_j) has between k and $2k-1$ items. Obviously, each $d_i \in d$ belongs to a specific group g_j.

Next, we compute the matrix of differences, $\mathcal{M}_{p\times 2}$, using each group of the partition. The first column contains the sum of differences between each element of the group and the arithmetic mean of all degrees that belong to the group, using floor function to round the mean value. The second column is computed in the same way, but using de ceiling function instead. Conceptually, $\mathcal{M}$ contains the number of degrees in the first column, which we should decrease in this group to fulfil k-degree anonymity. These values are always zero or positive. The second column contains the number of degrees which we should increase in this group to reach k-degree anonymity. These values are always negative or zero. Formally, for $j = \{1, \ldots, p\}$ each element m_{ji} is computed as:

$$m_{j1} = \sum_{d_i \in g_j} \left(d_i - \left\lfloor \langle g_j \rangle \right\rfloor \right) \tag{8.2}$$

$$m_{j2} = \sum_{d_i \in g_j} \left(d_i - \left\lceil \langle g_j \rangle \right\rceil \right) \tag{8.3}$$

where $\langle g_j \rangle$ is the average value of $d_i \in g_j$.

A group with zero values on both columns should not apply any modification on its items because it is already k-anonymous.

Example 1 *A group* $g_1 = \{2, 2, 2\}$ *generates* $[0, 0]$ *as the first row of* M*. So the items of the group* g_1 *do not need to modify their values. However, a group* $g_2 = \{3, 4, 4\}$ *generates the row* $[2, -1]$*. Thus, there are two possibilities to anonymize the group: decrease the values of the second and third items to 3 or increase the value of the first item to 4.*

Now we have to compute a solution, that is a p-sequence values where each $m_j \in \{m_{j1}, m_{j2}\}$. The closer $|\sum_{j=1}^{p} m_j|$ to 0, the better the solution. This research is a complex operation with cost $\mathcal{O}(2^q)$ where q is the number of groups with $m_j \neq 0$. Because of the power law on real networks, q is typically much smaller than p, but still too high for large networks.

For that reason, we propose two methods in order to achieve the best combination on reasonable time:

The Exhaustive Method

The first approximation is based on exhaustive search. This method selects the values m_{j1} or m_{j2} for $j = \{1, \ldots, p\}$ so that the minimum value of $|\sum_{j=1}^{p} m_j|$ is achieved. To do this selection, all possible combinations are considered unless a solution is found with $\sum_{j=1}^{p} m_j = 0$.

The Greedy Method

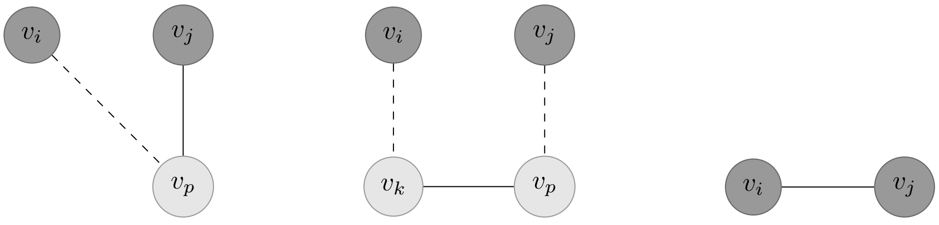
We define an iterative method and in each step values for m_j are selected according to a probability distribution based on the size of values m_{j1} and m_{j2}. The lower the value is, the more probability to be chosen. More specifically,

$$p(m_j = m_{j1}) = 1 - \left(\frac{m_{j1}}{m_{j1} + m_{j2}}\right) \tag{8.4}$$

$$p(m_j = m_{j2}) = 1 - \left(\frac{m_{j2}}{m_{j1} + m_{j2}}\right) \tag{8.5}$$

Iterations are finished when a solution is found with $\sum_{j=1}^{p} m_j = 0$ or when we have a fixed number of iterations without any improvement in the function $\sum_{j=1}^{p} m_j$.

Once exhaustive or greedy methods have finished, we have a k-degree anonymous sequence $(\widetilde{d})$ which minimizes the distance computed by Equation 8.1. However, notice that in case we start the process with a non-ordered degree sequence, we should apply here the inverse transform f^{-1} to obtain the correct degree sequence.



(a) Edge rotation (b) Edge removal (c) Edge addition

Figure 8.2: Basic operations for network modification with vertex invariability. Dashed lines represent deleted edges while solid lines are the added ones.

8.3.1.2 Graph modification

In the second step, changes are made on the original network in order to convert it to a k-degree anonymous graph. By Definition 2, a graph is k-degree anonymous if its degree sequence is k-anonymous. Therefore, we have to change its degree sequence from the original one (d) to the anonymous one ($\widetilde{d}$) which we computed in the step above. We have to modify the edge set to meet the k-anonymous degree sequence.

In order to modify the edge set of a given network, we define three basic operations:

- *Edge rotation* between three vertices can be defined as follows: if $v_i, v_j, v_p \in V$, $\{v_i, v_p\} \in E$ and $\{v_j, v_p\} \notin E$, we can delete $\{v_i, v_p\}$ and create $\{v_j, v_p\}$. Figure 8.2a shows this basic operation. Such modification can be translated in the degree sequence as $\widetilde{d_i} = d_i - 1$ and $\widetilde{d_j} = d_j + 1$, where $\widetilde{d}$ is the degree sequence after the basic operation. Notice that the number of edges in the graph remains the same, v_i decreases its degree by one, v_j increases its degree by one and v_p keeps the same degree.

- *Edge removal* is defined as follows: we select four vertices $v_i, v_j, v_k, v_p \in V$ where $\{v_i, v_k\} \in E$, $\{v_j, v_p\} \in E$ and $\{v_k, v_p\} \notin E$. We delete edges $\{v_i, v_k\}$ and $\{v_j, v_p\}$, and create a new edge $\{v_k, v_p\}$, as shown in Figure 8.2b. Note that $\widetilde{d_i} = d_i - 1$, $\widetilde{d_j} = d_j - 1$, $\widetilde{d_k} = d_k$ and $\widetilde{d_p} = d_p$.

- Finally, we define *Edge addition* as simply selecting two vertices $v_i, v_j \in V$ where $\{v_i, v_j\} \notin E$ and creates it. Note that $\widetilde{d_i} = d_i + 1$ and $\widetilde{d_j} = d_j + 1$. It is depicted in Figure 8.2c.

The vector $\widetilde{d} - d$ indicates which vertices have to modify their degree. In fact, $\widetilde{d} - d$ indicates precisely which vertices must increase or decrease their degree to fulfil the desired k-degree anonymity. The changes on original edges are performed using the basic operations depicted in Figure 8.2.

The graph modification step starts by obtaining the vector of changes between the k-degree anonymous sequence and the original one, $\delta = \widetilde{d} - d$. Vector δ allows to easily detect which vertices have to reduce or increase their degree. From δ, we create the list of vertices which must decrease their degree, $\delta^- = \{v_i : \delta_i < 0\}$ and the list of vertices which must increase their degree, $\delta^+ = \{v_i : \delta_i > 0\}$.

Let $\sigma(d)$ be the sum of each element in d, $\sigma(d) = \sum_{i=1}^{n} d_i$, and let $\sigma(\widetilde{d})$ be the sum of each element in $\widetilde{d}$, $\sigma(\widetilde{d}) = \sum_{i=1}^{n} \widetilde{d}_i$. Then, if $\sigma(d) = \sigma(\widetilde{d})$, it implies that there are the same number of edges in the original and in the anonymous graphs, and therefore, we do not need to increase or decrease the total number of edges and we only need to apply edge rotation modifications. Otherwise, we have to increase or decrease the total number of edges in order to reach the k-anonymous sequence.

If $\sigma(\widetilde{d}) < \sigma(d)$, we need to decrease $|\frac{\sigma(d)-\sigma(\widetilde{d})}{2}|$ edges from the graph, as we have shown on Figure 8.2b. In order to decrease by 1 the total number of edges from the graph, we choose $v_i, v_j \in \delta^-$ and find other two vertices v_k, v_p where $\{v_i, v_k\} \in E$ and $\{v_j, v_p\} \in E$. Then we delete these two edges and create a new one $\{v_k, v_p\}$. On the other hand, if $\sigma(\widetilde{d}) > \sigma(d)$, we need to increase $|\frac{\sigma(\widetilde{d})-\sigma(d)}{2}|$ edges to the graph, as we have seen on Figure 8.2c. To increase by 1 the total number of edges, we select $v_i, v_j \in \delta^+$ where $\{v_i, v_j\} \notin E$ and create it. Lastly, we have to modify the degree of some vertices, until $\sigma(\widetilde{d}) = \sigma(d) = 0$. This modification is done using edge rotation, Figure 8.2a. For each $v_j \in \delta^-$ and $v_p \in \delta^+$, we find another vertex v_i where $\{v_i, v_j\} \in E$. We delete this edge and create a new one $\{v_i, v_p\}$.

We suggest two approaches to select the auxiliary edges needed for graph modification process.

Random edge selection

We can simply select at random the auxiliary edges, that is, $\{v_i, v_k\}$ and $\{v_j, v_p\}$ for edge removal and $\{v_i, v_j\}$ for edge rotation. Obviously, this is the fastest way to select the needed auxiliary edges, but some important edges can be removed or new bridge-like edges can be created by this random approach, affecting considerably the local or global structure of the resulting network.

Neighbourhood centrality edge selection

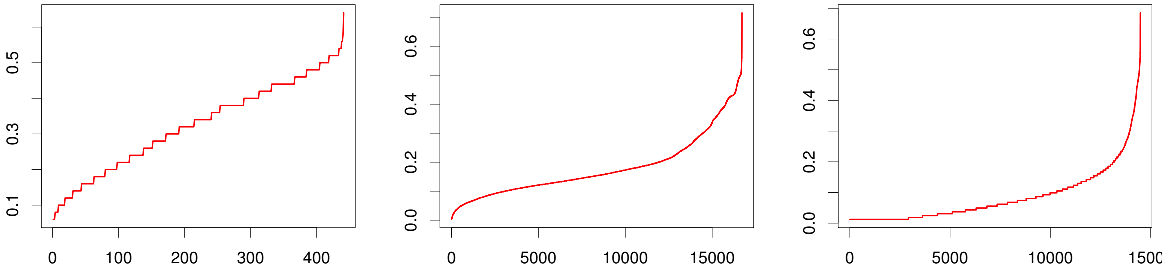
Alternatively, we can select the auxiliary edges by considering the relevance of each edge. Using this approach, we can remove or create new edges with low relevance, which leads the process to a low information loss results. For instance, if we remove edges with low value of edge betweenness, we will preserve the network information flow on the resulting network, since we preserve the most important edges using a measure related to information flow. Therefore, identifying a relevant measure is important for reducing the information loss. In addition, we have to choose a measure with a low complexity, since we will compute it many times. For instance, the measure commented above is not a good choice because calculating edge betweenness on all edges on a network involves calculating the shortest paths between all pairs of vertices. This takes $\mathcal{O}(n^3)$ time with the Floyd-Warshall algorithm. On a sparse network, Johnson's algorithm may be more efficient, taking $\mathcal{O}(n^2 log(n) + nm)$ time. Hence, edge betweenness is not useful for large networks, since every time we remove an edge we must re-calculate all edges' values.

We proposed a measure called *Edge neighbourhood centrality* (NC) (see Section 6.2 for details) for quantifying the edge's relevance on large networks. We have demonstrated that edge neighbourhood centrality identifies the most important edges on a network with low complexity, and therefore, this measure is able to work on medium or large networks.

8.3.1.3 Complexity

The first step, degree sequence anonymization, can be divided into two phases: the first one computes the optimal partition in a polynomial time, $\mathcal{O}(max(nlog(n), k^2 n))$ as the authors [43] stated. The second one computes the p-sequence values, which represents the k-anonymous degree sequence $(\widetilde{d})$. The problem is NP for the exhaustive method ($\mathcal{O}(2^q)$ where q is the number of groups with $m_j \neq 0$), but using the greedy approach we can get a quasi-optimal solution in polynomial time ($\mathcal{O}(wq)$ where w is the number of fixed iterations. For example, if we explore $w = log_2(2^q) = q$, then the complexity is polynomial $\mathcal{O}(q^2)$). So, the problem of degree sequence anonymization is a P-problem and can be resolved in polynomial time for a quasi-optimal solution.

The second step, graph modification, can be implemented by random edge selection or neighbourhood centrality edge selection. As we have seen, each single edge operation can be computed in linear time, and therefore random edge selection also can be computed in linear time. Neighbourhood centrality edge selection involves evaluating all, or at least, a fraction of all possible combinations to preserve the most important edges. For each



(a) Polbooks (b) Polblogs (c) GrQc

Figure 8.3: Neighbourhood centrality values of tested networks.

vertex v_i and v_j, we can generate several possible candidates:

- Edge rotation: Given v_i and v_j we select all $v_p \in \Gamma(v_i)$. Usually, we will find $\overline{deg}(G)$ candidates for v_p or $deg^{max}(G)$ candidates when v_i is a hub or a vertex with a high degree. Hence, we have to check $\overline{deg}(G)$ or $deg^{max}(G)$ possible combinations for each pair of vertices. It is important to note that, because of the power-law on real networks, the vertices with the highest degree do not usually meet the k-anonymity and they need to be modified.

- Edge removal: Given v_i and v_j we can select $v_k \in \Gamma(v_i)$ and $v_p \in \Gamma(v_j)$ as candidates. In average, there are $\overline{deg}(G)$ possible vertices for v_k and v_p, and $deg^{max}(G)$ in the worst case. For every pair, we have to check whether edge $\{v_k, v_p\} \notin E$. Therefore, we have to check $\overline{deg}(G)^2$ possible combinations in the average case, but it should be $deg^{max}(G)^2$ in the worst case, when v_i and v_j are hubs-like or vertices with a high degree value.

- Edge addition: Given v_i and v_j we only need to check whether edge $\{v_i, v_j\} \notin E$. Only one possible combination for every pair of vertices is found here.

The complexity of edge evaluation process is NP. Accordingly, we need to use an heuristic to reduce the complexity of evaluating all possible combinations for each basic edge operation. Figure 8.3 shows the neighbourhood centrality values of tested networks (see Table 2.2 for network details). The horizontal axis shows the edges while the vertical one presents the neighbourhood centrality value. As we can see, neighbourhood centrality values are distributed like a power-law, where most edges have low score values and only few edges have a high score. Figures 8.3b and 8.3c show clearly this phenomenon. Therefore, we focus on preserving the edges with high score of neighbourhood centrality.

According to the distribution observed on real networks, we do not have to evaluate all possible combinations, we only need to evaluate a few of them to preserve the most important edges. Evaluating only the log_2 of the total possible combinations we will find, with a very high probability, an edge with low NC score value and we will reduce the complexity of edge selection. The complexity reduction is very important to deal with large networks and so is our algorithm, as we will demonstrate on Section 8.3.3. Summarizing, the problem of graph modification is a P-problem for random edge selection and NP for neighbourhood centrality edge selection, but it can be resolved in polynomial time by computing the log_2 of all possible combinations.

8.3.2 Information loss and data utility

In this section we will compare the result of anonymizing several networks using UMGA algorithm with random edge selection (UMGA-R) and neighbourhood centrality edge selection (UMGA-NC). We apply both algorithms on the same data with the same parameters and compare the results in terms of information loss and data utility. We use several structural and spectral measures in order to quantify the information loss from distinct perspectives or network's characteristics. It is important to note that the privacy level is the same for all algorithms, as we compare results with the same k value.

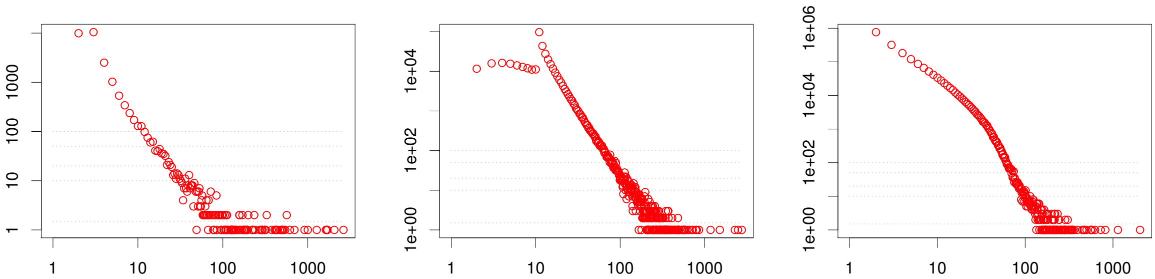
Results are disclosed in Table 8.1. For each dataset and algorithm, we vary k from 1 to 10 ($k = 1$ corresponds to original dataset) and compare the results obtained on λ_1, μ_2, $\overline{dist}$, h, Q, T and SC. The last column correspond to the average error $\langle \mathcal{E} \rangle$. Each row indicates the scored value for the corresponding measure and algorithm, and each column corresponds to an experiment with a different k-anonymity value. Each characteristic is reported twice, corresponding to UMGA-R (indicated by R) and UMGA-NC (indicated by NC). Bold rows indicate the algorithm that achieves the best results (i.e., lowest information loss) for each measure. Perfect performance in a row would be indicated by achieving exactly the same score as in the original network (the $k = 1$ column). Although deviation is undesirable, it is inevitable due to the edge modification processes.

The first tested network, Polbooks, is a small collaboration network. The results of the comparability are very encouraging. UMGA-NC outperforms on all measures, except on μ_2 where the values are close to UMGA-R and the average error $\langle \mathcal{E} \rangle$ is almost equal, as shown in Table 8.1. Polblog is the second tested network. UMGA-R and UMGA-NC obtain similar values. Finally, GrQc is larger than other networks (in terms of number of vertices), so results for this network are evaluated on k in range [1,50] instead of [1,10].

Polbooks		k=1	2	3	4	5	6	7	8	9	10	$\langle\mathcal{E}\rangle$
λ_1	R	11.93	12.11	12.00	12.17	11.82	11.84	12.18	12.25	12.25	11.89	0.163
	NC		**12.09**	**11.97**	**11.85**	**11.85**	**11.95**	**12.09**	**12.08**	**12.08**	**11.86**	**0.090**
μ_2	**R**	0.324	**0.359**	**0.428**	**0.497**	**0.330**	**0.398**	**0.653**	**0.593**	**0.593**	**0.495**	**0.143**
	NC		0.360	0.451	0.453	0.453	0.383	0.599	0.524	0.524	0.640	0.147
$\overline{dist}$	R	3.079	2.928	2.826	2.770	3.029	2.861	2.647	2.694	2.694	2.795	0.247
	NC		**2.987**	**2.883**	**2.896**	**2.896**	**2.988**	**2.765**	**2.856**	**2.856**	**2.762**	**0.182**
h	R	2.450	2.392	2.343	2.314	2.428	2.356	2.252	2.276	2.276	2.326	0.109
	NC		**2.416**	**2.371**	**2.379**	**2.379**	**2.418**	**2.312**	**2.350**	**2.350**	**2.312**	**0.077**
Q	R	0.402	0.400	0.396	0.388	0.398	0.389	0.377	0.379	0.379	0.394	0.012
	NC		**0.400**	**0.393**	**0.396**	**0.396**	**0.401**	**0.386**	**0.386**	**0.386**	**0.385**	**0.009**
T	R	0.348	0.350	0.330	0.316	0.330	0.325	0.298	0.304	0.304	0.313	0.027
	NC		**0.350**	**0.342**	**0.339**	**0.339**	**0.347**	**0.326**	**0.322**	**0.322**	**0.324**	**0.013**
$SC(\times 10^3)$	R	2.524	2.779	2.189	2.372	2.145	1.933	2.132	2.336	2.336	1.976	0.303
	NC		**2.774**	**2.358**	**2.224**	**2.224**	**2.338**	**2.363**	**2.389**	**2.389**	**2.110**	**0.204**
Polblogs		k=1	2	3	4	5	6	7	8	9	10	$\langle\mathcal{E}\rangle$
λ_1	R	74.08	73.88	73.78	73.95	73.93	73.67	73.70	73.82	73.74	73.63	0.260
	NC		**73.93**	**73.81**	**73.92**	**73.95**	**73.74**	**73.80**	**73.75**	**73.63**	**73.61**	**0.256**
μ_2	R	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.000
	NC		0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.000
$\overline{dist}$	**R**	2.738	**2.724**	**2.721**	**2.739**	**2.737**	**2.730**	**2.731**	**2.730**	**2.732**	**2.728**	**0.007**
	NC		2.733	2.729	2.725	2.724	2.724	2.732	2.726	2.731	2.727	0.009
h	**R**	2.506	**2.496**	**2.494**	**2.507**	**2.505**	**2.500**	**2.501**	**2.501**	**2.502**	**2.500**	**0.005**
	NC		2.501	2.499	2.496	2.496	2.496	2.502	2.498	2.502	2.499	0.006
Q	**R**	0.405	**0.403**	**0.403**	**0.405**	**0.405**	**0.403**	**0.403**	**0.403**	**0.403**	**0.402**	**0.001**
	NC		0.404	0.403	0.403	0.403	0.403	0.403	0.402	0.403	0.402	0.002
T	R	0.226	0.224	0.223	0.225	0.224	0.223	0.223	0.224	0.223	0.223	0.002
	NC		**0.224**	**0.224**	**0.224**	**0.224**	**0.223**	**0.225**	**0.224**	**0.223**	**0.224**	**0.001**
$SC(\times 10^{29})$	R	1.218	1.003	0.905	1.069	1.054	0.806	0.832	0.939	0.872	0.782	0.270
	NC		**1.052**	**0.932**	**1.040**	**1.068**	**0.871**	**0.921**	**0.875**	**0.776**	**0.765**	**0.266**
GrQc		k=1	5	10	15	20	25	30	35	40	50	$\langle\mathcal{E}\rangle$
λ_1	R	45.61	45.12	44.77	43.99	43.23	43.28	41.89	42.01	41.76	41.62	2.450
	NC		**45.37**	**45.28**	**44.78**	**44.14**	**44.49**	**43.02**	**43.72**	**43.55**	**43.05**	**1.353**
$\mu_2(\times 10^{-14})$	R	-1.26	-3.98	-2.37	-4.09	-3.33	-2.67	-3.55	-2.00	-1.91	-4.40	1.966
	NC		**-3.06**	**-2.71**	**-1.95**	**-2.01**	**-4.34**	**-1.57**	**-2.35**	**-2.76**	**-2.48**	**1.300**
$\overline{dist}$	R	6.049	6.002	5.942	5.918	5.862	5.861	5.864	5.892	5.816	5.915	0.150
	NC		**6.026**	**6.009**	**5.955**	**5.948**	**5.922**	**5.904**	**5.935**	**5.876**	**5.897**	**0.102**
h	R	8.863	8.784	8.696	8.650	8.568	8.571	8.571	8.607	8.597	8.639	0.229
	NC		**8.821**	**8.794**	**8.718**	**8.702**	**8.664**	**8.641**	**8.682**	**8.589**	**8.626**	**0.162**
T	R	0.630	0.622	0.612	0.603	0.588	0.585	0.579	0.571	0.560	0.548	0.044
	NC		**0.625**	**0.617**	**0.611**	**0.588**	**0.595**	**0.589**	**0.589**	**0.578**	**0.584**	**0.033**
$SC(\times 10^{16})$	R	1.235	0.754	0.530	0.244	0.114	0.120	0.030	0.034	0.027	0.023	0.951
	NC		**0.971**	**0.882**	**0.538**	**0.283**	**0.402**	**0.093**	**0.187**	**0.158**	**0.096**	**0.776**

Table 8.1: Data utility and information loss results for UMGA-R (R) and UMGA-NC (NC) algorithms.

In addition, Q is not evaluated since this network does not have community labels. On GrQc the UMGA-NC achieves the best results on all metrics, reducing considerably the information loss and raising the data utility.



(a) Caida (b) Amazon (c) Yahoo!

Figure 8.4: Degree histogram for tested networks (grey horizontal lines indicate the k values used in our experiments).

8.3.3 Results for large networks

After information loss and data utility analysis, we want to test our algorithm with large networks. We have tested our algorithm with three real and large networks. All these networks are undirected and unlabelled. All tests are made on a 4 CPU Intel Xeon X3430 at 2.40GHz with 32GB RAM running Debian GNU/Linux.

We use the *vertex refinement queries* to model the knowledge of the adversary and to analyse the network in terms of k-anonymity (see Section 3.4.2 for more details). Table 8.2 shows the candidate set ($cand_{\mathcal{H}_1}$) for the original networks. It shows interesting information about how re-identification risk is distributed over all vertices of the graph. Caida network has 70 re-identificable vertices, i.e, vertices with a unique degree's value, and 217 vertices in high risk of re-identification, i.e, vertices with candidate set size between 2 and 10. Amazon has 90 re-identificable vertices and 535 vertices in high risk of re-identification, and Yahoo! has 73 re-identificable vertices and 331 vertices in high risk of re-identification. Degree anonymity data is strictly related to the degree histogram. Figure 8.4 shows the log-log scale degree histogram, where our test networks show a characteristic power-law distribution for real networks. Degree values (horizontal axis) with a frequency (vertical axis) less than 10 are the ones on high risk of re-identification. We can see that Caida network has a sparse vertex distribution in this area. So, it will be harder to anonymize than other networks. In this context, harder means that we must modify more edges (in percentage) than on other networks.

Table 8.3 shows the results of our experiments. We apply the UMGA algorithm, using exhaustive and greedy search methods with random edge selection (UMGA-E-R and UMGA-G-R) and neighbourhood centrality edge selection (UMGA-E-NC and UMGA-G-NC), to the three selected networks. We test our algorithm for values of

Network	[1,1]	[2,10]	[11,20]	[21,50]	[51,100]
Caida	70 (0.264%)	217 (0.819%)	119 (0.449%)	294 (1.110%)	295 (1.114%)
Amazon	90 (0.022%)	535 (0.132%)	435 (0.107%)	711 (0.176%)	1,165 (0.288%)
Yahoo!	73 (0.003%)	331 (0.018%)	196 (0.010%)	381 (0.020%)	705 (0.037%)

Table 8.2: Candidate set size of $\mathcal{H}_1$ ($cand_{\mathcal{H}_1}$) for original tested networks.

General			**Exhaustive Method**				**Greedy Method**			
Network	k	2^q	**ed**	**%mod**	**R time**	**NC time**	**ed**	**%mod**	**R time**	**NC time**
Caida	10	2^{24}	0	6.06%	0:00:06	0:00:36	0	6.06%	0:00:06	0:00:34
	20	2^{21}	0	11.65%	0:00:07	0:01:05	0	11.65%	0:00:07	0:01:05
	50	2^{13}	-9	18.43%	0:00:08	0:01:45	-9	18.43%	0:00:08	0:01:45
	100	2^{9}	-9	25.81%	0:00:10	0:01:48	41	25.85%	0:00:10	0:01:46
Amazon	10	2^{54}	0	0.16%	0:14:16	0:22:47	0	0.16%	0:11:54	0:19:47
	20	2^{50}	0	0.26%	2:57:37	3:07:58	0	0.26%	0:11:55	0:22:25
	50	2^{32}	0	0.39%	0:12:07	0:33:07	0	0.39%	0:12:07	0:31:17
	100	2^{30}	-1	0.53%	2:11:47	2:42:48	-1	0.53%	0:13:57	0:53:37
Yahoo!	10	2^{38}	0	0.05%	1:34:34	1:52:25	0	0.05%	1:31:40	1:50:24
	20	2^{28}	0	0.07%	2:19:21	2:46:47	0	0.07%	1:31:57	1:59:34
	50	2^{19}	0	0.13%	1:33:09	2:22:59	0	0.13%	1:32:12	2:20:54
	100	2^{16}	0	0.18%	1:34:27	2:46:33	0	0.18%	1:34:32	2:47:58

Table 8.3: UMGA's parameters and execution time results for large tested networks.

$k = \{10, 20, 50, 100\}$ on each network and computes the number of possible combinations (2^q) in order to provide an approximation of the complexity. For each method we show the difference between the original edge set and the anonymous one ($ed = |E| - |\widetilde{E}|$), the percentage of modified edges ($\%mod = 1 - \frac{|E \cap \widetilde{E}|}{|E \cup \widetilde{E}|}$), and the computation time for UMGA-R (*R time*) and UMGA-NC (*NC time*).

Caida is a quite sparse network. More than 49% of their vertices have a degree between 1 and 10, 21.71% between 11 and 100, 18.66% between 101 and 1,000 and 10.26% have a degree greater than 1,000. The maximum degree is 2,628. Because of this, it is necessary to modify more than 6% of the edges in order to get a $k = 10$. This percentage grows, so does the value of k. For a $k = 50$ and $k = 100$ the algorithm needs to modify the size of the edge set removing 9 edges and adding new 41 edges. It is about 0.018% of total edges, so we believe that the noise introduced in the graph is minimum, specially using UMGA-NC as we demonstrated on Section 8.3.2. We can see similar times and results for exhaustive and greedy methods, but UMGA-R consumes much less time than UMGA-NC. The number of possible combinations is small and the exhaustive method can deal with it.

Amazon is a network larger than Caida, so we can see greater differences between exhaustive and greedy methods. The complexity rises to 2^{54} when $k = 10$. Notice that smaller k-values imply bigger complexity since more group of vertices (g_j) are possible, and therefore, more possible combinations. When $k = 100$ there is no solution with $\sum_{j=1}^{p} m_j = 0$, so the exhaustive method explores all the possible combinations. In this case it is 2^{30} and it can be done with relative small amount of time. For other values of k, there is a solution equal to 0, so the exhaustive method does not explore all combinations. Indeed, it explores less than 0.1% in all cases. However, the greedy method finds the same results on all experiments and it spends much less time. The time used by UMGA-NC is only two or three times larger than the one used by UMGA-R, much less than the difference on Caida network.

Yahoo! network is the largest test network, but it is less sparse than others. 99.21% of the vertices have a degree between 1 and 100, 0.75% between 101 and 1,000, and only 0.03% have a degree greater than 1,000. Vertices with a degree value less than 100 are well protected and they are more than 99%. These characteristics imply that k-degree anonymous networks from $k = 10$ to $k = 100$ can be achieved with less than 0.20% of modifications on edge set. Hence, the utility of the anonymous graphs will be almost intact. Of course, UMGA-NC consumes more time than UMGA-R, but time is reduced to less than double.

Notice that UMGA-NC spends more time in all experiments, but the time does not grow exponentially with the network because of the heuristic approach applied. On the contrary, we can see that neighbourhood centrality edge selection can deal with large networks, improving the data utility with reasonable time consuming.

8.4 Comparing k-degree anonymous algorithms

In this section we will compare our algorithms to two other well-known k-degree anonymous algorithms. We have selected two relevant k-degree anonymous methods, and in subsequent sections, we will analyse these methods and compare the empirical results to ours on real networks. The selected algorithms are:

1. *A dynamic programming algorithm*: Liu and Terzi defined the concept of k-degree

anonymity and presented their method in [59].

2. *Vertex addition method*: Chester et al. propounded an algorithm based on vertices and edges addition in [21].

All methods achieve the same privacy level, since they presuppose the same adversary knowledge and apply the same concept to preserve the network's privacy. Therefore, the evaluation of the results allows us to compare anonymous data in terms of data utility and information loss.

8.4.1 A dynamic programming algorithm

Liu and Terzi [59] developed a method based on adding and removing edges from the original graph $G = (V, E)$ in order to construct a new graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, which fulfil k-degree anonymity model and the vertex set remains the same, i.e, $V = \widetilde{V}$. Their approach is two-step based: in the first step the degree anonymization problem is considered, and in the second step the graph construction problem is dealt.

Degree anonymization

Given the degree sequence d of the original input graph $G = (V, E)$, the algorithm outputs a k-anonymous degree sequence $(\widetilde{d})$ such that the degree-anonymization cost Δ computed by Equation 8.1 is minimized. The authors proposed three approximation techniques to solve the degree anonymization problem. They first gave a dynamic-programming algorithm (DP) that solves the degree anonymization problem optimally in time $\mathcal{O}(n^2)$. Then, they showed how to modify it to achieve linear-time complexity. Finally, they also gave a greedy algorithm that runs in time $\mathcal{O}(nk)$.

Graph construction

The authors presented two approaches to resolve the graph construction problem. The first approach considers the following problem: Given the original graph $G = (V, E)$ and the desired k-anonymous degree sequence $\widetilde{d}$ (output by the the previous step), they construct a k-degree anonymous graph $\widetilde{G} = (V, \widetilde{E})$ with $\widetilde{E} \cap E = E$ and degree sequence equal to $\widetilde{d}$. Notice that the problem definition implies that only edge addition operations are allowed. The algorithm for solving this problem was called *SuperGraph*. It takes as inputs the original graph G and the desired k-degree anonymous sequence $\widetilde{d}$, operates on

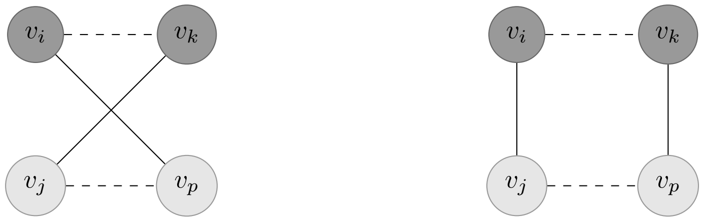


Figure 8.5: Valid switch operation among vertices v_i, v_j, v_k and v_p (dashed lines represent deleted edges while solid lines are the added ones).

the sequence of additional degrees $\widetilde{d} - d$ and outputs a super-graph of the original graph, if such graph exists.

The requirement that $\widetilde{E} \cap E = E$ may be too strict to satisfy is some cases. Thus, the second approach considers a relaxed requirement where $\widetilde{E} \cap E \approx E$, which means that most of the edges of the original graph appear in the degree-anonymous graph as well, but not necessarily all of them. The authors called this version of the problem the "Relaxed Graph Construction" problem. The *ConstructGraph* algorithm with input $\widetilde{d}$, outputs a simple graph $\widetilde{G}_0 = (V, \widetilde{E}_0)$ with degree sequence exactly $\widetilde{d}$, if such graph exists. Although $\widetilde{G}_0$ is k-degree anonymous, its structure may be quite different from the original graph $G = (V, E)$. The *GreedySwap* algorithm inputs $\widetilde{G}_0$ and G, and transforms $\widetilde{G}_0$ into $\widetilde{G} = (V, \widetilde{E})$ with degree sequence equal to $\widetilde{d}$ and $\widetilde{E} \cap E \approx E$ using greedy heuristic techniques. At every step i, the graph $\widetilde{G}_{i-1} = (V, \widetilde{E}_{i-1})$ is transformed into $\widetilde{G}_i = (V, \widetilde{E}_i)$ such that the degree sequences are equal and $|\widetilde{E}_i \cap E| > |\widetilde{E}_{i-1} \cap E|$. The transformation is made using *valid switch* operations, which are defined by four vertices v_i, v_j, v_k and v_p of $\widetilde{G}_i = (V, \widetilde{E}_i)$ such that $\{v_i, v_k\}$ and $\{v_j, v_p\} \in \widetilde{E}_i$, and $\{v_i, v_j\}$ and $\{v_k, v_p\} \notin \widetilde{E}_i$ or $\{v_i, v_p\}$ and $\{v_j, v_k\} \notin \widetilde{E}_i$. A valid switch operation transforms $\widetilde{G}_i$ to $\widetilde{G}_{i+1}$ by updating the edges $\widetilde{E}_{i+1} \leftarrow \widetilde{E}_i \setminus \{\{v_i, v_k\}, \{v_j, v_p\}\} \cup \{\{v_i, v_j\}, \{v_k, v_p\}\}$ or $\widetilde{E}_{i+1} \leftarrow \widetilde{E}_i \setminus \{\{v_i, v_k\}, \{v_j, v_p\}\} \cup \{\{v_i, v_p\}, \{v_j, v_k\}\}$, as we depicted in Figure 8.5.

8.4.2 Vertex addition method

Chester et al. [21, 20] focused on creating a k-degree anonymous graph $\widetilde{G} = (V \cup \widetilde{V}, E \cup \widetilde{E})$ from the original one $G = (V, E)$. In $\widetilde{G}$, the authors require that all the original vertices (V) are k-degree anonymous. They 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 G, i.e, $V \cup \widetilde{V}$ is k-degree-anonymous in $\widetilde{G}$. They seek to minimise $|\widetilde{V}|$, while maintaining the constraint that $E \subseteq \widetilde{V} \times (V \cup \widetilde{V})$.

Their method introduces fake vertices into the network and links them to each other and to real vertices in order to achieve the desired k-anonymity value. The authors introduced an $\mathcal{O}(kn)$ k-degree anonymization algorithm for unlabelled graphs based on dynamic programming and prove that, on any arbitrary graph, the minimisation of $|\widetilde{V}|$ is optimal within an additive factor of k. For a special class of graphs that is likely to include social networks, the algorithm is optimal within 1 for reasonable values of k.

At a high level, the algorithm proceeds in three stages. First, the authors designed a recursion to group the vertices of V by target degree (the degree they will have in $\widetilde{G}$). The recursion establishes a grouping such that the *max deficiency*, a parameter in determining with how many vertices V must be augmented, is minimised. A dynamic programming with cost $\mathcal{O}(nk)$ is used to evaluate the recursion. The second stage is to determine precisely how many vertices with which we wish to augment V in order to guarantee that they can k-anonymize all of $\widetilde{G}$. This number is a function of k and *max deficiency*. Finally, the algorithm introduces a particular means of adding new edges, each of which has at least one endpoint in $\widetilde{G}$, with the objective of satisfying all the target degrees established during the recursion stage and k-anonymizing the new vertices added during the second stage. A critical property of this approach is that the edges are added in such a manner as to guarantee tractability of the problem of k-anonymizing the new vertices, a problem which may be hard in the general case.

8.4.3 Experimental Results

Here we will compare the result of anonymizing processes using our k-degree anonymous algorithms and the two methods presented above. We apply all algorithms on the same data with the same parameters and compare the results in terms of information loss and data utility. We use several structural and spectral measures in order to quantify the information loss from distinct perspectives or network's characteristics. It is important to note that the privacy level is the same for all algorithms, as we compare results with the same k value. UMGA algorithm is applied using the neighbourhood centrality edge selection.

Results are displayed in Table 8.4. Each row indicates the scored value for the corresponding measure and algorithm, and each column corresponds to an experiment with a different k-anonymity value. For each dataset and algorithm, we vary k from 1 to 10 ($k = 1$ correspond to original dataset) and compare the results obtained on λ_1, μ_2, h, Q, T and SC. The last column correspond to the average error $\langle \mathcal{E} \rangle$. Each characteris-

Polbooks		k=1	2	3	4	5	6	7	8	9	10	$\langle\mathcal{E}\rangle$
λ_1	EAGA	11.93	12.04	12.01	12.04	11.95	12.05	12.01	11.72	10.84	11.45	0.230
	UMGA		**12.09**	**11.97**	**11.85**	**11.85**	**11.95**	**12.09**	**12.08**	**12.08**	**11.86**	**0.090**
	L&T		12.00	12.05	12.11	12.22	12.30	12.31	12.64	12.72	12.85	0.383
μ_2	EAGA	0.324	0.372	0.477	0.496	0.516	0.515	0.600	0.595	0.578	0.321	0.156
	UMGA		**0.360**	**0.451**	**0.453**	**0.453**	**0.383**	**0.599**	**0.524**	**0.524**	**0.640**	**0.147**
	L&T		0.430	0.450	0.600	0.600	0.790	0.630	0.650	0.970	0.880	0.312
h	EAGA	2.450	2.378	2.324	2.346	2.297	2.314	2.294	2.282	2.308	2.421	0.109
	UMGA		**2.416**	**2.371**	**2.379**	**2.379**	**2.418**	**2.312**	**2.350**	**2.350**	**2.312**	**0.077**
	L&T		2.350	2.320	2.280	2.280	2.230	2.270	2.260	2.200	2.190	0.167
Q	EAGA	0.402	0.399	0.387	0.387	0.383	0.387	0.379	0.379	0.387	0.389	0.014
	UMGA		**0.400**	**0.393**	**0.396**	**0.396**	**0.401**	**0.386**	**0.386**	**0.386**	**0.385**	**0.009**
	L&T		0.390	0.390	0.380	0.380	0.360	0.370	0.370	0.340	0.350	0.027
T	EAGA	0.348	0.343	0.330	0.324	0.281	0.300	0.288	0.283	0.245	0.299	0.044
	UMGA		**0.350**	**0.342**	**0.339**	**0.339**	**0.347**	**0.326**	**0.322**	**0.322**	**0.324**	**0.013**
	L&T		0.330	0.330	0.320	0.330	0.300	0.310	0.320	0.290	0.300	0.023
SC	EAGA	2.524	2.624	2.333	2.293	1.751	2.001	1.967	1.415	0.653	1.534	0.634
	UMGA		**2.774**	**2.358**	**2.224**	**2.224**	**2.338**	**2.363**	**2.389**	**2.389**	**2.110**	**0.204**
$(\times 10^3)$	L&T		2.480	2.560	2.530	2.760	2.440	2.680	3.600	3.580	4.120	0.431
Polblogs		k=1	2	3	4	5	6	7	8	9	10	$\langle\mathcal{E}\rangle$
λ_1	EAGA	74.08	73.13	70.26	55.61	53.09	49.33	46.89	44.44	42.88	44.08	18.703
	UMGA		**73.93**	**73.81**	**73.92**	**73.95**	**73.74**	**73.80**	**73.75**	**73.63**	**73.61**	**0.256**
	L&T		74.89	74.50	75.16	75.10	76.32	75.82	76.67	77.42	78.42	1.758
μ_2	EAGA	0.168	0.168	0.168	0.692	0.674	0.748	0.754	0.690	0.858	0.757	0.517
	UMGA		0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.000
	L&T		0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.000
h	EAGA	2.506	2.677	2.623	2.596	2.592	2.588	2.595	2.565	2.572	2.575	0.071
	UMGA		**2.501**	**2.499**	**2.496**	**2.496**	**2.496**	**2.502**	**2.498**	**2.502**	**2.499**	**0.006**
	L&T		2.500	2.484	2.494	2.475	2.469	2.461	2.462	2.486	2.458	0.026
	Chester		2.506	2.486	2.476	2.476	2.456	2.456	2.446	2.436	2.426	0.039
Q	EAGA	0.405	0.404	0.404	0.402	0.395	0.399	0.401	0.392	0.400	0.397	0.005
	UMGA		**0.404**	**0.403**	**0.403**	**0.403**	**0.403**	**0.403**	**0.402**	**0.403**	**0.402**	**0.002**
	L&T		0.402	0.401	0.401	0.396	0.394	0.395	0.389	0.387	0.385	0.010
T	EAGA	0.226	0.224	0.219	0.148	0.130	0.110	0.104	0.086	0.078	0.082	0.085
	UMGA		**0.224**	**0.224**	**0.224**	**0.224**	**0.223**	**0.225**	**0.224**	**0.223**	**0.224**	**0.001**
	L&T		0.225	0.223	0.224	0.221	0.222	0.220	0.219	0.221	0.221	0.004
	Chester		0.219	0.215	0.207	0.205	0.200	0.226	0.190	0.185	0.183	0.020
SC	EAGA	1.218	0.472	0.027	0.011	0.003	0.001	0.001	0.009	0.001	0.001	1.044
	UMGA		**1.052**	**0.932**	**1.040**	**1.068**	**0.871**	**0.921**	**0.875**	**0.776**	**0.765**	**0.266**
$(\times 10^{29})$	L&T		2.730	1.870	3.610	3.400	1.450	6.940	6.250	4.460	4.040	2.386
	Chester		1.300	1.410	2.160	2.880	2.660	5.550	5.370	11.000	8.250	2.969

Table 8.4: Results for EAGA, UMGA, Liu and Terzi (L&T) and Chester et al. (Chester) algorithms.

tic is reported from two to four times, corresponding to EAGA, UMGA, Liu and Terzi (indicated by L&T) and Chester et al. (indicated by Chester) algorithms. Bold rows indicate the algorithm that achieves the best results (i.e, lowest information loss) for each measure. Values of Liu and Terzi algorithm are taken from Ying et al. [87] and values of Chester et al. algorithm are taken from [21]. Unfortunately, values for all measures and algorithms are not available. Perfect performance in a row would be indicated by achieving exactly the same score as in the original network (the $k = 1$ column). Although deviation is undesirable, it is inevitable due to the edge or vertex modification process.

The first tested network, Polbooks, is a small collaboration network. We present values for EAGA, UMGA and Liu and Terzi algorithms. As shown in Table 8.4, UMGA algorithm introduces less noise on all measures. It outperforms on all measures, producing half of the average error in some measures, for instance, λ_1 or SC. EAGA algorithm achieves the second best results on λ_1, μ_2, h and Q, while Liu and Terzi algorithm carry out on T and SC.

Polblog is the second tested network, which is considerably larger than the first one. Values for Chester et al. algorithm are presented for h, T and SC (other values are not available from Chester et al. [21]). Like in the previous test, UMGA algorithm gets the best values on all measures, except on μ_2 where Liu and Terzi algorithm achieves the same value. For instance, the average error is 0.006 for UMGA on h, while it rises to 0.026 for Liu and Terzi algorithm, 0.039 for Chester et al. approach, and 0.071 for EAGA. Similar results appear on λ_1, T and SC. Liu and Terzi algorithm obtains the second best results on λ_1, h and T, while EAGA does on Q and SC. Chester et al. approach by vertex addition gets values close to others algorithms, though the predictable level of information loss is slightly larger than the ones obtained by UMGA and Liu and Terzi algorithms. Despite the fact that EAGA gets good results on some metrics, the average error outbursts in many others. For example, results on λ_1 and μ_2 are larger than others, pointing out a considerable spectral noise introduced by the anonymization process.

8.5 Conclusion

In this chapter we have proposed two algorithms for graph anonymization, both based on edge modification approach in order to fulfil the k-degree anonymity model.

The first algorithm, called Evolutionary Algorithm for Graph Anonymization (EAGA), is based on evolutionary algorithms to anonymize the degree sequence. The target of the evolutionary algorithm is to achieve a k-degree anonymous sequence and minimize the distance between the anonymous degree sequence and the original one, in order to preserve the data utility. The second step applies iterative Edge rotation to the original graph until the degree sequence is equal to the k-degree anonymous one. The results are favourable and indicate that the algorithm is able to provide anonymous graphs with a value of k-anonymity greater than the original one, and also keep a low perturbation level on anonymous data.

The second algorithm, called Univariate Micro-aggregation for Graph Anonymization (UMGA), is based on the modification of the degree sequence using univariate micro-aggregation technique. This process obtains an anonymous degree sequence which is k-degree anonymous and minimizes the distance from the original one. Then we use the three basic operations to translate the modifications made on anonymous degree sequence to graph's edge set.

In addition, we have proposed a method to preserve the most important edges on the network. Instead of modifying one of the possible edges at random, this method considers the edge's relevance and preserves the most important edges in the network. We have demonstrated that using this method we can clearly improve the data utility, reducing the information loss on anonymized data.

Furthermore, we have shown that UMGA algorithm is able to anonymize large networks. We have used three different real networks to test our algorithm based on the exhaustive and greedy methods. Both methods show good results on all networks, but the greedy method spends less time to get similar (in much cases, the same) results. In addition, the greedy method remains much more stable over time than the exhaustive one. The tests proved that our algorithm can anonymize large real networks based on k-degree anonymity concept.

Finally, four relevant methods of k-degree anonymity have been surveyed and compared. They are the algorithm by Liu and Terzi in [59], the method based on vertex addition instead of only changing the edge set by Chester et al. in [21], and our approaches using evolutionary algorithms (EAGA) and univariate micro-aggregation (UMGA). We have also probed that our algorithm based on univariate micro-aggregation is better, in terms of information loss and data utility, than the two other well-known k-degree anonymous algorithms.

Part IV

Conclusions

Chapter 9

Conclusions and further research

Several approaches to deal with data privacy in graph-formatted data have been proposed in the preceding chapters. In this last chapter we start by reviewing some conclusions to wrap up all the results obtained in Section 9.1, and finally we introduce some future research lines in Section 9.2.

9.1 Conclusions

In this thesis we have covered the fields of user's privacy-preserving in social networks and the utility and quality of the released data. As we have stated before, a trade-off between both fields is a critical point to achieve good anonymization methods for the subsequent graph mining processes.

Part of this thesis has focused on data utility and information loss. Firstly, in Chapter 5, we have studied the relation between the generic information loss measures and the clustering-specific ones, in order to evaluate whether the generic information loss measures are indicative of the usefulness of the data for subsequent data mining processes. We have found strong correlation between some generic information loss measures (average distance, betweenness centrality, closeness centrality, edge intersection, clustering coefficient and transitivity) and the precision index over the results of several clustering algorithms, demonstrating that these measures are able to predict the perturbation introduced in anonymous data. Furthermore, we have exposed that the perturbation method affects the correlation between generic and cluster-specific information loss measures. In addition, we have proved that the peculiarities of the graph structure and the perturbation method affect the quality of the released data and the relation between the generic and the specific information loss measures.

Secondly, two measures to reduce the information loss on graph modification processes have been presented in Chapter 6. The first one, *Edge neighbourhood centrality*, is based on information flow throw 1-neighbourhood of a specific edge in the graph. The second one is based on the *core number sequence* (also called *coreness*) and it preserves better the underlying graph structure, retaining more data utility. By an extensive experimental set up, we have demonstrated that both methods are able to preserve the most important edges in the network, keeping the basic structural and spectral properties close to the original ones. Both methods can be applied to any graph modification method, leading the process to a better data utility.

The other important topic of this thesis has been privacy-preserving methods. In Chapter 7 we have presented our random-based algorithm, which utilizes the concept of *Edge neighbourhood centrality* to drive the edge modification process to better preserve the most important edges in the graph, achieving lower information loss and higher data utility on the released data. Our method obtains a better trade-off between data utility and data privacy than other methods. In particular, it outperforms Random Perturbation in terms of data utility and Random Switch in terms of data privacy as empirical results showed.

Finally, two different approaches for k-degree anonymity on graphs have been developed in Chapter 8. First, an algorithm based on evolutionary computing has been presented and tested on different small and medium real networks. Although this method allow us to fulfil the desired privacy level, it presents two main drawbacks: the information loss is quite large in some graph structural properties and it is not fast enough to work with large networks. Therefore, a second algorithm has been presented, which uses the univariate micro-aggregation to anonymize the degree sequence and reduce the distance from the original one. This method is quasi-optimal and it results in lower information loss and better data utility. Additionally, it is much more faster and efficient, being able to anonymize large networks of millions of nodes and edges, as we have demonstrated in our experiments. Furthermore, we have also applied the concept of *Edge neighbourhood centrality* to this algorithm, producing an improvement in the quality and utility of the anonymous data, though it increases a little the time complexity of the algorithm. We have finalized this chapter with a comparison among our algorithms and two other well-know state of the art approaches, proving that our algorithm improves the others in terms of data utility.

9.2 Further research

In all topics described in this dissertation there are some open research lines for future work. In this section we briefly discuss some of them.

Referring to data utility, our analysis for correlation between measures and metrics can be extended to other graph's types, including for instance, directed networks. In many cases, authors deal with directed networks by simply removing their arcs directions and converting them to undirected networks. Although this approximation is simple and effective, it clearly destroys critical information inside the network. Thus, extending our experimental framework to work with directed networks can help us to better understand the behaviour of non-symmetrical relationships among users in a social network, like for instance, directed relations on Twitter.

In this text we have claimed that preserving the *coreness* of the network leads the edge modification process to a lower information loss. However, it is not clear the effects of preserving the coreness in terms of re-identification. A rigorous risk assessment analysis has to be performed to evaluate and quantify the knowledge an adversary can extract from the released network.

Lastly, regarding to privacy-preserving methods, it is obvious that our methods can be extended to work with richer graphs. As we have stated, undirected and unlabelled graphs are the simplest ones, but we need powerful methods to deal with different kinds of networks to avoid removing interesting information, i.e, the direction of the arcs, vertices' or edges' labels, etc. However, it is not trivial to modify our methods to work with richer graphs. As we have seen reviewing the state of the art, several problems with rich graphs are NP-hard, thus developing efficient heuristic methods is still an open but also hard problem.

Our contributions

[1a] J. Casas-Roma, J. Herrera-Joancomartí and V. Torra. Algoritmos genéticos para la anonimización de grafos. In *Spanish Meeting on Cryptology and Information Security (RECSI)*, pp. 243-248. Donostia-San Sebastián, Spain, 2012. Mondragon Unibertsitatea.

[2a] J. Casas-Roma, J. Herrera-Joancomartí and V. Torra. Comparing Random-based and k-Anonymity-based Algorithms for Graph Anonymization. In V. Torra, Y. Narukawa, B. López and M. Villaret, editors, *Proceedings of the 9th International Conference on Modeling Decisions for Artificial Intelligence*, volume 7647 of Lecture Notes in Computer Science, pp. 197-209. Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-34620-0_19

[3a] J. Casas-Roma, J. Herrera-Joancomartí and V. Torra. Evolutionary Algorithm for Graph Anonymization, 6. Databases. http://arxiv.org/abs/1310.0229, 2013.

[4a] J. Casas-Roma, J. Herrera-Joancomartí and V. Torra. An Algorithm For k-Degree Anonymity On Large Networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pp. 671-675. Niagara Falls, ON, Canada, 2013. IEEE Computer Society, doi:10.1145/2492517.2492643

[5a] J. Casas-Roma, J. Herrera-joancomartí and V. Torra. Analyzing the Impact of Edge Modifications on Networks. In V. Torra, Y. Narukawa, G. Navarro-Arribas and D. Megías, editors, *Proceedings of the 10th International Conference on Modeling Decisions for Artificial Intelligence*, volume 8234 of Lecture Notes in Computer Science, pp. 296-307. Berlin, Heidelberg, 2013. Springer Berlin Heidelberg, doi:10.1007/978-3-642-41550-0_26

[6a] J. Casas-Roma. Privacy-Preserving on Networks using Randomization and Edge-

Relevance. In V. Torra, Y. Narukawa and Y. Endo, editors, *Proceedings of the 11th International Conference on Modeling Decisions for Artificial Intelligence*, volume 8825 of Lecture Notes in Artificial Intelligence, pp. 204-216. Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[7a] J. Casas-Roma, J. Herrera-joancomartí and V. Torra. A Summary of k-Degree Anonymous Methods for Privacy-Preserving on Networks. In G. Navarro-Arribas and V. Torra, editors, *Advanced Research in Data Privacy*, volume 567 of Studies in Computational Intelligence, pp. 231-250. Berlin, Heidelberg, 2015. Springer International Publishing, doi:10.1007/978-3-319-09885-2_13

[8a] J. Casas-Roma, J. Herrera-joancomartí and V. Torra. Anonymizing Graphs: Measuring Quality for Clustering. *Knowledge and Information Systems* (KAIS), 2014, doi:10.1007/s10115-014-0774-7. (**SCI index 2013: 2.639, Q1**)

[9a] J. Casas-Roma, J. Herrera-joancomartí and V. Torra. k-Degree Anonymity And Edge Selection: Improving Data Utility On Large Networks. *Knowledge and Information Systems* (KAIS), 2014. (*Submitted*).

[10a] F. Rousseau, J. Casas-Roma and M. Vazirgiannis. Coreness-Preserving Edge Modification for Graph Anonymization. In *the 23rd ACM International Conference on Information and Knowledge Management* (CIKM), 2014. (*Submitted*).

Bibliography

[1] L. A. Adamic and N. Glance. The political blogosphere and the 2004 U.S. election. In *International Workshop on Link Discovery (LinkKDD)*, pages 36–43, New York, NY, USA, 2005. ACM.

[2] Y. Alufaisan and A. Campan. Preservation of Centrality Measures in Anonymized Social Networks. In *Proceedings of the 2013 International Conference on Social Computing (SOCIALCOM)*, pages 486–493, Washington, DC, USA, 2013. IEEE.

[3] J. I. Alvarez-Hamelin, L. D. Asta, A. Barrat, and A. Vespignani. K-core decomposition of Internet graphs: hierarchies, self-similarity and measurement biases. *Networks and Heterogeneous Media*, 3(2):371–393, 2008.

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *International Conference on World Wide Web (WWW)*, pages 181–190, New York, NY, USA, 2007. ACM.

[5] A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[6] V. Batagelj and M. Zaveršnik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5(2):129–145, 2011.

[7] M. Baur, M. Gaertler, G. Robert, and M. Krug. Generating Graphs with Predefined k-Core. In *Proceedings of the European Conference on Complex Systems (ECS)*, pages 1–15, 2007.

[8] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Class-based graph anonymization for social network data. *Proceedings of the VLDB Endowment*, 2(1):766–777, 2009.

[9] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment (JSTAT)*, 2008(10):P10008, 2008.

[10] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting Uncertainty in Graphs for Identity Obfuscation. *Proceedings of the VLDB Endowment*, 5(11):1376–1387, 2012.

[11] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In *2011 IEEE 27th International Conference on Data Engineering (ICDE)*, pages 924–935, Washington, DC, USA, 2011. IEEE.

[12] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. *Information Sciences*, 275:232–256, 2014.

[13] A. Budi, D. Lo, L. Jiang, and Lucia. *kb*-anonymity: a model for anonymized behaviour-preserving test and debugging data. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 447–457, New York, NY, USA, 2011. ACM.

[14] B.-J. Cai, H.-Y. Wang, H.-R. Zheng, and H. Wang. Evaluation repeated random walks in community detection of social networks. In *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1849–1854, Qingdao, China, 2010. IEEE.

[15] A. Campan and T. M. Truta. A Clustering Approach for Data and Structural Anonymity in Social Networks. In *ACM SIGKDD International Workshop on Privacy, Security, and Trust (PinKDD)*, pages 1–10, Las Vegas, Nevada, USA, 2008. ACM.

[16] A. Campan and T. M. Truta. Data and Structural k-Anonymity in Social Networks. In *Privacy, Security, and Trust in KDD (PinKDD)*, pages 33–54. Springer-Verlag, 2009.

[17] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir. A model of Internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences of the United States of America*, 104(27):11150–11154, 2007.

[18] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra. An Algorithm For k-Degree Anonymity On Large Networks. In *IEEE International Conference on Advances on Social Networks Analysis and Mining (ASONAM)*, pages 671–675, Niagara Falls, CA, 2013. IEEE.

[19] S. Chester, J. Gaertner, U. Stege, and S. Venkatesh. Anonymizing Subsets of Social Networks with Degree Constrained Subgraphs. In *IEEE International Conference on Advances on Social Networks Analysis and Mining (ASONAM)*, pages 418–422, Washington, DC, USA, 2012. IEEE.

[20] S. Chester, B. M. Kapron, G. Ramesh, G. Srivastava, A. Thomo, and S. Venkatesh. k-Anonymization of Social Networks By Vertex Addition. In *ADBIS 2011 Research Communications*, pages 107–116, Vienna, Austria, 2011. CEUR-WS.org.

[21] S. Chester, B. M. Kapron, G. Ramesh, G. Srivastava, A. Thomo, and S. Venkatesh. Why Waldo befriended the dummy? k-Anonymization of social networks with pseudo-nodes. *Social Network Analysis and Mining*, 3(3):381–399, 2013.

[22] S. Chester, B. M. Kapron, G. Srivastava, and S. Venkatesh. Complexity of social network anonymization. *Social Network Analysis and Mining*, 3(2):151–166, 2013.

[23] S. Chester and G. Srivastava. Social Network Privacy for Attribute Disclosure Attacks. In *2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 445–449, Kaohsiung, 2011. IEEE.

[24] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 70(6):1–6, 2004.

[25] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB Endowment*, 19(1):115–139, 2010.

[26] S. Das, O. Egecioglu, and A. E. Abbadi. Anonymizing weighted social network graphs. In *IEEE International Conference on Data Engineering (ICDE)*, pages 904–907, Long Beach, CA, USA, 2010. IEEE.

[27] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Data Privacy: Definitions and Techniques. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 20(6):793–818, 2012.

[28] C. Dwork. Differential Privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *International Conference on Automata, Languages and Programming (ICALP)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[29] C. Dwork. An Ad Omnia Approach to Defining and Achieving Private Data Analysis. In *Proceedings of the 1st ACM SIGKDD international conference on Privacy, security, and trust in KDD (PinKDD)*, pages 1–13, San Jose, CA, USA, 2008. Springer-Verlag.

[30] C. Dwork. Differential Privacy: A Survey of Results. In *International Conference on Theory and Applications of Models of Computation (TAMC)*, volume 4978, pages 1–19. Springer-Verlag, 2008.

[31] C. Dwork. The differential privacy frontier. In *Proceedings of the sixth theory of cryptography conference (TCC)*, pages 496–502. Springer-Verlag, 2009.

[32] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[33] T. Feder, S. U. Nabar, and E. Terzi. Anonymizing Graphs. *CoRR*, abs/0810.5:1–15, 2008.

[34] F. Ferri, P. Grifoni, and T. Guzzo. New forms of social and professional digital relationships: the case of Facebook. *Social Network Analysis and Mining (SNAM)*, 2(2):121–137, 2011.

[35] R. Ford, T. M. Truta, and A. Campan. P-Sensitive K-Anonymity for Social Networks. In *Proceedings of the 2009 International Conference on Data Mining (DMIN)*, pages 403–409, Las Vegas, USA, 2009. CSREA Press.

[36] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating Cooperation in Communities with the k-Core Structure. In *IEEE International Conference on Advances on Social Networks Analysis and Mining (ASONAM)*, pages 87–93, Kaohsiung, 2011. IEEE.

[37] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 99(12):7821–7826, 2002.

[38] P. M. Gleiser and L. Danon. Community Structure in Jazz. *Advances in Complex Systems*, 6(04):565–573, 2003.

[39] A. V. Goltsev, S. N. Dorogovtsev, and J. F. F. Mendes. k-core (bootstrap) percolation on complex networks: Critical phenomena and nonlocal effects. *Physical Review E*, 73(056101):1–11, 2006.

[40] D. Grady, C. Thiemann, and D. Brockmann. Robust classification of salient links in complex networks. *Nature communications*, 3(May):864:1–864:10, 2012.

[41] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 68(6):1–4, 2003.

[42] S. Hanhijärvi, G. C. Garriga, and K. Puolamäki. Randomization techniques for graphs. In *SIAM Conference on Data Mining (SDM)*, pages 780–791, Sparks, Nevada, USA, 2009. SIAM.

[43] S. L. Hansen and S. Mukherjee. A Polynomial Algorithm for Optimal Univariate Microaggregation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(4):1043–1044, 2003.

[44] M. Hay. *Enabling Accurate Analysis of Private Network Data*. PhD thesis, University of Massachusetts - Amherst, 2010.

[45] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate Estimation of the Degree Distribution of Private Networks. In *IEEE International Conference on Data Mining (ICDM)*, pages 169–178, Miami, FL, USA, 2009. IEEE.

[46] M. Hay, K. Liu, G. Miklau, J. Pei, and E. Terzi. Privacy-aware data management in information networks. In *International Conference on Management of Data (SIGMOD)*, pages 1201–1204, New York, NY, USA, 2011. ACM.

[47] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, 2008.

[48] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing Social Networks. Technical Report No. 07-19, Computer Science Department, University of Massachusetts Amherst, UMass Amherst, 2007.

[49] X. He, J. Vaidya, B. Shafiq, N. Adam, and V. Atluri. Preserving Privacy in Social Networks: A Structure-Aware Approach. In *International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 647–654, Milan, Italy, 2009. IEEE.

[50] J. Herrera-Joancomartí and C. Pérez-Solà. Online Social Honeynets: Trapping Web Crawlers in OSN. In V. Torra, Y. Narakawa, J. Yin, and J. Long, editors, *International Conference on Modeling Decisions for Artificial Intelligence (MDAI)*, volume 6820 of *Lecture Notes in Computer Science*, pages 115–131. Springer Berlin Heidelberg, 2011.

[51] B. M. Kapron, G. Srivastava, and S. Venkatesh. Social Network Anonymization via Edge Addition. In *IEEE International Conference on Advances on Social Networks Analysis and Mining (ASONAM)*, pages 155–162, Kaohsiung, 2011. IEEE.

[52] V. Krebs. http://www.orgnet.com, 2006.

[53] L. Lan, S. Ju, and H. Jin. Anonymizing Social Network Using Bipartite Graph. In *International Conference on Computational and Information Sciences (ICCIS)*, pages 993–996, Chengdu, China, 2010. IEEE.

[54] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[55] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5:1–5:39, 2007.

[56] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, pages 177–187, New York, NY, USA, 2005. ACM.

[57] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2:1–2:40, 2007.

[58] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *IEEE International Conference on Data Engineering (ICDE)*, pages 106–115, Istanbul, Turkey, 2007. IEEE.

[59] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 93–106, Vancouver, Canada, 2008. ACM.

[60] X. Lu, Y. Song, and S. Bressan. Fast Identity Anonymization on Graphs. In *23rd International Conference on Database and Expert Systems Applications (DEXA)*, pages 281–295, Vienna, Austria, 2012. Springer Berlin Heidelberg.

[61] Lucia, D. Lo, L. Jiang, and A. Budi. kb^e-Anonymity: Test Data Anonymization for Evolving Programs. In *International Conference on Automated Software Engineering (ASE)*, pages 262–265, New York, NY, USA, 2012. ACM.

[62] 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):3:1–3:12, 2007.

[63] F. D. Malliaros and M. Vazirgiannis. To Stay or Not to Stay : Modeling Engagement Dynamics in Social Graphs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (CIKM)*, pages 469–478, New York, NY, USA, 2013. ACM.

[64] F. McSherry and I. Mironov. Differentially private recommender systems. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 627–636, New York, NY, USA, 2009. ACM.

[65] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational Differential Privacy. In *Proceedings of the 29th international cryptology conference (CRYPTO)*, pages 126–142. Springer-Verlag, 2009.

[66] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 29–42, New York, NY, USA, 2007. ACM.

[67] F. Nagle. Privacy Breach Analysis in Social Networks. In T. Özyer, Z. Erdem, J. Rokne, and S. Khoury, editors, *Mining Social Networks and Security Informatics*, Lecture Notes in Social Networks, pages 63–77. Springer Netherlands, Dordrecht, 2013.

[68] F. Nagle, L. Singh, and A. Gkoulalas-divanis. EWNI : Efficient Anonymization of Vulnerable. In *Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 359–370, Kuala Lumpur, Malaysia, 2012. Springer-Verlag Berlin.

[69] A. Narayanan and V. Shmatikov. De-anonymizing Social Networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 173–187, Washington, DC, USA, 2009. IEEE.

[70] M. E. Nergiz and C. Clifton. Thoughts on k-anonymization. *Data & Knowledge Engineering (DKE)*, 63(3):622–645, 2007.

[71] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 69(2):1–16, 2003.

[72] P. Pons and M. Latapy. Computing communities in large networks using random walks. In p. Yolum, T. Güngör, F. Gürgen, and C. Özturan, editors, *Computer and Information Sciences - ISCIS 2005*, volume 3733 of *Lecture Notes in Computer Science*, pages 284–293. Springer Berlin Heidelberg, 2005.

[73] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(4):1118–1123, 2008.

[74] P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(6):1010–1027, 2001.

[75] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

[76] V. K. Sihag. A clustering approach for structural k-anonymity in social networks using genetic algorithm. In *CUBE International Information Technology Conference*, pages 701–706, Pune, India, 2012. ACM.

[77] L. Singh and C. Schramm. Identifying Similar Neighborhood Structures in Private Social Networks. In *International Conference on Data Mining Workshops (ICDMW)*, pages 507–516, Sydney, NSW, 2010. IEEE.

[78] L. Singh and J. Zhan. Measuring Topological Anonymity in Social Networks. In *2007 IEEE International Conference on Granular Computing (GRC)*, pages 770–774, Fremont, CA, 2007. IEEE.

[79] K. Stokes and V. Torra. On some clustering approaches for graphs. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 409–415, Taipei, Taiwan, 2011. IEEE.

[80] K. Stokes and V. Torra. Reidentification and k-anonymity: a model for disclosure risk in graphs. *Soft Computing*, 16(10):1657–1670, 2012.

[81] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 10(5):557–570, 2002.

[82] B. Tripathy and G. Panda. A New Approach to Manage Security against Neighborhood Attacks in Social Networks. In *IEEE International Conference on Advances on Social Networks Analysis and Mining (ASONAM)*, pages 264–269, Odense, Denmark, 2010. IEEE.

[83] T. M. Truta, A. Campan, A. Gasmi, N. Cooper, and A. Elstun. Centrality Preservation in Anonymized Social Networks. In *International Conference on Data Mining (DMIN)*, pages 1–7, Las Vegas, Nevada, 2011.

[84] S. M. van Dongen. *Graph clustering by flow simulation.* PhD thesis, University of Utrecht, 2000.

[85] H. Wu, J. Zhang, J. Yang, B. Wang, and S. Li. A Clustering Bipartite Graph Anonymous Method for Social Networks. *Journal of Information and Computational Science (JOICS)*, 10(18):6031–6040, 2013.

[86] Yahoo! Webscope. Yahoo! Instant Messenger friends connectivity graph, version 1.0, 2003.

[87] X. Ying, K. Pan, X. Wu, and L. Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *Workshop on Social Network Mining and Analysis (SNA-KDD)*, pages 10:1–10:10, New York, NY, USA, 2009. ACM.

[88] X. Ying and X. Wu. Randomizing Social Networks: a Spectrum Preserving Approach. In *SIAM Conference on Data Mining (SDM)*, pages 739–750, Atlanta, Georgia, USA, 2008. SIAM.

[89] X. Ying and X. Wu. Graph Generation with Prescribed Feature Constraints. In *SIAM Conference on Data Mining (SDM)*, pages 966–977, Sparks, Nevada, USA, 2009. SIAM.

[90] X. Ying and X. Wu. On Link Privacy in Randomizing Social Networks. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 28–39, Bangkok, Thailand, 2009. Springer-Verlag.

[91] W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[92] K. Zhang, D. Lo, E.-P. Lim, and P. K. Prasetyo. Mining indirect antagonistic communities from social interactions. *Knowledge and Information Systems (KAIS)*, 35(3):553–583, 2013.

[93] E. Zheleva and L. Getoor. Preserving the Privacy of Sensitive Relationships in Graph Data. In *ACM SIGKDD International Conference on Privacy, Security, and Trust (PinKDD)*, volume 4890 of *Lecture Notes in Computer Science*, pages 153–171. Springer-Verlag, 2007.

[94] B. Zhou and J. Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *IEEE International Conference on Data Engineering (ICDE)*, pages 506–515, Washington, DC, USA, 2008. IEEE.

[95] B. Zhou and J. Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems (KAIS)*, 28(1):47–77, 2011.

[96] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

[97] L. Zou, L. Chen, and M. T. Özsu. k-Automorphism: A General Framework For Privacy Preserving Network Publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.