University of Hradec Králové

Faculty of Informatics and Management

Department of Informatics and Quantitative Methods

Anonymization of social network datasets

Doctoral thesis

Author: Mgr. Jana Medková

Study programme: Applied Informatics

Study branch: 1802V001 Applied Informatics

Supervisor: prof. RNDr. Josef Hynek, MBA, Ph.D.

Supervisor's department: Department of informatics and quantitative methods

Hradec Králové April 2023

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

In ..... date ....

Author's signature

I am very grateful to my advisor, prof. RNDr. Josef Hynek, MBA, PhD, for much helpful advice regarding the research, writing papers and many other areas. I want to thank all members of my family, especially my husband and my kids, for all the support of various kinds, which I could hardly express in words, and for being patient with me.

Abstract: The thesis addresses the identity disclosure privacy problem in publishing social network datasets. It introduces the topic of anonymization, focusing on its application on social network datasets. The author reviews the state-of-the-art anonymization and deanonymization methods and detects three open problems in this field. This thesis aims to solve the detected problems and answer the formulated research questions by proposing new procedures and algorithms, their implementation in the MATLAB programming platform and testing them on sets of synthetic and real-world social network datasets. The author presents the composition attack as a novel threat to social network datasets. Moreover, the well-known k-degree anonymization algorithm is improved with the novel heuristic high-degree noise addition procedure. Finally, the author employs the genetic algorithm principles in the k-automorphism anonymization approach to develop a hybrid algorithm for k-automorphism anonymization. A significant aspect of the evaluation process is to verify the results on relevant data and to provide outputs that are easily comparable with other researchers. For this reason, the evaluation tool SecGraph is used to measure data utility and the vulnerability of anonymization methods to deanonymization attacks.

Keywords: privacy, anonymization, social networks, genetic algorithm, graph theory

Abstrakt: Disertační práce se zabývá problémem reidentifikace uživatelů při zveřejňování databází sociálních sítí. Představuje anonymizaci a její aplikaci na databáze sociálních sítí. Autorka prezentuje dosud publikované anonymizační a deanonymizační metody a stanovuje tři otevřené problémy v této oblasti. Cílem práce je vyřešit dané problémy a odpovědět na formulované výzkumné otázky navržením nových metod a algoritmů, jejich implementací v programovací platformě MATLAB a jejich testováním na syntetických a reálných datech. Autorka ukazuje, že kompoziční útok, který byl dříve představen jako informační hrozba pro relační databáze, může být aplikován i na databáze sociálních sítí. Dále autorka navrhuje novou heuristickou metodu přidávání šumu do dobře známého k-stupňového anonymizačního algoritmu. Nakonec prezentuje využití principů genetických algoritmů v k-automorfní anonymizační metodě a navrhuje hybridní k-automorfní anonymizační algoritmus. Důraz je kladen na testování algoritmů na relevantních datech a produkci výstupů, které jsou snadno porovnatelné s výsledky jiných výzkumů. Z tohoto důvodu je použit ohodnocovací nástroj SecGraph k měření užitečnosti dat či zranitelnosti anonymizačních metod vůči deanonymizačním útokům.

Klíčová slova: informační bezpečnost, anonymizace, sociální sítě, genetický algoritmus, teorie grafů

List of notation

G_A	social network graph with users' attributes
G	social network graph without users' attributes
G^*	anonymized social network graph
$\tilde{G}$	a released social network graph
$\mathcal{Q}$	a structural query
$E(G)$	the edge set of G
$V(G)$	the vertex set of G
$U(G_A)$	the set of attributes of G_A
$v \in V(G)$	a node of G
I	an user, an individual, an individual in genetic algorithm
$v(I) \in V(G)$	the node representing I in G
$(v_i, v_j) \in E(G)$	the edge between nodes v_i and v_j in G
u_i, u_{v_i}	r-tuple of attributes associated with the user v_i
$Att(G_A)$	the list of attributes associated with G_A
$\sim_d$	the equivalence on $V(G)$: $v \sim_d w \Leftrightarrow deg(v) = deg(w)$
$Q(G), Q(G, v)$	an $\sim_d$-equivalence class of G
$Q^a(G^*)$	an attribute equivalence class of G^*
$u(Q^a(G^*))$	the attributes of $Q^a(G^*)$
$deg_G(v)$	degree of the node v in G
d_G	degree sequence of G
$\delta^*_{G,s}$	degree sequence of G anonymized with heu-kDA using s
$N_G(v)$	neighborhood of v in G
$N^d_G(v)$	d-neighborhood of v in G
$AVD(G)$	average vertex degree in G
$APL(G)$	average shortest path length in G
$LCC(v, G)$	local clustering coefficient of v in G
$ACC(G)$	average clustering coefficient of G
$T(G)$	transitivity of G
$\mathbf{Adj}(G)$	the adjacency matrix of G
$\lambda(G)$	the largest eigenvalue of G
k	anonymization parameter
s	the modification parameter of heu-kDA
r_s	residue
f_a	anonymization function
L	the list of indices
F_j	an automorphisms on G^*
$g_f(s)$	a frequent subgraph with the minimal support s
H, P_{ij}	subgraphs of G
P'_{ij}	supergraphs of P_{ij}
Q_{ij}	subgraphs of P'_{ij}; $E(Q_{ij}) = E(P'_{ij}) \setminus E(P_{ij})$
$\mathbf{M}$	a matrix
$r_i(\mathbf{M})$	the i-th row of $\mathbf{M}$
$rc(\mathbf{M})$	the number of rows of $\mathbf{M}$
$\mathbf{Adj}_i$	the adjacency matrix of $P'_{i1}, \ldots, P'_{ik}$

CH	the bit part of the chromosome
$varCH$	the part of the chromosome representing the list of vertices
$Cost(G, G^*)$	the total anonymization cost
$VCost(G, G^*)$	the anonymization cost caused by vertex edits
$ExCost(G, G^*)$	the extension cost
$ExCost_i(H)$	the extension cost caused in the i-th round of HAkAu
$CECost(G, G^*)$	the crossing edges cost
$FF(I)$	the fitness function on the individual I

List of abbreviations

AS	Authorities Score
BC	Betweenness Centrality
CC	Closeness Centrality
CD	Community Detection
$Deg.$	Degree Distribution
ED	Effective Diameter
EV	Eigenvector
HS	Hubs Score
$Infe.$	Infectiousness
JD	Joint Degree Distribution
NC	Network Constraint
PR	Page Rank
RX	Role Extraction
GA	genetic algorithm
SN	social network
SNAP	Stanford Network Analysis Project
k-DA	k-degree anonymization algorithm [84]
heu-kDA	k-DA with the heuristic high-degree noise addition method
FkDA	fast k-degree anonymization algorithm [86]
tMean	t-Means Clustering [141]
Union	Union-split Clustering [141]
HAkAu	hybrid algorithm for k-automorphism anonymization
KM	the KM algorithm [163]
SecGraph	the SecGraph evaluation tool [60]
DUEF-GA	the DUEF-GA evaluation tool [19]
NS	Narayanan-Shmatikov's attack [104]
Per.	Yartseva-Grossglauser's attack [154]
Rec.	Korula-Lattanzzi's attack [68]
GraMi	the graph mining algorithm [31]
SiGraM	the single graph miner algorithm [72]

Contents

1. Introduction

A social network describes relationships between individuals or organizations. The social world can be viewed as an intertwined net of connections through which individuals are bound together [128]. The social network metaphor serves social scientists to make the complex and unfamiliar patterns of the social world comprehensible by relating them to well-understood concepts. Currently, social relationships are also shared online by online social network services. Benefiting from social network services is paid with handing over sensitive personal information to service providers. The growing popularity of online social network services has triggered the collection of large amounts of data. The gathered datasets are full of valuable information for researchers in different fields, such as product marketing, social psychology, information security, and healthcare. Thus, the service providers share their collected datasets with third parties. While publishing social network data brings significant advantages, it also causes privacy-preserving problems since it contains sensitive and private information.

Privacy is a complex concept of protecting sensitive data and information from unauthorized access. Because of the rapid expansion of computing and worldwide Internet usage, privacy-preserving techniques have become a frequently examined issue. Sharing and publishing users' data indeed threaten the privacy of individuals consuming online social network services.

Anonymization enables providers to publish their data while preserving individuals' privacy. Providers apply an anonymization technique to their dataset to protect the personal information of individuals and publish the anonymized version of the dataset. The aim of anonymization techniques is to prevent an adversary from distinguishing an individual from a group of others or revealing any sensitive information linked directly to them.

Preserving privacy in a dataset is managed by modifying the original dataset with a selected anonymization method to provide the required level of anonymity. Different anonymization methods and different input settings ensure different levels of privacy protection. However, the aim of anonymization is always to keep as much data utility as possible in the anonymized dataset such that the dataset is still valuable for data analysts. Providing a high level of privacy protection usually implies more dataset modifications or limits access to the released dataset. In other words, providing a high level of privacy causes more extensive information loss. Thus, the crucial issue in anonymization is finding the trade-off between privacy and data utility.

I decided to do my PhD research in the field of anonymization because my scientific interests are in information security, and I find anonymization to be a perfect field of study in which I can employ my knowledge of mathematical methods achieved in my Master's studies as well as my working experience in the cybersecurity software company. Since the anonymization of social network datasets offers many challenging issues and the opportunity for researchers to identify open problems, my research has been focused on this area.

This thesis focuses on anonymization and deanonymization approaches that address the privacy-preserving issue of identity disclosure in anonymized social network datasets. Anonymization approaches aim to protect the users of social

networks from being linked with the node representing them in the anonymized graph of the social network. On the other hand, studies in the deanonymization point out privacy threats and weak spots of anonymization methods to encourage researchers to improve the methods and develop new anonymization approaches.

In the early stage of my research, when I was preparing a comprehensive literature review, I identified three open problems. To solve the issues indicated, I propose three novel methods. All proposed methods are implemented in MATLAB, and their efficiency is evaluated by running experiments on synthetic or real social network datasets.

As the first contribution, I present the composition attack on social network datasets, a novel deanonymization approach based on the composition attacks applied to relational datasets. I propose the concept of the composition attack, implement it and test it on the set of synthetic networks. Furthermore, I introduce a heuristic noise addition procedure improving the state-of-the-art k-degree anonymization k-DA algorithm. The procedure is implemented in the greedy version of the k-DA algorithm. The resultant k-degree anonymization algorithm, named heu-kDA, is experimentally proved to preserve data utility better than the original algorithm. Finally, I focus on enhancing the k-automorphism anonymization method to keep data utility better. As a result, I revise the design of the original k-automorphism anonymization approach and propose the novel hybrid k-automorphism anonymization algorithm called HAkAu. In the design of HAkAu, I use a genetic algorithm to solve the NP-hard subtask of the k-automorphism anonymization problem. Both proposed anonymization methods, namely heu-kDA and HAkAu, are implemented and tested on real-world social network datasets. Moreover, the experimental results are evaluated with the SecGraph evaluation tool to make them easily comparable with any future research. Except for the mentioned methods, I present some minor findings that I also find beneficial for this field of study.

The rest of the thesis is organized into seven chapters. The comprehensive literature review of the state-of-the-art deanonymization and anonymization approaches is given in *Chapter 2*. In *Chapter 3*, the found open problems are described in detail, research questions are formally defined, and the goals of the thesis are stated. The necessary backgrounds in graph theory, equivalence classes, receiver operating characteristic analysis and genetic algorithms are given in *Chapter 4*. The proposed composition attack and the corresponding results are shown in *Chapter 5*. The heuristic noise addition method and results of the data utility analysis of heu-kDA are presented in *Chapter 6*. Finally, the k-automorphism anonymization algorithm HAkAu and the corresponding experimental results are described in *Chapter 7*. The thesis is concluded in *Chapter 8*.

2. The state of the art

In this chapter, the topic of anonymization is introduced. The emphasis is given to the anonymization of social network datasets. Furthermore, the system is formalized, the basic terms are defined, and the state-of-the-art anonymization and deanonymization methods and approaches are presented. Except for the studies and research focusing on anonymization and deanonymization methods, surveys and tools evaluating anonymized datasets, the literature review also contains studies focusing on genetic algorithms and their application in social network analysis and anonymization.

2.1 Social networks

The term *social network* has become a powerful image of social reality used by sociologists analyzing social structures since the 1930s [128]. Individuals are tied to each other by invisible bounds produced by human interactions and their unintended consequences. All springs between individuals can be joined into a vast network of connections describing human society [128].

After the rapid growth of computational technologies, the social interactions between individuals have been partially shifted into a virtual environment. The first online social networks *Classmates* and *SixDegrees* were launched in 1997. They set up the cornerstone for the multimillion business affecting the human population worldwide. The number of participants in online social networks (SNs) steadily grows and is expected to reach 4 billion in 2023 [134]. Since providers collect various kinds of data about each user, the amount of information in SN datasets is enormous. SN datasets have become a precious source of information about human behaviour, establishing relationships, shopping habits and mobility patterns for academic [90], medical [63, 103], and marketing [47] research all over the world.

2.1.1 History of online social networks

The first online social network *SixDegrees*, launched in 1997, was unable to create a profitable business model due to the poorly developed web technologies and the fact that the marketing industry was not yet prepared for the expansion to the online world [52]. In the following years, several more SNs with different functionalities arose like *AsianAvenue*, *Black-Planet* or *Live-Journal*. Creating profiles and making lists of friends or guestbooks became part of the offered service. While the early networks focused mainly on private networking, the first business networks were founded in 2001. The first business network was *Ryze*, which served as the role model for the successive business networks, including *LinkedIn*, which was founded in 2003 [52]. The first dating networks like *Match* and *Friendster* were launched too. In *Friendster*, people could make ties with friends of their friends. The network was based on the assumption that friends-of-friends are more likely to build romantic relationships. Its provider restricted access to other users in *Friendster*. Users were possible to make links with users within a four-degree distance. Until 2004, *Friendster* had been the largest SN

[52]. However, the popularity of *Friendster* caused technical problems since the providers were unprepared for the rapid growth of users and data.

The well-known *MySpace* network was established in 2003 as a reaction to users leaving *Friendster*. Its primary goal was to catch the users leaving from *Friendster*. Since the first participants in *MySpace* were musicians, the connections between music bands and their fans helped *MySpace* to build up a large user community involving young people [52]. In the next few years, many SNs were founded, aiming for a narrower audience and targeting groups of people with the same particular interest or living in a common demographic region.

A new era of SNs started with establishing *Facebook* in 2004. At first, *Facebook* was launched as the SN for Harvard students. A year later, the provider opened it for students from other universities and a broad audience followed shortly after. With the growing popularity of *Facebook*, SNs generated increasing economic interest among investors [52]. In 2010, *Facebook* was the most popular online SNs, with more than 800 million users used worldwide, providing communication platforms in 70 languages. As the response to privacy concerns connected with *Facebook*, *Unthink* and *Folksdirect* launched with the promise of offering a privacy-focused environment with easier control of privacy. One of the biggest attacks on *Facebook*'s hegemony was establishing *Google+* in 2011. However, its user engagement was lower than its competitor, and its benefit was mainly in connecting various Google services.

In 2022, *Facebook* was still the most popular online SN with nearly 3 billion monthly active users [135]. Closely behind there were *Youtube* with 2.5 billion and *Whatsapp* with two billion users while *Instagram*, *WeChat* and *TikTok* reached over one billion users [135].

2.1.2 Social network dataset with users' attributes

In online social networks, users make social ties with other users. Hence, the online social network can be represented as a graph, where nodes represent users and edges represent the connections between users. Moreover, users can share personal and non-personal information with their associates or all social network users. Personal information related to the user can be represented as the node labels in the graph.

Semantic anonymization methods, like the class-based anonymization algorithm [12], consider both the graph structure and the user information in their processes. The social network is represented as an undirected graph with node labels in this scenario.

Definition 1 (Social network with users' attributes [138])**.** *The* social network datasets with users' attributes *is represented by an undirected graph* $G_A = (V(G_A), E(G_A), U(G_A))$*, where* $V(G_A) = \{v_1, \ldots, v_n\}$ *is a set of vertices representing a set of individuals connected within the social network,* $E(G_A)$ *is a set of edges representing the relationships between the individuals and* $U(G_A) = \{u_1, \ldots, u_n\}$ *is a set of* r*-tuples representing the values of attributes characterizing the individuals. The* i*-th element* $u_i = (u_{i1}, \ldots, u_{ir})$ *is associated with the* i*-th vertex* v_i*,* $i \in \{1, \ldots, n\}$*.*

The terms "vertex" and "node" are used interchangeably in the remainder of the thesis. For simplicity, the graph G_A representing a social network is also

called "a social network G_A" or "SN dataset G_A". Without loss of generality, it is assumed that every individual participating in a social network has only one account in that social network. Therefore, for every individual I participating in the social network G_A, exactly one node $v(I) \in V(G_A)$ represents the individual I. Therefore, the expression "individual v" is also used in this thesis. As usual, the edge connecting vertices $v_1, v_2 \in V(G_A)$ is denoted by $(v_1, v_2) \in E(G_A)$. To simplify the notation, the r-tuple of values of attributes describing the individual v is denoted by u_v.

Let $Att(G_A)$ be a list of attributes associated with a graph G_A. Attributes in $Att(G_A)$ represent various user characteristics. Typical attributes are age, gender, postcode, birthday, home town, location or political affiliation. A special attribute, the *identifying attribute*, appears in every social network and identifies its users completely (i.e., username, login, first name and surname). Identifying attributes are always removed during the anonymization process before data publication. However, removing the identifying attributes is not a satisfactory anonymization method, as presented in [6]. A combination of *non-identifying attributes* or a linkage of some non-identifying attributes with available external data may also cause identity disclosure. The "potentially dangerous" attributes form *quasi-identifiers*, as introduced in [35].

Definition 2 (Quasi-identifier [35])**.** *A* quasi-identifier *is a set of non-identifying attributes such that at least one individual of the original social network can be uniquely identified by linking these attributes with an external data item.*

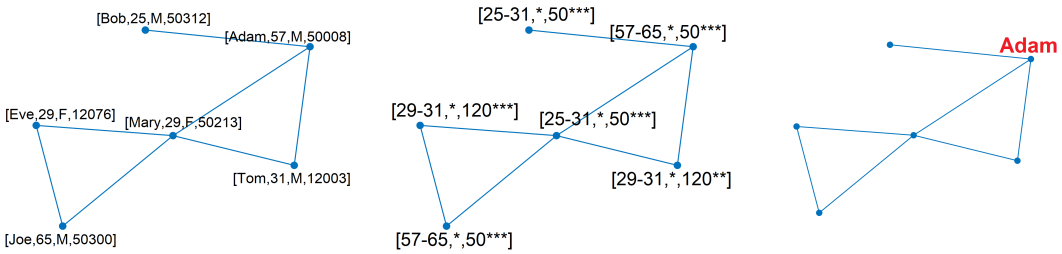
Quasi-identifiers constitute attributes that are not generally private; however, releasing them together in a non-anonymized form could cause information leakage, leading to a connection between the anonymized data record and a particular individual. The attributes belonging to the quasi-identifier are called *quasi-identifying attributes*. For instance, the combination of age, gender and postcode can reveal the individual's identity if the combination is unique in the dataset.

2.1.3 Social network dataset without users' attributes

So far, we have discussed the problem of re-identification of users in SN having users' attributes. As stated before, even if the identifiers like names or nicknames are removed from the data, the target user can be re-identified using the combination of quasi-identifying attributes and information collected from external data sources. The external information that the attacker has collected before the actual attack is called his or her *background knowledge*. The background knowledge can be extracted from the target SN itself or a source unrelated to the targeted SN.

However, even if the identifiers and quasi-identifiers are completely removed from the SN dataset, the privacy issue of users' re-identification persists. The graph structure itself can be considered a quasi-identifier. As shown in [6], the attacker can gather information about the graph structure around the target users. For instance, he or she can find the number of users linked to the target users, disclose their neighbourhoods or collect even more structural information. Hence, the graph structure can reveal the users identity. Thus, the graph structure can be considered a quasi-identifier and has to be modified before publishing

the SN dataset. The situation is illustrated in *Figure 2.1.* There is the example of the original network G_A with all attributes, G_A without identifiers and with anonymized quasi-identifiers and G_A without any attributes. Note that if the attacker knows that his or her target user (*Adam*) is linked with three other users, he or she can easily identify *Adam* in all three versions of G since there is only one vertex with degree equalling to three.



a) original network G_A b) G_A with anonymized attributes c) G_A without attributes

Figure 2.1: Social network with attributes. (Source: author's work [93].)

The deanonymization attacks exploiting only the knowledge about the graph structure are called *structural attacks*. Therefore, anonymization methods focused on preventing structural attacks consider only the graph's structure (nodes and edges). The user information represented with node labels is omitted. In this scenario, the social network dataset is assumed to be unlabelled and contains no identifying information about users. In other words, the assumption is that all user information has been removed before applying the anonymization method, and the anonymization method modifies only the graph structure. The social network without users' attributes can be depicted as an undirected unlabelled graph. Clearly, when the set of attributes is omitted, the labelled graph G_A turns into an unlabelled graph G.

Definition 3 (Social network without users' attributes [84])**.** *The* social network dataset without users' attributes *is represented by an undirected graph* $G = (V(G), E(G))$*, where* $V(G) = \{v_1, \ldots, v_n\}$ *is the set of nodes representing the participating users and* $E(G)$ *is the set of edges representing the social relationships between users. The edge between nodes* v_i *and* v_j *is denoted by* (v_i, v_j)*.*

In this thesis, the research focuses on SN datasets that can be represented well with undirected graphs. It means that the relationships between users are assumed to be bilateral. The user u is linked with the user v if and only if the user v is connected with the user u. Note that some SNs are better represented with directed graphs. For instance, in *Twitter*, the user u can follow the user v, but it is possible that the user v does not follow the user u. Hence, it is more suitable to represent the *Twitter* dataset with a directed graph. A similar situation is in citation networks where nodes represent scientific papers. The edge between nodes u and v is directed from the paper u toward the paper v in case v is cited in u [160].

It remains to mention that some social networks can also be described by edge-labelled graphs. The edge-labelled graphs represent the SNs having additional information about user interactions. For instance, *Bitcoin trust weighted*

signed network datasets [70, 71] can be represented with the directed edge-labelled graphs. The datasets contain information about who trusts whom among those who trade using Bitcoin on particular platforms. Since Bitcoin users are anonymous, there is a need for measuring the reputation of users to prevent transactions with fraudulent and risky users [74]. Users of the given Bitcoin platform rate other users of the platform on a scale of -10 to 10 in steps of 1. Hence, the directed edges of the corresponding graph are weighted with values in $\langle -10; 10 \rangle$ that represent the source user's rating for the target user. However, this model of SN is omitted since the thesis focus on anonymization problems in unlabelled or node-labelled SNs.

2.1.4 Real-world social network datasets

Real-world social networks are scale-free networks with power-law degree distribution, as presented in [99]. Especially in large SNs, the degrees of vertices follow the power law distribution. It means there are a few nodes with high degrees, and most of them have very low degrees. For example, there is shown the degree distribution of the SN *Email Enron* [67] with 36,692 nodes in *Figure 2.2*. For clarity of the illustration, nodes with the highest and lowest degrees were omitted in *Figure 2.2*. More precisely, five nodes with a higher degree than 100 are omitted, and 11,211 nodes with a degree equalling 1, 3,800 nodes with a degree equalling 2 and 5,167 nodes with a degree value of 3. Nodes with a higher degree than 100 correspond to 2% of all nodes, while nodes with a lower degree than 4 correspond to 55% of all nodes.

Consequently, many graphs describing SNs are sparse [106]. Thus, they have much fewer links than the possible maximum number of links within that network. Moreover, the real-world SNs demonstrate the "small-world phenomenon" [147]. It means that in each SN, any two individuals are likely to be connected through a short sequence of links. Therefore, the average shortest path between vertices is often surprisingly small in large SNs. The small-world phenomenon has been a subject of many sociologist studies [66].

The samples of real SN datasets are available online for academic and research purposes. The largest repository of graph data is the Stanford Large Network Dataset Collection [74], which is part of the Stanford Network Analysis Project (SNAP). The SNAP dataset collection consists of more than 50 large network datasets from tens of thousands of nodes and edges to tens of millions of nodes and edges. It includes social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks. Except for the dataset collection, SNAP provides a general-purpose network analysis and graph mining library which efficiently manipulates large graphs, calculates structural properties, generates regular and random graphs, and supports attributes of nodes and edges [75].

Another sizeable comprehensive collection of network data is gathered in Network Repository [122]. The network collection includes relational, attributed, heterogeneous, streaming, spatial, time series network data, and non-relational machine learning data. The repository was established to improve and facilitate the scientific study of networks by making it easier for researchers to interactively visualize, analyze, and ultimately download an extensive collection of networks.

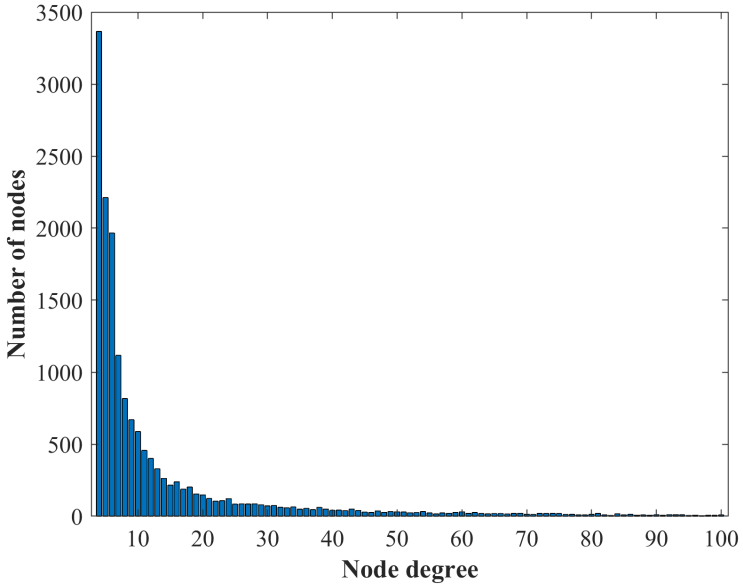


Figure 2.2: Degree distribution of Email Enron network. (Source: author's work.)

Except for SNAP and Network Repository, plenty of other sources provide smaller data collections like Pajek depository [7] or various GitHub projects.

In producing experimental results, I used 14 real SN datasets differing in size and network metrics available in the mentioned repositories. More precisely, from the SNAP library, I employed General Relativity and Quantum Cosmology collaboration network (GrQc)[78], Gnutella peer-to-peer network [78, 119], Wikipedia vote network [80], Caida AS Relationships Datasets [76], High-energy physics theory citation network (HepTh) [76, 37], Stanford web graph [79], Enron email network [79, 67], Amazon product co-purchasing network [77], Epinions social network [118] and the networks describing the social ties of geosocial networks Gowalla [27] and Brightkite [27]. Furthermore, I ran experiments also on three smaller networks: Polbooks [69], and Polblogs [2] stored in the Network Data Repository [122], and Prefuse network[50] stored in the GitHub repository of the Prefuse project (Java-based toolkit for building interactive information visualization application) [51]. The summary of sizes of the mentioned networks together with their average clustering coefficient (ACC) and diameter is given in *Table 2.1*[1]. Except for the values of ACC and diameter of PolBooks, Prefuse and Polblogs, all values in *Table 2.1* are taken from the website of repositories [51, 122, 74]. The remaining values were computed.

2.2 Anonymization

As mentioned in the previous section, SN datasets are of great use in academic and business research. However, sharing and publishing datasets are limited since the privacy of each individual has to be protected. Anonymization allows providers to share or publish their datasets while preserving users' privacy. The basic principle of anonymization is to modify the original dataset to satisfy the required level of

[1]The definitions of ACC and diameter can be found in *Chapter 4*

SN dataset	#nodes	#edges	ACC	Diameter
Polbooks	105	441	0.488	7
Prefuse	121	169	0.488	4
Polblogs	1,224	16,715	0.313	8
GrQc	5,242	14,496	0.53	17
Gnutella	6,301	20,777	0.011	9
Wiki-Vote	7,115	103,689	0.141	7
Gowalla	19,659	950,327	0.237	14
Caida	26,475	106,762	0.208	17
HepTh	27,770	352,807	0.312	13
Stanford-web	28,190	2,312,497	0.598	674
Email-Enron	36,692	183,831	0.497	11
Brightkite	58,228	214,078	0.172	16
Epinions	75,879	508,837	0.138	14
Amazon	403,394	3,387,388	0.418	21

Table 2.1: List of real SN datasets. (Source: author's work)

security and keep as much data utility as possible at the same time. The aim of anonymization techniques is to prevent an adversary from distinguishing an individual from a group of others. Formally, the anonymization procedure can be expressed as a function from the input graph to the anonymized one.

Definition 4 (Anonymization [138])**.** *Anonymization of social network data is a function $f_a : G_A \to G_A^*$, where G_A is a graph describing the original social network data and G_A^* is a graph describing the anonymized social network data. The function is denoted by $f_a = (f_a^V, f_a^E, f_a^U)$ such that $f_a^V : V(G_A) \to V(G_A^*)$, $f_a^E : E(G_A) \to E(G_A^*)$, $f_a^U : U(G_A) \to U(G_A^*)$. The anonymization of unlabelled graph G is the function $f_a = (f_a^V, f_a^E)$ such that $f_a^V : V(G) \to V(G^*)$, $f_a^E : E(G) \to E(G^*)$.*

The anonymization function f_a should have the following properties.

1. G^* does not contain any direct sensitive information of G
2. G^* protects the privacy of individuals involved in the social network against different kinds of attacks
3. The loss of data utility in G^* is the smallest possible under fulfilling the previous premises.

Adhere to the first property of f_a, identifying attributes, such as username, login, first name and surname, are always removed in each anonymization method. Nonetheless, the crucial property is the minimization of information loss caused by anonymization. Finding the most fitting anonymized graph G^* to the original one G is the optimization problem where the output solution G^* has to satisfy predefined anonymity property, and the data utility loss is minimal.

2.2.1 The complexity of anonymization

The anonymization problem can be viewed as the optimization problem of minimizing information loss where the constraint is the required level of privacy. The required level of privacy affects the complexity of the problem. The complexity of the optimization problem that arises from trying to find G^* that satisfies the k-anonymity property is proven to be NP-hard[2] in [45, 46, 44, 15, 8, 9]. The k-anonymity property and SN anonymization methods based on k-anonymity are correctly defined in *Section 2.3.8*.

Hartung et al. proved in [46] that finding k-degree anonymized graph was an NP-hard problem in case edge addition operations were applied on the input graph. In [8], Bazgan et al. focused on deletion operations. They showed that the k-anonymity problem is NP-hard when allowing edge or vertex deletion. Furthermore, they later proved in [9] that the problem was also NP-hard when edge rotation operations (edge switch operations) were applied. Bredereck et al. showed in [15] that the problem is NP-hard when considering vertex addition operations. The complexity of anonymization methods other than k-anonymity is not widely studied.

2.3 Anonymization methods

Generally, the anonymization methods can be categorized as semantic or structural. The semantic anonymization methods address the problem of anonymizing SN with attributes G_A. On the other hand, the structural anonymization methods focus on modifying the graph structure of G. Additionally, G_A can be anonymized by combining the structural SN anonymization method and the anonymization method for relational datasets [161, 25]. In that case, the set of vertices $V(G_A)$ and the set of edges $E(G_A)$ are anonymized using structural anonymization methods to hide the original graph structure. Then, the attribute set $U(G_A)$ is anonymized with the procedure for relational data anonymization to fulfil the required level of privacy.

2.3.1 Methods of relational data anonymization

In this section, the fundamental methods of relational data anonymization are presented. The methods are k-anonymity [125], l-diversity [89], t-closeness [82] and differential privacy [29]. Many extensions of those methods have been later published to improve those methods in terms of data utility [91]. However, the basic principles are still exploited in the state-of-the-art methods of anonymizing relational and SN datasets. This section uses the terms "table" and "relational dataset" interchangeably.

Definition 5 (k-anonymity [125])**.** *Let R be a relational dataset. Then R is said to be k-anonymous if R contains no identifying attributes, and for each record in R, there exist at least $k-1$ other records that have the same quasi-identifying attributes.*

[2]The definitions of the NP class and the NP-hard problem can be found in *Chapter 4*

The probability that an attacker identifies his or her target individual in the k-anonymized dataset equals $\frac{1}{k}$. The procedures used to achieve k-anonymity in the relational datasets are *generalization* and *suppression*. The principle of generalization is to substitute the concrete values (ex. *married*, *divorced*, *widow*) with the more general term (ex. *once_married*). The specific values of attributes described with numbers (ex. age: 28, 41, 35) can be replaced with intervals (ex. age: 25-45). Each quasi-identifying attribute in the dataset has several domains of generalized values that satisfy the domain generalization hierarchy [125]. During the anonymization, one domain for each quasi-identifying attribute is selected such that the table is k-anonymized. Since the goal of the anonymization is to preserve as much data utility as possible, the anonymized table should satisfy the minimal generalization property. Simplifying the minimal generalization property means that other k-anonymized versions of the input table contain more general values.

The complementary method to achieve k-anonymization is the *suppression*. Suppressing means removing data from the relational dataset such that they are not released at all. If the table contains n quasi-identifying attributes that are represented with n-tuples of values in the table, some elements of each n-tuples are omitted during suppression. The output of the suppression is the table with m columns, $m < n$, and satisfies the k-anonymity property. The best results are achieved with anonymization methods combining both approaches, generalization and suppression. Some attributes of R can be generalized, and others are suppressed.

An example of a relational dataset and its 3-anonymized version, obtained by generalization and suppression, is given in *Figure 2.3*. The datasets gather information about the spoken language in a social group. The identifying attribute is obviously removed when the dataset is anonymized. The attribute *Language* is the *sensitive attribute* in the dataset in *Figure 2.3a*. The sensitive attribute is the crucial information in the relational dataset and the focus of further data analysis. On the other hand, the identifiers and quasi-identifiers describe the individuals whose records are in the table. The sensitive attribute is not modified during the k-anonymization process. Usually, there is only one sensitive attribute in the dataset. The same attribute (*Language*) can play a different role in different datasets. In one dataset, it could be labelled as the sensitive attribute; in the other, it could be found to be a quasi-identifying attribute. Thus, *Language* is not anonymized, unlike the quasi-identifier (*Age, Gender, CCode, Marital*) in *Figure 2.3b*. The final table is 3-anonymized; hence for every record, there are two other records with the same quasi-identifiers.

The l-diversity, introduced in [89], focuses on protecting sensitive attributes. The l-diversity approach assumes that the relational dataset already satisfies the k-anonymity. Thus, there are groups of at least k records having the same values of quasi-identifiers. The aim of l-diversity is to provide sufficient diversity in the values of sensitive attributes in the group of k records with the same values of quasi-identifiers. The approach ensures for each group that the l most frequent values of the sensitive attributes have roughly the same frequencies [89].

The further extension of the l-diversity approach is t-closeness introduced in [82]. The t-closeness provides better privacy protection at the cost of more significant information loss. Given the threshold t, the t-closeness privacy model

Id	Age	Gender	CCode	Marital	Language
Barbara	37	F	804	Single	Spanish
Daniel	63	M	616	Widowed	Portugese
David	63	M	112	Divorced	Spanish
Elizabeth	37	F	348	Single	German
Charles	37	M	756	Widowed	Italy
Christopher	56	M	276	Widowed	German
Jennifer	47	F	643	Divorced	English
Jessica	49	F	756	Widowed	Spanish
John	43	M	642	Married	English
Joseph	63	M	203	Married	Russian
Linda	63	F	616	Married	English
Mary	41	F	703	Divorced	Italy
Michael	29	M	642	Married	Spanish
Patricia	28	F	756	Single	German
Richard	54	M	438	Single	German
Robert	31	M	804	Divorced	German
Sarah	28	M	528	Divorced	Russian
Susan	32	F	498	Single	French
Thomas	43	M	643	Single	Russian
William	45	M	203	Single	Russian

a) relational dataset

Age	Gender	CCode	Marital	Language
28-37	F	*	Single	Spanish
54-63	M	*	Once_married	Portugese
54-63	M	*	Once_married	Spanish
28-37	F	*	Single	German
37-56	M	*	Once_married	Italy
37-56	M	*	Once_married	German
41-63	F	*	Once_married	English
41-63	F	*	Once_married	Spanish
37-56	M	*	Once_married	English
54-63	M	*	Once_married	Russian
41-63	F	*	Once_married	English
41-63	F	*	Once_married	Italy
28-31	M	*	Once_married	Spanish
28-37	F	*	Single	German
37-56	M	*	Single	German
28-31	M	*	Once_married	German
28-31	M	*	Once_married	Russian
28-37	F	*	Single	French
37-56	M	*	Single	Russian
37-56	M	*	Single	Russian

b) 3-anonymized relational dataset

Figure 2.3: Relational dataset and its 3-anonymized version. The identifying attribute *Id* is removed in anonymization. The quasi-identifiers *Age* and *Marital* are generalized, and the quasi-identifier *CCode* is suppressed. The quasi-identifier *Gender* and the sensitive attribute *Language* are untouched. (Source: author's work.)

ensures that the distribution of the values of sensitive attributes in any group of k records with the same values of quasi-identifiers differs from the overall distribution in the table by at most t. In other words, the distance between these two distributions should be smaller than or equal to t.

The information, whether records of an individual are or are not included in a relational dataset, can also be viewed as sensitive information. This privacy-preserving issue is addressed by the differential privacy approach [29]. Differential privacy differs from previous methods by focusing on how the database behaves with and without an individual's data instead of comparing what can be learned about an individual with and without the database. The differential privacy approach makes no assumptions about the adversary's background knowledge. It means that the computational power of the attacker could be unlimited, and the attacker could have any external information or any structural knowledge about the original graph, and yet the database satisfying differential privacy is protected against his or her attack. In other words, the concept of differential privacy is independent of the attacker's auxiliary information and computational power and protects against any kind of attacks [6]. Differential privacy is a *perturbation* method. The anonymity property is achieved by adding noise such that the original data values are replaced with synthetically generated values. Furthermore, the synthetic values are generated so that statistical information does not differ much in both datasets [91]. Hence, it provides a solid privacy guarantee since it is a statistical property. Additionally, the number of queries on the data stored in the dataset satisfying differential privacy is limited [29].

Definition 6 (ϵ-differential privacy [29])**.** *A randomized function f ensures ϵ-differential privacy if for all relational datasets R_1 and R_2 that differs on at most*

one row holds that

$$\forall S \subseteq \mathcal{H}(f): \quad Pr[f(R_1) \in S] \leq e^{\epsilon} \cdot Pr[f(R_2) \in S],$$

where $\mathcal{H}(f)$ is the range of f

The parameter ϵ determines the degree of privacy. To achieve a high level of protection, ϵ should be relatively low. However, the low value of ϵ directly limits the number of queries that can be made on the stored data. The dataset with $\epsilon < 1$ can only be queried a few times. After that, access to the data is no longer permitted as privacy cannot be guaranteed [126]. If the data usability is preferred, ϵ is set to be higher, and the data protection is reduced.

2.3.2 Semantic anonymization methods

Semantic methods of SN anonymization focus on the issue of identity disclosure in SN with attributes. They exploit the principles of anonymization approaches for relational datasets described in the previous section. Zhou and Pei focused on anonymizing SN with attributes in [161, 162]. In [161], they introduced the k-neighbourhood method for anonymization of the graph structure and used generalization for anonymizing the attributes. The structural k-neighbourhood anonymity method is described in detail in *Section 2.3.8*.

Furthermore, they extended their approach with the principles of l-diversity in [162]. At first, the graph structure is modified to satisfy the k-neighbourhood anonymity. Let the groups of vertices having isomorphic neighbourhood form the equivalence class. Then, to protect the privacy of the individuals in the equivalence class, the distribution of the sensitive values in each equivalence class should be sufficiently diverse in the sense of l-diversity. Hence, an adversary with the background knowledge of the neighbourhood structure and no additional knowledge about the vertex labels only can infer the sensitive label for his or her target individual with the probability not greater than $\frac{1}{l}$.

The clustering approaches modifying SN with attributes were proposed in [17, 140, 12]. The aim of clustering methods is to aggregate nodes into clusters. Let $C = \{v_1, \ldots, v_n\}$ be a cluster aggregating nodes $v_1, \ldots, v_n$. Assume that there are m different quasi-identifiers in the network. Then the quasi-identifying attributes $u_1, \ldots, u_n$ describing nodes $v_1, \ldots, v_n$ are anonymized with generalization, such that for each $j = 1, \ldots, m$, the values $u_{1m}, \ldots, u_{nm}$ of the same quasi-identifier are replaced with more generalized value. Similarly, all attributes in all clusters are processed. As a result, each cluster is described with m-tuple of generalized values.

A completely opposite approach was introduced in [138]. Sun et al. introduced the splitting anonymization method, where the attributes were anonymized at first, and the graph structure was anonymized afterwards. In the splitting method, the attributes are generalized and split into groups. Each vertex is linked to one attribute group. Then each vertex v is divided into several vertices according to its degree. More precisely, $deg(v)-1$ artificial vertices are added into the graph structure, and the edges connecting the original vertex v to its neighbours are divided between the newly created nodes. It means that $deg(v) = 1$, and the degree of each copy of v equals one too. Each newly created vertex is

linked with the same attribute group as the vertex v. Finally, some edge editing operations are applied to obtain the final G_A^*.

The k-degree-l-diversity anonymity model protecting the structural information and sensitive labels of individuals was proposed by Yuan et al. in [157]. They introduced the methodology based on adding noise nodes. The method was extended in [23], where the noise node addition method is combined with the eigenvector centrality concept to preserve the social importance of real nodes.

2.3.3 Differential privacy

The differential privacy technique was also extended to the SN data in [48]. In SN data publishing and sharing, the aim of the differential privacy is to guarantee that an adversary having the published result will not be able to determine whether an individual participated in the network G or whether there exists the edge (u, v) in G for $u, v \in V(G)$ [1]. The differential privacy provides high-level security anonymization by adding noise to the query results and restricting the structure of queries on the output dataset. Unlike other anonymization techniques, differential privacy allows only constrained analysis of the SN datasets, and this approach can limit the successive data mining of the dataset. The differential privacy methods compute either node-related or edge-related statistics from G. Therefore, they are categorized into node-level and edge-level differential privacy methods. The main goal of the proposed methods is to reduce the amount of noise which has to be added to meet the requirements of differential privacy.

Sala et al. presented the node level differential privacy method called *Pygmalion* in [123]. Their main idea was to extract the graph structure into degree correlation statistics and introduce noise in the resulting dataset to produce the anonymized graph. They focused on reducing the noise necessary to add into G. Other node-level differential privacy methods were presented in [14, 114, 64].

The relaxed version of differential privacy is proposed in [117]. Rastogi et al. proposed the edge-level method that enabled sending more expressive queries to the anonymized dataset, which improved the data utility. Xiao et al. proposed a Hierarchical Random Graph model to satisfy the requirements of the edge level differential privacy in [151]. Instead of considering edges, they estimated the link probabilities among users.

2.3.4 Structural anonymization methods

The structural anonymization methods are categorized according to how it modifies the original graph structure. *Clustering methods* transform subgraphs into super-nodes and replace the original links between users with super-links connecting the super-nodes. *Noise addition methods* are based on adding artificial nodes into the input graph. *Edge editing methods* are based on adding, deleting or switching edges. The rest of this section establishes a brief description of all mentioned anonymization approaches and a review of the state-of-the-art methods.

2.3.5 Clustering methods

Clustering methods belong to the class of generalization methods. In the clustering algorithms, the input SN dataset is partitioned into different clusters, which are generalized into super-nodes [140]. Given the input graph G, the clustering algorithms solve the problem of finding the optimal partition of sets $V(G)$ and $E(G)$ into the minimal number of clusters in which any two vertices are indistinguishable from each other [1]. Clearly, the goal is to preserve as much structural information as possible from G in the clustered network G^*. More precisely, given the input graph G, the aim is to find the set of clusters $\mathcal{C} = \{C_1, \ldots, C_n\}$, $C_i \subseteq V(G)$, such that

1. $\forall v \in V(G) \quad \exists i \in \{1 \ldots, n\} : \quad v \in C_i$
2. $\mathcal{C} = \bigcup_{i=1}^{n} C_i = V(G)$
3. $\forall i, j = 1 \ldots, n,\ i \neq j : \quad C_i \cap C_j = \emptyset$

If the anonymized dataset should meet the requirement of k-anonymity, it means that each vertex should be indistinguishable in the group of k vertices, then $|C_i| \geq k$, $\forall i = 1, \ldots, n$. Hence, the vertices are divided into clusters such that each cluster contains a subset of at least k vertices having similar structural features such as a degree or isomorphic neighbourhood. Moreover, the cluster also contains the edges between those vertices. Those edges are called the inner edges of the cluster. Let $IE(C_i)$ be the set of inner edges of C_i. The edges connected vertices from different clusters are external edges. After the vertex partitioning, the clusters are generalized into super-nodes, and all external edges connecting C_i and C_j, $\forall i, j = 1 \ldots, n\ i \neq j$, are generalized into a single super-edge. Each super-node C_i is described by the tuple $(|V(C_i)|, |IE|)$ [1]. The resulting anonymized graph G^* consists of the super-edges (C_i, C_j), the super-nodes C_i and the tuples $(|V(C_i)|, |IE|)$.

Clustering methods were proposed in [140, 49, 141, 17]. The greedy algorithm *SaNGreeA* proposed in [17] built the clustering greedily, one cluster at a time. It selects the cluster, then it selects a node and adds the node to the cluster. The node is selected such that the node addition causes a minimal increment of the anonymization cost. The process continues until the cluster has the size of k.

The sequential clustering algorithm was proposed in [140]. The algorithm begins with partitioning vertices from G at random until there are $\frac{|V(G)|}{\alpha} \cdot k$ vertices in each cluster, where α is the clustering parameter. Afterwards, the algorithm repeatedly examines all vertices and moves them between clusters to minimize the total anonymization cost.

In [141], Thompson and Yao introduced two clustering algorithms, *tMeans* and *union-split*, to classify nodes into clusters based on similar structural features or similar social roles within SN. Moreover, they proposed an anonymization scheme for graph data in which edges were added and removed according to nodes' inter-cluster connectivity. The scheme was combined with the proposed clustering algorithms to obtain the anonymized network.

The proposed clustering methods protect against degree-based attacks. The attacker is considered to have background knowledge of the degrees of nodes within the given radius of the target node. To achieve the required level of k-degree anonymity, the graph G is divided into clusters having the size of at least

k vertices. The two algorithms differ in the way how the graph is divided into clusters.

Their first solution is built on the conventional t-means algorithm that has no minimum-size constraint [110]. The *tMean* algorithm first selects randomly t vertices to be the cluster centres. Each vertex $v \in V(G)$ is added to one of the clusters. The proper cluster is selected according to a chosen distance metric. If the selected cluster has $k+1$ members after the addition, then a vertex with the lowest marginal cost is bumped to its surrogate cluster.

The shortage in the design of *tMeans* is the random selection of the cluster centres in the first step. The *union-split* was designed to avoid this issue. In the first step, each vertex is set to be in its own cluster. Then, it computed all pairwise distances between cluster centres and fixed the next nearest cluster for each cluster. While the size of any cluster is lower than k, it chooses the cluster with the shortest distance to any other cluster and unions it with its nearest cluster. While the cluster size after the union is greater than $2k$, then the cluster is split into two.

Once the vertices of the whole graph are clustered, it remains to anonymize vertices within each cluster. The anonymization approach on the clustered graph, introduced in [141], was called inter-cluster matching. It takes a clustered graph, computes the average degree of nodes within each cluster and determines for each node how many edges it is sufficient to add or remove in order to match the average degree of its cluster.

Both algorithms, *tMeans* and *union-split* are implemented in the SecGraph evaluation tool [60] as one of the state-of-the-art anonymization methods. I use both algorithms and the k-DA algorithm to compare their behaviour with the behaviour of the anonymization method proposed in this thesis.

2.3.6 Noise node addition

As mentioned in *Section 2.3.2*, noise addition methods adding artificial noise nodes into the SN dataset with attributes were introduced in [157, 23]. The structural noise node addition method modifying the original social network to the k-anonymous one was proposed in [26] by Chester et al.. Their algorithm aimed to add artificial vertices and link them to the existing vertices. Unlike edge editing methods, the anonymization cost of noise node addition was computed as the number of added vertices since anonymity is achieved with vertex addition.

2.3.7 Edge editing methods

Edge editing methods are based on modifying the edge set of the original graph $E(G)$ to $E(G^*)$ with the sequence of edge editing operations. Modifying the vertex set $V(G)$ is not the prime intention of those methods. However, the desired anonymity property that should be held in G^* may also require minor modifications of the vertex set. Nevertheless, the number of vertex modifications is negligible to the number of edge set modifications.

Depending on the anonymization approach, $E(G) \subset E(G^*)$ or not. If the anonymization method applies only edge addition operation, then it holds that $E(G) \subset E(G^*)$ since it is not manipulated with any edge from the original graph.

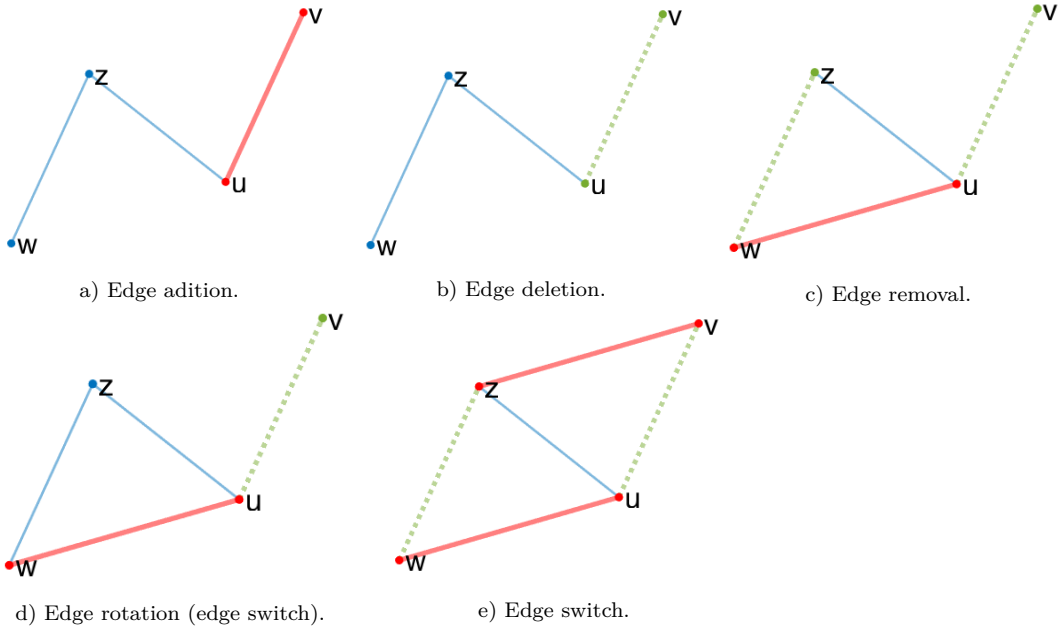


Figure 2.4: Edge editing operations. The red (bold) edges are added with the operations, and the green (dashed) edges are removed with the operations. (Source: author's work.)

In case that edge deletion or edge switches are allowed, then $E(G) \setminus E(G^*) \neq \emptyset$ or $E(G^*) \setminus E(G) \neq \emptyset$. However, the aim of most edge editing methods is to maximize $|E(G) \cap E(G^*)|$ and keep as many edges from the original graph as possible. Since the edge represents the link between users, which is the crucial information stored in SN, keeping as many edges as possible implies keeping as much data utility as possible in G^*.

The edge editing operations are illustrated in *Figure 2.4*. Assume that we modify the input graph G with a single edge edition operation to obtain the output graph G^*. Naturally, edge addition means adding a new edge to the graph structure. More precisely, $(u, v) \notin E(G) \ \wedge \ (u, v) \in E(G^*)$ (see *Figure 2.4a*). Therefore, the degree of both nodes u, v increases by one and $|E(G^*)| = |E(G)| + 1$. The edge deletion corresponds to removing an existing edge from the graph structure. Thus, $(u, v) \in E(G) \ \wedge \ (u, v) \notin E(G^*)$ (see *Figure 2.4b*). Afterwards, $deg(u)$ and $deg(v)$ decrease by one and $|E(G^*)| = |E(G)| - 1$.

More structured edition operations can be defined by combining edge addition and deletion. In [20], an edge removal operation consists of two edge deletions and one edge addition between four vertices. Assume $(u, v), (w, z) \in E(G)$ such that $(u, w) \notin E(G)$ (see *Figure 2.4c*). Then edge removal causes that (u, v) and (w, z) are deleted and (u, w) is added. Hence, $deg(u)$ and $deg(w)$ do not change, $deg(v)$ and $deg(z)$ decrease by 1 and $|E(G^*)| = |E(G)| - 1$.

The definition of edge switch operation varies in the literature. However, the aim of each edge switch operation is to keep the total number of edges, which means that $|E(G)| = |E(G^*)|$ after the edge switch operation. In [20], the edge switch is defined on three vertices. Assume $(u, v) \in E(G)$ and $(u, w) \notin E$ (see *Figure 2.4d*). After the edge switch $(u, v) \notin E(G^*)$ and $(u, w) \in E$. Hence, $deg(u)$ does not change, $deg(v)$ decreases and $deg(w)$ increases. The same operation is called edge rotation in [9].

On the other hand, the edge switch is defined on four vertices in [3]. The advantage of the second definition is that the degree of all involved vertices is preserved. Assume $(u, v), (w, z) \in E(G)$ such that $(u, w), (v, z) \notin E(G)$ (see *Figure 2.4e*). After the edge switch, (u, v) and (w, z) are deleted and (u, w) and (v, z) are added, thus $(u, v), (w, z) \notin E(G^*)$ and $(u, w), (v, z) \in E(G^*)$.

The special edge editing method is the anonymization method established on the random walk proposed in [113] by Mittal et al.. They introduced the algorithm where every edge $(u, v) \in E(G)$ was deleted and replaced with two edges $(u, z), (v, z) \in E(G^*)$ where z was found using the random walk in G. A random walk from u to v in G is a sequence of steps from u to its random neighbour u_1, then to the random neighbour of u_1 and so forth. More precisely, the random walk is a Markov chain with the transition probability depending on the node's degree. The probability that the chain continues from the vertex u to the vertex u_1 equals $P_{u,u_1} = \frac{1}{deg(u)}$ if u_1 is the neighbour of u and $P_{u,u_1} = 0$ otherwise [113]. The length of the random walk used in the anonymization procedure is given by the input parameter t. The larger t corresponds to adding more noise to G^*. To minimize the effect on the degree distribution, the later edge (v, z) is added into G^* with a probability smaller than 1.

2.3.8 Methods based on k-anonymity

Most of the edge editing methods are based on the concept of k-anonymity. The k-anonymity method was initially designed for relational datasets in [125]. The aim of the k-anonymity is to modify the original dataset such that for every record, there are at least $k-1$ other records in the anonymized dataset that have the same quasi-identifiers, as mentioned in *Section 2.3.1*. Hence, the probability that the individual is identified in the anonymized dataset is $\frac{1}{k}$ since the user is unrecognizable in the group of k users having the same quasi-identifiers. The value of the anonymization parameter k is usually chosen by the dataset provider depending on the dataset's size and the requirements of further analysis. To show the similarity between k-anonymity and the definitions of SN k-anonymization methods, *Definition 5* of k-anonymity introduced in *Section 2.3.1* is repeated.

Definition 5 (k-anonymity [125])**.** *Let R be a relational dataset. Then R is said to be k-anonymous if R contains no identifying attributes, and for each record, at least $k-1$ other records have the same quasi-identifying attributes.*

The graph structure can be viewed as a quasi-identifier as revealed in *Section 2.1.3*. This concept led to the extension of the k-anonymity approach from the relational dataset to social network datasets [84, 48].

Hay et al. proposed in [48] the anonymity model in which the SN graphs satisfied the k-candidate anonymity if, for every structure query over the graph, there existed at least k nodes that match the query. The structure queries checked the existence of neighbours of the node or the structure of the subgraph linked to the node. However, they did not develop methods to find k-candidate graphs.

Liu and Terzi introduced in [84] the k-degree anonymity where for each node in G, there should be at least $k-1$ other nodes with the same degree value. The proper definition of k-degree anonymity requires introducing the k-anonymous vector. Note that d_G denotes the degree sequence of the graph G, where the

degree sequence is the list of $deg(v)$, $\forall v \in V(G^*)$

$$d_G = (deg(v_1), deg(v_2), \ldots, deg(v_n).$$

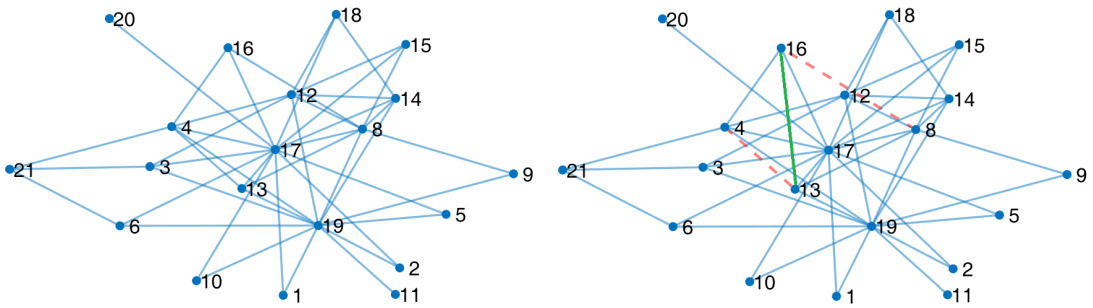
Definition 7 (k-anonymous vector [84]). *A vector of integers $r = (r_1, \ldots, r_n)$ is* k-anonymous, *if every distinct value r_i, $i = 1, \ldots, n$, appears in r at least k times.*

Definition 8 (k-degree anonymous graph [84]). *A graph G^* is* k-degree anonymous *if the degree sequence d_{G^*} is k-anonymous. It means, that for every vertex $v \in V(G^*)$, there exist at least $k-1$ other vertices $v_1, \ldots v_{k-1} \in V(G^*)$ with the same degree, $deg(v) = deg(v_1) = \cdots = deg(v_{k-1})$.*

An example of the 2-degree anonymous graph with its degree sequence is given in *Figure 2.5.* There is a graph G and its 2-degree anonymized version G^*. Two edges are removed, and one edge is added to make G^* 2-degree anonymous. The degree sequences of both graphs are

$$d_G = (14, 14, 8, 7, 6, 5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 1, 1)$$

$$d_{G^*} = (14, 14, 7, 7, 5, 5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 1, 1)$$



a) Original graph G. b) 2-degree anonymous G^*.

Figure 2.5: Graph G and its 2-degree anonymized graph G^*. The red (dashed) edges $(4, 13)$ and $(8, 16)$ were removed, the green (bold) edge $(13, 16)$ was added to make G^* 2-degree anonymous. (Source: author's work.)

Moreover, a systematic framework for graph anonymization is proposed in [84]. Liu and Terzi decomposed the problem of finding k-degree anonymous graph into two parts: *degree anonymization* and *graph construction.*

In the degree anonymization procedure, the degree sequence of G, denoted by d_G, is changed to obtain k-anonymous degree sequence d^*_G. In the graph construction procedure, the original graph G is modified with the sequence of edge editing operations to find the k-degree anonymized graph G^* corresponding to the found k-anonymous degree sequence d^*_G, $d_{G^*} = d^*_G$. They also proposed the k-DA algorithm solving both degree anonymization and graph construction tasks.

The original k-DA algorithm applied only the edge addition operations. However, the idea of using simultaneous edge additions and deletions was also outlined in [84]. Moreover, they proposed two procedures for the degree anonymization:

Dynamic programming algorithm and *Greedy* algorithm. The dynamic programming algorithm is deterministic and finds the optimal solution in $\mathcal{O}(n^2)$, where $|V(G)| = n$. The greedy algorithm is a linear-time alternative with the complexity in $\mathcal{O}(kn)$. Even though the greedy algorithm does not guarantee finding the optimal solution, experiments on real networks proved that the anonymization cost of the found solution was very close to the optimal one. Hence, the greedy algorithm was shown to be faster and more efficient for larger networks.

Lie and Terzi provided several variants of the graph construction problem in [84]. The *ConstructGraph* algorithm requires the anonymized degree sequence d_G^* and ensures G^* with exactly this degree sequence if such a graph exists. Otherwise, it outputs *"No"* if such a graph does not exist. Hence, it decides whether the given d_{G^*} is realizable. However, the *ConstructGraph* algorithm does not guarantee that $E(G) \subset E(G^*)$ or at least that $|E(G) \setminus E(G^*)|$ is negligible. To solve this issue, the *ConstructGraph* was extended to *Supergraph* algorithm in [84]. The *Supergraph* searches for the set of edges $\tilde{E}$, that should be added into G, thus $E(G^*) = E(G) \cup \tilde{E}$. Nevertheless, it is not an oracle like the original construction algorithm. If *Supergraph* does not find proper G^*, it does not mean that the graph does not exist.

Finally, the *SimultenousSwap* algorithm was proposed that solved the relaxed version of the k-degree anonymity problem such that $E(G) \cap E(G^*) \approx E(G)$ instead of $E(G) \subset E(G^*)$. Hence, it enabled to use also the edge switch operation in all procedures. Experimental results showed that the power-law distribution of G^* found by *SimultaneousSwap* was much closer to that from G than the one obtained by algorithms applying only edge additions.

The degree anonymization and graph construction procedures are connected to the probing scheme, as proposed in [84]. The simple outline of the probing scheme using the *Supergraph* construction algorithm is given in *Figure 2.6*.

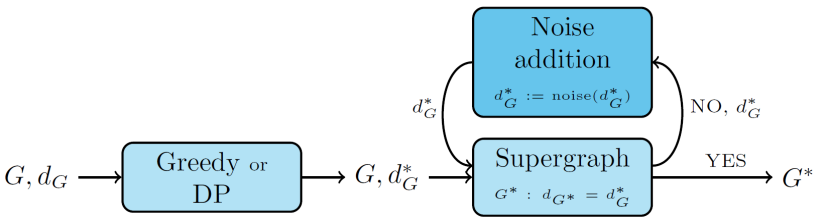


Figure 2.6: Scheme of k-DA algorithm. (Source: author's work.)

Using the probing scheme, the k-DA algorithm performs on G in two steps. At first, *Greedy* or *Dynamic programming* algorithm takes as the input the degree sequence d_G and the anonymization parameter k and finds the k-anonymous degree sequence d_G^*. Then *Supergraph* algorithm tries to modify G to construct G^* such that $d_{G^*} = d_G^*$. In case that the *Supergraph* algorithm fails, d_G^* is slightly modified by adding noise and *Supergraph* runs repeatedly until G^* is found such that $d_{G^*} = d_G^*$. Since the design of the *Greedy* procedure is essential for describing the improvement of the k-DA algorithm proposed in this thesis, the procedure is outlined in *Algorithm 1*. Note that $d_G[i]$ stands for the i-element of d_G.

Later published studies preserved the approach of anonymizing the degree sequence to obtain d_G^* and then modified to graph structure to fit d_G^* [45, 86, 20, 88, 146]. Although the k-DA algorithm was improved in terms of speed [45, 86]

Algorithm 1 Greedy procedure in k-DA [84]

Require: anonymization parameter k, the degree sequence d_G
Ensure: the k-anonymized degree sequence d_G^*

Sort d_G in the descending order.
Let n be the length of d_G.
Set d_G^* to be an empty vector of the length of n.
Set $i := 1$ and $l := k$.
while $l \leq n$ **do**
    $m := \texttt{median}(d_G[i], \ldots, d_G[l])$
    Set $d_G^*[i] := m$, $d_G^*[i+1] := m, \ldots,$ $d_G^*[l] := m$.
    repeat
        Compute $C_{merge} := d_G[i] - d_G[l+1] + \sum_{j=l+2}^{l+k+1}(d_G[l+2] - d_G[j])$.
        Compute $C_{new} := \sum_{j=l+1}^{l+k}(d_G[l+1] - d_G[j])$.
        Set $d_G^*[l] := m$ and $l := l + 1$.
    until $C_{merge} > C_{new}$
    Set $i := l$ and $l := i + k - 1$.
end while
Set $d_G^*[i] := m$, $d_G^*[i+1] := m, \ldots,$ $d_G^*[n] := m$.
Return d_G^*.

or its usability in larger networks [20] by applying different kinds of heuristics, it is still considered to be the basis of the k-degree anonymization. It is one of the state-of-the-art anonymization methods implemented in the evaluation tools SecGraph [60], and ShareSafe [139]. Casas-Roma et al. compared their UMGA algorithm with k-DA in [20]. Zhang et al. referred to k-DA while mentioning the k-degree algorithm in the comparison of several SN anonymization techniques in [158]. Alavi et al. compared their genetic GAGA graph anonymizer with k-DA and other well-known anonymization approaches in [3]. Finally, k-DA was one of the examined anonymization algorithms in [42], where the clique-destroying problem in the network was addressed. Hence, k-DA is still a contemporary algorithm.

The k-degree anonymization method and k-DA algorithm were the subjects of theoretical studies investigating its complexity [45, 46]. Moreover, it was presented in [46] that high-degree nodes increased the anonymization cost significantly. The idea was expressed that the high-degree nodes could be potentially removed from the anonymization process. As far as I know, this idea was not further investigated.

Lu et al. introduced in [86] *Fast k-degree anonymization* algorithm (FkDA), which clustered and anonymized the vertices of $V(G)$ into several anonymization groups with at least k members. The original graph was modified, so vertices in each group have the same degree. To reduce the anonymization cost, the vertices in each group should have similar degrees in the original graph. For this reason, FkDA clustered vertices in descending order according to their degree in the original graph, which also corresponded with the scale-free property of real SNs.

In the degree anonymization procedure, the edges were added simultaneously to all vertices. Inside one group, FkDA linked vertices with insufficient degrees

with each other. However, the new degree value should still be lower than the highest degree in the group. After the edge addition, vertices in the group were reordered according to the new degrees, and other vertices outside the group could be added to it as well. The design of the procedure is greedy, and its implementation has been made to minimize the need for reordering the groups. Furthermore, there is a relaxed edge addition procedure that was launched where the deterministic procedure failed to find enough vertices for adding more edges, and the output graph is still not k-degree anonymous. In summary, the advantage of FkDA over the original k-DA was that testing the realizability of the degree sequence and repeating runs of graph construction procedure, which were time-consuming operations, were avoided; hence FkDA was faster on larger datasets.

Casas-Roma et al. introduced the k-degree anonymization algorithm based on univariate micro-aggregation to anonymize large SN datasets in [20]. They preserved the two-step approach and split the task into the problem of degree sequence anonymization and graph modification. The aim of their degree anonymization method was to modify the values of d_G to create sets of k or more elements. The method used the optimal univariate micro-aggregation [43] to achieve the optimal set distribution. Then, it computes the values for each set that minimizes the distance of d_G^* from d_G. The greedy approach was also introduced to reduce the search complexity. In the graph modification algorithm, the neighbourhood centrality measure was used to quantify the edge relevance in the network and select auxiliary edges in the edge editing operations (addition, removal and switch). I used the modified version of their UMGA algorithm for anonymizing social ties in the geosocial networks in [94]. Instead of using neighbourhood centrality in the graph modification procedure, I exploited the location information stored in the geosocial network and quantified the importance of edges with the location entropy metric.

The k-degree anonymity is the method with the limited model of the attacker. The k-degree anonymized dataset preserves privacy against the attacker, knowing the degree of their target node. If the attacker has larger background knowledge about the graph structure of the original network, then the individual's privacy is not ensured. To address this issue, other methods based on k-anonymity were proposed to provide more privacy protection after anonymization.

To protect the anonymized network against the attacker who knows the target's neighbourhood in the original networks, the k-neighbourhood anonymity method was introduced in [161, 162]. The proposed approach was extended to $k(d)$-neighbourhood anonymity in [3, 101]. The $k(d)$-neighbourhood anonymity protects against the attacker with the structural background knowledge about nodes in the distances up to d from the target node. Before giving the proper definition of k-neighbourhood and $k(d)$-neighbourhood anonymity, the terms of the neighbourhood of the vertex and d-neighbourhood of the vertex are formally defined.

Definition 9 (Neighbourhood)**.** *The* neighbourhood *of the vertex $u \in V(G)$ in G, denoted by $N_G(u)$, is a subgraph of G such that*

- $V(N_G(u)) = \{v \in V(G)\,;\, (u,v) \in E(G)\}$
- $E(N_G(u)) \subseteq E(G)$ *such that* $\forall (u,v) \in E(N_G(u)) :\ u, v \in V(N_G(u))$

Definition 10 (d-neighbourhood)**.** *The* d-neighbourhood *of the vertex* $u \in V(G)$ *in* G*, denoted by* $N_G^d(u)$*, is a subgraph of* G *such that*

- $V(N_G^d(u)) = \{v \in V(G)\,;$ *there is a path of the lenght* d *from* u *to* v *in* $G\}$
- $E(N_G^d(u)) \subseteq E(G)$ *such that* $\forall (u,v) \in E(N_G^d(u)) : u, v \in V(N_G^d(u))$

Definition 11 (k-neighbourhood anonymous graph [161])**.** *A vertex* $u \in V(G^*)$ *is* k-neighbourhood anonymous *in* G^* *if there are at least* $k-1$ *other vertices* $v_1, \ldots, v_{k-1} \in V(G^*)$ *such that* $N_G(u), N_G(v_1), \ldots, N_G(v_{k-1})$ *are isomorphic to each other. The graph* G^* *is* k*-neighbourhood anonymous if* $\forall u \in V(G^*)$*,* u *is* k*-neighbourhood anonymous.*

Definition 12 ($k(d)$-neighbourhood anonymous graph [3])**.** *A vertex* $u \in V(G^*)$ *is* $k(d)$-neighbourhood anonymous *in* G^* *if there are at least* $k-1$ *other vertices* $v_1, \ldots, v_{k-1} \in V(G^*)$ *such that* $N_G^d(u), N_G^d(v_1), \ldots, N_G^d(v_{k-1})$ *are isomorphic to each other. The graph* G^* *is* $k(d)$*-neighbourhood anonymous if* $\forall u \in V(G^*)$*,* u *is* $k(d)$*-neighbourhood anonymous.*

Clearly, the values of both parameters k, d are set before anonymization and can not be changed afterwards. The value of k indicates the probability that an individual can be re-identified in G^*, the value of d corresponds to the size of the attacker's knowledge that is not sufficient for the re-identification attack. However, the size of the background knowledge can change over time. If the attacker obtains the structure of G including $N_G^{d+1}(v)$, G^* becomes vulnerable against his or her structural attack.

To address this privacy-preserving issue, there were published anonymization methods that protected against any structural attack: k-isomorphism [25], k-symmetry [150] and k-automorphism [163]. The protection against any structure attack in terms of those methods means that for any subgraph of G^*, at least other $k-1$ disjoint subgraphs are isomorphic to it. In other words, even if the attacker knows the structure of an arbitrarily large subgraph of G or the structure of several subgraphs of G, then he or she is not able to recognize his or her target user in the anonymized data G^*. Of course, suppose the attacker gains sufficiently extensive knowledge about its target from the structure of the original graph G. In that case, he or she does not need to gain more information about the target user from the anonymized G^*. However, the leakage of non-anonymized data is not the subject of this thesis.

The k-isomorphism, k-symmetry and k-automorphism are very similar approaches. Cheng et al. proved in [25] that k-isomorphic graph is k-automorphic as well. The hypothesis that k-automorphic graph is k-isomorphic and that both approaches are equivalent to k-symmetry has not been rigorously proved.

In [25] Cheng et al. addressed not only the problem of identity disclosure but also the privacy issue of link disclosure which was addressed neither in [163] nor [150]. They introduced the k-isomorphism anonymization method and proved its persistence against identity and link disclosure privacy issues. Furthermore, they proposed a dynamic release mechanism that addresses the privacy-preserving issue caused by repeated releases of the same database over time. The k-isomorphic graph is defined after the formal definition of graph isomorphism.

Definition 13 (Isomorphic graphs [163]). *Given two graphs P_1 and P_2, P_1 is* isomorphic to P_2 *(denoted by $P_1 \backsimeq P_2$), if and only if there exists at least one bijective function $F: V(P_1) \longrightarrow V(P_2)$ such that for any edge $(u, v) \in E(P_1)$, there is an edge $(F(u), F(v)) \in E(P_2)$. The function F is called the* isomorphism *of P_1 and P_2.*

Definition 14 (k-isomorphic graph [25]). *A graph G^* is k-isomorphic, if G^* consists of k disjoint subgraphs $P_1, \ldots, P_k$, where P_i and P_j are isomorphic, $i \neq j$, $\forall i, j = 1, \ldots, k$.*

The k-isomorphism algorithm proposed in [25] by Cheng et al. partitions the input graph into k disjoint subgraphs and makes them isomorphic to each other with edge additions and deletions. The crucial part is to identify those k subgraphs. They proposed a heuristic procedure where subgraphs P_i, such that $|E(P_i)| \leq m$, were selected as potential anonymization subgraphs. The threshold m was experimentally set as the average degree in G. Afterwards, at least k disjoint matches of the selected P_i's are found in G. Those matches are removed from G, added into G^*, and the procedure is repeated. If there are not enough matches of the particular embedding, G is modified with edge additions and deletions. After the whole G is processed, the disjoint components of G^* are linked together such that the final G^* is k-isomorphic.

The k-isomorphism approach was improved with the method better preserving communities in the graph in [120]. Rong et al. proposed the k^+-isomorphism method detected communities at first, then partitioned them into n similar subgraph clusters. Afterwards, in each cluster, the subgraphs were modified so that at least k communities are isomorphic to each other. They applied both edge addition and deletion operations as well as vertex addition and deletion operations.

Wu et al. proposed the k-symmetry method in [150] where the vertices' property "being automorphic" is viewed as the equivalence relation, called *automorphism equivalence*, on the vertex set $V(G)$. The vertex set $V(G)$ is split into vertex partitions induced by the automorphism equivalence. The graph G^* is said to be k-symmetric if every vertex partition contains at least k vertices. The basic idea of the k-symmetry anonymization algorithm is repeating the procedure of making duplicate copies of the vertex partitions until the size of each vertex partition combined with its copies is equal to or larger than k. In [150], Wu et al. also presented the extension of the algorithm that improved the algorithm in terms of preserving data utility. Since the formal definition of k-symmetry requires formally defining several other terms not used in the rest of the thesis, the definition is omitted with reference to [150].

The k-automorphism approach introduced in [163] is defined with the automorphisms on $V(G^*)$.

Definition 15 (Graph automorphism). *Let G be a graph, and the bijective function $F: V(G) \longrightarrow V(G)$ be the isomorphism. Then F is the* automorphism *on G.*

Definition 16 (k-automorphic graph [163]). *Let G^* be a graph. If there exist at least $k-1$ automorphisms F_j, $j = 1, \ldots, k-1$ in G^*, and*

$$\forall v \in V(G^*): \; F_{j_1}(v) \neq F_{j_2}(v) \qquad \forall j_1, j_2 \in \mathcal{N} : 1 \leq j_1 < j_2 \leq k-1,$$

then G^* is called a k-automorphic graph.

As mentioned before, each k-isomorphic graph is k-automorphic as well. The k-automorphism KM anonymization algorithm, proposed by Zou et al. in [163], is constructed on similar principles as the k-isomorphism algorithm introduced in [25]. Furthermore, it is also proven to guarantee privacy under dynamic releases. The problem of finding the k-automorphism graph is divided into two subtasks: finding the optimal graph partitioning and finding the optimal block alignment. Both subtasks are proved to be NP-hard in [163] and are solved by heuristic procedures.

Since one algorithm proposed in this thesis is based on the KM algorithm, its brief description is given in *Algorithm 2*. I refer to [163] for its detailed description. The KM algorithm transforms the input graph G to the k-automorphism graph G^*. It solves the task as two separate NP-hard problems: finding the optimal graph partitioning (lines 2-6 in *Algorithm 2*), finding the optimal graph alignment (lines 7-9). Both problems are correctly defined in *Chapter 7*.

Algorithm 2 KM algorithm [163]

Require: anonymization parameter k, input network G
Ensure: k-automorphism network G^*

1: Set $i = 1$ and C to be an empty set.
2: Find the frequent subgraph $g_f(k)$ in G with the minimal support equals to k. Denote each match of the frequent subgraph $g_f(k)$ in G as P_{ij}, $j = 1 \dots k$. Let U_i be the set of all subgraphs P_{ij}.
3: Expand subgraphs P_{ij} with some other vertices and edges from G such that the anonymization cost of the whole block U_i is minimal.
4: Remove all subgraphs P_{ij} in U_i from G: $G = G \setminus U_i$.
5: Store all crossing edges into the set C. Crossing edges are edges (v, w) from G such that $v \in P_{ij}$ and $w \notin P_{ij}$ (for some j).
6: Set $i = i + 1$.
7: Repeat steps 2-7 until there is no edge in G. After that there are m blocks $U_1, \dots U_m$ such that for each $i = 1 \dots, m$ the block U_i contains k subgraphs $P_{i1}, \dots, P_{ik}$.
8: For each set U_i: perform graph alignment procedure on all graphs P_{ij} to obtain k isomorphic graphs P'_{ij}. Denote F_a to be the isomorphism between $P'_{i,a}$ and $P'_{i,a+1}$, $a = 1, \dots, k-1$.
9: Replace each block P_{ij} by its alignment block P'_{ij} to obtain the anonymized network G^*. Remove all crossing edges from G^*.
10: For each crossing edge (v, w) from the set C: add (v, w) and $k-1$ other edges $(F_j(v), F_j(w))$ into G^*.
11: Return G^*.

At first, the algorithm starts with finding the frequent subgraph with the given minimal support by running the grow-and-store SiGraM algorithm [72]. Then, for fixed i, the KM algorithm first finds k matches of the frequent subgraph $g_f(k)$ and denotes them P_{ij} (line 2). The subgraphs P_{ij} are isomorphic to each other. After that, the subgraphs P_{ij} are expanded, which means that some neighbourhood vertices and edges are added to each P_{ij} (line 3). The expansion is necessary for decreasing the total anonymization cost. The expanded blocks P_{ij} have fewer crossing edges than the original frequent subgraphs; hence the anonymization

cost caused by the crossing procedure is smaller. Thus, the subgraphs P_{ij} are larger but are not isomorphic to each other anymore. However, since the aim of the algorithm is to obtain k-automorphic graph G^*, its subgraphs have to be isomorphic. Thus, dummy edges are added to make the expanded subgraphs P_{ij} isomorphic to each other again (line 8). Subgraphs P_{ij} are removed from G, i is increased, new frequent subgraphs in G are found and the whole process is repeated until G is the empty graph. Finally, all disjoint expanded isomorphic subgraphs P_{ij}, $\forall i, j$, are isomorphically reconnected with the crossing edges and their copies (line 10).

Neither Cheng et al. nor Wu et al. did not compare their experimental results with the results of other known k-anonymity algorithms as Zou et al. did. Zou et al. compared their results with the results of k-DA algorithm [84], k-candidate anonymity approach [48] and k-neighbourhood method [161]. Since the KM algorithm provides higher privacy protection than other algorithms, the k-neighbourhood approach and k-DA were shown to preserve the data utility better than KM in [161].

2.3.9 Surveys

A summary of the surveys focusing on the SN anonymization approaches is provided to complete the literature review of SN anonymization methods. The state-of-the-art anonymization techniques and approaches were summarized in comprehensive surveys [34, 1, 21, 61, 91].

In [34], Fung et al. evaluated various approaches to privacy-preserving data publishing and studied the challenges in practical data releasing. They clarified the characteristics and requirements that distinguished those privacy-preserving issues in data publishing from other related problems.

Recent social network data publishing developments were reviewed in [1]. Abawajy et al. presented state-of-the-art privacy-preserving approaches for publishing social network data and summarised privacy attacks on anonymized social network datasets. The survey presented by Casas-Roma et al. in [21] is focused only on graph modification methods altering the graph structure of the input network.

Both anonymization and deanonymization techniques were classified in [61]. Existing anonymization methods were classified into six categories with respect to graph and application utility metrics. Ji et al. examined the performance of known deanonymization attacks with respect to scalability, practicability and robustness. Moreover, they also analyzed the resistance of the anonymization methods against the attacks.

Recently, Majeed and Lee presented in [91] the comprehensive survey of all known anonymization techniques for privacy-preserving data publishing including methods for anonymization relational datasets, SN anonymization methods, anonymization approach used for application-oriented scenarios as well as the review of evaluation metrics for anonymization algorithms. They also pointed out the challenges for future research in both fields of anonymization.

2.4 Deanonymization attacks

Deanonymization attacks are attacks against published anonymized datasets. This thesis focuses on attacks and deanonymization approaches aiming to re-identify users in the anonymized data. It means that the aim of the adversary is to link the target individual I with the node $v(I) \in V(G^*)$ that represents the individual in the anonymized network G^*. This privacy-preserving issue is called the *identity disclosure problem*. There are also techniques causing other kinds of privacy leakage, for example leaking the link privacy; however, there are not involved in the review since the focus of the thesis is the identity disclosure problem.

The deanonymization attack can be active, passive, or a combination of both types. The assumption for active attacks is that the attacker can modify the target SN before the provider publishes it. For instance, the attacker can create many fake accounts (Sybil accounts) and link them with each other and with the target user. When the SN is published, he or she attempts to recognize the embedded part of the SN, and the recognition simplifies the re-identification of the target users. However, providers nowadays possess various Sybil defence techniques and tools for detecting Sybil accounts that can recognize and delete fake accounts, making this type of attack difficult [60].

On the other hand, passive attacks do not require access to the network before publishing. Furthermore, they can aim at a large part of SN, and the same procedure can be repeated, seeking different SN datasets. Before the actual attack on the target SN dataset, the attacker is assumed to gain some information that enables him or her to perform the attack. Such information is called the attacker's *background knowledge*. The format of the background knowledge depends on the kind of attack. In structural attacks, the background knowledge is usually a graph, for example, the subgraph of G. In this case, the background knowledge is traditionally called an *auxiliary graph*.

Assume the provider possesses the graph G representing the SN network. The provider desires to share or publish the data. The version of G shared or published is called the *released graph* $\tilde{G}$. The released graph $\tilde{G}$ equals G or any anonymized version of G depending on the privacy level guaranteed by the provider. The identity disclosure occurs if an attacker can identify the target individual in the released dataset $\tilde{G}$. In other words, the identity is disclosed if the attacker can link $v \in V(\tilde{G})$ with the particular individual represented with v.

Definition 17 (Query [163])**.** *Given a social network G, a query $\mathcal{Q}$ represents any information the attacker can exploit to extract private information from G. The result of $\mathcal{Q}$ is a set of vertices $V' \subset V(G)$. Each $v \in V'$ is called a match vertex.*

Definition 18 (Structural attack [163])**.** *Given a released network $\tilde{G}$, if a query $\mathcal{Q}$ over $\tilde{G}$ launched by an attacker has a limited number of match vertices in $\tilde{G}$, then target individual t might be uniquely identified. The attack is called structural if $\mathcal{Q}$ is based on the structural information about t in $\tilde{G}$.*

Structural attacks include degree attacks, subgraph attacks, neighbour-graph attacks and hub fingerprint attacks [163]. Many sophisticated structural attacks on SN datasets have been published. In [6], Backstrom et al. introduced the

model of adversary that could deduce the users' identity by solving a set of restricted graph isomorphism tasks. Thus, it was proved that the naive anonymization of only removing identifiers and attributes set $U(G)$ was insufficient for preserving users' privacy in the SN dataset. Their work was extended in [6], where the family of attacks on a single anonymized dataset was presented. It involved both passive and active attacks, in which the adversary could find whether edges existed between pairs of targeted nodes.

Narayanan and Shmatikov presented a structural seed-based deanonymization scheme [104], where the attacker exploited the knowledge of a small part of the original network called *seed* and background knowledge about the anonymized dataset. They also showed how the adversary could gain such a structural background knowledge which was represented as the auxiliary graph. The simplified version of their attack was presented in [105].

The divide-and-conquer algorithm based on attacking the network at first on a community level is introduced in [107]. Nilizadeh et al. performed that the community-based approach improved the seed-based deanonymization techniques. Peng et al. presented the seed-growing algorithm in [111]. The identified seed subgraph grew larger based on the attacker's background knowledge. Similar attacks were proposed in [68, 154], where the seed subgraph grew iteratively, mapping a pair of nodes according to the number of their neighbour nodes.

The seed-free deanonymization attack was introduced in [109]. Based on Bayesian probability, the seed-free attack matched two nodes, one from the attacked network and the other from the auxiliary graph. Instead of exploiting the side information like the seed, node features such as degree and distances were used as nodes' fingerprints. The algorithm ran in rounds where the most likely pairs were mapped first. Afterwards, these paired nodes subsequently generated additional features in the fingerprints of other nodes.

Semantic-based deanonymization attacks are proposed in [149, 115]. Wondracek et al. exploited the group member information stored in social networks in [149]. They performed that group memberships could significantly increase the probability of the correct re-identification of the target user. The web browser history stealing attacks were used to determine the group membership. Quin et al. set up knowledge graphs to increase identity disclosure risk in anonymized networks in [115].

Recently, Ma et al. introduced the random forest classifier in [87] that used the spectral partition method to partition large graphs into several small subgraphs. Its output matched candidate nodes from the anonymized network with the ones from the auxiliary network. A framework examining the interplay between graph properties and the vulnerability to deanonymization attacks is proposed in [55]. In [24], Chen et al. proposed three heuristic attack strategies: the community detection attack, the degree-based attack and the modularity Q-Attack based on a genetic algorithm. They showed that Q-Attack achieved better results than the other two strategies in reducing normalized mutual information.

The deanonymization approaches mentioned above focused on issues connected to sharing or publishing data from a single social network. They examined a single anonymized dataset and took advantage of some background knowledge about the single anonymized dataset. The background knowledge of the attacker in the form of an auxiliary graph can be a subgraph of the attacked graph or any

external information gained by the attacker to perform the attack. The external information could be any data, including the graph of another anonymized dataset [104]. However, as far as I know, no research has been yet focused on the particular situation where the background knowledge is only another anonymized dataset with the overlapping user community. In this scenario, both anonymized datasets are threatened since the first can be the auxiliary graph for the latter and vice versa.

On the other hand, there were published studies that address the problem of privacy leakage involved in publishing two or more relational datasets with overlapping records [35, 81, 127]. This attack combing the knowledge of two anonymized datasets is called the *composition attack*.

In [35], Ganta et al. investigated an intersection attack, a version of the composition attack, and experimentally demonstrated the attack's severity for many previously proposed anonymization techniques. Their study proved that several partition-based anonymization schemes, including the k-anonymity and its variants, are vulnerable to composition attacks.

Sattar et al. proposed a probabilistic (d, α)-linkability method, an efficient extension of k-anonymization, in [127]. They presented the theoretical background for reducing the risk of a composition attack on relational datasets. They experimentally proved that the methods based on k-anonymization protect data privacy adequately and can preserve more utility than alternative privacy models.

Moreover, Li et al. introduced a hybrid algorithm to protect the privacy of relational datasets against composition attacks in [81]. They proposed an anonymization algorithm that reduced the risk of composition attacks. They considered an adversary who knew some personal information about the victim, while the victim's records are included in two anonymized datasets independently published. The attacker exploits the occurrence of the plain value of a sensitive attribute in anonymized relational datasets. The considered datasets had the particular property that the value of the sensitive attribute was constant for each individual over all the anonymized datasets.

2.5 Evaluation tools

Different anonymization approaches provide different levels of security. A high level of security protects against the attacker with more extensive background knowledge. However, a higher level of anonymization usually requires more modifications in the input dataset, causing larger information loss in anonymized data. Maintaining a balance between privacy and utility loss was addressed in [108, 146].

Evaluating anonymization methods is a challenging issue as well. Anonymization algorithms are usually implemented and tested on synthetic or real-world datasets. Then, the selected network metrics are measured in the anonymized datasets. Experimental results published in different research papers are usually not comparable. The problem of missing methodology for evaluating other anonymization methods is addressed in [60, 19].

Ji et al. proposed an evaluation tool called SecGraph in [60]. It can be used by different researchers to analyze the performance of their anonymization algorithms, evaluate anonymized data concerning utility and application metrics,

anonymize social network datasets or examine the vulnerability of anonymized data to state-of-the-art deanonymization attacks. Three modules are implemented in SecGraph: anonymization, utility, and deanonymization.

In the anonymization module, graph data anonymization schemes are implemented. The module can be used to anonymize raw graph data. The utility module can evaluate anonymized data utility concerning utility and application metrics. Therefore, it can determine how an anonymization algorithm preserves data utility. In the deanonymization module, data security can be evaluated with real-world deanonymization algorithms before publishing or sharing. The effectiveness of an anonymization algorithm can also be examined in this module. Researchers can test whether the anonymized data of an anonymization algorithm is resistant to attacks. SecGraph has been recently improved by adding two other modules, recommendation and security quantification. The second version of the tool is called ShareSafe [139]. Both tools are available online [59].

A different privacy evaluation framework for graph anonymization is proposed in [19]. The DUEF-GA framework includes generic and task-specific information loss measures as well as metrics for the examination of re-identification and risk assessment. It was designed to help researchers and experts to select the best parametrization or algorithm to reduce information loss and maximize data utility. The tool is available online [18]. DUEF-GA was published after the publication of SecGraph; however, the existence of the SecGraph tool was not mentioned in [19].

SecGraph and DUEF-GA tools commonly focus on evaluating and measuring data utility in anonymized SN datasets. However, their approach differs in many aspects. DUEF-GA is focused mainly on data utility and information loss measurement. To evaluate the vulnerability against deanonymization attacks, it computes only the candidate sets for each vertex to measure the feasibility of degree-based attacks and computes proportions of vertices that change their set of neighbours at a distance one to measure the feasibility of 1-neighbourhood-based attacks. DUEF-GA implementation contains no anonymization algorithm that can do the actual anonymization of the SN dataset. To produce anonymized data for testing in [19], Casas-Roma employed the UMGA anonymization algorithm [20]. On the other hand, as mentioned above, SecGraph is focused equally on evaluating preserving data utility and vulnerability against state-of-the-art deanonymization techniques.

Focusing on the data utility measurement, both tools provided the measurement of some structural metrics and some application metrics. Furthermore, both tools require the same input: the original graph G and its anonymized version G^*, where $V(G) = V(G^*)$. Even though some metrics are measured with both tools, the methodology is entirely different. For instance, both tools deal with the shortest path and the clustering coefficient[3]. DUEF-GA compute the average shortest path and global clustering coefficient for whole graphs G, G^* and then outputs the difference of the measured values. SecGraph calculates the global clustering coefficient for complete graphs G, G^* and outputs the ratio of the measured values. Additionally, it computes the distribution of the shortest path and local clustering coefficient in G, G^* and outputs its cosine similarity.

[3]Metrics mentioned in this section as well as cosine and Jaccard similarities are properly defined in *Chapter 4*

Focusing on vertex-level structural metrics, both tools compute betweenness and closeness centrality for each vertex. SecGraph again computes the output value with the cosine similarity of vectors of vertex values, DUEF-GA combines the values with root mean square[4]. Eigenvalues of the adjacency matrices of graphs are also handled in both tools. SecGraph computes the eigenvalue vectors, while DUEF-GA measures the largest eigenvalues.

The approaches differ a lot in the measurement of the application metrics. Page rank is the only application metric computed in both tools, except that both aim to detect communities and compute the information flow. The author of DUEF-GA defined the precision index metrics and combined them with four clustering algorithms to detect communities. He introduced his own metric for measuring the information flow. On the other hand, SecGraph computes the authorities and hubs scores and uses the method published in [152] to detect communities. The information flow is given by infectiousness, influence maximization, secure routing and the computation of the minimum-sized influential node set [60].

Except for the mentioned, in SecGraph there are additionally implemented four other structural metrics (effective diameter, network constraint, degree and joint degree distribution) and three additional application metrics (role extraction, reliable email detection and Sybil account detection) [60]. On the other hand, structural metrics that are implemented only in DUEF-GA are transitivity, and edge intersection [19].

Both tools were tested on real-world SN datasets; however, the datasets used for evaluating differ in size. Authors of SecGraph tested the tool on large SN datasets up to 0.8 million users, while DUEF-GA was tested on smaller graphs up to 2000 users. In this thesis, the SecGraph tool is used for evaluating the efficiency of the proposed anonymization method since it also focuses on evaluating the vulnerability to attacks and includes the implementation of state-of-the-art deanonymization algorithms.

2.6 Genetic algorithms in social network analysis

Genetic algorithms based on mimicking the processes of natural evolution are powerful tools while addressing NP-hard problems. They were developed by Holland in [54]. They have been successfully applied to various issues belonging to the NP class like the travelling salesman problem [41, 56], scheduling problems [28], or bin packing problem [136]. Many studies describe the application of genetic algorithms on a wide range of NP-hard problems. However, since the thesis is focused on social network anonymization, the review contains only studies dealing with the application of genetic algorithms in social network analysis.

Genetic algorithms have been recently exploited in the SN analysis for community detection in large networks [5, 112], graph clustering [11, 16] and predicting the dynamics of SN [22]. Since the problem of modifying the SN graph with the minimal amount of edge editing operations to become k-anonymous is NP-hard

[4]For the definition of root mean square see [19].

[45], several SN anonymization methods based on genetic algorithms have been recently presented too.

Sihag proposed the genetic algorithm anonymizing SN by clustering nodes into super-nodes in [130]. However, the algorithm was tested only on small SNs (up to 67 nodes) and caused more significant information loss than the previously published deterministic clustering algorithm Sangreea [17].

Another genetic clustering algorithm was proposed by Yazdanjue et al. in [155]. They optimized the clustering procedure in the k-anonymity method employing particle swarm optimization. They presented a hybrid solution that combined particle swarm optimization with genetic algorithms. Their solutions were represented with binary matrices describing which node belonged to which super-node. Each chromosome contained all nodes and all super-nodes of the whole graph. Hence, while anonymizing large networks, it can be used only large values of the anonymizing parameter k since larger k means fewer super-nodes and smaller chromosomes.

The genetic k-degree edge modification was introduced by Rajabzadeh et al. in [116]. At first, the algorithm detected communities in the SN graph and then modified each community's edge set with the genetic algorithm. Hence, the SN graph was anonymized by adding edges between vertices inside detected communities. The algorithm was tested on real datasets up to 23,133 nodes. However, the anonymized SN satisfied only k-degree anonymity.

Alavi et al. introduced the $k(d)$-neighbourhood anonymity approach and presented GA for graph anonymization called GAGA. The GAGA was the edge editing algorithm that prioritized edge switching over edge adding or removing. They showed that the genetic algorithm was an efficient tool for anonymizing large SNs. Using the SecGraph framework [60] GAGA algorithm proved resistant against five deanonymization attacks. Moreover, the SecGraph was used to measure information loss after anonymization and compare GAGA with existing approaches. I agree that evaluating anonymization algorithms with an independent tool like SecGraph can lead to a better comparison of proposed algorithms. However, GAGA is tested only on the subgraph of the DBLP co-authorship network [153], not on the whole DBLP dataset. Since other researchers can not find which subgraph they used, they can not compare their results with those published in [3].

3. Research questions and objectives

My research in SN anonymization was motivated by the growing need for privacy protection of personal information in published data from social networks. Anonymization of social networks is still a promising field of study and gives researchers the opportunity to find open problems. The most important open problem is indeed finding a robust SN anonymization method that preserves privacy such that the anonymized dataset is not vulnerable against the state-of-art deanonymization attacks and keeps enough data utility such that the anonymized dataset is still useful for data analysts [61].

Finding the universal anonymization method for SN datasets would be a too ambitious goal for this thesis. I started my research by studying the state-of-the-art techniques of SN anonymization and developing the comprehensive literature review given in *Chapter 2*. In this early phase of my research, I identified three nontrivial open problems, the solutions of which are achievable in PhD research and contribute to the progress in SN anonymization.

In this section, I present the identified problems, formulate the research questions and formally define the objectives of this thesis. Furthermore, I outline the proposed solution for each stated problem. The complete solutions are described in detail in the rest of the thesis.

3.1 Study and research questions

My PhD research mainly focuses on anonymization methods and approaches to modifying social network datasets. However, since anonymization was initially developed as the solution to privacy-preserving issues in relational datasets, my research in anonymization techniques started by studying anonymization and deanonymization approaches in tabular data. As stated in *Section 2.4*, several versions of composition attacks on relational datasets were published. The composition attack benefits from combining records from two anonymized datasets that are published independently of each other. The scenario is as follows.

Let R_1, R_2 be two relational datasets. Each dataset contains information about a group of individuals. Let $I(R_1)$ and $I(R_2)$ be the set of individuals whose data are included in R_1 and R_2, respectively. Let R_1^* and R_2^* be the anonymized versions of R_1 and R_2, respectively. Assume that R_1^* and R_2^* were released by different providers. The data provider publishing R_1^* does not have the knowledge about releasing R_2^* such that $I(R_1) \cap I(R_2) \neq \emptyset$. Since generalization, suppression, or perturbation do not change the number of rows in datasets, it holds that $I(R_i^*) = I(R_i)$, $i = 1, 2$. Hence, there may exist two published anonymized datasets R_1^* and R_2^* and the non-empty set of individuals $I(R_1^*, R_2^*)$ such that $I(R_1^*, R_2^*) = I(R_1^*) \cap I(R_2^*)$. In other words, there exist some individuals whose records are included in the two anonymized databases. This situation significantly decreases the level of security provided by the applied anonymization methods. Even if R_1 and R_2 were anonymized well and R_1^* and R_2^* provide high privacy protection, combining anonymized records from R_1^* and R_2^* can lead to

re-identification of users in $I(R_1^*, R_2^*)$ [81].

This problem has not been widely studied in social network datasets. Assume social network datasets $G_{A,1}, G_{A,2}$ such that $V(G_{A,1}) \cap V(G_{A,2}) \neq \emptyset$ and their anonymized versions $G^*_{A,1}, G^*_{A,2}$. As there are several social networks and more than half of Internet users participate in more than one social network [144, 148], an individual is likely to appear in more than one anonymized social network dataset. Thus, two anonymized social network datasets published independently of each other will likely have overlapping user communities.

Assume an attacker that wants to re-identify the user $v(I) \in V(G_{A,1})$ in the released $G^*_{A,1}$. The background knowledge of the attacker in the form of an auxiliary graph $Aux(G^*_{A,1})$ can be a subgraph of $G_{A,1}$ or any external information gained by the attacker which includes $G^*_{A,2}$ or its subgraphs [104].

However, as far as I know, no research has been yet focused on the particular situation where $Aux(G^*_{A,1}) = G^*_{A,2}$. When two anonymized datasets of social networks having overlapping user communities are published independently of each other, similarly to relational datasets, they can be vulnerable to a composition attack. Formally, the goal is to answer the following research question:

Question 1. What modifications should be done to the design of the composition attack such that the attack becomes applicable to the social network data?

To solve this issue, I study the published composition attacks on relational datasets [35, 81, 127], analyze in detail their approaches and apply them to SN datasets. I indicate the privacy risk associated with the situation and describe the attacker's goals and motivations for realizing the attack. Moreover, I present and implement a new composition attack algorithm. The algorithm takes as input two graphs representing the anonymized social network data and finds pairs of vertices, one from each graph, describing the same individual participating in both social networks. Its feasibility is experimentally proved by testing it on synthetic scale-free networks generated using the Barabasi-Albert (B-A) model. The detailed description of the proposed approach, the composition attack, and the experimental results are given in *Chapter 5*.

While studying SN anonymization methods, it is crucial to know the k-DA algorithm proposed in [84]. The k-DA algorithm is considered to be the basis of the k-degree anonymization. Moreover, it is still a contemporary algorithm, as shown in *Section 2.3.8*.

Except for the k-DA algorithm, I have also studied its improved versions [86, 20] and its complexity [45, 46]. As mentioned in *Section 2.3.8*, Hartung et al. presented in [46] that high-degree nodes significantly increased the anonymization cost of k-DA. Furthermore, they proposed removing the high-degree nodes from the anonymization process since they might be well-known anyway. It did not mean the high-degree nodes should be removed from the graph during the anonymization procedure. The high-degree nodes would be kept in the graph, but they would not be anonymized. In other words, the few high-degree values would remain untouched by the anonymization procedure.

However, the assumption that high-degree nodes represent well-known individuals is too strong in the case that we focus on developing an anonymization

method that is independent of the dataset itself. Nevertheless, the idea motivated me to study the degree anonymization procedure of the k-DA algorithm more closely.

In the degree anonymization procedure of k-DA, the degree sequence is ordered decreasingly, and degree values are split according to merging rules into several groups of at least k elements. In each group, all elements are set to have the same anonymized value. In the original k-DA algorithm, the anonymized value is the median of the k values in the group. However, the median can be distant from the highest values in the group. The problem is the most significant in the groups where high-degree nodes are put together with lower-degree nodes. If the median value belongs to a lower-degree node, setting the higher values to the median causes a significant decrement in degree. In that case, many nodes linked to the high-degree nodes in G cannot be connected to them in G^*. Hence, the d^*_G is often unrealizable since there are not enough potential neighbours for all those nodes.

The noise addition strategy significantly impacts the speed and efficiency of the whole k-DA algorithm. It corrects d^*_G so that it becomes realizable and helps keep data utility in the anonymized graph. I propose the improvement of the noise addition procedure such that the anonymized degree value of high-degree nodes is kept closer to their original values. The noise addition procedure corrects the anonymized value only in those groups of d^*_G where the difference between the highest and the lowest original value is significant. Moreover, the correction is not constant for all groups, but it also depends on the values in the group. It could decrease the probability that *Supergraph* fails to find G^*. Thus, it could positively affect the efficiency of the whole k-DA algorithm. I formulate the hypothesis into the following research question:

Question 2. What effect does the correction of the anonymized values in groups of d^*_G have on the efficiency of the k-DA algorithm?

To solve this issue, I propose the heuristic improvement of the noise addition procedure of k-DA. The proposed noise addition approach considers the power-law distribution of real social networks. The proposed procedure is shown to better handle the anonymization of high-degree nodes by flexibly correcting the anonymized values in their groups. Moreover, I implement the procedure in the greedy version of the k-DA algorithm. To show the applicability and efficiency of the improved k-DA algorithm, I run experiments on a set of different real-world social networks. The detailed description of the heuristic high-degree noise addition method and its usability, information loss and data utility analysis is given in *Chapter 6*.

Since the model of the attacker is limited in the k-degree method and the method was proven to be vulnerable against deanonymization attacks [60], I also studied methods providing privacy protection against any structural attack. I focused mainly on k-automorphism since its authors compare their results with the results of k-DA.

The k-automorphism approach was proposed by Zou et al. in [163], where the k-automorphism anonymization algorithm called KM algorithm was also designed. Zou et al. proved that the proposed k-automorphism KM algorithm

provides higher privacy protection than the other algorithms. Nevertheless, k-DA and k-neighbourhood methods were shown to preserve the data utility better.

The KM algorithm proceeds in the following steps. At first, KM finds at least k isomorphic subgraphs in the given graph. The isomorphic subgraphs are isomorphically extended such that the anonymization cost of the final anonymized graph is minimized. Then the extended isomorphic subgraphs are removed from the input graph, and the process is rerun on the smaller graph. After the whole input graph is processed, we get the set of disconnected graphs such that for every graph, there are at least $k-1$ other graphs that are isomorphic to it. The disconnected graphs are linked together in such a way that the final graph is k-automorphic.

Both subtasks, finding the isomorphic subgraphs and extending them isomorphically, are proved to be NP-hard in [163]. They are solved with heuristic approaches in the KM algorithm. Moreover, the KM algorithm applied only edge addition operation to modify the graph structure. Therefore, the anonymization significantly affected the degree distribution of the final anonymized network. The simultaneous use of edge addition and deletion operations in anonymization methods reduces the number of edges added to the resultant network. Thus, the degree distribution of the graph that will be anonymized with a k-automorphism method applying both edge edition operations is likely to be closer to the degree distribution of the original graph. Other network metrics could be positively affected too. Moreover, since no deterministic algorithms exist to solve the NP-hard subtask in the k-automorphism method, there is also scope for improving the KM algorithm. Overall, the aim of my research in k-automorphism anonymization is to improve the k-automorphism method to preserve data utility better and therefore raise its usability in real SN datasets, as formulated in the following research question:

Question 3. What modifications improve the k-automorphism method in terms of preserving data utility?

To address this question, I propose and implement a novel Hybrid Algorithm for k-Automorphism anonymization (HAkAu). The algorithm uses both edge addition and deletion operations and addresses NP-hard tasks with a genetic algorithm. More precisely, the NP-hard subtasks of finding isomorphic subgraphs and extending them isomorphically are merged into a single NP-hard subtask which is solved with the genetic algorithm.

The motivation for using the genetic algorithm to solve the NP-hard task arises from the fact that genetic algorithms are powerful tools while addressing NP-hard problems, and they have already been successfully applied to improve the k-degree anonymization method [116], k-neighborhood anonymization method [3] and a clustering anonymization algorithm [155].

The proposed HAkAu algorithm modifies the given social network to obtain the k-automorphism one. The algorithm is based on the structure of the KM algorithm. Unlike the KM algorithm, it solves the NP-hard subtask of finding isomorphic graph extensions with the genetic algorithm, uses edge deletion operation and employs the known GraMi algorithm [31] for finding frequent subgraphs. The HAkAu algorithm is evaluated by running experiments on real social networks. The data utility in the anonymized network is measured with the

SecGraph tool [60]. Moreover, the SecGraph tool is also used to analyze the protection against deanonymization attacks. Using the SecGraph tool and running tests on the datasets available online makes the experimental results comparable to any future research. The detailed description of the proposed solution and all findings and experimental results are given in *Chapter 7*.

3.2 Objectives of the dissertation thesis

In my PhD research, I focus on anonymization methods and approaches that modify social network datasets to preserve individuals' privacy in the published data. During my research, I have identified three open privacy-preserving issues related to k-anonymization methods, their ability to preserve data utility well and their vulnerability to deanonymization attacks.

The goal of this thesis is to solve the detected problems and answer the formulated research questions by the proposal of new procedures and algorithms and their implementation in the MATLAB programming platform. The work included testing the applicability and efficiency of the proposed algorithms on sets of synthetic and real-world social network datasets, measuring the data utility in the anonymized networks and comparing the experimental results with the results of the state-of-the-art algorithms. A significant aspect of the evaluation process is to verify the results on relevant data and to provide outputs that are easily comparable with other research. For this reason, great efforts have been made to find a proper evaluation method. Therefore, the evaluation tool SecGraph is used to measure data utility and the vulnerability of anonymization methods to deanonymization attacks.

The main contributions of the thesis are the improvement of two well-known anonymization methods and the proposal of the new deanonymization method. More precisely, the work introduces the composition attack as a novel type of deanonymization attack on social network datasets and improves the k-DA algorithm and the k-automorphism anonymization method in terms of preserving the data utility.

Furthermore, while designing the HAkAu algorithm, some minor findings were made that can be beneficial for this field of study. A novel chromosome representation for k-anonymization problems solved by genetic algorithms was proposed. The chromosome representation preserves the k-anonymity property of the chromosomes by "design"; hence, testing the k-anonymity property with fitness or selection function is unnecessary. Then the novel "divide and conquer" procedure for effective selection of vertex-disjoint subgraphs is introduced. Additionally, as far as I know, this is the first work where the GraMi algorithm is applied in the anonymization method.

Finally, even if the evaluation of data utility measurement and testing the resistance against deanonymization attacks with the SecGraph tool has been used in [3], the usage of an external evaluation tool and the emphasis on making the experimental results comparable in any future research is not very common in published anonymization studies.

4. Preliminaries

In this chapter, there are given definitions of terms that are used in the rest of this thesis. It starts with providing terms from graph theory, including the Barabasi-Albert model, which is used to produce synthetic SN networks in experiments in *Chapter 5* and introducing all network metrics used in evaluating results in *Chapters 6, 7*. Then, it defines equivalence classes in G_A, establishes matrix notation and introduces receiver operating characteristic analysis that is used for evaluating the efficiency of the proposed composition attack in *Chapter 5*. Since few NP-hard problems are discussed in *Chapters 7* and one stated problem is proved to be NP-hard in the same chapter, this chapter also includes the definition of NP-hard problems and polynomial reducibility. Finally, in order not to burden the basics while defining the special features of the proposed genetic algorithm in *Chapters 7*, this chapter ends with describing the basic principles of the genetic algorithm.

4.1 Graph theory

The definitions of well-known terms like graph, degree of the vertex, degree distribution, and adjacency matrix of the graph are omitted with reference to [13]. This section begins with the definitions of the subgraph, supergraph and frequent subgraph with minimal support.

Definition 19 (Subgraph [13])**.** *Let $G = (V(G), E(G))$ be a graph. Then $H = (V(H), E(H))$ is said to be a* subgraph *of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The property of "being a subgraph" is denoted by $H \subseteq G$.*

Definition 20 (Supergraph [13])**.** *If H is a subgraph of G, then G is said to be a supergraph of H.*

The terms isomorphism and automorphism are frequently used in *Chapter 7*. The same function F_j is marked as isomorphism in one paragraph and automorphism in another. To explain the double marking, *Lemma 1* is formulated below. Due to the lemma and its proof, graph automorphism and isomorphism definitions from *Section 2.3.8* are recapitulated.

Definition 13 (Isomorphic graphs [163])**.** *Given two graphs P_1 and P_2, P_1 is* isomorphic to P_2 *(denoted by $P_1 \backsimeq P_2$), if and only if there exists at least one bijective function $F : V(P_1) \longrightarrow V(P_2)$ such that for any edge $(u, v) \in E(P_1)$, there is an edge $(F(u), F(v)) \in E(P_2)$. The function F is called the* isomorphism *of P_1 and P_2.*

Definition 15 (Graph automorphism)**.** *Let G be a graph, and the bijective function $F : V(G) \longrightarrow V(G)$ be the isomorphism. Then F is the* automorphism *on G.*

Lemma 1. *Let $P_1, P_2 \subset G$ be disjoint subgraphs of G and let $F : G \longrightarrow G$ be an automorphism on G such that $F(P_1) = P_2$. Then the restriction $F|_{P_1}$ of F on P_1 $F|_{P_1} : P_1 \longrightarrow P_2$ is the isomorphism from P_1 to P_2.*

Proof. If F is the automorphism of G and P_1 is the subgraph of G, then for all edges $(u, v) \in E(P_1)$ holds that $(F(u), F(v)) \in E(P_2)$. The same condition holds under the restriction of F: for all edges $(u, v) \in E(P_1)$ holds that $(F|_{P_1}(u), F|_{P_1}(v)) \in E(P_2)$. Moreover, since $F(P_1) = P_2$, then for all $u \in V(P_1)$ its image $F(u) \in V(P_2)$ and for all $v \in V(P_2)$ its preimage $F^{-1} \in V(P_1)$. Hence, the restriction $F|_{P_1}$ maps P_1 to P_2, and it is a bijection, since F is a bijection. Thus, $F|_{P_1}$ is the isomorphism from P_1 to P_2. □

Thus, if F is the automorphism of G, then its restriction on a particular subgraph P_1 is the isomorphism of P_1 and P_2. To simplify the notation in *Chapter 7*, the notation for the restriction is omitted, and the same functions F_j are characterized both as automorphisms and isomorphisms. Note that F_j are automorphisms on G^*, but when we focus on subgraphs displayed with F_j, then F_j are isomorphisms of these subgraphs.

Definition 21 (Frequent subgraph with minimal support [72])**.** *Given a graph G and the minimum support s, a graph $g_f(s)$ is called a* frequent subgraph *of G if and only if there exist s subgraphs of G, $P_1, \ldots, P_s$, that are isomorphic to $g_f(s)$ and*

$$E(P_i) \cap E(P_j) = \emptyset \qquad i \neq j \qquad \forall i, j \in \mathcal{N} : 1 \leq i < j \leq s.$$

The graphs $P_1, \ldots, P_s$ are called the matches *of $g_f(s)$ in G.*

In large graphs, there can be several frequent subgraphs with given support s. In that case, the k-automorphism algorithms take the frequent subgraph with the largest number of edges. Note that matches of the frequent subgraphs are isomorphic to each other since there are instances of the graph $g_f(s)$.

Definition 22 (Crossing edge)**.** *Let G be a graph and P the subgraph of G. Each edge $(u, v) \in E(G)$ such that $u \in V(P)$ and $v \in V(G) \setminus V(P)$ is called a* crossing edge *between P and G.*

The crossing edges are essential when subgraph P is separated from G. The situation is illustrated in *Figure 4.1*. Assume that all edges and all vertices depicted in *Figure 4.1* belong to the graph G. When its subgraph P depicted with red (bold) lines is separated from G, the crossing edges have to be removed from G since one of their ending vertices belongs to P and is removed. The crossing edges are depicted with dashed lines in *Figure 4.1*.

In case two graphs P_1, P_2 are separated from G, then the set of crossing edges can be defined as $\{(u, v) \in E(G);\ \exists j \in \{1, 2\} :\ u \in V(P_j) \wedge v \notin V(P_j)\}$. A similar situation happens when more than two graphs are separated from G.

Finally, I briefly describe the *Barabasi-Albert model* for generating random scale-free networks using a preferential attachment mechanism. Social networks are proven to belong to the class of scale-free networks having power-law degree distribution [99]. The Barabasi-Albert model is one of several proposed models that generate synthetic scale-free networks. Hence, this model is used to generate synthetic testing datasets in this thesis.

According to the model, the graph generation process sequentially adds vertices. The model includes two important concepts, growth and preferential attachment, differentiating the real networks from the purely random ones. Growth is represented by adding a new vertex at each step. The new vertex is linked to

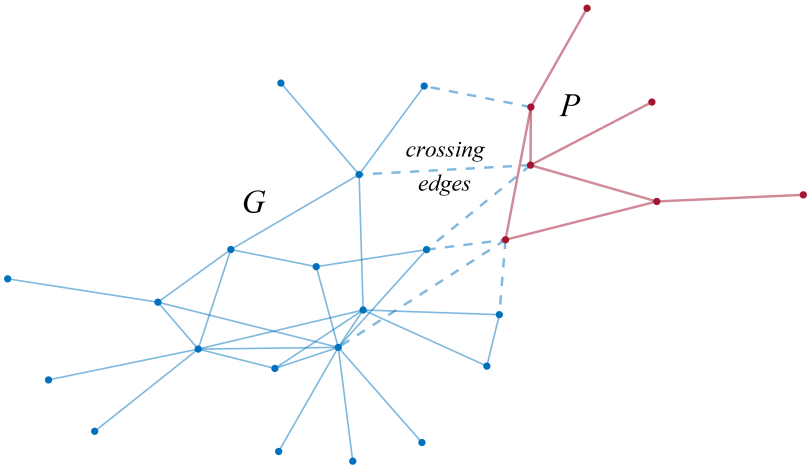


Figure 4.1: Separation of the subgraph P from G. (Source: author's work.)

the network with m new edges connecting it to m vertices already present in the network. Preferential attachment is a probabilistic rule determining that the new vertex connects to an existing vertex v with a probability proportional to its degree. More precisely, the probability that a new vertex is connected to the node v_i is given by the following formula

$$p_i = \frac{deg(v_i)}{\sum_{j=1}^{n} deg(v_j)}$$

Hence, the vertices with higher degrees are likely to quickly accumulate even more new links, while nodes with a lower degree are unlikely to receive new neighbours.

4.2 Utility metrics

This section contains the definitions of structural and application utility metrics used to evaluate experimental results. In all following definitions $G = (V(G), E(G))$ is a graph with the vertex set $V(G) = \{v_1, \ldots, v_n\}$ and the edge set $E(G)$.

Definition 23 (Average vertex degree [20])**.** *The* average vertex degree *in* G *is defined as*

$$AVD(G) = \frac{\sum_{j=1}^{n} deg(v_j)}{n}.$$

The average vertex degree is necessary for computing the sensitive value in graphs representing social networks with vertex labels in *Chapter 5*. In the rest of the thesis, the average vertex degree is computed using the number of its vertices and the number of its edges (see *Lemma 3*). The formula is derived from the "handshaking lemma", which was proposed and proved in [10].

Lemma 2 (Handshaking lemma [10])**.**

$$\sum_{j=1}^{n} deg(v_j) = 2|E(G)|$$

Proof. Since each edge connects two vertices, it increases the degree of two vertices by 1. Thus, each edge increases the sum of all degrees by two and $\sum_{j=1}^{n} deg(v_j) = 2|E(G)|$. □

Lemma 3. *Let G be a graph with the vertex set $V(G)$, and the edge set $E(G)$. Then*

$$AVD(G) = \frac{2|E(G)|}{|V(G)|}.$$

Proof. Combining the *Definition 23* and *Lemma 2* we get $AVD = \frac{\sum_{j=1}^{n} deg(v_j)}{n} = \frac{2|E(G)|}{n} = \frac{2|E(G)|}{|V(G)|}$ □

Definition 24 (Average shortest path length [20])**.** *The* average shortest path length $APL(G)$ *of the graph G is defined as*

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

where $dist(v_i, v_j)$ is the length of the shortest path from v_i to v_j, meaning the number of edges along the path from v_i to v_j.

Definition 25 (Clustering coefficient [65])**.** *The* local clustering coefficient *of the vertex v denoted $LCC(v)$ is defined as follows*

$$LCC(v, G) = \frac{\textit{the number of triangles connected to } v \textit{ in } G}{\textit{the number of triplets centred on } v \textit{ in } G}$$

The average clustering coefficient $ACC(G)$ *of the graph G is defined as the average of local clustering coefficients of all vertices $v \in V(G)$:*

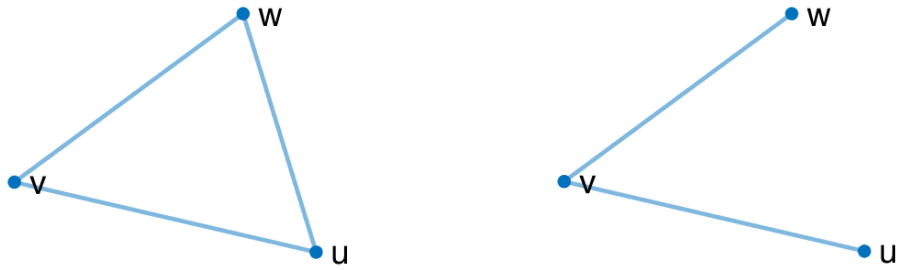
$$ACC(G) = \frac{\sum_{i=1}^{n} LCC(v, G)}{n}$$

Note the difference between a triangle connected to v and a triplet centred on it. A *triangle* connected to v is the subgraph of G consisting of three vertices $v, u, w \in V(G)$ such that $(u, v), (v, w), (u, w) \in E(G)$ (see *Figure 4.2a*). On the other hand, a *triplet* centred on v is the subgraph of G consisting of three vertices $v, u, w \in V(G)$ such that $(u, v), (v, w) \in E(G)$ (see *Figure 4.2b*). In other words, a triplet includes v and its two neighbour vertices u, w with the edges $(u, v), (v, w)$ linking u and w to v. Hence, $LCC(v)$ measures the neighbourhood density of v. The parameter $LCC(v)$ is higher in the case that for many pairs of v's neighbours u, w, there exists the edge $(u, w) \in E(G)$ too. Thus, $LCC(v)$ is higher if many triplets centred on v are also triangles.

Definition 26 (Transitivity [20])**.** *The* transitivity $T(G)$ *of the graph G is defined as*

$$T(G) = \frac{3 * \textit{(number of triangles on } G)}{\textit{(number of connected triples of vertices in } G)}$$

The transitivity is the probability of revealing the existence of tightly connected communities in the network. It measures the presence of local loops near the vertex [39] and describes the graph interconnectedness [62].



a) a triangle connected to v b) a triplet centered on v

Figure 4.2: Example of a triangle and a triplet. (Source: author's work.)

Definition 27 (Largest eigenvalue [20])**.** *Let G be a graph and* $\mathbf{Adj}(G)$ *be its adjacency matrix. The non-zero vector* $\boldsymbol{b}$ *is called* eigenvector *if*

$$\mathbf{Adj}(G) \cdot \boldsymbol{b} = \lambda \cdot \boldsymbol{b}$$

where λ is a scalar multiplier. The scalar λ is an eigenvalue *of* $\mathbf{Adj}(G)$*. Then the* largest eigenvalue *of G, denoted by $\lambda(G)$, is such an eigenvalue that $\lambda(G) \geq \lambda_i$ for all eigenvalues of* $\mathbf{Adj}(G)$*. The corresponding eigenvector is denoted with EV.*

The largest eigenvalue $\lambda(G)$ of the adjacency matrix of G is a spectral measure which encodes the information about the cycles of the networks and their diameter [20]. Now, there are definitions of network metrics implemented in the SecGraph evaluation tool [60].

Definition 28 (Betweenness centrality [33])**.** Betweenness centrality *of the vertex v_i in the graph G is given by*

$$BC(v_i, G) = \frac{\sum_{j<l} d_{jl}(v_i)}{d_{jl}}, \quad j, l = 1, \ldots, n$$

where d_{jl} is the number of shortest paths linking vertices v_j and v_l and $d_{jl}(v_i)$ is the number of the shortest paths containing the vertex v_i.

Definition 29 (Closeness centrality [33])**.** Closeness centrality *of the vertex v_i in the graph G is given by*

$$CC(v_i, G) = \frac{1}{\sum_{j=1}^{n} dist(v_i, v_j)}, \quad j \neq i$$

where $dist(v_i, v_j)$ is the distance of v_i from all other vertices in the graph.

The betweenness centrality is higher when the vertex is more frequently in-between the shortest paths that connect every other couple of vertices [33]. The closeness centrality expresses the inverse of the distance of a vertex from all the others in the network, considering the shortest paths that connect each couple of vertices [33].

Definition 30 (Effective diameter [62])**.** Diameter *is the length of the maximum shortest path between any pair of connected vertices.* Effective diameter *ED is the length of the path separating a given percentage of connected vertex pairs (commonly 90%).*

The effective diameter is more robust than the diameter, and both metrics express the graph width. Moreover, visualizing the relationship between the chosen threshold and the effective diameter may provide additional insight into the patterns of graph connectivity [62].

Definition 31 (Joint degree distribution [133])**.** Degree distribution *(Deg.) is the probability that the vertex degree equals* l, $l = 1, \ldots, max_{v \in V(G)}(deg(v))$. Joint degree distribution *(JD) is the probability that a randomly selected edge will be between vertices of degree* j *and* l, $j, l = 1, \ldots, max_{v \in V(G)}(deg(v))$.

The joint degree distribution uniquely defines the degree distribution of the graph up to isolated vertices [133]. However, graphs with the same degree distribution may have very different joint degree distributions [133].

Definition 32 (Network constraint coefficient [145, 159])**.** *The* network constraint coefficient *for the vertex* v_i *is defined as*

$$NC(v_i, G) = \sum_{j \neq i} \Big(p_{ij} + \sum_{q \neq i, q \neq j} p_{iq} \cdot p_{qj} \Big), \quad i, j, q = 1 \ldots, n$$

where p_{ij} *is the strength of the tie between vertex* v_i *and its neighbour* v_j*. In the case of unweighed graphs,* $p_{ij} = 1$ *for all ties. The network constraint coefficient for the whole graph is defined as*

$$NC(G) = \sum_{i=1}^{n} NC(v_i, G)$$

Higher values of $NC(v_i, G)$ indicate that v_i acts as less of a structural hole [145]. Moreover, $NC(v_i, G)$ represents the information and control advantages of the vertex v_i in the network structure [159]. The lower the total $NC(G)$, the more structural holes are occupied by vertices [159].

In the experimental results in *Chapter 7*, a few application metrics are measured with SecGraph. Since their formal definition would require defining terms not further discussed in the thesis, I describe them briefly, emphasizing their significance for network analysis.

Infectiousness ($Infe.$) measures the number of users infected by the disease in an infectious disease spreading model where each user transmits the disease to its neighbours with some infection rate [3]. Hence, it characterizes the communication channels in the network. *Page rank* (PR) measures the importance of each vertex within the graph based on the number of links and the importance of the linked vertices. A high PR value means the vertex is connected to many other vertices with high PR. The page rank metric is another measure of centrality for the vertices of the graph [40]. *Role extraction* (RX) approach summarized the behavior of vertices in large graphs [53]. Without any background knowledge, it determines the underlying roles in the network and assigns the mixed membership of these roles to each vertex in the graph [53]. It benefits applications like network transfer learning, measuring structural similarity, understanding the underlying behaviour in a network or network visualization [53].

Hubs score (HS) and *Authorities score* (AS) are defined together in a recursive way. The vertex is assigned a hub score equalling the sum of authorities scores of all neighbour vertices [137]. On the other hand, a vertex is assigned an authority

score equalling the sum of hubs scores of all neighbour vertices [137]. A vertex is given the authority score based on the number of hub vertices connected to it (and vice versa). For instance, assume that G is the graph of webpages and their links ($u \in V(G)$ is a webpage, $(u, v) \in E(G)$ if there is a link leading to v on u). Then $HS(u, G)$ is the number of links to other webpages that occurs on u and $AS(u, G)$ is the number of webpages with the link leading to u.

Furthermore, SecGraph detects and compares communities in the given pair of graphs. The comparison is summarized in the *community detection* metric (CD). A community is a group of vertices with more connections amongst its members than between its members, and the remainder of the graph [152]. Communities in a single graph overlap since vertices usually belong to more than one group at once [152]. The detection of communities enables the comprehensive analysis of a network structure and supports applications line classification or information propagation in the network [60].

The SecGraph measures the difference in preserving a particular metric m in the original graph G and the anonymized graph G^*. It measures the values of m in both G and G^*. The result of the measurement is the distribution of the metric between all vertices (or all pairs of vertices) in the graph. Hence, it is the vector of values. Let denote the distribution of the metric m in the graph G as $m(G)$. Then, for most metrics, SecGraph measures the cosine similarity between $m(G)$ and $m(G^*)$ to evaluate the similarity in preserving m in both graphs. Only for CD, it measures the Jaccard similarily between $m(G)$ and $m(G^*)$. Exceptions from this practice are ED and EV metrics, which are measured for the whole graph. SecGraph computes the ratio of $m(G)$ and $m(G^*)$ for those metrics. For completeness, there are the definitions of cosine and Jaccard similarities.

Definition 33 (Cosine similarity [124])**.** *Let* $\mathbf{a} = (a_1, \ldots, a_n)$ *and* $\mathbf{b} = (b_1, \ldots, b_n)$ *be two non-zero vectors. Then the* cosine similarity *is defined as*

$$S_C(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \circ \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Definition 34 (Jaccard similarity [143])**.** *For any two finite sets* A *and* B*,* Jaccard index *is defined as the ratio of the size of the intersection over union:*

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

For both similarities, their values lie between 0 and 1, $0 \leq S_C(\mathbf{a}, \mathbf{b}) \leq 1$ and $0 \leq J(A, B) \leq 1$. The higher value displays more similarities between vectors (sets), and the value 1 indicates that both vectors (sets) are equal [143].

4.3 Equivalence classes in G_A

This section begins with defining the equivalent class from the set theory.

Definition 35 (Equivalence class [98])**.** *Let* S *be a set, and* $\sim$ *be an equivalence relation on* S*. Then the* $\sim$-equivalence class *of the element* $a \in S$ *is the set*

$$Q(S, a, \sim) = \{b \in S;\ a \sim b\}$$

The equivalence $\sim$ splits the elements of S into several $\sim$-equivalence classes. When necessary, they are distinguished by indexing: $Q_i(S, \sim)$, $i \in \mathbb{N}$.

Assume a vector $\mathbf{z} = (z_1, \ldots, z_m)$ of integer values that are ordered decreasingly, $z_i \geq z_j$ for each $1 \leq i < j \leq m$. Then the vector can be viewed as the ordered list of $=$-equivalence classes where each equivalence class contains the elements of z having the same value, $Q_1(\mathbf{z}, =), Q_2(\mathbf{z}, =), \ldots, Q_l(\mathbf{z}, =)$ for some $l \leq m$. For example, let $\mathbf{z} = (5, 5, 4, 3, 3, 1, 1, 1)$, then $Q_1(\mathbf{z}, =) = \{5, 5\}, Q_2(\mathbf{z}, =) = \{4\}, Q_3(S, =) = \{3, 3\}$ and $Q_1(\mathbf{z}, =) = \{1, 1, 1\}$. These equivalence classes are used in the description of the algorithm proposed in *Chapter 6*.

Assume a graph G and the relation $\sim_d$ on $V(G)$ "having the same degree". More precisely, for two vertices $v, w \in V(G)$ $\sim_d$ is defined on $V(G)$ such that $v \sim_d w$ iff $deg(v) = deg(w)$. Clearly, $\sim_d$ is transitive, symmetric and reflexive; thus, it is the equivalence on $V(G)$. We can consider the $\sim_d$-equivalence classes

$$Q(V(G), v, \sim_d) = \{w \in V(G);\ v \sim_d w\} = \{w \in V(G);\ deg(w) = deg(v)\}$$

Given that usually not all vertices have the same degree, there are several equivalence classes in the graph. The degree of vertices in $Q(V(G), \sim_d)$ is denoted by $deg(Q(V(G), \sim_d))$, and therefore, $deg(Q(V(G), v, \sim_d)) = deg(v)$.

Equivalence classes $Q(V(G), v, \sim_d)$ can be considered in labelled and unlabelled SN graphs as well as in original and anonymized SN graphs. They are significant for demonstrating k-anonymity in SNs in *Chapter 5*.

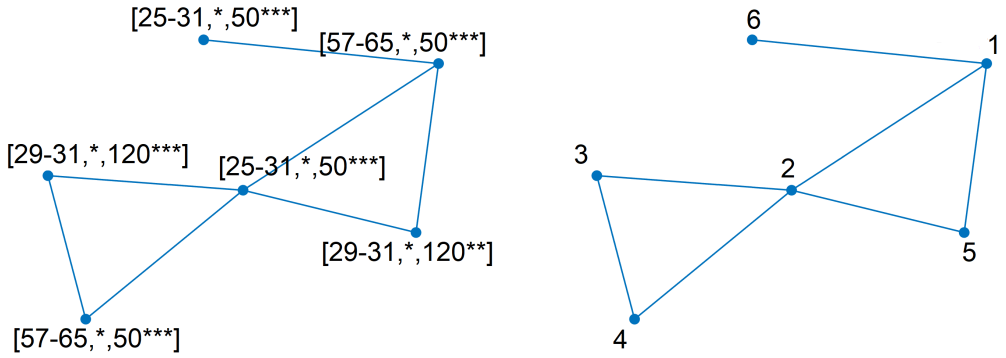
Unlike $\sim_d$-equivalence classes, which are worth defining for each graph structure, the term *attribute equivalence class* is worth defining only in anonymized graphs with attributes G_A^*. It is based on the definition of an equivalence class over an anonymized relational dataset introduced in [81].

Definition 36 (Attribute equivalence class [81])**.** *Let G_A^* be an anonymized graph with attributes. An* attribute equivalence class $Q^a(G_A^*)$ *is a set of vertices having the same anonymized r-tuple $u \in U(G_A^*)$. Let $v_i \in V(G_A^*)$, then the attribute equivalence class containing the r-tuple u_{v_i} is denoted by $Q^a(G_A^*, v_i)$; more precisely,*

$$Q^a(G_A^*, v_i) = \{v_j \in V(G_A^*);\ u_{v_i} = u_{v_j}\}.$$

The same tuple of values of attributes describes all vertices belonging to the same attribute equivalence class. Therefore, the value of the tuple of attributes of vertices in $Q^a(G_A^*)$ is denoted by $u(Q^a(G_A^*))$. Thus, for a fixed $v \in V(G_A^*)$ it holds that $u(Q^a(G_A^*, v)) = u_v$, $u_v \in U(G_A^*)$. For simplicity, $u(Q^a(G_A^*))$ is called the *attributes of* $Q^a(G_A^*)$.

The equivalence classes are based on the graph structure, whereas attribute equivalence classes are associated with the attribute table $U(G_A^*)$. There is no relationship between the number of $\sim_d$-equivalence and attribute equivalence classes. For example, the anonymized graph G_A^* in *Figure 4.3* has four equivalence classes $Q_1(G_A^*) = \{6\}$, $Q_2(G_A^*) = \{3, 4, 5\}$, $Q_3(G_A^*) = \{1\}$ and $Q_4(G_A^*) = \{2\}$, and three attribute equivalence classes $Q_1^a(G_A^*) = \{1, 4\}$, $Q_2^a(G_A^*) = \{2, 6\}$ and $Q_3^a(G_A^*) = \{3, 5\}$. Their attributes are $u(Q_1^a(G_A^*)) = (57 - 65, *, 50 * **)$, $u(Q_2^a(G_A^*)) = (25 - 31, *, 50 * **)$ and $u(Q_3^a(G_A^*)) = (57 - 65, *, 120 * *)$ respectively. A close relationship exists between the size of the smallest attribute equivalence class and k-anonymity.



a) G_A^* with attributes b) G_A^* without attributes

Figure 4.3: Social network G_A^*. (Source: author's work [93].)

Proposition 1. *Let G_A^* be an anonymized graph. Then, an anonymized set $U(G_A^*)$ satisfies the k-anonymity iff the size of the smallest attribute equivalence class is at least k.*

Proof. The attribute set $U(G_A^*)$ satisfies the k-anonymity $\Leftrightarrow$ for every $u_i \in U(G_A^*)$: $|\{u_j \in U(G_A^*);\ u_i = u_j\}| \geq k \Leftrightarrow |\{v_j \in V(G_A^*);\ u_i = u_j\}| \geq k \Leftrightarrow |Q_i^a(G_A^*, v)| \geq k,\ \forall i \in \{1, \ldots, n\}$. □

4.4 Matrix notation

Since the description of the algorithm proposed in *Chapter 7* uses matrices and manipulates with their rows, the following definition introduces the necessary notation.

Definition 37 (Matrix notation)**.** *Let $\mathbf{M}(m, n)$ be a matrix with m rows and n columns. Then $rc(\mathbf{M})$ denotes the number of rows of $\mathbf{M}$ and $r_i(\mathbf{M})$ denotes the i-th row of $\mathbf{M}$. The fact that the element e is contained in the i-th row of $\mathbf{M}$ is denoted by $e \in r_i(\mathbf{M})$. Adding the vector r as the last row if $\mathbf{M}$ is denoted by $\mathbf{M} \cup r$. Removing the row r from $\mathbf{M}$ is denoted by $\mathbf{M} \setminus r$.*

4.5 Receiver operating characteristic analysis

The receiver operating characteristic (ROC) analysis introduced in [32] is used to evaluate the accuracy of the deanonymization algorithm in *Chapter 5.5*. Thus, a classifier and true and false positive rates are defined in this section.

Definition 38 (True/False Positive/Negative [32])**.** *Consider a classification problem using two predicted classes $\{Y, N\}$ (Y equals positive and N negative). Let $\{I;\ I$ instance$\}$ be a set of instances. Some instances are 'true', and others are 'false'. Formally, each instance I is mapped to one element of the set of positive and negative class labels $\{p, n\}$. A* classifier *provides a mapping from a set of instances to predicted classes $\{Y, N\}$. Given an instance I, the classifier has four possible outcomes. If I has positive label p and is classified as Y, it is counted as* true positive *(TP). If I has positive label p and is classified as N, it*

is counted as false negative *(FN). If I has negative label n and is classified as Y, it is counted as* false positive *(FP). If I has negative label n and is classified as N, it is counted as* true negative *(TN).*

Given a classifier and the set of instances, a confusion matrix can be composed (see *Figure 4.4*). It represents the distribution of the set of instances and forms the basis for many metrics of the classifier, including the true and false positive rates. The true positive rate is estimated as the ratio of positive instances correctly classified and the total positive instances. On the other hand, the false positive rate is given by the ratio of negative instances correctly classified to all negative instances.

		True classes: p	True classes: n
Predicted classes	Y	TP	FP
	N	FN	TN

Figure 4.4: Confusion matrix of a classifier. (Source: author's work.)

Definition 39 (True/False positive rate [32])**.** *A* true positive rate *of a classifier is estimated as*

$$TPrate = \frac{TP}{TP + FN} .$$

A false positive rate *of a classifier is estimated as*

$$FPrate = \frac{FP}{FP + TN} .$$

4.6 NP-hard problems

In this section, I define the NP-hard and NP-complete problems, the NP class and polynomial reducibility. The definition of the Turing machine is omitted with reference to [131].

Definition 40 (NP class [131])**.** NP *is the class of decision problems that can be decided by the nondeterministic Turing machine in polynomial time.*

A decision problem is a question whose answer is "yes" or "no" [13]. A solution to the problem in NP is impossible to find in polynomial time. However, if for any instance of the problem whose answer is "yes", there is a certificate validating the fact, it is possible to verify the answer in the polynomial time [13]. For instance, deciding whether two graphs are isomorphic is a well-known NP problem [13]. It is impossible to decide in polynomial time whether two graphs are isomorphic. However, if someone gives us the isomorphism, we can verify in polynomial time whether it really maps the given graphs on each other or not.

Definition 41 (Polynomial reduction [13]). *A* polynomial reduction *of a Problem P to a Problem Q is a pair of polynomial-time algorithms, one of which transforms each instance of Problem P to an instance of Problem Q. The other one transforms a solution for the instance of Problem Q to a solution of the instance of Problem P.*

Definition 42 (NP-hard problem [131]). *Problem is* NP-hard *if all problems in NP class are polynomial time reducible to the Problem, even though Problem may not be in NP itself.*

The NP-hard term is used for measuring the complexity of the optimization problems in which the optimal solution to the given problems is searched. For instance, optimization NP-hard problems connected with the graph theory are travel salesman problem, finding the maximum clique or the maximum stable set in the given graph [13].

Lemma 4. *For showing that Problem P is NP-hard, it is enough to reduce a known NP-hard problem to the Problem P in the polynomial time.*

Proof. Assume that *Problem Q* is a known NP-hard problem. According to the *Definition 42*, all problems in the NP class are polynomial reducible to *Problem Q*. Hence, for each NP problem, there exists an algorithm working in polynomial time that transforms its instances to instances of *Problem Q*.

Pick an arbitrary *NP problem* and polynomial reduction Alg_{NP} that transforms instances of *NP problem* to the instances of *Problem Q*. We can show that *Problem Q* is polynomially reducible to *Problem P*. Therefore, there exists the algorithm Alg_1 working in polynomial time that transforms instances of *Problem Q* to instances of *Problem P*. Now, define algorithm Alg_2 such that on each instance of the *NP problem* it works as Alg_{NP} and on the found instances of *Problem Q*, it works as Alg_1 outputting instances of *Problem P*. Hence, Alg_2 can transform all instances of *NP problem* to instances of *Problem P*. The same reasoning can be given for the polynomial reduction of results. Hence, all problems in the NP class are polynomial time reducible to *Problem P*; thus, *Problem P* is NP-hard. □

Definition 43 (NP-complete problem). *A problem is* NP-complete *if it is NP-hard and belongs to NP.*

4.7 Genetic algorithm

Genetic algorithms are a heuristic approach based on evolutionary principles: stronger individuals are more likely to participate in creating a new generation of individuals, and each generation is better than the previous one. The search for a suitable solution to a given problem is viewed as the competition amongst the whole population of *individuals* representing potential solutions to the problem. These individuals are encoded as *chromosomes* of fixed length.

Finding the optimal solution to the given problem with the genetic algorithm begins with creating *initial population* of individuals. The initial population can be generated randomly or according to fixed rules. All individuals in the initial population are evaluated with the *fitness function*. The evaluation of individuals forms the basis of their chance to be selected for survival and reproduction. The

selection function is employed to emulate the processes of natural selection where the fitter individuals are given a higher chance to be selected. Individuals with a higher fitness function value are more likely to be selected; however, "worse" individuals also have a nonzero chance to participate in reproduction. Consequently, the selected individuals go through the process of reproduction.

The reproduction operators used in this thesis are crossover and mutation. *Crossover* is two-parent operation. It combines two parent individuals into a single child individual. When the individual in the new generation is the offspring of two parent individuals, it naturally inherits its features from its parents. *Mutation* is an asexual recombination operator maintaining genetic diversity between generations. If the individual was created with a single-parent individual mutation, its features were affected by the random process of mutating.

After a new population is created, the process is repeated. All individuals are evaluated, a subset is selected to participate in reproduction, and the next generation of individuals is produced. When the evolutionary process is repeated many times, the current generation contains individuals with high fitness function values. Such individuals represent the acceptable or even optimal solution to the given problem. The number of repetitions of the evolutionary process depends on the given problem and the computational and time capacities. The process can be repeated at least ten times, usually a hundred or a thousand times [57].

Randomness is contained in the reproduction process and the selection procedure or the creation of the initial population. It plays an essential role in the design of genetic algorithms. It helps keep the diversity in the population and prevents from converting to the local optimal solution of the given problem instead of the global optimal one. The randomness naturally causes each run of the genetic algorithm offers a unique solution, even if it is run several times on the same input. Hence, the behaviour of the genetic algorithm on the given problem is evaluated with the statistics summarizing its best, worst and average values of the monitored parameters [58]. The principles of genetic algorithms taken from [58] are formally described in *Algorithm 3*.

Algorithm 3 Basic scheme of a genetic algorithm.

Require: optimalization problem $\mathcal{P}$, a set of termination criteria $\mathcal{TC}$
Ensure: the acceptable solution of $\mathcal{P}$

1: Set $t = 0$.
2: Generate (randomly) the initial population $P(0)$.
3: Compute the evaluation of each individual in $P(0)$ with the fitness function.
4: **while** any element in $\mathcal{TC}$ is true **do**
5: Select pairs of individuals from $P(t)$ and create their offsprings. Let $P'(t)$ be the set of offspring.
6: Evaluate each individual in $P'(t)$.
7: Create the new population $P(t+1)$ from $P(t)$ and $P'(t)$.
8: $t = t + 1$
9: Evaluate each individual in $P(t)$.
10: **end while**
11: **Return** $P(t)$.

The termination criteria stopping the evaluation process can be various. The

most common criterium is setting the maximal number of generations [58]. More precisely, it is defined $t_{MAX} \in \mathbb{N}$ and the genetic algorithm stopps if $t = t_{MAX}$. The insufficient diversity in $P(t)$ is the other termination criteria [58]. It depends on the representation of the problem and the individuals and how the diversity is measured. However, when the diversity in $P(t)$ is low, the individuals in $P(t+1)$ are too similar to the individuals in $P(t)$. In other words, there is no significant evolution in the process and no need to keep it on. The genetic algorithm is also stopped if it has already found the acceptable solution in $P(t)$. The suitable termination criteria are based on the feature of the problem and the required quality of the wanted solution. Various termination criteria have been widely studied in [85].

The complexity of the genetic algorithm is determined by the complexity of its fitness function, selection function and reproduction operators. Let g denote the number of generations the genetic algorithm runs and n the number of chromosomes in a generation. Let FF represent the complexity of the fitness function, SF the complexity of selecting one individual with the selection function and RO the complexity of reproduction operators. Since the fitness function is applied on each chromosome in each generation, the evaluation takes $\mathcal{O}(g \cdot n \cdot FF)$ in total. The reproduction operators are also applied in each generation. The number of reproduction operations in one generation depends on the methodology of selecting and pairing parents. However, the set of parents usually equals to $c \cdot n$, $c \in \mathbb{R}$; thus, the complexity of reproduction operations is in $\mathcal{O}(g \cdot n \cdot RO)$. Similarly, the selection procedure is applied for selecting parents in each generation; hence its total complexity is $\mathcal{O}(g \cdot n \cdot SF)$. Finally, it is necessary to consider the complexity of creating the initial population. When the population is created randomly, its complexity is in $\mathcal{O}(n \cdot l_{CH})$ where l_{CH} is the length of a chromosome. Total, the complexity of the whole genetic algorithm is in $\mathcal{O}(n \cdot l_{CH} + g \cdot n \cdot (SF + FF + RO))$.

5. Composition attack

In this chapter, a potential threat for anonymized social network datasets called a *composition attack* is presented and described in detail. The proposed composition attack is designed as a potential threat to SNs with attributes. Therefore, the two attacked SNs with attributes should be denoted by $G_{A,1}$ and $G_{A,2}$. However, such a notation would make the formulas in this chapter too complicated to read. Thus, to simplify the notation in this chapter, instead of $G_{A,1}$ and $G_{A,2}$, the attacked pair of SNs with attributes are denoted by G_1 and G_2. At first, the motivation for considering such a threat and its effects on the privacy of SNs are described. Before giving a detailed description of the proposed attack, there are presented assumptions about SN datasets and the behaviour of their users under which the proposed composition attack works. The attack is implemented in MATLAB and tested on synthetic datasets to test its success rate. Finally, the design of the proposed composition attack against the SN dataset is compared with the design of the composition attack against the relational dataset presented in [81], and *Question 1* is answered. This chapter is based on [93].

5.1 Motivation

Recently presented deanonymization methods were focused on issues connected to sharing or publishing data from a single social network. As there are several social networks and more than half of Internet users participate in more than one social network [144, 148], an individual is likely to appear in more than one anonymized social network dataset. Thus, two anonymized social network datasets published independently of each other likely have overlapping user communities.

In general, a data publisher providing a collection of anonymized data of individuals is primarily unaware of a second anonymized dataset containing records of the same individuals. Even if the anonymized dataset preserves privacy well, there is no guarantee combining its records with records of a second anonymized dataset satisfies the same level of anonymity, as considered in [81]. An adversary who knows that two published anonymized datasets contain the records of the same individual and has access to each of them can conduct a composition attack.

The composition attack was presented as the privacy threat for relational datasets with overlapping records [35, 81, 127]. In this chapter, I show that this kind of attack is also applicable to SNs with attributes having overlapping user communities.

5.2 Effects of the attack on preserving privacy

The composition attack can be applied to a pair of SNs with overlapping user communities. It is possible that the user communities of three or more social networks also overlap, although the attack could always be applied to a pair of them. The presented composition attack does not completely deanonymize the networks. Still, it recognizes a subset of users participating in both anonymized networks and obtains pairs of vertices, one from each network, representing the

same individual. Thus, the two vertices, one from each network, representing the same individual, are called the *corresponding vertices*.

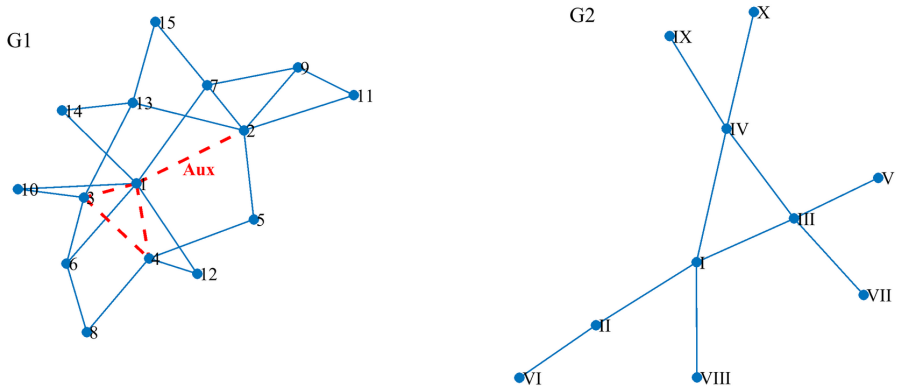
Each social network gathers users' data on its own and obtains data of different types from its users. When the adversary connects the records of an individual from two separate sources, it clearly gains more information about the individual. Although the obtained data are still anonymized, the combination of records from two different anonymized sources does not always satisfy the same level of anonymity as when the records are tested separately. Thus, the composition attack could simplify a further reidentification attack.

Additionally, when one of the considered SNs is compromised by another deanonymization method, the second social network could also be compromised. The situation is illustrated on *Figure 5.1*. Assume that the attacker aims to deanonymize the social network G_2, although the attacker cannot access any auxiliary information connected to this network. However, the attacker can link a subset of nodes with identifying information in a second anonymized social network G_1 (see *Figure 5.1a*). In other words, the attacker possesses an auxiliary subgraph Aux to deanonymize G_1. Then, the attacker applies the composition attack on G_1 and G_2 (see *Figure 5.1b*). If the attacker finds the corresponding vertices in G_2 for nodes in Aux, he or she can gain the auxiliary subgraph of G_2 to deanonymize G_2 (see *Figure 5.1c*).

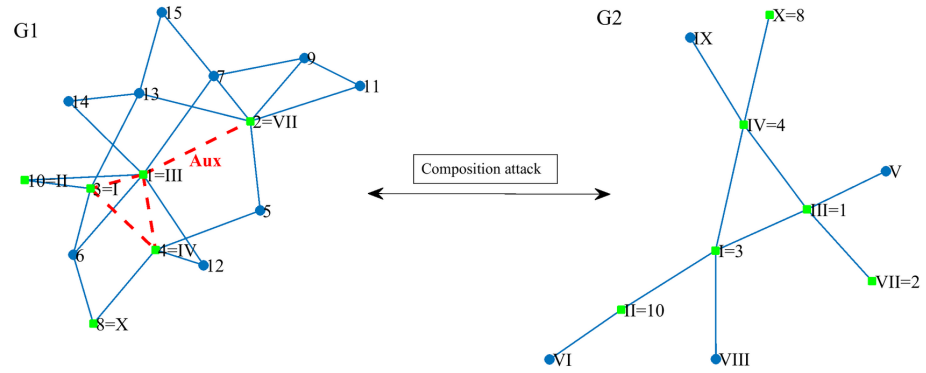
The magnitude of the attack is based on the fact that the attacker does not have to possess any special background knowledge to realize the composition attack. The attacker knows only that two anonymized social networks with overlapping user communities are anonymized with the same anonymization technique. The attacker has no access to any non-anonymized information. Moreover, the attacker has no knowledge about any user or the graph structure of the attacked networks.

5.3 Assumptions about the social behaviour of social network users

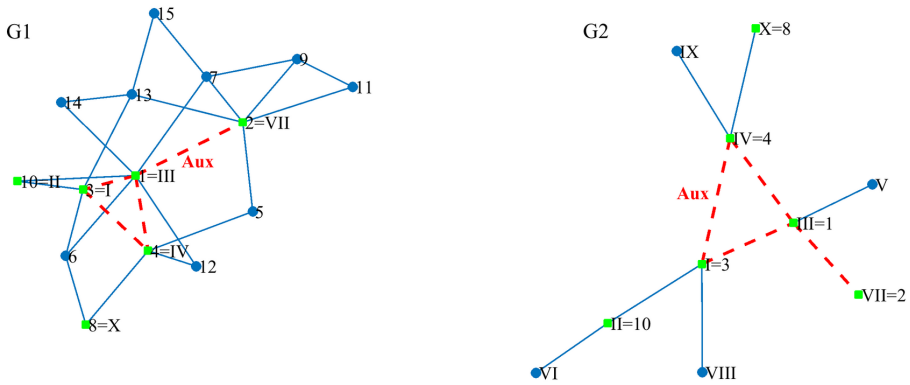
More than half of Internet users participate in more than one SN, as noted in [144, 148]. Therefore, two independent SNs are likely to have overlapping user communities. Users' online behaviour on social network sites varies among user communities. Some users are more active than others, visiting their profiles daily and spending considerable time communicating with other users, and frequently sharing personal information. Many factors influence the usage of SNs. However, a user's personality and gender are highly related to social network use, as demonstrated in [4, 102]. This thesis considers that a user's characteristics, such as personality or gender, influence the user's behaviour in the same manner on all SNs in which the user participates. For example, extroverts tend to be members of more groups on Facebook, as presented in [121]. Thus, extroverts are assumed to behave similarly to other SNs. It is assumed that if an individual tends to make an above-average number of relationships on a particular SN, he or she will act similarly on other SNs. Additionally, the number of individuals who can be reached through the SN depends on the size of the social network, as mentioned in [100]. Thus, the probability of finding users with similar interests is higher in



a) Two anonymized social network graphs with overlapping user communities. The attacker owns auxiliary information (the subgraph *Aux* with red dashed edges), which could be used to deanonymize G_2.



b) Attacker finds pairs of corresponding vertices by the composition attack.



c) Attacker gains auxiliary information for deanonymization of G_2.

Figure 5.1: Gaining auxiliary information on G_2 by a composition attack. (Source: author's work [93].)

larger networks, and an individual is likely to have more relationships in larger SNs.

Based on the previous considerations, it is assumed that, in proportion to the size of the social network, the number of users with whom an individual has a relationship is nearly constant over all social networks in which the individual participates. In other words, for every individual, the ratio of the number of relationships to the average number of relationships in the network is constant up to small δ over all social networks in which the individual participates. For example, let G_1 be a network with $|V(G_1)| = 100$ and $|E(G_1)| = 500$ and G_2 be a network with $|V(G_2)| = 50$ and $|E(G_2)| = 300$. Then, the average degree $AVD(G_1) = \frac{2*500}{100} = 10$ and the average degree $AVD(G_2) = \frac{2*300}{50} = 12$. If an individual I is connected with 10 other people in G_1, then $deg_{G_1}(I) = 10$. I assume that $\frac{deg_{G_1}(I)}{AVD(G_1)} = \frac{deg_{G_2}(I)}{AVD(G_2)} \pm \delta$, and therefore,

$$deg_{G_2}(I) = \frac{deg_{G_1}(I)}{AVD(G_2)} \cdot AVD(G_2) \pm \delta = 12 \pm \delta$$

and the individual I is likely to have 12 ($\pm\delta$) connections in G_2.

In addition, I assume that if an individual I has a relationship with an individual J on a social network G_1 and both individuals participate in the second social network G_2, they will have a relationship also in G_2. In other words, if there exist v_1, v_2 in G_1 and w_1, w_2 in G_2 such that $v_1 = v(I)$, $v_2 = v(J)$, $w_1 = w(I)$ and $w_2 = w(J)$ and there exists the edge $(v_1, v_2) \in E(G_1)$, then there exists the edge $(w_1, w_2) \in E(G_2)$. The assumptions about the social behaviour of social network users are summarized as follows.

Assumption 1. *For every individual I and every G, the ratio of the number of edges $(v(I), \cdot) \in E(G)$ to the average number of relationships in G is constant (up to small δ) over all social networks G in which I participates.*

Assumption 2. *Let I, J be individuals and G_1,G_2 be SNs. Let $\exists v_1, v_2 \in V(G_1)$, $\exists w_1, w_2 \in V(G_2)$ such that $v_1 = v(I)$, $v_2 = v(J)$, $w_1 = w(I)$, $w_2 = w(J)$ and $(v_1, v_2) \in E(G_1)$. Then, $(w_1, w_2) \in E(G_2)$.*

The composition attack on relational datasets exploits the fact that both independently published datasets have plain sensitive values in their anonymized versions. The sensitive value is assumed to be constant for each individual in both anonymized datasets. Moreover, values of sensitive attributes are not modified during the anonymization.

The attribute table $U(G_1)$ and $U(G_2)$ usually contain information describing the users of SNs. Hence, all their attributes are usually considered quasi-identifiers and are anonymized during the process. The crucial information in SN datasets is the graph structure. Therefore, a new sensitive value based on the graph structure is defined to perform a composition attack on SN data.

Definition 44. *Let G_A^* be an anonymized social network and $AVD(G_A^*)$ the average vertex degree in G_A^*. Then, for every $v \in V(G_A^*)$ a* sensitive value *$S(v)$ is defined as*

$$S(v) = \frac{deg(v)}{AVD(G_A^*)}.$$

Based on *Assumption 1*, the sensitive value $S(v(I))$ is assumed to be constant (up to small δ) overall SNs in which the individual I participates. Although the

sensitive value $S(v)$ is artificially created and represents no real characteristic of the individual v, it introduces a new feature of v. Every vertex is now described by edges $(v, \cdot)$ connecting the vertex with the rest of the network, its degree $deg(v)$, its attributes u_v and its sensitive value $S(v)$.

5.4 The proposed algorithm

The detailed description of the proposed composition attack algorithm is presented in *Algorithm 4*. It consists of the *preprocessing* stage (see lines 1-9 in *Algorithm 4*), the *composition* stage (see lines 10-20) and the *postprocessing stage* (see lines 21-25). The algorithm requires two anonymized SNs G_1^* and G_2^* and the parameter δ as the input. It returns the set $\mathcal{R}$ of pairs of corresponding vertices $[v, w]$, $v \in V(G_1^*)$, $w \in V(G_2^*)$ that are suspected to represent the same individual $v(I) = w(I)$.

Algorithm 4 Composition attack

Require: social networks G_1^*, G_2^*, the parameter δ
Ensure: the set of pairs of matching vertices $\mathcal{R}$

1: Set $\mathcal{R} = \emptyset$.
2: Find $Att(G_1^*, G_2^*) = Att(G_1^*) \cap Att(G_2^*)$ and set $r = |Att(G_1^*, G_2^*)|$.
3: **for** each $u \in U(G_i^*)$, $i \in \{1, 2\}$ **do**
4: Order the elements of the r_i-tuple u such that its first r elements correspond to the ordered list $Att(G_1^*, G_2^*)$.
5: **end for**
6: **for** each $v \in V(G_i^*)$, $i \in \{1, 2\}$ **do**
7: Compute $S(v)$.
8: **end for**
9: Divide the elements of $V(G_i^*)$ into particular attribute equivalence classes $Q_1^a(G_i^*), \ldots, Q_{m_i}^a(G_i^*)$, $i \in \{1, 2\}$
10: **for** $j := 1, \ldots, m_1$ **do**
11: **for** $l := 1, \ldots, m_2$ **do**
12: **if** $u_z(Q_j^a(G_1^*)) \cap u_z(Q_l^a(G_2^*)) \neq \emptyset$, $\forall z \in \{1, \ldots, r\}$ **then**
13: **for** each $v \in Q_j^a(G_1^*)$ and each $w \in Q_l^a(G_2^*)$ **do**
14: **if** $S(v) \in [S(w) - \delta; S(w) + \delta]$ **then**
15: the pair of vertices $[v, w]$ is added into $\mathcal{R}$
16: **end if**
17: **end for**
18: **end if**
19: **end for**
20: **end for**
21: **for** each $[v_1, w_1]$ in $\mathcal{R}$ **do**
22: **if** $\nexists [v_2, w_2] \in \mathcal{R} : (v_1, v_2) \in E(G_1^*) \ \wedge \ (w_1, w_2) \in E(G_2^*)$ **then**
23: remove $[v_1, w_1]$ from $\mathcal{R}$
24: **end if**
25: **end for**
26: **Return** $\mathcal{R}$.

To simplify the notation, $Q(G, v) = Q(V(G), v, \sim_d)$, since $\sim_d$ is the only

considered equivalence in the chapter. Thus, $Q(G, v)$ is the set of the vertices from G having the same degree as the vertex v. For simplicity, $\sim_d$-equivalence classes are called simply equivalence classes.

The first step in the preprocessing stage is the unification of $U(G_1^*)$ and $U(G_2^*)$. The number and character of attributes vary from network to network. To execute the composition attack, it is necessary to select a common subset from the set of attributes of both target networks. The common subset of attributes contains all the same attributes in both target networks. For example, if $Att(G_1^*) = \{Birthday, Age, Gender, Hometown, Languages, Work\}$ and $Att(G_2^*) = \{gender, age, spoken\_languages, education, marital\_status\}$, then the appropriate subset of attributes that could be used for the composition attack is $Att(G_1^*, G_2^*) = \{Age, Gender, Languages\}$. In the case that for two social networks G_1^*, G_2^* $Att(G_1^*) \cap Att(G_2^*) = \emptyset$, then the composition attack can not be executed on G_1 and G_2. Furthermore, $Att(G_1, G_2)$ can be viewed as the ordered list of the names of attributes. Let r be the number of names in the list; it means $r = |Att(G_1, G_2)|$. Then the attributes of $U(G_1^*)$ and $U(G_2^*)$ have to be ordered such that their first r elements correspond to the ordered list $Att(G_1, G_2)$ (see line 4). The output of the preprocessing stage is the set of the attribute equivalence classes $Q_1^a(G_i^*), \ldots, Q_{m_i}^a(G_i^*)$, $i \in \{1, 2\}$ (see line 9).

In the composition stage, the attribute classes of G_1^* and G_2^* are paired such that the values of their attributes have nonempty intersections (see line 12). Every $Q^a(G_i^*)$ can be described with its attributes $u(Q^a(G_i^*))$. For instance, assume $Att(G_1^*, G_2^*) = \{Age, Gender, Languages\}$, $u(Q_j^a(G_1^*)) = \{\langle 20; 30\rangle, F, \{English, German\}\}$ and $u(Q_l^a(G_2^*)) = \{\langle 25; 35\rangle, F, \{English, French\}\}$. Then

$$u_1(Q_j^a(G_1^*)) \cap u_1(Q_l^a(G_2^*)) = \langle 25; 30\rangle$$
$$u_2(Q_j^a(G_1^*)) \cap u_2(Q_l^a(G_2^*)) = F$$
$$u_3(Q_j^a(G_1^*)) \cap u_3(Q_l^a(G_2^*)) = \{English\}$$

Hence, all values of attributes have the nonempty intersection and $Q_j^a(G_1^*)$ and $Q_l^a(G_2^*)$ can be further processed together. For every $v \in Q_j^a(G_1^*)$ and for every $w \in Q_l^a(G_2^*)$, the closeness of the sensitive values $S(v)$ and $S(w)$ is tested. The test is based on *Assumption 1* (see line 14). In the case of the successful check, the pair $[v, w]$ is added into $\mathcal{R}$ (see line 15).

In the postprocessing stage, the cardinality of $\mathcal{R}$ is reduced. The graph structure is considered, and the neighbourhoods of particular vertices belonging to $\mathcal{R}$ are examined. Then, some false positive pairs of vertices are deleted, and the cardinality of $\mathcal{R}$ is decreased. Based on *Assumption 2*, the focus is on the edges of $E(G_1^*)$, $E(G_2^*)$ such that

$$\exists [v_1, w_1], [v_2, w_2] \in \mathcal{R} : \ (v_1, v_2) \in E(G_1^*) \ \wedge \ (w_1, w_2) \in E(G_2^*).$$

All pairs of vertices $[v_1, w_1]$ not fulfilling the previous relationship with any other pair $[v_2, w_2] \in \mathcal{R}$ are removed from $\mathcal{R}$ (see line 22 and 23). Then, the result is the final set $\mathcal{R}$ of pairs of corresponding vertices, one vertex from G_1^* and the second one from G_2^*.

5.4.1 Complexity

The most time-consuming part of *Algorithm 4* is reducing the cardinality of the set $\mathcal{R}$. The time complexity of the reducing stage is $\mathcal{O}(|\mathcal{R}|^2)$. The remaining stages are asymptotically less difficult. Therefore, the time complexity of the entire algorithm equals $\mathcal{O}(|\mathcal{R}|^2)$. The cardinality of $\mathcal{R}$ depends not only on the graph size but also on the anonymity level k, the graph structure and the variability of the attributes describing particular vertices. The anonymity level k, the graph structure and the table of attributes together determine the total number of attribute equivalence classes. In cases with only a few attribute equivalence classes, they are large, and many pairs of vertices are added to the set $\mathcal{R}$. As the high value of k implies a small number of attribute equivalence classes, the cardinality of $\mathcal{R}$ increases with an increasing value of k while the graph size remains constant. Similarly, when the level of anonymization is constant, the cardinality of $\mathcal{R}$ increases with an increase in the graph size.

5.5 Experimental results

In this section, the implementation of the algorithm and the results of the experiments conducted on synthetic networks are presented. All experiments were performed on a laptop computer running Microsoft Windows 7 operating system with 8 GB RAM and a 2.60 GHz processor. The programs were written in MATLAB 9.2.0.538062 (R2017a). The implementation is included in the attached CD, and the overview of corresponding MATLAB files is given in *Attachment B*.

5.5.1 Generation of synthetic scale-free networks

Real social networks are scale-free networks with power-law degree distribution, as presented in [99]. Therefore, the synthetic networks used for experiments must also fulfil the scale-free property. The Barabasi–Albert model described in *Section 4.1* was used to generate synthetic networks with the scale-free property. The existing implementation of the model, SFNG.m MATLAB function from the B-A Scale-Free Network Generation and Visualization MATLAB toolbox introduced in [38], was used. The input parameters of $SFNG(n, m, seedM)$ function are the number of nodes of the output graph n, the number of links added at every step m and the seed matrix $seedM$. The seed matrix is an adjacency matrix of a small initial graph to which the B-A algorithm links additional nodes. Each node of the initial graph must have at least one link. The initial graph, represented by the seed matrix, is called a *seed subgraph*. The number of links m affects the graph's density: the more links added in each step, the denser the output graph.

As the algorithm is applicable to two social networks, two scale-free graphs G_1 and G_2, representing two synthetic social networks, were generated before every run of the algorithm. To fulfil the required *Assumptions 1,2* described in *Section 5.3*, the same seed matrix was used for creating both graphs. The same seed guaranteed both networks would have an overlapping user community with the assumed properties. Unless stated otherwise, the size of the seed matrix was approximately 10% of the size of the smaller input graph.

For each vertex of each graph, its identifying attribute *Id* as well as its non-identifying attributes *Age, Gender, Country code, Marital status, Education, Language* were assigned a random value from their domains. All attributes were discrete; the size of their domains is included in *Table 5.1.* All the values of attributes belonging to the vertices of one graph were stored in a relational table called an *attribute table.* During the anonymization process, the identifying attributes were removed from the attribute tables, and non-identifying attributes were anonymized by applying the greedy data anonymization method for relational datasets called Mondrian, introduced in [73]. Therefore, the anonymized attribute tables satisfied k-anonymity for a chosen k. The existing implementation of the Mondrian Multidimensional k-Anonymity method, included in the UTD anonymization toolbox available in [142], was used in the anonymization process. Unless stated otherwise, the attribute tables satisfied 2-anonymity.

The graph structure was not anonymized during the experiments, but the original networks with an unchanged set of vertices and an unchanged set of edges were used.

Attribute	Domain size
Id	26^4
Age	100
Gender	2
Country code	19
Marital status	4
Education	6
Language	7

Table 5.1: Attribute domain size. (Source: author's work [93].)

5.5.2 Definition of accuracy

The parts of the graphs being unequal to the seed and the attributes describing vertices outside the seed subgraph were generated pseudorandomly. Therefore, it was assumed that no pair of corresponding vertices could be found outside the seed subgraph. The results evaluation was restricted to only the vertices of the seed subgraphs. However, for every vertex v of the seed subgraph of G_1, there existed a vertex w of the seed subgraph of G_2 such that v and w represented the same individual. Therefore, every vertex of the seed subgraph of G_1 should be paired with a vertex of the seed subgraph of G_2, and vice versa. The seed subgraph of G_i is denoted by $seed(G_i)$.

After one run of the algorithm, all vertices of $seed(G_1)$ and all vertices of $seed(G_2)$ were divided into three sets: *truly paired vertices* (TPV), *falsely paired vertices* (FPV) and *missed vertices* (MV). The set TPV included all vertices of both seed subgraphs correctly paired with the corresponding vertex. The set FPV contained all vertices of both seed subgraphs that were paired with another vertex, but the paired vertex was not the true corresponding vertex. The set MV

contained all vertices from both the seed subgraphs for which the algorithm did not (and should) find a pair.

The classifier could be defined using the results of one run of the algorithm and *Definition 38*. For simplicity, the classifier is described in detail with vertices only from $seed(G_1)$. A similar set of instances was defined for vertices of $seed(G_2)$, and they were classified similarly. Instances of two types were defined as follows:

I=*a vertex* $v \in seed(G_1)$ *is paired with a corresponding vertex* $w \in seed(G_2)$
$\hat{I}$=*a vertex* v *is paired with any other vertex* $\hat{w} \in seed(G_2) \setminus \{w\}$ *that does not represent the same individual.*

Clearly, all instances I had a positive label p, and all instances $\hat{I}$ had a negative label n. If the algorithm paired a vertex $v \in seed(G_1)$ with the corresponding vertex $w \in seed(G_2)$, the classifier marked the instance I as Y and the instance $\hat{I}$ as N. Suppose the algorithm paired the vertex v with any other vertex from $seed(G_2) \setminus \{w\}$, the classifier marked the instance I as N and the instance $\hat{I}$ as Y. If the algorithm did not find any pair for the vertex v, both instances, I and $\hat{I}$ were classified as N because each vertex of both seed subgraphs should be paired.

The rates $TPrate$ and $FPrate$ of the described classifier using the sizes of sets TPV, FPV and MV are defined as

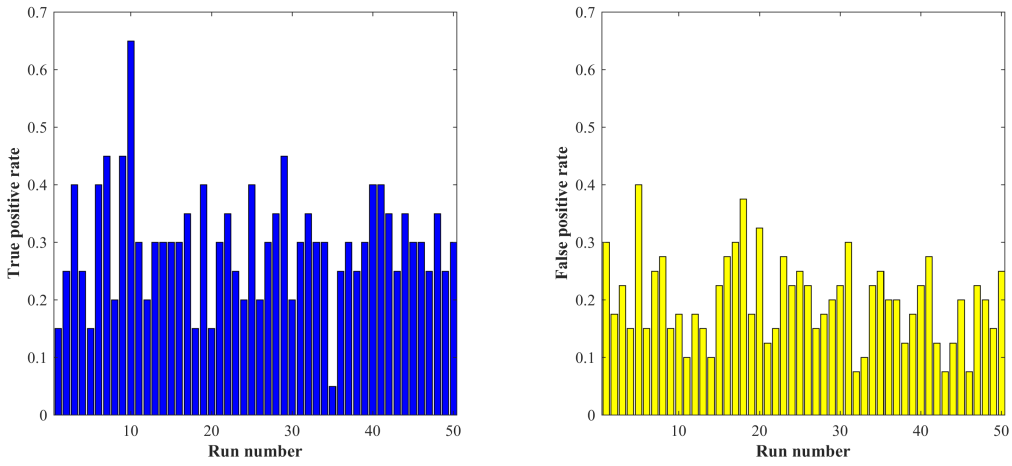
$$TPrate = \frac{|TPV|}{|TPV| + |FPV| + |MV|}$$

$$FPrate = \frac{|FPV|}{|TPV| + |FPV| + |MV|}$$

The accuracy of the proposed algorithm is a measure of a potential privacy risk caused by the execution of the composition attack. The accuracy is defined as $TPrate$ in the evaluation of the results. It corresponds to the definition of the accuracy of the composition attack on relational datasets presented in [81]. $FPrate$ represents the error rate of the algorithm because it indicates the expectancy of false positive errors of the classifier. The success of the algorithm is also measured by the comparison of $TPrate$ and $FPrate$. In cases where $TPrate > FPrate$, the classifier makes decisions better than a random guessing classifier, as presented in [32]. Thus, if $TPrate > FPrate$, the proposed algorithm pairs vertices more successfully than another classifier based only on random guessing.

5.5.3 Results evaluation

I examined the accuracy of the implemented algorithm for pairs of synthetic scale-free networks differing in size and density. As every run of the algorithm included a pseudorandom generation of two scale-free networks, the two networks and their graphs G_1 and G_2 were different in every run, even if the input parameters of the algorithm remained constant. Therefore, the algorithm was run 50 times for every combination of parameters. The average $TPrate$ and average $FPrate$ of the 50 runs of the algorithm were computed as the average accuracy and average error rate. $TPrate$s and $FPrate$s did not vary considerably during the 50 runs of the algorithm. An example of the variability of $TPrate$ and $FPrate$ is given



a) $TPrate$ b) $FPrate$

Figure 5.2: Variability of $TPrate$ and $FPrate$ in 50 runs of the algorithm with parameters $|V(G_1)| = 200$, $|V(G_2)| = 250$ and $m_1 = 4$, $m_2 = 4$ and k=2. (Source: author's work [93].)

in *Figure 5.2*. In general, the variance of $TPrate$ in the 50 runs was up to 2%, and the variance of $FPrate$ was up to 1%.

The following parameters of the algorithm were considered: the number of nodes of G_1 $|V(G_1)|$, the number of links m_1 added to G_1 at every step during the generating procedure, the number of nodes of G_2 $|V(G_2)|$, the number of links m_2 added to G_2 at every step during the generating procedure, the number of nodes of the seed subgraph $|V(seed)|$ and the required level of anonymization k. The domains of all parameters are summarized in *Table 5.2*. Some parameters were fixed to analyze the algorithm's accuracy, and the rest took values from their domains.

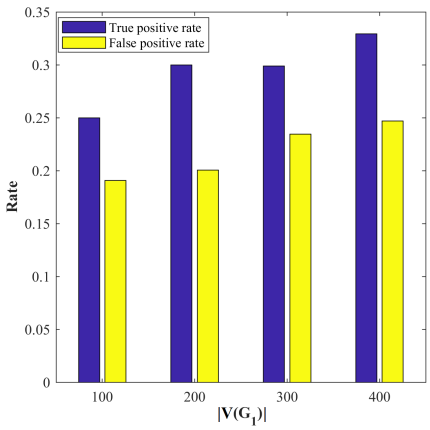
Parameter	Domain
$\|V(G_1)\|$	$\{100, 150, 200, 300, 400\}$
m_1	$\{2, 3, 5, 10\}$
$\|V(G_2)\|$	$\{150, 200, 250, 350, 450\}$
m_2	$\{2, 3, 5, 10\}$
$\|V(seed)\|$	$\{10, 15, 20, 25, 30, 35, 40\}$
k	$\{1, 2, 3, 5\}$

Table 5.2: Parameter domains. (Source: author's work [93].)

Dependence of the accuracy on the graph size. Initially, the parameters $m_1 = 4$, $m_2 = 4$ and $k = 2$ were fixed, $|V(seed)| = |V(G_1)| \cdot 0.1$ was set and the values of $|V(G_1)|$ and $|V(G_2)|$ were changed to determine whether the accuracy of the algorithm depends on the size of the networks. In *Figure 5.3.a*, it can be seen that better results were obtained with larger graphs. For $|V(G_1)| = 400$ and $|V(G_2)| = 450$, $TPrate$ achieves its maximum value, 33%.

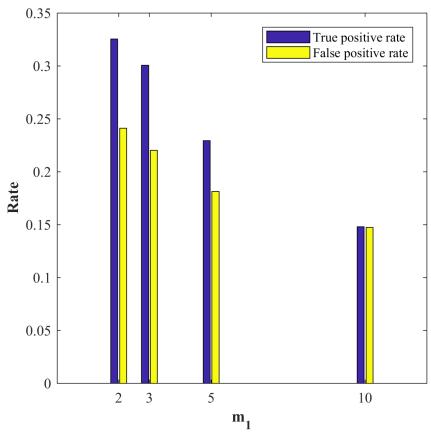
Dependence of the accuracy on the graph density. Moreover, the graph sizes $|V(G_1)| = 150$, $|V(G_2)| = 200$ and $|V(seed)| = 15$ and $k = 2$ were fixed to determine whether the accuracy is dependent on the density of the networks. In *Figure 5.3b*, it can be observed that $TPrate$ becomes worse with an increase in the values of m_1 and m_2 and almost achieves the level of average $FPrate$ with $m_1 = m_2 = 10$ ($TPrate = 16\%$, $FPrate = 15\%$). Adding more links at every step of the graph generation process causes the seed subgraph to drown in the rest of the network and makes finding the corresponding vertices more difficult.

Dependence of the accuracy on the level of anonymization. Finally, all values, except the level of anonymization, were fixed. $|V(G_1)| = 150$, $|V(G_2)| = 200$, $|V(seed)| = 15$, $m_1 = 4$ and $m_2 = 4$ were set to observe the dependence of the accuracy on the level of anonymization of the attribute tables. In this case, the decreasing trend of the accuracy is the most visible (see *Figure 5.3.c*). As expected, the accuracy of the algorithm decreased with a higher level of anonymization. With no anonymization ($k = 1$), $TPrate$ was equal to $48,8\%$, and with $k = 5$ $TPrate$ fell to 15%.



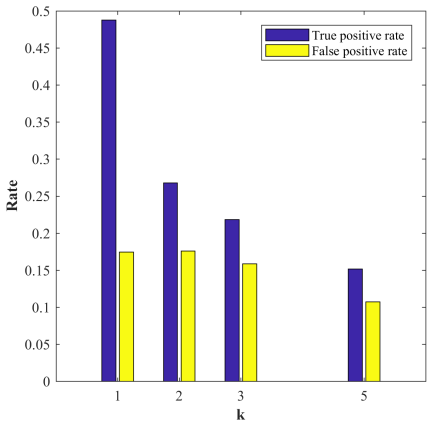
a) Dependence of the accuracy on the graph size.

Note: $|V(G_2)| = |V(G_1)| + 50$.



b) Dependence of the accuracy on the graph density.

Note: $m_2 = m_1$.



c) Dependence of the accuracy on the level of anonymization.

Figure 5.3: Dependence of the accuracy on parameters. (Source: author's work [93].)

In summary, the accuracy of the algorithm increased with larger input graphs, whereas it decreased with increasing graph density and anonymization level. However, the algorithm found pairs of corresponding vertices more successfully than methods based on random guessing because the average $TPrate$ is always higher than the average $FPrate$. A summary of all the achieved average true and false positive rates is provided in *Table 5.3*. I consider a correct pairing of $20 - 30\%$ vertices from all appropriate vertices in synthetic scale-free networks to be a good starting point for further research on composition attacks against social network data.

$\lvert V(G_1)\rvert$	m_1	$\lvert V(G_2)\rvert$	m_2	k	$\overline{TPrate}$	$\overline{FPrate}$
100	4	150	4	2	25%	19%
200	4	250	4	2	30%	20%
300	4	350	4	2	30%	23%
400	4	450	4	2	33%	25%
150	2	200	2	2	33%	24%
150	3	200	3	2	30%	22%
150	5	200	5	2	23%	18%
150	10	200	10	2	16%	15%
150	4	200	4	1	49%	17%
150	4	200	4	2	27%	18%
150	4	200	4	3	22%	15%
150	4	200	4	5	15%	11%

Table 5.3: Average true positive and false positive rates from 50 runs of the algorithm with given parameters. $\overline{TPrate}$ and $\overline{FPrate}$ denote average $TPrate$ and $FPrate$ respectively. (Source: author's work [93].)

Experimental results indicated that the weak points of the proposed algorithm are the rules determining whether the pair of vertices is false positive or true positive. During several algorithm runs, the true pair of corresponding vertices was found; however, it was incorrectly removed in the composition stage. The true pair of corresponding vertices was often nearly accepted as a true positive but ultimately dismissed. Therefore, improving the rules would render the algorithm more efficient when it takes real social networks as input. The possibility of improving the rules based on the graph structure or specific features of the particular input networks will be studied in future research. However, in this case, it must be assumed that the attacker has background knowledge about the graph structure.

5.6 Discussion

In this section, it was innovatively presented that social network datasets could be attacked by the composition attack, just like relational datasets. Furthermore, a new sensitive value in a social network graph that enables the attacker to perform the attack was introduced, and a new composition attack algorithm against two

social network datasets was proposed. Its feasibility was experimentally proved by running the algorithm on a set of synthetic scale-free networks. Its ability to find $20 - 30\%$ of corresponding vertices in synthetic scale-free networks confirms that the algorithm can be suggested for testing as a privacy threat on real social network datasets.

Now let me address the problem stated in *Question 1* and summarize the modifications that must be done to apply the composition attack to SN datasets. The proposed composition attack aims at two datasets with overlapping users community, similar to the composition attack against relational data introduced in [81]. Moreover, the attribute equivalence classes are exploited in the same way as the equivalence classes in [81], and the same knowledge of the adversary is considered.

Unlike the relational datasets, all attributes of $U(G_A^*)$ are usually found to be quasi-identifiers and do not contain any sensitive attribute. The sensitive attribute is important in composition attacks on relational datasets [81]. Hence, the artificial sensitive value must be defined before applying the composition attack to the SN data. The sensitive value of the individual in SN data has to be determined so that it is nearly constant over all SN datasets in which the individual participates. It is essential for performing the composition attack on the SN dataset. In the proposed attack, the sensitive value is based on the graph structure of SN datasets. The proposed sensitive value is assumed to be nearly constant in SNs making two assumptions about users' behaviour in SNs. The assumptions are based on published findings about SN users, and their online behaviour [121, 4, 102]. Hence, the sensitive value of all vertices of both attacked datasets has to be computed in the preprocessing stage of the attack, which is not required when attacking relational datasets.

Furthermore, reducing the cardinality of $\mathcal{R}$ based on the graph structure can not be naturally applied in relational datasets. This step is added to the algorithm to balance the inaccuracy caused by using the artificial sensitive value instead of the real one. The artificial sensitive attribute is less reliable than the real one in relational datasets. Using the artificial sensitive attribute in the composition stage of the attack produces more false positives that have to be removed after the composition stage.

To summarize the answer to *Question 1*, the composition attack was proved to be applicable to social network data. The crucial modifications required to apply the attack to SN data are computing the sensitive value based on the graph structure and removing the false positives from the set of found corresponding vertices.

6. Heuristic noise addition method

This chapter addresses *Question 2* and presents a novel approach for noise addition operation in the k-degree anonymization algorithm k-DA. The proposed high-degree noise addition method modifies the degree sequence anonymized by the degree anonymization procedure of k-DA before it is processed by the graph construction procedure of k-DA. The method is implemented in the greedy version of the k-DA algorithm. The whole algorithm based on k-DA with the novel noise addition method is denoted by heu-kDA. The usability of the heu-kDA is demonstrated by running experiments on the set of different real-world social network datasets. This chapter is based on [95].

6.1 Motivation

The k-DA algorithm, introduced in [84] and described in *Section 2.3.8*, is considered to be the basis of the k-degree anonymization. Its goal is to modify the original SN graph G to the k-degree anonymous graph G^* with edge editing operations. The fewer changes made in the graph, the less information loss caused by the anonymization.

The k-DA algorithm performs on G in two steps. At first, *Greedy* or *Dynamic programming* algorithm takes as the input the degree sequence d_G and the anonymization parameter k and finds the k-anonymous degree sequence d^*_G. Then *Supergraph* algorithm tries to modify G to construct G^* such that $d_{G^*} = d^*_G$. If the *Supergraph* algorithm fails, d^*_G is unrealizable and has to be slightly modified by adding noise. Then, *Supergraph* runs repeatedly until G^* is found such that $d_{G^*} = d^*_G$. To recall the design of the whole k-DA algorithm, I repeat *Figure 2.6* from *Chapter 2*. Since a similar scheme is made for the novel heu-kDA algorithm later in this chapter, repeating the figure also emphasizes the differences and similarities between both algorithms.

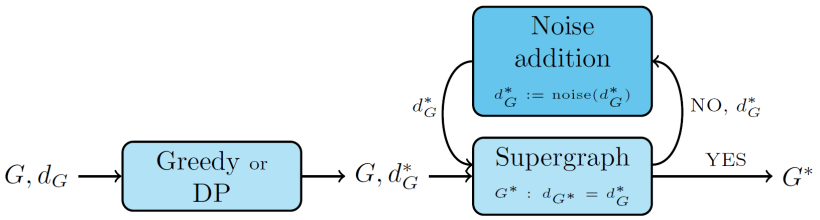


Figure 2.6: Scheme of k-DA algorithm. (Source: author's work.)

The noise addition strategy significantly affects both the speed and efficiency of the whole k-DA algorithm. Actually, there was no noise addition procedure in the original k-DA algorithm. In case d^*_G was unrealizable, k-DA added a random noise into d^*_G. The noise addition approach proposed in this thesis is based on the power-law distribution of real SNs. It modifies the degree of high-degree nodes since high-degree nodes are proven to significantly increase the total anonymization cost of k-DA [46].

6.2 Greedy version of the k-DA algorithm

In this thesis, I work with the k-DA algorithm using *Greedy* algorithm and the simultaneous edge addition and deletion in *Supergraph*. The detailed description of *Greedy* including the definition of C_{merge} and C_{new} is given in *Algorithm 1* in *Section 2.3.8*. The main idea of the *Greedy* degree anonymization algorithm is summarized here since it is crucial for explaining the proposed noise addition method.

The degree sequence d_G, sorted in descending order, is required as the input to *Greedy*. The *Greedy* algorithm first forms a group consisting of the first k vertices with the highest degrees. Then it computes the median value of their degrees and changes the degree of the first k vertices to the median value. After that, it evaluates whether it is better to join the $(k+1)$-th vertex into the previously formed group or start a new group at position $(k+1)$. The decision is based on the computation and the comparison of two costs: the cost of merging the $(k+1)$-th node to the first group C_{merge} and the cost of creating a new group starting with the $(k+1)$-th node C_{new}.

After proceeding recursively with all elements of d_G, all vertices are divided into groups about at least k elements. All vertices in one group have the same degree. Hence, the k-anonymized degree sequence d_G^* is created. The groups of vertices having the same degree correspond to the $\sim_d$-equivalence classes.

Let us look at the sequences d_G, d_G^* as the sequences of values rather than the sequence of vertex degrees. Then, d_G^* can be divided into $=$-equivalence classes

$$Q(\mathbb{N}, a, =) = \{a \in \mathbb{N};\ a = b\} \quad \text{where} \quad deg(Q(\mathbb{N}, a, =)) = a.$$

Then, the degree sequence d_G^* equals to the ordered list of $=$-equivalence classes $Q_1, Q_2, \ldots, Q_m$ where $deg(Q_i) \geq deg(Q_j)$ for each $1 \leq i < j \leq m$ and $|Q_i| \geq k$. Since $\sim_d$-equivalence classes are not used in the rest of this chapter, $=$-equivalence classes are simply called equivalence classes in the rest of this chapter.

For example, let me show how the *Greedy* algorithm proceeds on $k = 2$ and $d_G = (6, 4, 4, 2, 2, 1, 1, 1)$. The median value of the first $k = 2$ elements of d_G equals 5. Hence, d_G^* is set to $(5, 5, 4, 2, 2, 1, 1, 1)$. The third element equals 4. The algorithm has to decide whether it is better to add the element to the first equivalence class Q_1 and increase its value to 5 or to start Q_2 starting with the third element. Since the merging cost $C_{merge} = 2$ and the cost of starting new equivalence class $C_{new} = 2$, the third element is added to Q_1 and $d_G^* = (5, 5, 5, 2, 2, 1, 1, 1)$. Then the fourth element equalling 2, is evaluated. Since $C_{merge} = 5$ and $C_{new} = 0$, the fourth element is not merged to the Q_1 and the *Greedy* algorithm is recursively run on the subsequence $(2, 2, 1, 1, 1)$. The final 2-anonymized degree sequence is $d_G^* = (5, 5, 5, 2, 2, 1, 1, 1)$ where $Q_1 = \{5, 5, 5\}$, $Q_2 = \{2, 2\}$ and $Q_3 = \{1, 1, 1\}$.

The list L of indices of elements starting the equivalence classes can be easily derived from the degree sequence d_G^*. Since the first element of d_G always starts Q_1, $L = (1)$ at the beginning. If the $(k+1)$-th element is not merged to Q_1, $L = (1, k+1)$ and then the algorithm runs recursively on the subsequence of d_G. In other words, $L = (1, |Q_1| + 1, |Q_2| + 1, \ldots, |Q_m| + 1)$. In the above example, $L = (1, 4, 6)$.

6.3 High-degree noise addition heuristic

This section contains the description of the high-degree noise addition heuristic, which is applied on d_G^* after the *Greedy* algorithm, in case d_G^* is unrealizable. The degree distribution in real SNs is the power-law distribution, as presented in [99]. Thus, there are a few nodes with a very high degree and many with a very low degree. If the high-degree nodes are merged into one equivalence class with lower-degree ones, the highest degrees are very distant from the median of the class. Therefore, the values of the few high-degree nodes decrease greatly by the *Greedy* algorithm. It implies removing a large number of edges from G in the graph construction algorithm, which processes d_G^* after the *Greedy* algorithm. This reduction is impossible if the total number of required degree reductions in the rest of d_G^* is lower. Thus, the d_G^* becomes unrealisable.

Depending on the size and the structure of the network as well as the value of k, the high-degree nodes can meet the lower-degree nodes in some of the first few equivalence classes. Considering the power-law distribution, the problem can not arise in the middle or end of d_G^* where the degree values are closer to each other. Thus, the noise addition method can deal only with several first classes at the beginning of d_G^*.

The problem arises in classes with a mixture of high- and lower-degree nodes. More precisely, there is a significant difference between the maximum values in the class and the median value in the class. The proposed method is not based on the exact identification of the problematic classes and the exact correction computation for every problematic class, which would be too difficult and time-consuming. It is based on the simple heuristic that meets the core of the problem well and is efficient for computation demonstrated with the experimental results. In summary, the proposed heuristic increases the anonymized value in the first few equivalence classes of d_G^*, which reduces the number of required edge deletions. Thus, it increases the probability that modified d_G^* is realizable. The procedure for correcting medians is detailed in *Algorithm 5*. Note that $d_G[i]$, $\delta^*_{G,s}[i]$ and $L[i]$ are the i-th elements of d_G, $\delta^*_{G,s}$ and L respectively.

The crucial idea behind the whole high-degree noise addition method is to decrease the difference between the maximum values and the median values in $Q_1, \ldots, Q_{max}$ for some $max < m$ where m is the total number of equivalence classes in d_G^*. Moreover, since the unrealizability of d_G^* is often caused by the lack of existing edges for edge removal, the median should be moved closer to the high-degree values and not vice versa. Thus, the median of Q_i is increased for each $1 \leq i \leq max$ by multiplying it with the parameter x that is greater than one (see line 10 in *Algorithm 5*). The formulas for calculating x and max were found experimentally. The parameter x is computed separately for each $Q_1, \ldots, Q_{max}$, but the median correction is done only if $x > 1$ for Q_i.

The values of x for $Q_1, \ldots, Q_{max}$ depend on the modification parameter s (see line 7). The parameter s is usually a small real number, and the method of how to find the most suitable s for a given dataset and a given k is presented in the next section. The parameter s is constant for one run of *Algorithm 5*. Hence, in the single run of the algorithm, the same s is used in the computation of x for all classes $Q_1, \ldots, Q_{max}$.

Algorithm 5 The correction of the median

Require: the modification parameter s, the degree sequence d_G, the degree sequence returned from *Greedy* d_G^*, the anonymization parameter k
Ensure: the degree sequence $\delta_{G,s}^*$

```
Find the list of indices L from d*_G.
Set n = |d_G|
Set δ*_{G,s} = d*_G
Set max = ⌊ln(n/k)⌋
for i := 1, ..., max do
    Set j_1 = L[i] and j_2 = L[i+1] − 1
    Compute x = (d_G[j_1] / d_G[j_2])^{1/2s}
    if x > 1 then
        for l := j_1, ..., j_2 do
            Set δ*_{G,s}[l] := ⌊δ*_{G,s}[l] · x⌋
        end for
    end if
end for
Return δ*_{G,s}.
```

6.4 The modification parameter setting

The scheme of the whole heu-kDA algorithm is illustrated in *Figure 6.1.* As mentioned above, the *Greedy* algorithm first runs on G and d_G to produce d_G^*. Afterwards, *Supergraph* is applied on d_G^*. In case d_G^* is unrealizable, then d_G^* enters the first run of the *High-degree noise addition procedure.* In its first run, the modification parameter s is set to one. Then, the *Algorithm 5* runs with $s = 1$ and modifies the given d_G^* to $\delta_{G,1}^*$. Hence, the anonymized degree sequence is changed, $d_G^* := \delta_{G,1}^*$ and *Supergraph* runs again and tries to find G^* with $d_{G^*} = d_G^*$. If the new d_G^* is still unrealizable, *High-degree noise addition* algorithm runs again. The value of the modification parameter is changed, and the correction algorithm runs with the new value of s. The cycle continues until d_G^* is found realizable and G^* is produced by *Supergraph.* In all accomplished experiments with different real SNs, the suitable s was always found, and it took at most seven repetitions of *Supergraph* and *High-degree noise addition.*

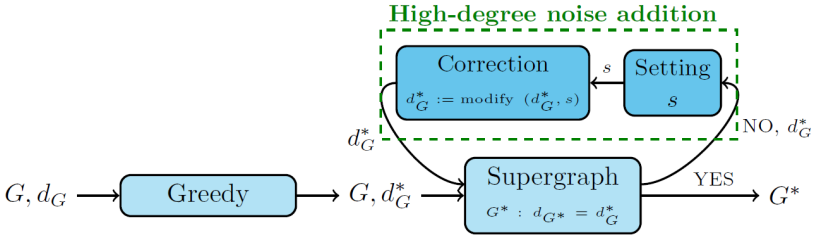


Figure 6.1: Scheme of heu-kDA algorithm. (Source: author's work.)

It remains to describe how to set the values of s in the different runs of *High-degree noise addition* method. The *High-degree noise addition* method with the method of finding the most suitable modification parameter is described in detail

in *Algorithm 6*. As mentioned above, it is crucial to find the most suitable s for a given G and a given k.

Algorithm 6 High-degree noise addition

Require: the degree sequence d_G, the degree sequence returned from *Greedy* d^*_G, the anonymization parameter k
Ensure: the k-degree anonymized G^*

Run $S(d^*_G)$ ensuring G^* and d_{G^*}.
Set $r_0 = \sum_{i=1}^{n} d_{G^*}[i] - \sum_{i=1}^{n} d^*_G[i]$.
if $|r_0| \leq 1$ **then**
    Quit successfully and **return** G^*.
else
    Set $s = 1$.
    Set $r_s = C(1, d_G, d^*_G, k) + S(d^*_G)$ and $previous\_r = r_s$.
    if $\mathrm{sgn}(r_0) \neq \mathrm{sgn}(r_s)$ **then**
        Set $dec = 1$, $cont = 1$.
        while $cont < 10$ **do**
            if $|r_s| \leq 1$ **then**
                Quit successfully and **return** G^*.
            else
                Modify $s = s \pm dec$.
                Set $r_s = C(s, d_G, d^*_G, k) + S(d^*_G)$.
                if $\mathrm{sgn}(r_s) \neq \mathrm{sgn}(previous\_r)$ **then**
                    Modify $dec = dec \cdot 0.1$ and set $cont = 0$
                else
                    Set $previous\_r = r_s$.
                end if
            end if
            Set $cont = cont + 1$.
        end while
        Quit unsuccessfully.
    else
        Quit unsuccessfully.
    end if
end if

The *Supergraph* method implemented in heu-kDA is improved. Unlike the original *Supergraph* that outputs either the proper k-degree anonymized G^* or `Unknown` [84], its version implemented in heu-kDA always returns a graph G^*. In case that d^*_G is realisable, than G^* is k-degree anonymous graph and $d_{G^*} = d^*_G$. In case d^*_G is unrealizable, then the outputted G^* is some supergraph of G but not k-degree anonymous. Certainly, $d_{G^*} \neq d^*_G$.

Hence, let G^* be a graph found by *Supergraph* on the input G. Let $\delta^*_{G,s}$ be the degree sequence returned by *Algorithm 5* and d_{G^*} be the degree sequences of G^*. I define

$$r_s = \sum_{i=1}^{n} d_{G^*}[i] - \sum_{i=1}^{n} \delta^*_{G,s}[i]$$

to be the residue after the run of *High-degree noise addition* with s and *Super-*

graph. If $r_s = 0$, then $\delta^*_{G,s}$ is realisable and $\delta^*_{G,s} = d_{G^*}$. Furthermore, set

$$r_0 = \sum_{i=1}^{n} d_{G^*}[i] - \sum_{i=1}^{n} d^*_G[i]$$

to be the residue after the first run of *Supergraph* that is not preceded by the run of the noise addition method.

Thus, the most suitable modification parameter is such a value of s on which the *Algorithm 5* outputs $\delta^*_{G,s}$ with $r_s = 0$. Since we deal with real networks with thousands of nodes, this condition is relaxed. The value of s is also considered suitable if $r_s = \pm 1$. It means that the degree of a single node is left greater or smaller by one than it should be according to $\delta^*_{G,s}$.

Note that sgn denotes the signum function in *Algorithm 6*. Moreover, the notation $C(s, d_G, d^*_G, k)$ stands for the run of *Algorithm 5* on the input s, d_G, d^*_G, k and $S(d^*_G)$ stands for the run of *Supergraph*. Surely, *Algorithm 5* ensures $\delta^*_{G,s}$ while *Supergraph* ensures G^* and d_{G^*}. The degree sequences $\delta^*_{G,s}, d_{G^*}$ can be used for the computation of r_s, hence the expression $r_s = C(s, d_G, d^*_G, k) + S(d^*_G)$ means running both algorithm sequentially producing their outputs and computing r_s afterwards.

The case that $s = 1$ corresponds to the most significant modification possible by *High-degree noise addition*. If $s > 1$, then the change is smaller. The modification parameter setting procedure is inspired by the root-finding with the bisection method. There exists a direct or indirect dependency between s and r_s. Depending on G and k, r_s grows or falls linearly with the growing s. Thus if r_s and $previous\_r$ have different signs, then $r_{s'} = 0$ lies in the interval between r_s and $previous\_r$ and the required s' lie between the value of s and the previous value of s.

Hence, the algorithm recurs and tries to find the s between the real numbers with one decimal digit between s and the previous s. It is possible to recur deeper into decimal numbers; hence the algorithm can additionally require setting the maximal possible depth of the recursion as the input parameter. The most suitable s was always found in real numbers with no or just one decimal digit in all performed experiments. The procedure always quit successfully for all tested datasets and all tested k values.

6.4.1 Complexity

The time complexity of *Algorithm 6* is determined by the complexity of *Algorithm 5* and *Supergraph* and the number of attempts needed for finding the proper s. Let the number of attempts needed to find the proper s be denoted by p. The complexity of *Algorithm 5* is in $\mathcal{O}(n)$, where $n = |d_G| = |d^*_G| = |V(G)|$, since each value of d^*_G is modified at most once. As presented in [84], the complexity of *Supergraph* is in $\mathcal{O}(a \cdot n)$, where $a = max_{i=1,\dots,n}|d^*_G[i] - d_G[i]|$. Thus, the whole complexity of *Algorithm 6* is in $\mathcal{O}(p \cdot a \cdot n)$.

6.5 Experimental results

In the experiments, the presented heu-kDA algorithm was examined. All experiments were performed on a Windows 10 operating system PC with 16 GB RAM

and a 3,2 GHz processor. The programs were written in MATLAB 9.6.0.1072779 (R2019a). The implementation is included in the attached CD, and the overview of corresponding MATLAB files is given in *Attachment B*. Data utility was measured using the SecGraph evaluation tool available at [59].

6.5.1 Tested datasets

The proposed algorithm was tested on the following real-world network datasets stored in SNAP library [74]: General Relativity and Quantum Cosmology collaboration network (GrQc)[78], Gnutella peer-to-peer network [78, 119], Wikipedia vote network [80], Caida AS Relationships Datasets [76], High-energy physics theory citation network (HepTh) [76, 37], Stanford web graph [79], Enron email network [79, 67], Amazon product co-purchasing network [77], Epinions social network [118] and the networks describing the social ties of geosocial networks Gowalla [27] and Brightkite [27]. Furthermore, experiments were also run on two smaller networks, Polbooks [69] and Polblogs [2], stored in the Network Data Repository [122]. The experiments were run on the network samples containing between 10^3 and 10^5 nodes. From the datasets containing more than 10^5 nodes (Stanford-web, Amazon, Gowalla), the subnetwork consisting of 10% of their nodes was tested.

6.5.2 Usability analysis

At first, the usability of the algorithm was demonstrated by running experiments on 12 real SN datasets differing in size (see *Table 6.1*). The algorithm was tested for values of the anonymization parameter $k = \{10, 20, 50, 100\}$ for each dataset. The algorithm found the most suitable s value for all datasets and all k values. For instance, the particular s values, total runtime, r_s values and the number of runs of the *Supergraph* algorithm for $k = 50$ are summarized in *Table 6.1*.

Dataset G	$\|V(G)\|$	#runs	s	r_s	runtime
Polblogs	1,224	3	2	1	12s
GrQc	5,242	2	1	0	4s
Gnutella	6,301	2	1	0	5s
Wiki-Vote	7,115	3	2	1	7min 41s
Gowalla	19,659	5	2.9	0	46min 29s
Caida	26,475	6	2.2	0	4min 25s
HepTh	27,770	3	2	1	22min 52s
Stanford-web	28,190	5	2.1	0	1min 4s
Email-Enron	36,692	3	2	0	10min 20s
Amazon	40,340	3	2	1	1min 53s
Brightkite	58,228	3	2	1	14min 35s
Epinions	75,879	4	3	0	1h 41min 49s

Table 6.1: Usability analysis of heu-kDA for $k = 50$. (Source: author's work [95].)

Since the anonymized degree sequence differs for various k in the same dataset, the suitable s differs for various k. Hence, it is not possible to set the same s for all runs of k-DA on the same dataset. The distribution of the s value over all datasets and all k values are shown in *Figure 6.2*. The most common value of s equalled 2. In 20% of computations, d_G^* needs no modification, and the resultant s is set to 0. However, $k \leq 20$ in all those cases. The d_G^* modification was always required for $k > 20$. In 17% of computations, the most suitable s was a decimal number, and the algorithm had to recur once. Although s equalled or grew with the growing k for 58% datasets, no dependency was found between parameters s and k, which would hold for all datasets.

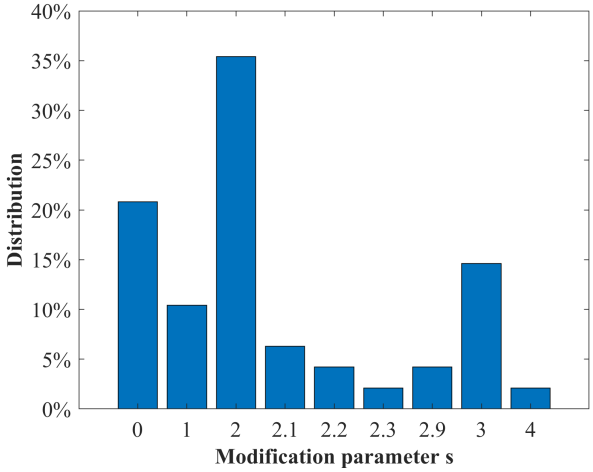


Figure 6.2: The distribution of the s values for all k and all datasets. (Source: author's work [95].)

The runtime of one algorithm run increased expectantly with the growing k and the growing number of nodes. In the worst case, the total computation took 8.17 hours (for Epinions and $k = 100$). The total runtime also depended on the number of necessary repetitions of the graph construction algorithm for different s. However, the graph construction algorithm has to be repeated only seven times in the worst case (for Gowalla and $k = 20$), and the average number of repetitions equalled 3.08. Hence, it shows the efficiency of the heuristic in the anonymized degree sequence anonymization for different datasets.

6.5.3 Information loss analysis

In this section, the results of anonymizing various networks with the proposed heu-kDA algorithm are compared with the results of three other algorithms: original Liu and Terzi k-degree anonymization algorithm (k-DA) [84], univariate micro-aggregation anonymization algorithm with neighbourhood centrality edge selection (NC)[20] and Fast k-degree anonymization algorithm (FkDA) [86].

Wiki-Vote and Email-Enron network. The data utility improvement is demonstrated on the results of experiments with Wiki-Vote and Email-Enron networks shown in *Table 6.2* where heu-kDA is compared with k-DA and FkDA. Unlike heu-kDA, both FkDA and k-DA consider only the edge addition operation

Wiki-Vote					
nED	$k=1$	10	20	50	100
heu-kDA	**0**%	**2**%	**9**%	**4**%	**15**%
FkDA		4%	12%	40%	80%
k-DA		8%	20%	64%	136%
$ACC(G)$	$k=1$	10	20	50	100
heu-kDA	**0.141**	**0.140**	**0.144**	**0.148**	**0.153**
FkDA		0.167	0.183	0.2	0.183
k-DA		0.367	0.467	0.559	0.651
$APL(G)$	$k=1$	10	20	50	100
heu-kDA	**3.203**	**3.202**	3.192	**3.173**	**3.173**
FkDA		3.24	**3.2**	3.12	3.04
k-DA		2.68	2.52	2.24	2.08
Email-Enron					
nED	$k=1$	10	20	50	100
heu-kDA	**0**%	**2**%	**3**%	**13**%	**8**%
FkDA		**2**%	4%	**13**%	27%
k-DA		2.5%	6%	20%	47%
$ACC(G)$	$k=1$	10	20	50	100
heu-kDA	**0.497**	0.486	0.485	0.470	0.465
FkDA		**0.5**	**0.5**	**0.506**	**0.5**
k-DA		0.513	0.613	0.625	0.625

Table 6.2: Wiki-Vote and Enron-Email network analysis. (Source: author's work [95].)

in *Supergraph*. The values of both algorithms are taken from [86]. The examined metrics were average distance $APL(G)$, the average clustering coefficient $ACC(G)$ and the normalized edit distance $nED = \frac{|E \cup E^*| - |E \cap E^*|}{|E|}$. The metric values for $k = 1$ correspond to the metric values of the original datasets.

Both metric values of heu-kDA were closer to the original values than FkDA and k-DA in Wiki-Vote dataset. Moreover, heu-kDA kept the normalized edit distance significantly lower than both other algorithms in both datasets. It is caused by the application of simultaneous edge addition and deletion in heu-kDA.

Polbooks, Polblogs, GrQc network. The structural metric values in Polbooks, Polblogs, and GrQc networks are compared as shown in *Table 6.3*. The values of k-DA are taken from [156] while the values of NC are taken from [20]. The examined structural metrics are the average distance $APL(G)$, transitivity $T(G)$ and the largest eigenvalue of the adjacency matrix λ. The metric values for $k = 1$ correspond to the metric values of the original datasets.

Polbooks										
λ	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**11.93**	**11.98**	12.03	**12.01**	12.70	11.75	**11.94**	12.21	**11.84**	14.20
NC		12.09	**11.97**	**11.85**	**11.85**	**11.95**	12.09	**12.08**	12.08	**11.86**
k-DA		12.00	12.05	12.11	12.22	12.30	12.31	12.64	12.72	12.85
$APL(G)$	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**3.079**	2.938	2.786	2.894	2.652	2.749	2.735	2.661	2.627	2.530
NC		**2.987**	**2.883**	**2.896**	**2.896**	**2.988**	**2.765**	**2.856**	**2.856**	**2.762**
$T(G)$	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**0.348**	0.345	0.332	0.337	0.332	0.314	0.318	0.299	0.294	0.322
NC		**0.350**	**0.342**	**0.339**	**0.339**	**0.347**	**0.326**	**0.322**	**0.322**	**0.324**
k-DA		0.330	0.330	0.320	0.330	0.300	0.310	0.320	0.290	0.300
Polblogs										
λ	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**74.08**	74.32	73.77	74.54	**74.18**	**74.13**	76.67	**74.10**	**74.33**	**74.22**
NC		**73.93**	**73.81**	**73.92**	73.95	73.74	**73.80**	73.75	73.63	73.61
k-DA		74.89	74.50	75.16	75.10	76.32	75.82	76.67	77.42	78.42
$APL(G)$	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**2.729**	2.706	2.713	2.719	**2.727**	2.716	2.671	2.718	2.715	2.713
NC		**2.733**	**2.729**	**2.725**	2.724	**2.724**	**2.732**	**2.726**	**2.731**	**2.727**
$T(G)$	$k=1$	2	3	4	5	6	7	8	9	10
heu-kDA	**0.226**	**0.225**	**0.224**	**0.227**	**0.226**	**0.225**	0.224	**0.224**	**0.224**	**0.224**
NC		0.224	0**.224**	0.224	0.224	0.223	**0.225**	**0.224**	0.223	**0.224**
k-DA		**0.225**	0.223	0.224	0.221	0.222	0.220	0.219	0.221	0.221
GrQc										
λ	$k=1$	5	10	15	20	25	30	35	40	50
heu-kDA	**45.62**	45.26	44.18	43.43	43.39	47.45	**44.88**	**45.13**	**44.24**	48.21
NC		**45.37**	**45.28**	**44.78**	**44.14**	**44.49**	43.02	43.72	43.55	**43.05**
$T(G)$	$k=1$	5	10	15	20	25	30	35	40	50
heu-kDA	**0.630**	0.621	0.612	0.598	**0.594**	0.554	0.567	0.576	0.562	0.515
NC		**0.625**	**0.617**	**0.611**	0.588	**0.595**	**0.589**	**0.589**	**0.578**	**0.584**

Table 6.3: Polbooks, Polblogs and GrQc network analysis. (Source: author's work [95].)

In 86% computations, heu-kDA kept the values of λ and $T(G)$ closer to the values of the original networks than k-DA. The values of heu-kDA were even closer to the original ones than the NC values in a few cases, although the NC was proved to reduce the information loss much better than k-DA in [20]. Hence, the proposed heuristic improves k-DA in terms of data utility.

Caida				
$\|E\| - \|E^*\|$	$k = 10$	20	50	100
heu-kDA	-116	-661	-1919	-3223
NC	0	0	-9	-9
$\%mod$	$k = 10$	20	50	100
heu-kDA	11.14%	20.34%	30.48%	39.48%
NC	6.06%	11.65%	18.43%	25.81%
time	$k = 10$	20	50	100
heu-kDA	25s	1min 13s	4min 25s	5min 27s
NC	36s	1min 5s	1min 45s	1min 48s

Table 6.4: Caida network analysis. (Source: author's work [95].)

Caida network. Finally, the performance of heu-kDA is compared with the performance of NC on the larger network Caida with 26,475 nodes. The values of NC are taken from [20]. For each algorithm, the total runtime was measured, as well as the difference between the original and anonymized edge set $|E| - |E^*|$ and the percentage of modified edges $\%mod = 1 - \frac{|E \cap E^*|}{|E \cup E^*|}$. As shown in *Table 6.4*, NC is faster and modifies the network better than heu-kDA. Thus, although the proposed heuristic was shown to improve the k-DA significantly, heu-kDA would not achieve as good results as the recently proposed NC algorithm aiming at the anonymization of large networks.

6.5.4 Data utility measurement

Finally, the proposed heu-kDA is compared with the k-DA implemented in the SecGraph evaluation tool [60, 59] in preserving data utility metrics. As mentioned in previous chapters, SecGraph includes the implementation of several anonymization methods, and the k-degree algorithm proposed by Liu and Terzi is one of them. Furthermore, SecGraph was also used for the measurement as the external independent tool.

The methodology of this measurement is as follows. Three datasets differing in size (Polblogs, Wiki-Vote and Caida) were inputs to heu-kDA and k-DA together with the value of the anonymization parameter $k \in \{5, 10, 15, 20, 50\}$. For each algorithm and each parameter setting, SecGraph measures the utility metrics in the outputted anonymized graph and the original graph and returns the comparison of the metric values. How metrics are compared with SecGraph is explained in detail in *Section 4.2*. When SecGraph returns 1, it means that the value of the particular metric is the same in both the anonymized and original graph. The more similar the metrics in the anonymized and original graph, the better the data utility preservation. Hence, values close to 1 are preferable.

Results with $k = 15$ are given in *Table 6.5*. Due to the space limits, the complete results are contained in *Attachment A*. The proposed heu-kDA preserved most of the metrics better than k-DA. This outcome corresponds with the results presented in previous paragraphs and proves that heu-kDA preserves data utility

better than the original k-DA. Moreover, AS, HS and PR were preserved better with heu-kDA for all parameter settings. The heuristic noise addition procedure heavily affects only the RX metric since SecGraph returned zero each time the metric was computed. The reason why this particular metric is so modified will be studied in future research since it requires a deep understanding of the metric and its implementation in the SecGraph tool.

	Polblogs		Wiki-Vote		Caida	
k=15	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.849	**0.960**	0.834	**0.987**	0.491	**0.789**
BC	0.904	**0.908**	0.885	**0.910**	**0.838**	0.734
CC	0.999	**1.000**	0.999	**1.000**	0.996	**0.999**
CD	0.857	**0.888**	0.838	**0.868**	0.067	**0.549**
Deg.	0.977	**0.990**	0.965	**0.991**	0.847	**0.887**
ED	0.955	**0.956**	0.982	**1.004**	0.817	**0.985**
EV	0.957	**0.984**	0.977	**0.991**	0.807	**0.867**
HS	0.814	**0.984**	0.876	**0.993**	0.470	**0.676**
Infe.	**0.914**	0.886	**0.848**	0.805	0.700	**0.719**
JD	**0.232**	0.110	**0.318**	0.304	**0.487**	0.141
LCC	0.955	**0.984**	0.988	**0.998**	0.887	**0.938**
NC	0.998	**1.000**	**1.000**	**1.000**	0.999	**1.000**
PR	0.518	**0.996**	0.607	**0.945**	0.309	**0.803**
RX	**0.335**	0.000	**0.757**	0.000	**0.245**	0.000

Table 6.5: Heu-kDA: utility measurement for $k = 15$. (Source: author's work.)

6.6 Discussion

In this chapter, I presented the improvement of the noise addition procedure in the k-DA algorithm. The noise addition procedure slightly modified the degree sequence found by the greedy degree anonymization algorithm in case the sequence is unrealizable and the graph construction procedure is not able to make the k-degree anonymization graph. The original noise addition procedure just added random noise into the sequence and did not consider the features of input graphs.

The known property of all graphs representing SN datasets is their power-law degree distribution. The result of the power-law degree distribution is that there are few nodes with high degrees, and most have very low degrees. Depending on the value of k and the network structure, there may be fewer than k nodes with a high degree. In this case, those high values are combined with lower values in the degree anonymization procedure, and the high-degree values are significantly decreased. When the values in the Q_1 are increased with the proposed heuristic algorithm, the distance between the original high-degree values in the class and their values after degree anonymization is smaller, which causes fewer modifications that have to be done in the graph construction procedure. If the number of high-degree nodes is greater than k, a similar situation can happen in Q_2, Q_3

or the Q_i formed by the degree anonymization procedure. The flexible correction of medians in equivalence classes of d_G^* kept the anonymized values closer to the original degree of high-degree nodes. Hence, fewer edges have to be removed during the graph modification procedure, and d_G^* is more likely to be realizable.

More importantly, the procedure is flexible. The modification parameter s and the parameter of correction x are dependent on both d_G and d_G^*. The modification parameter is computed with respect to the whole degree sequences. All equivalence classes of d_G^* are modified with the same value of s, but the correction parameter x differs for each class. The parameter x depends on the class's highest and lowest values. Moreover, the correction is applied if the difference between the highest and lowest value is significant. Otherwise, the anonymized value of the class is the median.

To finally answer *Question 2*, let me summarize the effects of implementing the high-degree noise addition method on the behaviour of the whole k-DA algorithm. As proved by the experimental results presented above, the high-degree noise addition method and edge deletion operation improve preserving the data utility in the k-DA algorithm. Moreover, the proposed high-degree noise addition procedure improves k-DA in terms of speed too, since it makes the computation of k-DA more efficient and reduces the number of repetitions of the graph modification algorithm.

7. Hybrid algorithm for k-automorphism anonymization

The k-automorphism anonymization approach protects against any structural attack on SN datasets. However, providing a high level of privacy protection entails extensive modifications in the graph structure during the anonymization procedure. Due to the significant information loss, the k-automorphism methods have not been widely developed. The recent research in edge editing SN anonymization methods focused primarily on k-degree and k-neighbourhood methods.

In this chapter, *Question 3* is addressed by proposing the Hybrid Algorithm for k-Authomorphism anonymization (HAkAu). The k-automorphism is demonstrated as a competitive SN anonymization approach providing a high-security level. HAkAu is based on the structure of the KM algorithm proposed in [163]. It combines the original approach with the genetic algorithm (GA). This chapter is based on [97].

7.1 Motivation

The privacy-preserving problem addressed in this chapter is the identity disclosure problem in SN datasets. I recall the problem stated in *Chapter 2*. The provider desires to share or publish the data. Thus, he or she released the graph $\tilde{G}$. Assume that $\tilde{G}$ is an anonymized version of the input graph G, hence $\tilde{G} = G^*$. The identity disclosure occurs if an attacker can identify the target individual in the released dataset G^*. In other words, the identity is disclosed if the attacker can link $v \in V(G^*)$ with the particular individual represented with v.

The k-automorphism concept defends against all kinds of structural attacks. For every vertex $v \in V(G^*)$, there are at least other $k-1$ vertices with the same l-neighbourhood in G^* for any $l \in \mathcal{N}$. Therefore, the result of any structural query $\mathcal{Q}$ on G^* contains at least k vertices. The attacker possessing any structural background knowledge can not identify his or her target node with a higher probability than $\frac{1}{k}$ in the k-automorphic graph G^*.

The motivation for applying GA arises from the fact that the k-automorphism anonymization problem is proven to be NP-hard, and GA is a powerful tool for solving search-based optimization NP-hard problems. The novel chromosome presentation proposed in this chapter enables us to solve the problem of finding the k-automorphism graph as an optimization problem of minimizing information loss. Moreover, the solution space can be modelled very well, and the particular solutions are easily comparable with respect to the introduced fitness function. Thus, GA is a good fit for finding the optimal solution to such a problem. Moreover, GA has been successfully used to improve the k-degree anonymization method [116], k-neighborhood method [3] and clustering k-anonymization methods [132, 155, 130].

7.2 Theoretical part

Before the proposed HAkAu is introduced, some theoretical aspects of the proposed solution are presented.

7.2.1 Anonymization cost

The main tools of HAkAu are edge editing operations. The input edge set $E(G)$ is modified with both edge addition and edge deletion operations; hence the original edge set $E(G)$ is not the subset of $E(G^*)$. The information loss caused by the anonymization is called the *anonymization cost*. In the edge editing anonymization methods, the anonymization cost corresponds to the number of edge edits made during the anonymization process. Since making a graph k-automorphic also requires node edits in the input graph, the total anonymization cost of HAkAu equals the sum of edge edits and node edits.

The number of node edits depends on $|V(G)|$. If G^* is k-automorphic, then for each node v, there exist $k-1$ nodes that are isomorphic to v. Hence, $|V(G^*)|$ is divisible by k. If $|V(G)|$ is not divisible by k, adding or removing some vertices is necessary. Let us denote $z := mod(|V(G)|, k)$. In case $z \leq \frac{k}{2}$, then z vertices are removed by HAkAu, otherwise, $k - z$ dummy vertices are added. Thus, the number of node edits, denoted by $VCost(G, G^*)$, is equal to or less than $\frac{k}{2}$.

Edge edits have a more significant impact on the anonymization cost. HAkAu uses both edge-deleting and edge-adding operations. Edge modifications are applied in two parts of HAkAu: when chosen subgraphs of G are extended by GA and in the adding crossing edges procedure. Hence, the total anonymization cost caused by modifying G to G^* with HAkAu is

$$Cost(G, G^*) = VCost(G, G^*) + ExCost(G, G^*) + CECost(G, G^*)$$

where $CECost(G, G^*)$ is the number of edge edits made in the adding crossing edges procedure, and $ExCost(G, G^*)$ is the number of edge edits made by GA. Both costs, $ExCost(G, G^*)$ and $CECost(G, G^*)$, are computed in *Section 7.3*, after HAkAu is explained.

7.2.2 NP-hard problems

The issue addressed in this chapter contains NP-hard problems. At first, I outline the problem of finding a frequent subgraph with the given minimal support. The problem is NP-hard since it relies on the NP-hard subgraph isomorphic problem, as proved in [31]. The problem is addressed in both KM and HAkAu with the external algorithm. KM employs SiGraM [72] while HAkAu uses a more efficient GraMi algorithm [31].

Problem 1. *Let G be a graph and $s \in \mathcal{N}$. Find the frequent subgraph of G with the minimal support s and its matches in G.*

Now, I recapitulate the problems solved with the KM algorithm since they play a role in further proof.

Problem 2 (Finding optimal graph partitioning)**.** *Let G be a graph. Find the sets $U_1, \ldots, U_m$, where $U_i = \{\overline{P_{i1}}, \ldots, \overline{P_{ik}}, \overline{P_{ij}} \subseteq G, j = 1, \ldots, k\}$ for each $i = 1, \ldots, m$ and such that*

- *$G = \bigcap_{i=1}^{m} U_i$*
- *$Cost(G, G^*)$ is minimal, where G^* is the k-automorphic version of G obtained by KM algorithm*

Problem 2 is NP-complete as proved in [163]. *Problem 2* is addressed in KM algorithm as follows. KM algorithm builds the sets U_i sequentially. For each i, it first finds the frequent subgraph of G with the minimal support k and its matches in G, denoted by $P_{i,1}, \ldots, P_{i,k}$. Hence, P_{ij}'s are isomorphic to each other. Then graphs $P_{i,1}, \ldots, P_{i,k}$ are extended to graphs $\overline{P_{i,j}}$, which are subgraphs of G but are not isomorphic to each other any more. The extension is made so that $Cost(G, G^*)$ is minimal. Then $E(G) := E(G) \setminus \bigcup_{j=1}^{k} E(\overline{P_{i,j}})$ and the procedure is repeated with $i := i + 1$ until G is completely partitioned into sets U_i.

Then, for each set U_i, some edges are added into $\overline{P_{i,j}}$ to create new graphs $\overline{P'_{ij}}$ that are isomorphic to each other again. The edge addition procedure requires minimizing $Cost(G, G^*)$. The alignment vertex table defined in [163] described how vertices are mapped to each other under the isomorphism. The issue of finding the optimal $\overline{P'_{i1}}, \ldots, \overline{P'_{ik}}$ is proven to be NP-hard in [163] and is formalized in the following problem. The resultant $\overline{P'_{ij}}$ is the basis of the final anonymized graph G^*.

Problem 3 (Finding optimal graph alignment)**.** *For each U_i from* Problem 2*, $i = 1, \ldots, m$, find graphs $\overline{P'_{i1}}, \ldots, \overline{P'_{ik}}$ such that $\overline{P_{ij}} \subseteq \overline{P'_{ij}}$ and for each $j = 1, \ldots, k$ and $Cost(G, G^*)$ is minimal.*

Now, I state the problems that are addressed by HAkAu and prove their NP-hardness. Methods for solving them are proposed in the next sections, where HAkAu is described in detail. The following problem is the core of finding k-automorphic graph G^* for the given G. It is the combination of *Problem 2,3*. Naturally, G^* is not known before the anonymization procedure, and the real goal is to find G^* such that $Cost(G, G^*)$ is minimal.

Problem 4. *Let G be a graph and $P_{i,1}, \ldots, P_{i,k}$ be subgraphs of G such that $P_{i,j}$ is isomorphic to $P_{i,l}$, $\forall j, l := 1, \ldots, k$. For $P_{i,1}, \ldots, P_{i,k}$ find graphs $P'_{i,1}, \ldots, P'_{i,k}$ and isomorphisms $F_{i,j}$ such that*

- *$P'_{i,j} \subset G^*$*
- *$P'_{i,j}$ is the supergraph of $P_{i,j}$*
- *$F_{i,j}(P'_{i,j}) = P'_{i,j+1},\ j = 1, \ldots, k-1,$*
- *$F_{i,k}(P'_{i,k}) = P'_{i,1}$*
- *$Cost(G, G^*)$ is minimal*

where G^* is a k-automorphism graph obtained by HAkAu.

Problem 4 repeatedly arises in HAkAu, and the index i denotes the number of its repetitions. The repetitions correspond to finding U_i defined in *Problem 2*. Again, the graphs $P_{i,1}, \ldots, P_{i,k}$ are the matches of the frequent subgraph with the minimal support k in G. Unlike KM algorithm, HAkAu expands the subgraphs $P_{i,1}, \ldots, P_{i,k}$ "isomorphically". Thus the found graphs $P'_{i,j}$ are still isomorphic to each other, and it is not necessary to make them isomorphic in the next step, which is the main advantage of this approach.

Theorem 5. Problem 4 *is NP-hard.*

Proof. According to *Lemma 4*, to prove the NP-hardness, it is enough to reduce an NP-hard problem to Problem 4. In this proof, *Problem 3* is reduced to *Problem 4*. Denote $P_{i,1}, \ldots, P_{i,k}$ to be the matches of the frequent subgraph with the support k. Then set $U_i := \{P_{i,j};\ j := 1, \ldots, k\}$ before the extension is applied. After all $P'_{i,j}$ and $F_{i,j}$ are found for each i, j with HAkAu, then subgraphs $\overline{P'_{i,j}}$ requested in *Problem 3* are $\overline{P'_{i,j}} := P'_{i,j}$. The graphs $\overline{P'_{i,j}}$ are isomorphic. Moreover, the isomorphisms $F_{i,j}$ give the alignment vertex table. □

At the end of this section, I give the problem addressed in the preprocessing stage of HAkAu. In the preprocessing stage, the set of vertex-disjoint graphs is selected from the set of edge-disjoint graphs, and the resultant set becomes the input to the genetic algorithm.

Problem 5. *Let G be a graph and M be a set of subgraphs of G. Select $S \subseteq M$ such that $\forall P_1, P_2 \in S : |V(P_1)| \cap |V(P_2)| = \emptyset$.*

Theorem 6. Problem 5 *is NP-hard.*

Proof. Since all subgraphs in M are isomorphic, they have the same number of vertices. Let the matrix $\mathbf{M}$ be defined such that the i-th row of $\mathbf{M}$ is the list of vertices of the i-th subgraph of M. The selection of the subset S corresponds to the selection of the set of rows in $\mathbf{M}$ that do not contain any identical number. The selected rows represent the isomorphic subgraphs with the mutually vertex-disjoint set of vertices.

Now let me show that the problem of selecting matrix rows that do not contain identical numbers equals the maximum independent set problem. This problem is known to be NP-hard [36]. Let us suppose that the individual rows in $\mathbf{M}$ will be represented by vertices of a completely new graph (without any assumption of what kind of graph is represented by the specific row). Assume adding edges in the newly created graph so there will be an edge between two vertices when the intersection between the respective rows is non-empty. This simple transformation changed the problem of selecting matrix rows that do not contain identical numbers into the maximum independent set problem. Hence, the problem is NP-hard, and *Problem 5* is NP-hard, as well. □

7.3 HAkAu algorithm

In this section, the novel Hybrid Algorithm for k-Automorphism anonymization is described in detail. The proposed HAkAu modifies the graph G representing the

given SN to the anonymized k-automorphism graph G^*. The final G^* is resistant to any structural attack.

The crucial idea of the approach is as follows. The k isomorphic subgraphs are found in G. They are isomorphically extended to minimize $Cost(G, G^*)$. Then the extended isomorphic subgraphs are removed from the input graph, and the process is rerun on the smaller graph. After the whole input graph is processed, we get the set of disconnected graphs such that for every graph, there are at least $k-1$ other graphs that are isomorphic to it. The disconnected graphs are linked together, making the final graph k-automorphic.

The detailed description of HAkAu is given in *Algorithm 7*. The graph H is the rest of the input graph after the i-th round of the *while* cycle (see line 2). While there are at least k vertices in H, H is partitioned. GraMi algorithm is run to find the frequent subgraph $g_f(s)$ and s matches of $g_f(s)$ in H (see line 3). If GraMi finds more than one frequent subgraph, $g_f(s)$ is the largest. The set of subgraphs matching $g_f(s)$ in H is denoted by M. After selecting k vertex-disjoint subgraphs P_{ij} from M (see line 4), the GA is run to find isomorphic supergraphs P'_{ij} (see line 5). The proposed GA is used to tackle the NP-hard *Problem 4*. The graphs P'_{ij} are removed from H (see lines 8 and 9) and the crossing edges between P'_{ij} and H are added into the set of crossing edges C (see line 7). When $|V(H)| < k$, the edges remaining in H are added into C, and the remaining vertices become the last found isomorphic subgraphs (see lines 14-20). All found subgraphs P'_{ij} are the core of the anonymized graph G^*. To make G^* connected, selected crossing edges and their copies are isomorphically added into G^* (see line 24). The adding crossing edges procedure is detailed in *Section 7.3.2*.

Note the following interesting aspects of the algorithm. In the first round of the *while* cycle GA finds k isomorphic graphs $P'_{11}, \ldots, P'_{1k}$. The set of k isomorphic graphs determines k isomorphisms $F_{11}, \ldots, F_{1k}$ as shown in line 5. Since F_j is a zero homomorphism at the beginning, $F_j := F_{1j}$ after the first round. After the second round, F_j is the extension of F_{1j} and F_{2j} and it is still the isomorphism, since $V(P'_{1j}) \cap V(P'_{2j}) = \emptyset$, $\forall j = 1, \ldots, k$. Similarly, after the i-th round F_j is extended with F_{ij} (see line 10). After the *while* cycle ends, there are $m \cdot k$ graphs P'_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, k$, and k isomorphisms F_j:

- $F_j(P'_{i,j}) = P'_{i,j+1}, \quad i = 1, \ldots, m\,, j = 1, \ldots, k-1,$
- $F_k(P'_{i,k}) = P'_{i,1}, \qquad i = 1 \ldots, m$

Secondly, note that P_{ij} are subgraphs of H, but P'_{ij} are not subgraphs of H. Naturally, $V(P_{ij}) \subseteq V(P'_{ij})$ and $E(P_{ij}) \subseteq E(P'_{ij})$, since P_{ij} are supergraphs of P_{ij}. Moreover, $V(P'_{ij}) \subseteq V(H)$, but $E(P'_{ij}) \not\subseteq E(H)$.[1] Some edges from H are preserved in P'_{ij}, some new edges can be added between nodes of $V(P'_{ij})$ by GA, some edges that were between some nodes of $V(P'_{ij})$ in H do not exist in P'_{ij}.

Thirdly, GA aims to set up graphs P'_{ij} so that they are isomorphic to each other and $Cost(G, G^*)$ is minimal. Putting back a single crossing edge caused adding other $k-1$ edges that are isomorphic to the graph. By finding larger graphs, P'_{ij} and replacing them with smaller P_{ij}, the amount of crossing edges is reduced; hence $CECost(G, G^*)$ is significantly reduced.

[1]The statement $V(P'_{ij}) \subseteq V(H)$ simplifies slightly the real situation. In some cases, a few new nodes have to be added, and for some $j : V(P'_{ij}) \not\subseteq V(H)$ (see *Section 7.4.2)*

Algorithm 7 HAkAu algorithm

Require: anonymization parameter k, input network G, minimal support s
Ensure: k-automorphism network G^*

Set $i := 1$, $H := G$, $C := \emptyset$, $M := \emptyset$, $S := \emptyset$ and G^* to be an empty graph. Set $F_j := 0$, $j := 1, \ldots, k$.
while $|V(H)| \geq k$ **do**
    Run GraMi on (H, s) to find the frequent subgraph $g_f(s)$ and the set $M := \{I_1, \ldots, I_s;\ I_j \text{ is a match of } g_f(s) \text{ in } H, j := 1, \ldots, s\}$.
    Run *Algorithm 8* on (M, s, k) to find the set of k vertex-disjoint matches $S := \{P_{ij} \in M; V(P_{ij}) \cap V(P_{il}) = \emptyset,\ j \neq l,\ j, l = 1, \ldots, k\}$ (see *Section 7.3.1*)
    Run GA on (S, H, k) to find graphs P'_{ij} and isomorphism F_{ij} such that
    - P'_{ij} is the supergraph of P_{ij}
    - $F_{ij}(P'_{i,j}) = P'_{i,j+1},\ j = 1, \ldots, k-1,$
    - $F_{ik}(P'_{i,k}) = P'_{i,1}$
    - $Cost(G, G^*)$ is minimal

    (see *Sections 7.4*)
    $C_i := \{(u, v) \in H : \exists j \in \{1, \ldots, k\} : u \in V(P'_{ij}) \wedge v \notin V(P'_{ij})\}$
    $C := C \cup C_i$
    $V(H) := V(H) \setminus \bigcup_{j=1}^{k} (V(P'_{ij}) \cap V(H))$
    $E(H) := E(H) \setminus (C \cup \bigcup_{j=1}^{k} (E(P'_{ij}) \cap E(H)))$
    $F_j := F_j \oplus F_{ij}\ j := 1, \ldots, k$
    $i := i + 1$.
end while
$m := i - 1$
$C := C \cup E(H)$
if $|V(H)| \geq \frac{k}{2}$ **then**
    $m := m + 1$
    Add $k - |V(H)|$ dummy edges in $V(H)$.
    for $j := 1, \ldots, k$ **do**
        Select $v \in V(H)$ and set $P'_{mj} := v$
        $V(H) := V(H) \setminus \{v\}$
    end for
end if
$G^* := \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} P'_{ij}$
Run *Algorithm 9* on (C, G, G^*, k, F_j) to add selected crossing edges and their isomorphic copies in G^* (see *Section 7.3.2*).
Return G^*.

Finally, the HAkAu algorithm requires the input dataset and two parameters: the anonymization parameter k and the minimal support parameter s. The anonymization parameter is the independent parameter corresponding to the required anonymization level, and the minimal support s is the dependent parameter. It holds that $s > k$ since k vertex-disjoint subgraphs are selected from the set of s edge-disjoint subgraphs.

7.3.1 Finding the subset of vertex-disjoint subgraphs

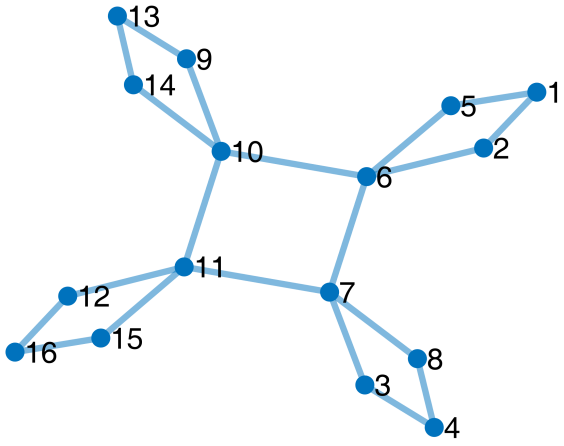
GraMi algorithm has been utilized to find the largest frequent subgraph $g_f(s)$ and its matches in H. Let denote

$$M := \{I_1, \ldots, I_s;\ I_j \text{ is a match of } g_f(s) \text{ in } H, j := 1, \ldots, s\}.$$

Naturally, there might be partially overlapping subgraphs in M having one or more common vertices. To proceed further, it is necessary to find k mutually vertex-disjoint subgraphs in M. The task is formulated in *Problem 5* and is proved to be NP-hard in *Theorem 6*.

The output of the GraMi algorithm is in tabular form with s rows, where each row represents one subgraph I_j as the list of its vertices. Since $I_1, \ldots, I_s$ are isomorphic, $|V(I_1)| = \cdots = |V(I_s)|$. The set of the rows can be represented by a matrix $\mathbf{M}$ with s rows and $|V(I_1)|$ columns.

To illustrate the situation, a simple graph G, the GraMi output on G and the matrix M compiled from the GraMi output are given in *Figure 7.1*. On G, there can be easily identified five four-squared subgraphs I_1, I_2, I_3, I_4, I_5 that are the matches of the found frequent graph. It is visible on *Figure 7.1* that the $I_1, \ldots, I_5$ are edge-disjoint, but not vertex-disjoint. This fact is reflected on the GraMi output so that the same number (label of the relevant vertex) is included in two or more rows. For example, vertex 6 can be found on the first and third rows.



GraMi output:
$I_1 = (1, 2, 5, 6)$
$I_2 = (3, 4, 7, 8)$
$I_3 = (\mathbf{6}, \mathbf{7}, \mathbf{10}, \mathbf{11})$
$I_4 = (9, 10, 13, 14)$
$I_5 = (11, 12, 15, 16)$

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ \mathbf{6} & \mathbf{7} & \mathbf{10} & \mathbf{11} \\ 9 & 10 & 13 & 14 \\ 11 & 12 & 15 & 16 \end{pmatrix}$$

Figure 7.1: Simple graph G, the GraMi output on G and the corresponding matrix $\mathbf{M}$. (Source: author's work.)

The task to select the largest subset of mutually disjunct subgraphs is equal to the task of selecting the maximal number of rows of matrix $\mathbf{M}$ that do not contain any identical number, as shown in the proof of *Theorem 6*. Considering the graph at *Figure 7.1*, we can either select the third row (inner four-squared subgraph), and there is nothing more that could be added there. The second option is to select four "outside" four-squared subgraphs (rows 1,2,4 and 5 from M), and this attempt will give us the optimal solution comprising of four vertex-disjoint subgraphs I_1, I_2, I_4, I_5.

This way, the problem of finding the subset of vertex-disjoint graphs was transformed into the issue of selecting matrix rows that do not contain any iden-

tical number. As proved in *Theorem 6*, this problem is NP-hard. Therefore, a naive approach based on a brute-force algorithm that successively compares each row with another one is impractical, and a heuristic approach is needed to speed up the search.

The proposed heuristic algorithm, described in *Algorithm 8*, is based on the "divide and conquer" design paradigm where the frequency of the individual vertices (numbers) in matrix $\mathbf{M}$ is utilized to break down the main task into two subtasks. In the first subtask, the rows of $\mathbf{M}$ containing the most frequent vertices are step by step removed and stored in auxiliary matrix $\mathbf{R}$ (see lines 12 and 13 in *Algorithm 8*). Therefore, the first step of the algorithm is to calculate the frequency of the numbers included in matrix $\mathbf{M}$ (see line 4).

Algorithm 8 Finding the subset of mutually vertex-disjoint subgraphs

Require: the set of s isomorphic subgraphs $M = \{I_1, \ldots, I_s\}$, the anonymization parameter k

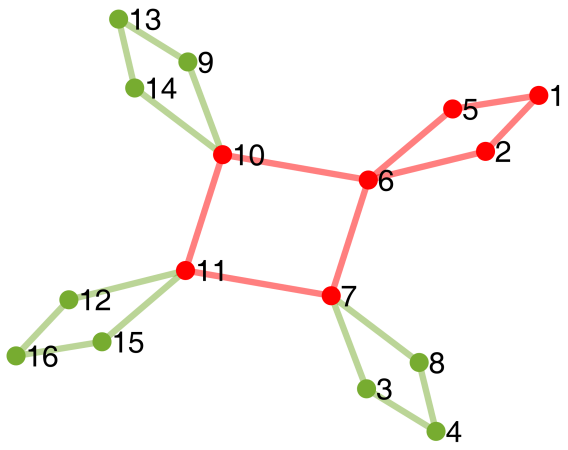
Ensure: the set of k vertex-disjoint isomorphic subgraphs S

1: Set $\{v_1^j, \ldots, v_n^j\} := V(I_j)$ and $\mathbf{M} := \{v_i^j\}_{i,j}$ to be a matrix, $i := 1, \ldots, |V(I_1)|$, $j := 1, \ldots, s$.
2: $\mathbf{R} := \emptyset$, $\mathbf{S} := \emptyset$, $cont := 1$.
3: **while** $rc(\mathbf{M}) > 1 \wedge cont = 1$ **do**
4: Calculate the frequency of all vertices in $\mathbf{M}$.
5: **if** the frequency of all vertices equals 1 **then**
6: $\mathbf{S} := \mathbf{M}$
7: $cont := 0$
8: **else**
9: Set v to be the vertex with the highest frequency of occurrence in $\mathbf{M}$.
10: **for** $j := 1, \ldots, rc(M)$ **do**
11: **if** $v \in r_j(M)$ **then**
12: $\mathbf{M} := \mathbf{M} \setminus r_j(\mathbf{M})$
13: $\mathbf{R} := \mathbf{R} \cup r_j(\mathbf{M})$
14: **end if**
15: **end for**
16: **if** $rc(\mathbf{M}) = 1$ **then**
17: $\mathbf{S} := \mathbf{M}$.
18: **end if**
19: **end if**
20: **end while**
21: **repeat**
22: Calculate the frequency of vertices contained in $\mathbf{S}$.
23: Find $r := r(\mathbf{R})$ such that $\forall v \in r(\mathbf{R})$ frequency of v in $\mathbf{S}$ equals 0.
24: $\mathbf{R} := \mathbf{R} \setminus r$
25: $\mathbf{S} := \mathbf{S} \cup r$
26: **until** $r = \emptyset$ or $rc(\mathbf{S}) = k$
27: Return $S := \{I \in M; \exists i : r_i(S) = V(I)\}$

Let n rows in matrix $\mathbf{M}$ and c be the highest frequency found in $\mathbf{M}$. Then it is evident that the maximal number of mutually disjunct rows will be less or equal to $n - c + 1$. In the example shown in *Figure 7.1*, the maximal number of

mutually disjunct rows is computed with $6-2+1=4$. For instance, if the vertex number 6 has two occurrences and only mutually disjunct rows are required, then the only option is either the row $(1,2,5,6)$ or the row $(6,7,10,11)$ can be a part of the solution. Hence, the task can be divided into two subtasks:

1. pick up one row amongst the two we have just identified
2. remove the rows considered in the previous step from $\mathbf{M}$ and solve the remaining task represented by the smaller $\mathbf{M}$ (see *Figure 7.2*))



Select one row from
$(1,2,5,6)$
$(6,7,10,11)$

Solve the subtask

$$\mathbf{M} = \begin{pmatrix} 3 & 4 & 7 & 8 \\ 9 & 10 & 13 & 14 \\ 11 & 12 & 15 & 16 \end{pmatrix}$$

Figure 7.2: Dividing the main task into subtasks. (Source: author's work.)

The same procedure will be applied to the smaller matrix $\mathbf{M}$: calculate the frequencies, select the vertex with the highest frequency or one of those in the case of multiple vertices with the same frequency, and divide the table into two. At each iteration of the algorithm, at least two rows will be removed from the matrix, and at the end, all the rows in the matrix $\mathbf{M}$ will be disjunct. The remaining rows of $\mathbf{M}$ are set to be the cornerstone of output matrix $\mathbf{S}$. There are three possible options:

1. the final matrix $\mathbf{M}$ is empty (no row)
2. the final matrix $\mathbf{M}$ contains only one row (see line 16)
3. the frequency of all the vertices involved is equal to 1 (see line 5)

In the first case, $\mathbf{S} = \emptyset$, and in the second and the third cases, $\mathbf{S} := M$. Our illustrative example has reached the third option, as all three rows are vertex-disjoint.

The second phase of the algorithm (beginning at line 21) takes the matrix $\mathbf{S}$ as the basis of the solution, and going backwards, the algorithm tries to add to it one row from the previously deleted ones that are stored in the auxiliary matrix $\mathbf{R}$. Clearly, it is possible to add one row maximally; otherwise, the disjunctivity would be compromised. The task is done relatively efficiently. There is no need to compare the added row with each one already in the solution matrix $\mathbf{S}$. A more efficient process is to compute the frequency of vertices contained in $\mathbf{S}$ (see

line 22) and check whether the frequency of all vertices added by the relevant row is equal to 0 (see line 23). If this condition is met for some row, the row should be added to the solution matrix (see line 25). Then, the frequency table is updated (see line 22), and the procedure moves one step backwards. The process terminates if it can find no such a row in $\mathbf{R}$, or $\mathbf{S}$ have k rows (see line 26).

In the illustrative case, adding the second row to the partial solution matrix is impossible as vertices 7,10, and 11 are already included there. On the contrary, the first row $(1, 2, 5, 6)$ can be added, as the frequency of all these vertices equals zero. Hence, there exists the solution consisting of four independent rows in the matrix representing four subgraphs.

Algorithm 8 solves the problem of finding the subset of vertex-disjoint subgraphs directly. Hence, it requires the set of isomorphic subgraphs, which is transformed to the matrix $\mathbf{M}$ (see line 1), and delivers a subset of mutually disjunct subgraphs which is built from the final matrix S (see line 27). There is no guarantee that the optimal solution (the largest subset) has been found, but the algorithm is fast ($\mathcal{O}(N^3)$) and found the acceptable solution in all executed experiments.

7.3.2 Adding crossing edges

Adding crossing edges procedure, including the computation of $CECost(G, G^*)$, is described in detail in *Algorithm 9*.

Algorithm 9 Adding crossing edges

Require: the set of crossing edges C, input network G, anonymized network G^*, anonymized parameter k, isomorphisms F_j $(j := 1, \ldots, k)$
Ensure: k-automorphism network G^*, $CECost(G, G^*)$

1: $CECost(G, G^*) := 0$
2: **while** $C \neq \emptyset$ **do**
3: Select (v, w) from C.
4: Set $\tilde{C} := \{(F_j(v), F_j(w)) \in C\}$.
5: **if** $|\tilde{C}| \geq \frac{k}{2} - 1$ **or** $(deg_{G^*}(v) < deg_G(v)$ **or** $deg_{G^*}(w) < deg_G(w))$ **then**
6: $E(G^*) := E(G^*) \cup \{(F_j(v), F_j(w)),\ j := 1, \ldots, k\}$
7: $C := C \setminus \{\tilde{C} \cup (v, w)\}$
8: $CECost(G, G^*) := CECost(G, G^*) + k - 1 - |\tilde{C}|$
9: **else**
10: $C := C \setminus \{(v, w)\}$
11: $CECost(G, G^*) := CECost(G, G^*) + 1$
12: **end if**
13: **end while**
14: Return G^*, $CECost(G, G^*)$.

The crossing edges are edges stored in the set C. Note that a crossing edge (v, w) is the edge that connects vertices from different subgraphs, $v \in V(P'_{ij})$, $w \in V(P'_{ab})$, where $i \neq a$ or $j \neq b$. If (v, w) is added into G^*, then it is necessary to add other $k - 1$ edges $(F_j(v), F_j(w))$ into G^*, $j = 1, \ldots, k - 1$, to keep G^* k-automorphic.

The crossing edge (v, w) is added into G^* if there are at least $\frac{k}{2} - 1$ other crossing edges in C that are isomorphic to (v, w). Hence, adding (v, w) into G^* causes fewer edge edits than not adding it. Except that, HAkAu adds crossing edges that are significant for matching the structure of G^* to the structure of G. Since at most k operations are done with each crossing edge, and each crossing edge is considered once, the complexity of *Algorithm 9* is in $\mathcal{O}(|C| \cdot k)$.

7.3.3 Computing the extension cost

The extension cost $ExCost(G, G^*)$ is the anonymization cost caused by replacing P_{ij} with P'_{ij}, $i := 1, \ldots, k$, $j := 1, \ldots, m$. Let $ExCost_i(H)$ denote the extension cost caused in the i-th round of *while* cycle in *Algorithm 7* where H is processed. Then

$$ExCost(G, G^*) := \sum_{i=1}^{m} ExCost_i(H)$$

$$ExCost_i(H) := \sum_{j=1}^{k} ExCost(P_{ij}, P'_{ij})$$

where $ExCost(P_{ij}, P'_{ij})$ is the anonymization cost caused by extending P_{ij} to P'_{ij}, i, j fixed. The cost $ExCost(P_{ij}, P'_{ij})$ equals to the number of edges that exist in P'_{ij} and not exist in G plus the number of edges $(u, v) \in E(G)$ such that $u, v \in V(P'_{ij})$ and $(u, v) \notin E(P'_{ij})$. More precisely,

$$\begin{aligned} ExCost(P_{ij}, P'_{ij}) := {} & |E(P'_{ij}) \setminus (E(Pij') \cap E(G))| \\ & + |\{(u, v) \in E(G);\ u, v \in V(P'_{ij}) \wedge (u, v) \notin E(P'_{ij})\}| \end{aligned}$$

7.3.4 The comparison with the design of KM algorithm

Before the detailed description of the genetic algorithm, I summarize the differences between KM and HAkAu and enhance the improvements in HAkAu. As mentioned in *Section 7.2*, HAkAu solves the NP-hard problem, combining two problems addressed in the KM algorithm. As a result, KM finds the extended graphs that are not isomorphic by solving *Problem 2* and make them isomorphic again by solving *Problem 3*. Contrarily, HAkAu extended the graphs "isomorphically" by solving *Problem 4*.

Furthermore, the KM algorithm uses only edge addition operation. HAkAu employs edge addition and edge deletion operations, leading to better preservation of the degree distribution, which is documented with experimental results in *Section 7.5.2*. It also leads to the improvement of the procedure of adding crossing edges. The KM algorithm puts back all crossing edges and all their copies. On the other hand, HAkAu adds a crossing edge and its copies only when adding them causes fewer edge edits than not, or the crossing edge was significant in the structure of G. Thus, HAkAu adds fewer edges in the adding crossing edges procedure than the KM algorithm.

Both algorithms need to find the frequent subgraph with the given minimal support several times. The KM algorithm runs the SiGraM algorithm published [72]. The SiGraM algorithm is based on the grow-and-store method. It stores each examined subgraph's appearance, which requires much space and computational time.

HAkAu employs a recent GraMi algorithm proposed in [31]. GraMi stores only the templates of the frequent subgraphs, not the whole subgraphs, and models the frequency evaluation as the constraint satisfaction problem. The GraMi algorithm is proven to be faster than the SiGraM algorithm in [31].

7.4 Genetic algorithm

I use the model of the genetic algorithm described in *Section 4.7*. In this section, I describe the chromosome representation, fitness and selection function, and how genetic operators are applied to the used representation.

7.4.1 Chromosome representation

In this section, the novel chromosome representation used in GA is introduced. The goal of GA is to find k graphs P'_{ij}, $j := 1, \ldots, k$, i fixed, that are isomorphic to each other (line 5 in *Algorithm 7*). Therefore, in the rest of this section, the index i is fixed, and indices $j, l := 1, \ldots, k$.

Each individual in GA represents one solution; hence each individual represents all graphs P'_{ij}. For all j, the graph P'_{ij} is the supergraph of P_{ij}. I denote $Q_{ij} = P'_{ij} \setminus P_{ij}$ to be the subgraph of P'_{ij} with $V(Q_{ij}) = V(P'_{ij})$ and $E(Q_{ij}) = E(P'_{ij}) \setminus E(P_{ij})$ (see *Figure 7.3*). The graphs P_{ij} are found with *Algorithm 8*, and they do not change during the run of GA. Thus, GA aims to find the optimal Q_{ij} for each j. Since $P'_{ij} \backsimeq P'_{il}$ and $P_{ij} \backsimeq P_{il}$, then $Q_{ij} \backsimeq Q_{il}$.

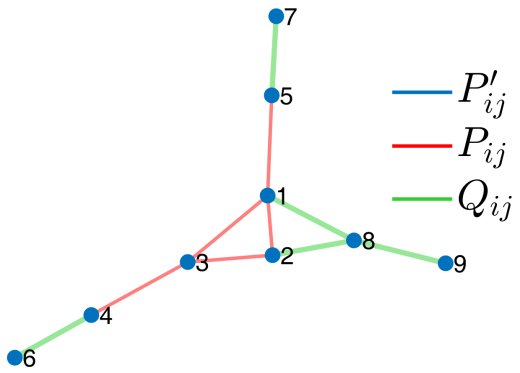


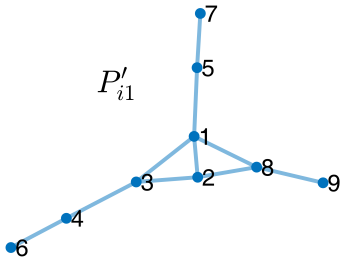
Figure 7.3: Subgraphs of P'_{ij}. Vertex sets of particular subgraphs are $V(P'_{ij}) = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $V(P_{ij}) = \{1, 2, 3, 4, 6\}$, $V(Q_{ij}) = \{1, 2, 4, 5, 6, 7, 8, 9\}$. (Source: co-authored work [97].)

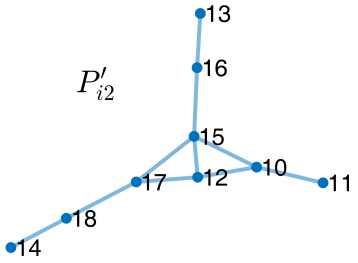
Furthermore, since $P'_{ij} \backsimeq P'_{il}$, then $P'_{i1}, \ldots, P'_{ik}$ have the same adjacency matrix (see the example with P'_{i1} and P'_{i2} in *Figure 7.4*). Let $\mathbf{Adj}_i$ denote the adjacency matrix of $P'_{i1}, \ldots, P'_{ik}$. The part of $\mathbf{Adj}_i$ representing edges of P_{ij} is known before GA is run and corresponds to $g_f(s)$ found with GraMi (line 3 in *Algorithm 7*). Thus, the part of $\mathbf{Adj}_i$ representing edges of P_{ij} is constant in all possible solutions of GA and all individuals in all generations. Therefore, to encode the individuals in chromosomes, it is enough to encode some part of $\mathbf{Adj}_i$ and the ordered lists of vertices of Q_{ij}. Hence, each chromosome consists of two parts

- CH = bits representing elements of $\mathbf{Adj}_i$

- $varCH$ = ordered lists of vertices from $V(Q_{i1}), \ldots, V(Q_{ik})$

More precisely, CH represents the elements of $\mathbf{Adj}_i$ corresponding to $E(Q_{ij})$ and $(u_{ij}, v_{ij}) \in E(P'_{ij}) : u_{ij} \in V(Q_{ij}) \wedge v_{ij} \in V(P_{ij})$ (see *Figure 7.5*). The chromosome representation guarantees that each individual corresponds to graphs $P'_{i1}, \ldots, P'_{ik}$ that are supergraphs of $P_{i1}, \ldots, P_{ik}$ and $P'_{ij} \backsimeq P'_{il}$. Hence, the representation keeps the k-automorphism in the final solution G^*, and it is unnecessary to check the k-anonymity property during GA processing.





Adjacency matrix of P'_{i1}:

	1	2	3	4	5	6	7	8	9
1	0	1	1	0	1	0	0	1	0
2	1	0	1	0	0	0	0	1	0
3	1	1	0	1	0	0	0	0	0
4	0	0	1	0	0	1	0	0	0
5	1	0	0	0	0	0	1	0	0
6	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0
8	1	1	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	1	0

Adjacency matrix of P'_{i2}:

	15	12	17	18	16	14	13	10	11
15	0	1	1	0	1	0	0	1	0
12	1	0	1	0	0	0	0	1	0
17	1	1	0	1	0	0	0	0	0
18	0	0	1	0	0	1	0	0	0
16	1	0	0	0	0	0	1	0	0
14	0	0	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0
10	1	1	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	1	0

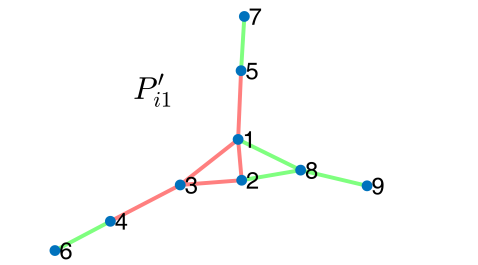
Graph isomorphism $F_1 : P'_{i1} \longrightarrow P'_{i2}$:

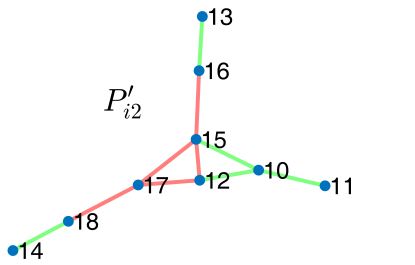
$F_1(1) = 15$ $F_1(4) = 18$ $F_1(7) = 13$
$F_1(2) = 12$ $F_1(5) = 16$ $F_1(8) = 10$
$F_1(3) = 17$ $F_1(6) = 14$ $F_1(9) = 11$

Figure 7.4: Adjacency matrix of isomorphic graphs P'_{i1} and P'_{i2} ($k = 2$). (Source: co-authored work [97].)

7.4.2 Fitness function

A fitness function $FF(I)$ evaluates how close the solution represented by the individual I is to the optimal solution of the problem. The aim of GA in HAkAu is to find the solution with minimal $Cost(G, G^*)$. GA runs several times in HAkAu (see line 5 in *Algorithm 7*). In each run GA finds the optimal graphs $P'_{i1}, \ldots P'_{ik}$ for fixed i. Thus, minimizing $ExCost_i(H)$ in each run of GA is necessary. Except that, $CECost(G, G^*)$ should be minimized. However, during the particular run of GA, it is impossible to compute $CECost(G, G^*)$ properly since the decision, whether to add a crossing edge in G^* or not, is made after all runs of GA. $CECost(G, G^*)$ is directly proportional to the number of crossing edges between $V(P'_{ij})$ and other nodes of H. Hence, the number of crossing edges is minimized





$\mathbf{Adj}_i$	1	2	3	4	5	**6**	**7**	**8**	**9**
1	0	1	1	0	1	0	0	1	0
2	1	0	1	0	0	0	0	1	0
3	1	1	0	1	0	0	0	0	0
4	0	0	1	0	0	1	0	0	0
5	1	0	0	0	0	0	1	0	0
6	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0
8	1	1	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	1	0

$\mathbf{Adj}_i$	15	12	17	18	16	**14**	**13**	**10**	**11**
15	0	1	1	0	1	0	0	1	0
12	1	0	1	0	0	0	0	1	0
17	1	1	0	1	0	0	0	0	0
18	0	0	1	0	0	1	0	0	0
16	1	0	0	0	0	0	1	0	0
14	0	0	0	1	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0
10	1	1	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	1	0

$CH = 00010 \quad 000010 \quad 1100000 \quad 00000001$
$varCH = 6\,7\,8\,9 \quad 14\,13\,10\,11$
Whole chromosome: $0001000001011000000000000016\,7\,8\,9\,14\,13\,10\,11$

Figure 7.5: Chromosome representation for $k = 2$. The part of $\mathbf{Adj}_i$ representing $E(P_{ij})$ is highlighted with red (dark) colour, and the part of $\mathbf{Adj}_i$ representing $E(Q_{ij})$ is highlighted with green (light) colour. The part of $\mathbf{Adj}_i$ in the black tetragon makes the bit part of the chromosome CH. The bits are taken in columns. Ordered lists of nodes from $V(Q_{i1})$ and $V(Q_{i2})$ makes $varCH$. (Source: co-authored work [97].)

instead of minimizing $CECost(G, G^*)$ in GA. Thus, the fitness function of the individual I is defined as the two-tuple

$$FF(I) = \Big[nCE(H, I);\ ExCost(H, I)\Big]$$

where $ExCost(H, I)$ means $ExCost_i(H)$ and $nCE(H, I)$ means the number of crossing edges between P'_{ij} and H, $j := 1, \ldots, k$, i fixed, where P'_{ij} **are constructed according to the individual** I. Referring to the line 6 in *Algorithm 7*, $nCE(H, I) = |C_i|$. If two values $FF(I_1)$ and $FF(I_2)$ are compared in GA, then

$$FF(I_1) \leq FF(I_2)$$

$$\begin{aligned} \textbf{if} \quad & nCE(H, I_1) < nCE(H, I_2) \\ \textbf{or} \quad & nCE(H, I_1) = nCE(H, I_2) \ \wedge\ ExCost(H, I_1) \leq ExCost(H, I_2) \end{aligned}$$

The number of crossing edges has more weight while comparing FF values. It has three reasons. Firstly, the aim of GA is to search for the expansion of subgraphs found with GraMi. Why are the subgraphs P_{ij} found with GraMi not used? By the expansion of P_{ij}, $CECost(G, G^*)$ is reduced significantly. Thus,

the focus is to minimize $CECost(G, G^*)$ as much as possible during GA runs. Moreover, it has been experimentally found that $ExCost(H, I)$ is much smaller than $nCE(H, I)$. Finally, if one crossing edge is added in G^*, other $k-1$ copies must be added in G^* to keep G^* k-automorphic. On the other hand, increasing $ExCost(H, I)$ by one corresponds only to the single change in G^*.

7.4.3 Selection function

The selection function chooses individuals from the current population to create the next generation. The two-step selection function is proposed. Let N be the number of requested parents. In the first step, $2N$ individuals are selected from the whole current population using the roulette wheel selection where the expectations are computed with $nCE(H, I)$. In the second step, N parents are selected from the $2N$ chosen individuals using tournament selection where expectations are computed with $ExCost(H, I)$. The definitions of the roulette wheel and tournament selection functions are omitted with reference to [58].

7.4.4 Genetic operators

The usual genetic operations are used: crossover and mutation. Crossover combines two parent individuals I_1 and I_2 into a single child individual I_{1+2}. Let CH_1, CH_2 and CH_{1+2} be the bit parts of chromosomes corresponding to I_1, I_2 and I_{1+2} respectively and $varCH_1$, $varCH_2$ and $varCH_{1+2}$ be the "node" parts of chromosomes corresponding to I_1, I_2 and I_{1+2} respectively. CH's and $varCH$'s are crossed over separately.

The two-point crossover is applied on CH_1 and CH_2 [58]. CH_1 and CH_2 have the same length l_{CH}. Select two random integers $a, b \in (1; l_{CH})$. The bit part of the child individual CH_{1+2} gets 1st, ..., a-th bit from CH_1, $(a+1)$-th, ..., b-th bit from CH_2 and $(b+1)$-th, ..., l_{CH}-th bit from CH_1 (see *Figure 7.6*).

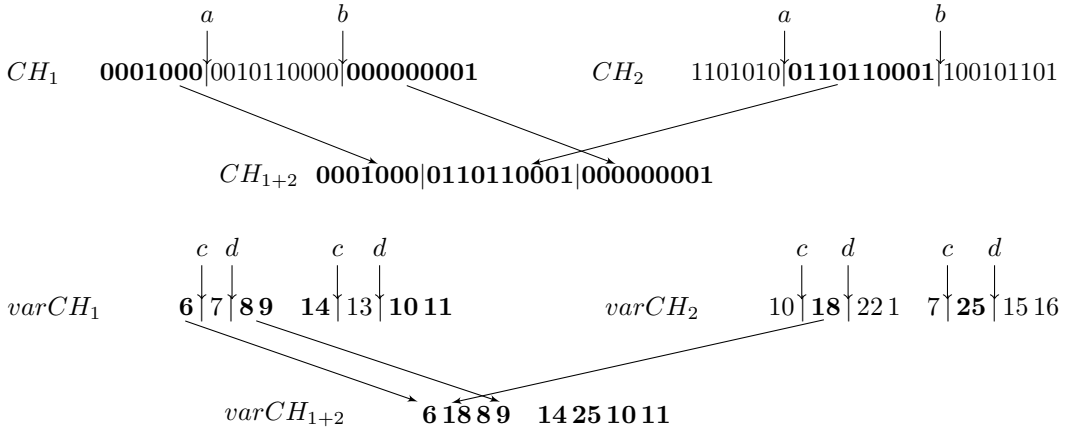


Figure 7.6: Crossover operation in GA. (Source: co-authored work [97].)

Before $varCH_1$ and $varCH_2$ are crossed over, $varCH_1$ and $varCH_2$ are cut into k segments corresponding to $Q_{i1}, \ldots, Q_{ik}$. The two-point crossover is applied to every segment. The i-th segment of $varCH_1$ is crossed with the i-th segment of $varCH_2$, $i = 1, \ldots, k$. Select two random integers c, d between 1 and the length

of the segment. Then all segments are crossed at the same points corresponding to c and d (see *Figure 7.6*).

The probability that an individual is mutated is given by the mutation rate. Then it is randomly decided whether CH or $varCH$ is mutated. When CH is mutated, one random bit is reversed in CH. When $varCH$ is mutated, one random vertex from $varCH$ is replaced with a new one.

7.4.5 Selecting new vertices proportionally to their degree

If it is necessary to add a new vertex in $varCH$, then the vertex is selected from the set of unused vertices $V(H) \setminus V(P'_{ij})$, i fixed, $j = 1, \ldots, k$. There are three situations when vertices are added to $varCH$:

- Creating initial population. The bit parts of chromosomes CH are generated randomly. The vertices are added one by one to $varCH$ from the unused vertices.
- When CH_1 and CH_2 are crossed over, $\mathbf{Adj}_i$ corresponding to CH_{1+2} can lead to graphs with more vertices than the parents' graphs. Hence, adding new vertices to $varCH_{1+2}$ is necessary. Similarly, CH after mutation can lead to a graph with more vertices.
- $varCH$ is mutated.

Selecting new vertices proportionally to their degree reduces $nCE(H, I)$. The bit string CH is usually spare after several generations of GA. Most vertices in $varCH$ are connected to P_{ij} with a single edge. Hence, their other edges become crossing edges. The fewer links the vertices in $varCH$ have, the fewer crossing edges are produced. Thus, a stochastic selection method where the probability of selecting the vertex is proportional to its degree is employed. More precisely, the new vertex to $varCH$ is chosen with roulette selection where expectations are computed with the metric $\frac{1}{deg(v)}$, where $deg(v)$ is the vertex's degree.

7.4.6 Complexity

As mentioned in *Section 4.7*, the complexity of GA is determined by the complexity of the fitness function, selection function and reproduction operators. Let l_{CH} be the length of the chromosome. Since the crossover operation is linear in l_{CH}, its complexity is in $\mathcal{O}(l_{CH})$. The mutation is in $\mathcal{O}(1)$. Let n denote the number of individuals in one generation. The roulette wheel selection is in $\mathcal{O}(n)$ as mentioned [83]. The tournament selection is also in $\mathcal{O}(n)$ as mentioned [129].

The complexity of the fitness function is determined by the complexity of compiling graphs P'_{ij} ($j = 1, \ldots, k$, i fixed) from the chromosome and the computations of $ExCost$ and nCE. Compiling all graphs P'_{ij} takes $\mathcal{O}(k \cdot l_{CH})$ times since $j = 1, \ldots, k$. Computing $ExCost$ for a single P'_{ij} requires considering all its edges and all edges linking its vertices with the rest of the graph. Thus, the complexity of $ExCost$ is in $\mathcal{O}(|V(P'_{ij})| \cdot m_{ij})$, where $m_j = \max_{v \in V(P'_{ij})} deg(v)$. Computing nCE for a single P'_{ij} requires also considering all edges linking vertices from $V(P'_{ij})$, thus its complexity is also $\mathcal{O}(|V(P'_{ij})| \cdot m_j)$. Computing both

metrics for all graphs P'_{ij} ($j = 1, \ldots, k$, i fixed) takes $\mathcal{O}(k \cdot |V(P'_{ij})| \cdot M)$, where $M = \max_{j=1,\ldots,k} m_j$. Since $l_{CH} < \frac{1}{2}|V(P'_{ij})|^2$, then the total complexity of the proposed fitness function is in

$$\mathcal{O}(k \cdot l_{CH} + k \cdot |V(P'_{ij})| \cdot M) = \mathcal{O}(k \cdot |V(P'_{ij})|^2 \cdot M).$$

Considering the formula for the complexity of GA in *Section 4.7*, where g is the number of generations, the total complexity of the proposed GA is in

$$\mathcal{O}(n \cdot l_{CH} + g \cdot n \cdot (n + k \cdot |V(P'_{ij})|^2 \cdot M + l_{CH})) = \mathcal{O}(g \cdot n \cdot (n + k \cdot |V(P'_{ij})|^2 \cdot M)).$$

7.5 Experimental results

This section presents the results of accomplished experiments with real-world networks. All experiments were performed on a Windows 10 operating system PC with 8 GB RAM and a 3.2 GHz processor. The programs were written in Matlab 9.7.0.1261785 (R2019b). The implementation is included in the attached CD, and the overview of corresponding MATLAB files is given in *Attachment B*. The used implementation of the GraMi algorithm is available at [30]. The evaluation tool SecGraph is available at [59].

7.5.1 Tested datasets and parameter setting

HAkAu was tested on three real-world datasets of different sizes that are free to use: Prefuse [50], Polblogs [2] and Wiki-Vote [80]. Three kinds of results are produced. At first, HAkAu is compared with the KM algorithm. I provide the comparison on the Prefuse dataset since the results of KM on Prefuse were presented in [163]. Then, data utility measurement is provided on all three datasets. Finally, I tested the resistance against deanonymization attacks. Since the Prefuse dataset was too small to accomplish this testing, results are provided only on Polblogs and Wiki-Vote.

Both tested datasets and the parameter settings were chosen with respect to the computation capabilities of the single PC and the fact that the computation had to be repeated at least ten times for each parameter setting since the algorithm is non-deterministic. The largest tested dataset was the Wiki-Vote network with 7,115 nodes. Depending on the other parameters, a single run of the HAkAu algorithm on the Wiki-Vote network takes between 4.5 and 9.1 hours. Hence, the computation takes more than 90 hours in the most demanding case.

The minimal support $s = 2k$ in all experiments. The parameter setting of GA is based on the computation capabilities of a single PC and the limitations of MATLAB. The genetic algorithm ran in two phases. At first, it created a population of 1,000 individuals and ran for 20 generations. Then, it picked the 20 best individuals with respect to the fitness function and ran for 100 generations. In both phases, the cross-over operation was applied to all individuals, and the mutation rate equals 2%.

Moreover, each time the mutation should be applied, the algorithm flips the coin to decide whether to mutate CH or $varCH$ in the chromosome. The best individual from each generation survived for the next generation. In other words, the elite count equals 1. The initial population in the first phase of GA was

generated randomly. However, each time the new vertex should be selected from the set of unused vertices, it is chosen with the roulette selection proportionally to its degree.

All chromosomes in all generations had the same length, which depended on the anonymization parameter k and the number of unused nodes $nn = |V(H)| - \sum_{j=1^m} |V(P_{ij})|$. Hence, $varCH$ equaled to the list of $\lfloor \frac{nn}{k} \rfloor$ nodes and CH was the bit string of the length equalling $|varCH|^2 + \sum_{i=1}^{|varCH|} 1$ (the formula is deduced from the proposed representation see *Figure 7.5*).

It was experimentally found that the acceptable solution contains a lot of zeros, and the number of vertices in the acceptable solution was smaller than $|varCH|$. Thus, the starting chromosome length has been reduced to one-third for the largest Wiki-Vote network. The reduction gave even more importance to selecting new vertices proportionally to their degree while creating the initial population.

7.5.2 The comparison of HAkAu and KM algorithm

The performance of the HAkAu algorithm is compared with the performance of the original k-automorphism algorithm KM algorithm. To compare the produced results with the ones presented by [163], experiments were run on the Prefuse dataset with the anonymization parameter $k \in \{5, 10, 15, 20\}$ and the following network metrics in the anonymized network were computed: average clustering coefficient $ACC(G^*)$, average shortest path length $APL(G^*)$ and the total degree difference. The total degree difference equals the sum of the difference between the node degree in the original graph and its degree in the anonymized one.

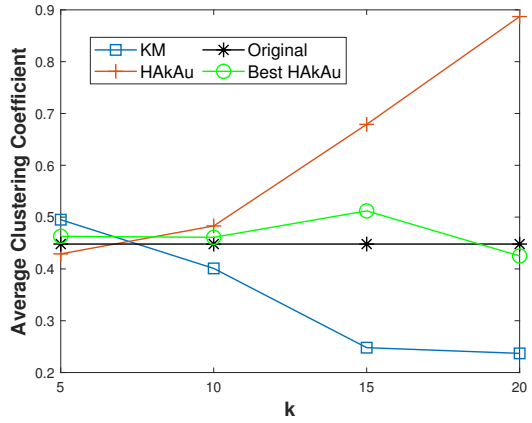
Since HAkAu is a non-deterministic algorithm, it was run ten times on each parameter setting. The experimental results are shown in *Figure 7.7*. There is the mean of the ten metric values and the metric value of the best run.

The clustering coefficient describes how well the neighbourhood of a node is connected. If it is fully connected, the clustering coefficient is 1, whereas a value close to 0 implies hardly any connection [65]. Hence, when k is larger, more edges are added by HAkAu, and ACC increases on average. However, in the best case, the algorithm can compile an anonymized graph with ACC very close to the original network. Interestingly, the ACC values of KM get lower with larger k even though KM only adds edges.

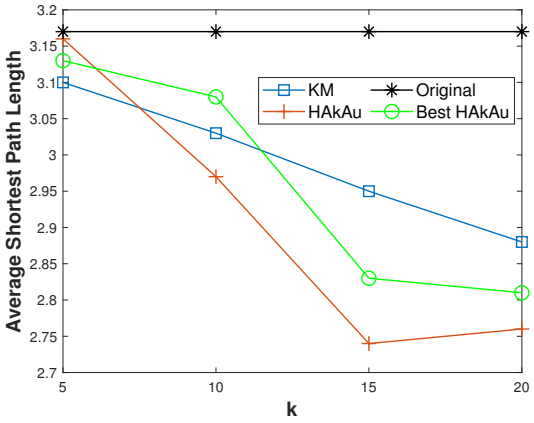
While the original graph is modified by adding edges, the distance between each pair of nodes is reduced (see *Figure 7.7b*). If $k \geq 15$, the GraMi algorithm finds no frequent subgraph appearing at least 15 times in the original graph of the Prefuse network since the dataset is small. Hence, the subgraphs $P_{1,1}, \ldots, P_{1,15}$ inputting GA are only isolated vertices selected randomly.

The experiments show that the fitness function could be designed to keep the APL property better. If APL was the critical property that had not changed during the anonymization process, then the fitness function would have to be modified to consider distances between vertices.

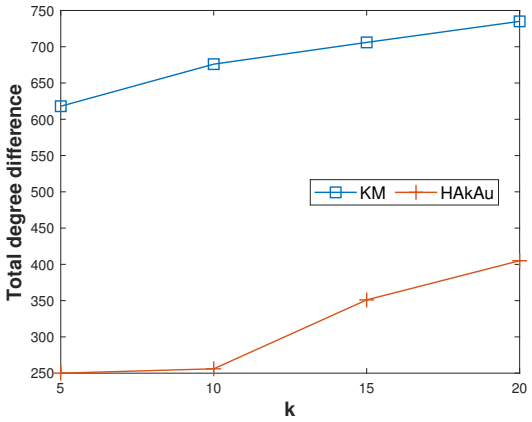
The total degree difference is significantly lower while applying HAkAu. Using edge deletion operation in the procedure of adding crossing edges decreases the amount of added edges and the final degree of all nodes.



a) Average clustering coefficient.



b) Average shortest path length.



c) Total degree difference.

Figure 7.7: Comparison of HAkAu and KM algorithm while anonymizing Prefuse network with the anonymization parameter $k \in \{5, 10, 15, 20\}$. (Source: co-authored work [97].)

7.5.3 Data utility measurement

The HAkAu algorithm was further evaluated in preserving other network and application metrics. Three real social networks were anonymized with HAkAu, and the anonymized networks were evaluated using the SecGraph tool [60, 59].

The following methodology of experiments was used. The dataset and the anonymization parameter were selected: $D \in \{\text{Prefuse, Polblogs, Wiki-Vote}\}$, $k \in \{5, 10, 15, 20\}$. HAkAu was run ten times on D with k. Utility metrics were measured in all ten output graphs by the SecGraph utility module, and the mean value of each metric is shown in *Table 7.1*. The dataset D was further anonymized using three schemes from the SecGraph anonymization module and the heu-kDA algorithm proposed in the previous chapter. The anonymization methods selected from the SecGraph tool are k-degree anonymization method k-DA [84] with the anonymization parameter k and clustering methods tMean and Union [141] with parameter $t = k$. Since those algorithms are deterministic, they ran once on each parameter setting. Utility metrics were measured in the graphs anonymized with k-DA, heu-kDA, tMean and Union as well (see *Table 7.1*). The values in the table describe how each metric is preserved in the anonymized graph compared to the

	Prefuse					Polblogs					Wiki-Vote				
k=15	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
AS	0.157	0.155	0.199	**0.383**	0.219	0.849	0.960	0.858	**0.901**	0.608	0.834	0.987	**0.886**	0.881	0.799
BC	0.528	0.526	**0.834**	**0.834**	0.615	0.904	0.908	0.971	**0.994**	0.420	0.885	0.910	0.966	**0.991**	0.477
CC	**0.995**	0.961	0.913	0.943	**0.995**	0.999	1.000	**1.000**	**1.000**	0.993	0.999	1.000	**1.000**	**1.000**	0.981
CD	0.214	0.509	**0.810**	0.715	0.102	0.857	0.888	0.756	**1.000**	0.061	0.838	0.868	0.755	**0.960**	0.018
Deg.	0.795	0.784	0.973	**0.983**	0.779	0.977	0.990	0.991	**0.998**	0.826	0.965	0.991	0.992	**0.999**	0.817
ED	0.802	1.164	1.374	1.381	**0.826**	0.955	0.956	1.132	**0.987**	0.844	0.982	1.004	**1.013**	0.984	0.911
EV	0.904	0.718	0.411	0.692	**0.905**	0.957	0.984	0.966	**0.997**	0.792	0.977	0.991	0.989	**0.998**	0.800
HS	0.276	0.022	0.201	**0.381**	0.041	0.814	0.984	0.816	**0.881**	0.806	**0.876**	0.993	0.846	0.865	0.674
Infe.	**0.694**	0.644	0.617	0.602	0.646	0.914	0.886	0.896	0.895	**0.928**	0.848	0.805	0.835	0.850	**0.870**
JD	**0.660**	0.215	0.029	0.124	0.005	**0.232**	0.110	0.230	0.224	0.080	0.318	0.304	0.353	**0.397**	0.087
LCC	0.972	0.974	0.970	**0.996**	0.880	0.955	0.984	0.905	**0.983**	0.827	0.988	0.998	0.988	**1.000**	0.774
NC	0.982	0.986	0.982	**0.993**	0.957	0.998	1.000	0.947	**1.000**	0.781	**1.000**	1.000	0.996	**1.000**	0.712
PR	0.572	0.976	0.537	0.544	**0.949**	0.518	0.996	0.498	0.487	**0.666**	0.607	0.945	**0.690**	0.678	0.588
RX	0.460	0.000	0.937	**0.933**	0.490	0.335	0.000	0.332	0.346	**0.398**	0.757	0.000	0.889	**0.971**	0.379

Table 7.1: HAkAu: utility measurement for $k = 15$. (Source: author's work.)

original one. The methodology of how SecGraph computes the metrics is given in *Section 4.2*. Due to the space limits, *Table 7.1* contains only metric values for $k = 15$; all results are included in *Attachement A*.

As we see in the next section, HAkAu offers a much higher level of security than other algorithms. Since deanonymization attacks exploit structural metrics, keeping a high level of protection is paid with worse data utility preservation. The metrics affected the most by HAkAu anonymization are metrics based on the degree of nodes, like degree and joint degree distributions ($Deg.$, JD).Communities in the graph are also heavily altered with HAkAu, which was proved by the low values of the community detection metric (CD).

On the other hand, HAkAu is best in preserving infectiousness ($Infe.$) in 9 parameter settings (Prefuse: $k = 10$, Polblogs for all k, Wiki-Vote for all k) and page rank (PR) in 6 parameter settings (Prefuse for all k, Polblogs: $k \in \{5, 15\}$). Both metrics are centrality metrics that can identify influential users in the graph.

The PR metric measures the importance of each node within the graph based on the number of links and the importance of the linked nodes. Thus, the importance of nodes is preserved while HAkAu modifies the edge set. The final k-isomorphism tend to map important nodes to each other. Preserving infectiousness indicates that the communication channels in the anonymized network are kept very well even though the graph structure is changed significantly by HAkAu.

Since HAkAu is non-deterministic, the metric values varied in ten anonymized networks with the same parameter settings obtained in ten runs. The coefficient of variation was lower in the metrics values that HAkAu better preserved. Except for JD and CD metrics, the coefficient of variation was up to 10%. Furthermore, the coefficient of variation was lower in larger networks.

7.5.4 Resistance to deanonymization attacks

In this section, HAkAu is shown to be resistant to deanonymization techniques, and the resistance is guaranteed in every algorithm run. The methodology of experiments is similar to the one utilized in the previous measurement. The dataset and the anonymization parameter were selected: $D \in \{$Polblogs, Wiki-Vote$\}$, $k \in \{5, 10, 15, 20\}$. The HAkAu was run ten times on each parameter setting, and each output network was attacked with three deanonymization algorithms implemented in SecGraph: Narayanan-Shmatikov's attack (NS) [104], Yartseva-Grossglauser's attack (Per.) [154] and Korula-Lattanzzi's attack (Rec.) [68]. The results are given in *Table 7.2*.

These seed-based passive attacks employ the structural similarity between the anonymized and auxiliary graphs to break the anonymity. The input of their procedures is the anonymized network G^*, auxiliary network G_{aux} and seed mapping s. The network G_{aux} is a fraction of the original network gained by the attacker before the passive attack. The seed mapping s is the mapping that links some nodes from G_{aux} with the ones in G^*. In the experiment, G_{aux} was sampled randomly with the probability of 90% from the original network. The seed s was set as 50 links between randomly selected nodes from G_{aux} and G^*.

The output of deanonymization procedures in SecGraph is the ratio of successfully deanonymized users. The results are given in percentages in *Table 7.2*.

	Polblogs					Wiki-Vote				
k=5	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
NS	92.42%	82.84%	90.61%	92.42%	**4.64**%	72.43%	63.75%	72.61%	72.90%	**0.81**%
Per.	48.83%	56.13%	11.64%	60.02%	**4.96**%	9.57%	3.88%	14.46%	37.40%	**0.92**%
Rec.	96.00%	96.73%	48.94%	98.94%	**4.74**%	98.30%	99.58%	30.02%	99.55%	**0.79**%
k=10	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
NS	90.88%	82.35%	91.88%	92.33%	**4.53**%	72.03%	63.74%	72.67%	72.90%	**0.80**%
Per.	18.77%	55.80%	53.16%	63.00%	**4.65**%	2.18%	10.79%	7.97%	33.11%	**0.89**%
Rec.	91.99%	90.36%	63.48%	97.14%	**4.28**%	92.92%	96.96%	43.22%	98.40%	**0.75**%
k=15	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
NS	91.79%	82.43%	91.25%	92.42%	**4.56**%	71.45%	63.75%	72.84%	72.90%	**0.81**%
Per.	20.76%	47.22%	43.23%	36.73%	**4.65**%	8.32%	22.05%	4.67%	35.42%	**0.85**%
Rec.	81.54%	86.03%	27.70%	84.72%	**4.29**%	90.89%	92.89%	22.98%	97.44%	**0.75**%
k=20	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
NS	90.97%	82.92%	88.09%	92.15%	**4.51**%	71.31%	63.49%	72.53%	72.90%	**0.81**%
Per.	17.51%	32.79%	61.19%	56.32%	**4.59**%	2.83%	2.11%	22.28%	8.47%	**0.84**%
Rec.	69.12%	75.00%	25.65%	86.36%	**4.23**%	81.26%	93.30%	24.53%	94.35%	**0.74**%

Table 7.2: Resistance against deanonymization attacks. (Source: author's work.)

The HAkAu values are the average ratios in the ten runs with the same parameter settings. I also tested the graphs anonymized with k-DA, heu-kDA, tMean and Union.

All results prove that the security level of the HAkAu algorithm is much higher than the security level of other tested algorithms. In the Wiki-Vote network, the percentage of deanonymized users was up to 1% for all k. The resistance level is the same for all k values in both networks. The k-automorphism approach achieves better resistance against attacks compared to k-degree and clustering methods.

Although HAkAu is non-deterministic, the percentage of deanonymized users did not vary in the ten runs with the same parameter settings. The coefficient of variation within the ten runs was up to 4% in all cases, except the instance with D =Wiki-Vote and $k = 10$, where the coefficient of variation equals 10%. However, the level of security was the same since the best value equalled 0.85% and the worst one to 1.09% when D =Wiki-Vote and $k = 10$.

Let me compare k-DA, and heu-kDA in resistance against deanonymization attacks since this topic was omitted in *Chapter 6*. Both algorithms have similar resistance levels since they are both based on the same anonymization approach (k-degree anonymity). Both algorithms are very vulnerable against NS and *Rec.* attacks. The proposed heu-kDA is a bit more resistant against NS attack while k-DA is slightly better in resisting against *Rec.*. Both algorithms are better at resisting against *Per.*. It also holds for the tested clustering algorithms. Union and tMeans achieved a higher level of resistance while facing *Per.* then NS or *Rec.*. On the other hand, HAkAu achieved the same level of resistance to each tested deanonymization algorithm.

7.6 Discussion

In this chapter, the novel k-automorphism anonymization algorithm HAkAu was introduced. The HAkAu algorithm improves the previously published KM algorithm by employing the genetic algorithm and edge deletion operation.

The employment of GA in the k-automorphism method enables the reduction of two NP-hard problems into a single one. In the KM algorithm, the isomorphic graphs were found, then extended so that the isomorphism was lost, and then it was necessary to make them isomorphic again. HAkAu makes the process more efficient by extending the isomorphic subgraphs "isomorphically" with GA. Hence, the algorithm solves only one NP-hard problem instead of two.

The comparison of KM and HAkAu is limited since only two structural metrics and the total degree difference are measured in [163]. The total degree difference is significantly lower while applying HAkAu. Using edge deletion operation in the procedure of adding crossing edges decreases the amount of added edges and the final degree of all nodes. The application of edge deletion operation was enabled by improving the design of the k-automorphism algorithm.

Data utility measurement demonstrated that HAkAu does not exceed in preserving all utility metrics; however, it keeps Page Rank and Infectiousness very well. Both metrics are centrality metrics that can identify influential users in the graph. Thus, the importance of nodes is preserved in the network anonymized by HAkAu. The final k-automorphism maps important nodes to each other. Preserv-

ing infectiousness indicates that the communication channels in the anonymized network are kept very well even though the graph structure is changed significantly by HAkAu.

The k-automoprhism method was designed to protect against any structural attack; however, its actual resistance against deanonymization methods has not yet been proven. The presented experiments demonstrate that HAkAu is resistant well, unlike other tested algorithms. It proves that the k-automorphism approach provides a higher level of security than other solutions. Moreover, HAkAu delivers the same level of resistance for all kinds of tested attacks and all values of the anonymization parameter.

Summarize the above findings and answer *Question 3*. Applying GA, employing edge deletion operation and the novel adding crossing edges operation improves the k-automorphism approach in data utility preservation. Moreover, the proposed algorithm was proven to be resistant to deanonymization attacks. Providing the resistance disables the ability to keep all utility metrics well. However, the experiments showed that even the algorithm providing a high level of protection could preserve some metrics better than other solutions. It supports the idea of implementing a framework for application-oriented anonymization where the input data are modified with respect to specific metrics chosen by the data recipient [96].

8. Conclusion

This thesis focused on the privacy-preserving issue in publishing social network datasets called the identity disclosure problem. It introduced anonymization as the practical approach enabling providers to publish their datasets and preserve users' privacy at the same time. The thesis started with a comprehensive literature review on anonymizing social network datasets, emphasizing k-anonymization methods based on edge editing. Then, three open problems in this field were identified, and the objectives of the thesis were introduced.

The first goal was to show that social network datasets could be vulnerable to composition attacks. So far, the composition attack was presented only as a privacy threat against relational datasets. The composition attack algorithm was proposed and tested on a set of synthetic scale-free networks to solve this issue. Its capability to correctly find 20-30% of corresponding vertices proved that attacks of this kind could threaten privacy in social network datasets similar to relational datasets.

The second issue was improving the noise addition method in the well-known k-degree anonymization algorithm. The novel heuristic approach was based on the power-law degree distribution of social networks. By running experiments on the set of real social network datasets, it was proved that the version of the algorithm employing the proposed procedure was feasible and improved the algorithm in terms of preserving data utility.

Finally, the thesis focused on the k-automorphism method. Since modifying the original dataset to the k-automorphism one includes addressing NP-hard subtasks, the proposal of a genetic algorithm was part of the solution. Moreover, except for the main contribution, proposing and implementing the hybrid k-automorphism anonymization algorithm, the novel chromosome representation and the procedure for finding the subset of vertex-disjoint graphs were introduced. The hybrid algorithm was experimentally proved to preserve some network metrics better than the original solution. Compared with other state-of-the-art anonymization techniques, the algorithm exceeded in resisting deanonymization attacks.

In producing the experimental results, the focus was made on their comparability. The aim was to present results so that any researcher in the future could easily compare their results with the presented ones. Hence, the external evaluation tool SecGraph was used for measuring the data utility of anonymization methods and their resistance against deanonymization attacks. Moreover, while employing SecGraph, the input dataset to the anonymization process was always the whole dataset available online since if only some subset of an available dataset was used, other researchers could not determine which subset it was.

The findings addressing the third research question highlight several new avenues that could be explored in future studies. Implementing the HAkAu algorithm can be further improved with more sophisticated parameter settings in the genetic algorithm. Estimating the optimal running strategy could raise the quality of the search process and the found results. The proposed chromosome representation can be applied to other genetic algorithms dealing with anonymization tasks. The possibility of exploiting the representation in other

tasks handling graphs can also be explored. Similarly, the procedure for finding the vertex-disjoint subgraphs has the potential for broader application.

This thesis's findings prove, similarly to some other recent results, that the proposal of the robust and universal anonymization method providing a high level of security and preserving all data utility well is a nearly impossible task. When a method is improved in terms of preserving data utility, it is less resistant to attacks. On the other hand, even methods providing high-level security can keep some metrics well, like HAkAU, that preserves infectiousness and page rank. Thus, I find application-oriented anonymization the promising research direction in this field. This approach is based on the idea that the anonymization method is not required to preserve all data utility but only data utility specified by the data recipient. Therefore, it enables to apply of techniques that provide high-level security but preserve only some metrics. Since evaluation tools play an essential role in this approach, comparing the two recently published tools, SecGraph and DUEF-GA, is also an interesting research challenge.

References

[1] Jemal H. Abawajy, Mohd Izuan Hafez Ninggal, and Tutut Herawan. Privacy preserving social network data publication. *IEEE Commun Surv Tut*, 18(3): 1974–1997, 2016. doi: 10.1109/COMST.2016.2533668.

[2] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd International workshop on Link discovery*, pages 36–43, Chicago, IL, USA, 2005. ACM Press. doi: 10.1145/1134271.1134277.

[3] Arash Alavi, Rajiv Gupta, and Zhiyun Qian. When the attacker knows a lot: The GAGA graph anonymizer. In *Information Security*, pages 211–230. Springer, Cham, 2019. doi: 10.1007/978-3-030-30215-3_11.

[4] Yair Amichai-Hamburger and Gideon Vinitzky. Social network use and personality. *Comput Human Behav*, 26(6):1289–1295, 2010. doi: 10.1016/j.chb.2010.03.018.

[5] Mehdi Azaouzi, Delel Rhouma, and Lotfi Ben Romdhane. Community detection in large-scale social networks: State-of-the-art and future directions. *Soc Netw Anal Min*, 9(1):1–32, 2019. doi: 10.1007/s13278-019-0566-x.

[6] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. *Commun ACM*, 54(12):133–141, 2011. doi: 10.1145/2043174.2043199.

[7] Vladimir Batagelj and Andrej Mrvar. Pajek. `http://vlado.fmf.uni-lj.si/pub/networks/data/`, 2014. Accessed: 2021-11-13.

[8] Cristina Bazgan, Robert Bredereck, Sepp Hartung, André Nichterlein, and Gerhard J. Woeginger. Finding large degree-anonymous subgraphs is hard. *Theor Comput Sci*, 622:90–110, 2016. doi: 10.1016/j.tcs.2016.02.004.

[9] Cristina Bazgan, Pierre Cazals, and Janka Chlebíková. Degree-anonymization using edge rotations. *Theor Comput Sci*, 873:1–15, 2021. doi: https://doi.org/10.1016/j.tcs.2021.04.020.

[10] Mehdi Behzad and Gary Chartrand. *Introduction to the Theory of Graphs*. Allyn and Bacon, 1972.

[11] Gema Bello-Orgaz, Héctor D. Menéndez, and David Camacho. Adaptive K-means algorithm for overlapped graph clustering. *Int J Neur Syst*, 22(05), 2012. doi: 10.1142/S0129065712500189.

[12] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc VLDB Endow*, 2(1):766–777, 2009.

[13] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory*. Springer Publishing Company, Incorporated, 2008.

[14] Christian Borgs, Jennifer Chayes, and Adam Smith. Private graphon estimation for sparse graphs. *Adv Neur In*, 28, 2015.

[15] Robert Bredereck, Vincent Froese, Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. The complexity of degree anonymization by vertex addition. *Theor Comput Sci*, 607:16–34, 2015. doi: 10.1016/j.tcs.2015.07.004.

[16] Qing Cai, Maoguo Gong, Lijia Ma, Shasha Ruan, Fuyan Yuan, and Licheng Jiao. Greedy discrete particle swarm optimization for large-scale social network clustering. *Inform Sciences*, 316:503–516, 2015. doi: 10.1016/j.ins.2014.09.041.

[17] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *International Workshop on Privacy, Security, and Trust*

in KDD, pages 33–54, Berlin, Heidelberg, 2008. Springer. doi: 10.1007/978-3-642-01718-6_4.

[18] Jordi Casas-Roma. GitHub - jcasas/DUEF-GA: Data utility and privacy evaluation framework for graph anonymization. `https://github.com/jcasasr/DUEF-GA`, 2019. Accessed: 2023-01-13.

[19] Jordi Casas-Roma. DUEF-GA: data utility and privacy evaluation framework for graph anonymization. *Int J Inf Secur*, 19:465–478, 2020. doi: 10.1007/s10207-019-00469-4.

[20] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. K-degree anonymity and edge selection: Improving data utility in large networks. *Knowl Inf Syst*, 50(2):447–474, 2017. doi: 10.1007/s10115-016-0947-7.

[21] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. A survey of graph-modification techniques for privacy-preserving on networks. *Artif Intell Rev*, 47(3):341–366, 2017. doi: 10.1007/s10462-016-9484-8.

[22] Maria Chiara Caschera, Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. MONDE: A method for predicting social network dynamics and evolution. *Evolving Systems*, 10(3):363–379, 2019. doi: 10.1007/s12530-018-9242-z.

[23] Saptarshi Chakraborty and B. K. Tripathy. Alpha-anonymization techniques for privacy preservation in social networks. *Soc Netw Anal Min*, 6(1):1–11, 2016. doi: 10.1007/s13278-016-0337-x.

[24] Jinyin Chen, Lihong Chen, Yixian Chen, Minghao Zhao, Shanqing Yu, Qi Xuan, and Xiaoniu Yang. GA-based Q-attack on community detection. *IEEE Trans Comp Soc Syst*, 6(3):491–503, 2019. doi: 10.1109/TCSS.2019.2912801.

[25] James Cheng, Ada Wai-Chee Fu, and Jia Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 459–470, New York, NY, USA, 2010. ACM Press. doi: 10.1145/1807167.1807218.

[26] Sean Chester, Bruce M. Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and Sistla Venkatesh. Why waldo befriended the dummy? k-anonymization of social networks with pseudo-nodes. *Soc Netw Anal Min*, 3 (3):381–399, 2013. doi: 10.1007/s13278-012-0084-6.

[27] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1082–1090, New York, NY, USA, 2011. ACM Press.

[28] A. I. Diveev and O. V. Bobr. Variational genetic algorithm for NP-hard scheduling problem solution. *Procedia Comput Sci*, 103:52–58, 2017. doi: 10.1016/j.procs.2017.01.010.

[29] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer. doi: 10.1007/978-3-540-79228-4_1.

[30] Mohammed Elseidy and Ehab Abdelhamid. GraMi. `https://github.com/ehab-abdelhamid/GraMi`, 2014. Accessed: 2021-11-13.

[31] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. GraMi: Frequent subgraph and pattern mining in a single large graph. *Proc VLDB Endow*, 7(7):517–528, 2014. doi: 10.14778/2732286.2732289.

[32] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn Lett*, 27(8): 861–874, 2006. doi: 10.1016/j.patrec.2005.10.010.

[33] Andrea Fronzetti Colladon and Elisa Remondi. Using social network analysis to

prevent money laundering. *Expert Syst with Appl*, 67:49–58, 2017. doi: 10.1016/j.eswa.2016.09.029.

[34] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput Surv*, 42(4): 1–53, 2010. doi: 10.1145/1749603.1749605.

[35] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, New York, NY, USA, 2008. ACM Press. doi: 10.1145/1401890.1401926.

[36] Michael R. Garey and David S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *J ACM*, 25(3):499–508, 1978.

[37] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 KDD Cup. *SIGKDD Explorations*, 5(2):149–151, 2003. doi: 10.1145/980972.980992.

[38] Mathew George. B-A scale-free network generation and visualization. `http://www.mathworks.com/matlabcentral/fileexchange/11947-b-a-scale-free-network-generation-and-visualization`, 2006. Accessed: 2017-03-20.

[39] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *P Natl A Sci*, 99(12):7821–7826, 2002. doi: 10.1073/pnas.122653799.

[40] Mustafa Hajij, Eyad Said, and Robert Todd. Pagerank and the k-means clustering algorithm. *arXiv preprint arXiv:2005.04774*, 2020.

[41] A. Hanif Halim and I. Ismail. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. *Arch Comput Method E*, 26(2): 367–380, 2019. doi: 10.1007/s11831-017-9247-y.

[42] Meng Han, Dongjing Miao, Jinbao Wang, and Liyuan Liu. A balm: defend the clique-based attack from a fundamental aspect. *J Comb Optim*, pages 1–22, 2020. doi: 10.1007/s10878-020-00527-x.

[43] S.L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE T Knowl Data En*, 15(4):1043–1044, 2003. doi: 10.1109/TKDE.2003.1209020.

[44] Sepp Hartung and Nimrod Talmon. The complexity of degree anonymization by graph contractions. In *Theory and Applications of Models of Computation*, pages 260–271, Cham, 2015. Springer. doi: 10.1007/978-3-319-17142-5_23.

[45] Sepp Hartung, Clemens Hoffmann, and André Nichterlein. Improved upper and lower bound heuristics for degree anonymization in social networks. In *International Symposium on Experimental Algorithms*, pages 376–387, Copenhagen, 2014. Springer. doi: 10.1007/978-3-319-07959-2_32.

[46] Sepp Hartung, André Nichterlein, Rolf Niedermeier, and Ondřej Suchý. A refined complexity analysis of degree anonymization in graphs. *Inform Comput*, 243:249–262, 2015. doi: https://doi.org/10.1016/j.ic.2014.12.017.

[47] John Harvey, Andrew Smith, James Goulding, and Ines Branco Illodo. Food sharing, redistribution, and waste reduction via mobile applications: A social network analysis. *Ind Market Manag*, 88:437–448, 2020. doi: 10.1016/j.indmarman.2019.02.019.

[48] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer science department faculty publication series*, 180, 2007.

[49] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Chao Li. Resisting

structural re-identification in anonymized social networks. *ACM VLDB*, 19(6): 797–823, 2010. doi: 10.1007/s00778-010-0210-x.

[50] Jeffrey Heer, Stuart K. Card, and James A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM Press. doi: 10.1145/1054972.1055031.

[51] Jeffrey Heer, Stuart K. Card, and James A. Landay. Prefuse data. `https://github.com/prefuse/Prefuse/blob/master/data/socialnet.xml`, 2007. Accessed: 2021-11-13.

[52] Julia Heidemann, Mathias Klier, and Florian Probst. Online social networks: A survey of a global phenomenon. *Computer Networks*, 56(18):3866–3878, 2012. doi: 10.1016/j.comnet.2012.08.009.

[53] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: Structural role extraction; mining in large graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1231–1239, New York, NY, USA, 2012. ACM Press. doi: 10.1145/2339530.2339723.

[54] John H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J Comput*, 2(2):88–105, 1973.

[55] Sameera Horawalavithana, Juan G. Arroyo Flores, John Skvoretz, and Adriana Iamnitchi. Behind the mask: Understanding the structural forces that make social graphs vulnerable to deanonymization. *IEEE Trans Comp Soc Syst*, 6(6): 1343–1356, 2019. doi: 10.1109/TCSS.2019.2951330.

[56] Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Comput Intel Neurosc*, 2017. doi: 10.1155/2017/7430125.

[57] Josef Hynek. Genetic algorithms in a nutshell. *Economics and Management*, 5: 48–54, 2002.

[58] Josef Hynek. *Genetické algoritmy a genetické programování*. Grada Publishing a.s., Prague, Czech Republic, 2008.

[59] Shouling Ji and Weiqing Li. SecGraph home. `https://nesa.zju.edu.cn/secgraph_pages/home.html`, 2015. Accessed: 2021-11-13.

[60] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem Beyah. SecGraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *24th USENIX Security Symposium*, pages 303–318, 2015.

[61] Shouling Ji, Prateek Mittal, and Raheem Beyah. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Commun Surv Tut*, 19(2):1305–1326, 2017. doi: 10.1109/COMST.2016.2633620.

[62] Sanjay Kairam, Diana MacLean, Manolis Savva, and Jeffrey Heer. Graphprism: Compact visualization of network structure. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, page 498–505, New York, NY, USA, 2012. ACM Press. doi: 10.1145/2254556.2254651.

[63] Ryota Kanai, Bahador Bahrami, Rebecca Roylance, and Geraint Rees. Online social network size is reflected in human brain structure. *P Roy Soc B-Biol Sci*, 279(1732):1327–1334, 2012. doi: 10.1098/rspb.2011.1959.

[64] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476, Berlin, Heidelberg, 2013. Springer.

[65] Andreas Kemper. *Valuation of network effects in software markets: A complex networks approach*. Springer Science & Business Media, Berlin, Heidelberg, 2009.

[66] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32rd annual ACM symposium on Theory of computing*, pages 163–170, Portland, Oregon, USA, 2000.

[67] Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the 1st conference on email and anti-spam*. CEAS, 2004.

[68] Nitish Korula and Silvio Lattanzi. An efficient reconciliation algorithm for social networks. *Proc VLDB Endow*, 7(5):377–388, 2014. doi: 10.14778/2732269.2732274.

[69] Valdis Krebs. Books about US politics. `http://www.orgnet.com`, 2004. Accessed: 2020-11-10.

[70] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

[71] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341. ACM, 2018.

[72] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph. *Data Min Knowl Disc*, 11(3):243–271, 2005. doi: 10.1007/s10618-005-0003-9.

[73] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 25–25. IEEE, 2006. doi: 10.1109/ICDE.2006.101.

[74] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014. Accessed: 2021-11-13.

[75] Jure Leskovec and Rok Sosič. SNAP: A general-purpose network analysis and graph-mining library. *ACM T Intel Syst Tec*, 8(1):1, 2016.

[76] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, Chicago, IL, USA, 2005. ACM Press.

[77] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans Web*, 1(1):5–es, 2007. doi: 10.1145/1232722.1232727.

[78] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans Knowl Disc Data*, 1(1), 2007. doi: 10.1145/1217299.1217301.

[79] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[80] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, New York, NY, USA, 2010. ACM Press.

[81] Jiuyong Li, Muzammil M. Baig, A. H. M. Sarowar Sattar, Xiaofeng Ding, Jixue Liu, and Millist Vincent. A hybrid approach to prevent composition attacks for independent data releases. *Inform Sciences*, 2016. doi: 10.1016/j.ins.2016.05.009.

[82] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007. doi: 10.1109/ICDE.2007.367856.

[83] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A*, 391(6):2193–2196, 2012. doi: https://doi.org/10.1016/j.physa.2011.12.004.

[84] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–106, Vancouver, Canada, 2008. ACM Press. doi: 10.1145/1376616.1376629.

[85] Yanfeng Liu, Aimin Zhou, and Hu Zhang. Termination detection strategies in evolutionary algorithms: A survey. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 1063–1070, New York, NY, USA, 2018. ACM Press. doi: 10.1145/3205455.3205466.

[86] Xuesong Lu, Yi Song, and Stéphane Bressan. Fast identity anonymization on graphs. In *Database and Expert Systems Applications*, pages 281–295. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-32600-4_21.

[87] Jiangtao Ma, Yaqiong Qiao, Guangwu Hu, Yongzhong Huang, Arun Kumar Sangaiah, Chaoqin Zhang, Yanjun Wang, and Rui Zhang. De-anonymizing social networks with random forest classifier. *IEEE Access*, 6:10139–10150, 2018. doi: 10.1109/ACCESS.2017.2756904.

[88] Tinghuai Ma, Yuliang Zhang, Jie Cao, Jian Shen, Meili Tang, Yuan Tian, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. KDVEM: a k-degree anonymity with vertex and edge modification algorithm. *Computing*, 97(12):1165–1184, 2015. doi: 10.1007/s00607-015-0453-x.

[89] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Trans Knowl Discov Data*, 1(1):3–es, 2007. doi: 10.1145/1217299.1217302.

[90] Maria Macià and Iolanda García. Informal online communities and networks as a source of teacher professional development: A review. *Teach Teach Educ*, 55: 291–307, 2016. doi: 10.1016/j.tate.2016.01.021.

[91] Abdul Majeed and Sungchang Lee. Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey. *IEEE Access*, 9:8512–8545, 2021. doi: 10.1109/ACCESS.2020.3045700.

[92] MathWorks. Global optimization toolbox. `https://www.mathworks.com/products/global-optimization.html`, 2019. Accessed: 2023-02-26.

[93] Jana Medková. Composition attack against social network data. *Computers & Security*, 74:115–129, 2018. doi: 10.1016/j.cose.2018.01.002.

[94] Jana Medková. Anonymization of geosocial network data by the (k, l)-degree method with location entropy edge selection. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–8, New York, NY, USA, 2020. ACM Press. doi: 10.1145/3407023.3409184.

[95] Jana Medková. High-degree noise addition method for the k-degree anonymization algorithm. In *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems*, pages 1–6, Hachijo Island, Japan, 2020. IEEE. doi: 10.1109/scisisis50064.2020.9322670.

[96] Jana Medková and Josef Hynek. Application-oriented framework for social network datasets and IoT environments. In *Accepted for the publication in Proceedings of the 15th International Conference, SecITC 2022*. Springer.

[97] Jana Medková and Josef Hynek. HAkAu: Hybrid algorithm for effective k-automorphism anonymization of social networks. *Soc Netw Anal Min*, 13(63), 2023. doi: 10.1007/s13278-023-01064-1.

[98] Elliott Mendelson. *Introduction to mathematical logic*. Chapman and Hall/CRC, 2009.

[99] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM Press. doi: 10.1145/1298306.1298311.

[100] Monika Mital and Sumit Sarkar. Multihoming behavior of users in social networking web sites: a theoretical model. *Inform Technol Peopl*, 24(4):378–392, 2011. doi: 10.1108/09593841111182250.

[101] R. Mortazavi and S.H. Erfani. GRAM: An efficient (k,l) graph anonymization method. *Expert Syst with Appl*, 153:113454, 2020. doi: https://doi.org/10.1016/j.eswa.2020.113454.

[102] Nicole L. Muscanell and Rosanna E. Guadagno. Make new friends or keep the old: Gender and personality differences in social networking use. *Comput Human Behav*, 28(1):107–112, 2012. doi: 10.1016/j.chb.2011.08.016.

[103] Sahiti Myneni, Brittney Lewis, Tavleen Singh, Kristi Paiva, Seon Min Kim, Adrian V. Cebula, Gloria Villanueva, and Jing Wang. Diabetes self-management in the age of social media: Large-scale analysis of peer interactions using semi-automated methods. *JMIR Med Inform*, 8(6), 2020. doi: 10.2196/18441.

[104] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, CA, USA, 2009. IEEE. doi: 10.1109/SP.2009.22.

[105] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *The 2011 International Joint Conference on Neural Networks*, pages 1825–1834. IEEE, 2011. doi: 10.1109/IJCNN.2011.6033446.

[106] Mina Nasrazadani, Afsaneh Fatemi, and Mohammadali Nematbakhsh. Sign prediction in sparse social networks using clustering and collaborative filtering. *J Supercomput*, 78:596–615, 2022. doi: 10.1007/s11227-021-03902-5.

[107] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, page 537–548, New York, NY, USA, 2014. ACM Press. doi: 10.1145/2660267.2660324.

[108] Mohd Izuan Hafez Ninggal and Jemal H. Abawajy. Utility-aware social network graph anonymization. *J Netw Comput Appl*, 56:137 – 148, 2015. doi: 10.1016/j.jnca.2015.05.013.

[109] Pedram Pedarsani, Daniel R. Figueiredo, and Matthias Grossglauser. A bayesian method for matching two similar graphs without seeds. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1598–1607, 2013. doi: 10.1109/Allerton.2013.6736720.

[110] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000.

[111] W. Peng, F. Li, X. Zou, and J. Wu. A two-stage deanonymization attack against anonymized social networks. *IEEE T Comput*, 63(2):290–303, 2014. doi: 10.1109/TC.2012.202.

[112] Clara Pizzuti. A multiobjective genetic algorithm to find communities in complex networks. *IEEE T Evolut Comp*, 16(3):418–430, 2012. doi: 10.1109/TEVC.2011.2161090.

[113] Charalampos Papamanthou Prateek Mittal and Dawn Song. Preserving link privacy in social network based systems. In *Proceeding of 20th Annual Network and Distributed System Security Symposium*, San Diego, California, USA, 2013. The Internet Society.

[114] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc VLDB Endow*, 7(8):637–648, 2014. doi: 10.14778/2732296.2732300.

[115] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, and Linlin Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016. doi: 10.1109/INFOCOM.2016.7524578.

[116] Sara Rajabzadeh, Pedram Shahsafi, and Mostafa Khoramnejadi. A graph modification approach for k-anonymity in social networks using the genetic algorithm. *Soc Netw Anal Min*, 10(1):1–17, 2020. doi: 10.1007/s13278-020-00655-6.

[117] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: Output perturbation for queries with joins. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, page 107–116, New York, NY, USA, 2009. ACM Press. doi: 10.1145/1559795.1559812.

[118] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *Proceedings of the 2nd International Semantic Web Conference*, pages 351–368, Berlin, Heidelberg, 2003. Springer. doi: 10.1007/978-3-540-39718-2_23.

[119] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Comput*, 6, 2002.

[120] Huan Rong, Tinghuai Ma, Meili Tang, and Jie Cao. A novel subgraph k^+-isomorphism method in social network based on graph similarity detection. *Soft Comput*, 22(8):2583–2601, 2018. doi: 10.1007/s00500-017-2513-y.

[121] Craig Ross, Emily S. Orr, Mia Sisic, Jaime M. Arseneault, Mary G. Simmering, and R. Robert Orr. Personality and motivations associated with Facebook use. *Comput Human Behav*, 25(2):578–586, 2009. doi: 10.1016/j.chb.2008.12.024.

[122] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015. doi: 10.1609/aaai.v29i1.9277. URL `http://networkrepository.com`. Accessed: 2021-11-13.

[123] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, page 81–98, New York, NY, USA, 2011. ACM Press. doi: 10.1145/2068816.2068825.

[124] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[125] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*, Palo Alto, CA, 1998. Computer Science Laboratory, SRI International.

[126] Nicolas Sartor. A brief history of data anonymization. https://aircloak.com/history-of-data-anonymization/, 2019. Accessed: 2022-02-17.

[127] A. H. M. Sarowar Sattar, Jiuyong Li, Jixue Liu, Raymond Heatherly, and Bradley Malin. A probabilistic approach to mitigate composition attacks on privacy in non-coordinated environments. *Knowl Based Syst*, 67:361–372, 2014. doi: 10.1016/j.knosys.2014.04.019.

[128] John Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988.

[129] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 515–519, 2015. doi: 10.1109/ABLAZE.2015.7154916.

[130] Vikas Kumar Sihag. A clustering approach for structural k-anonymity in social networks using genetic algorithm. In *Proceedings of the CUBE International Information Technology Conference*, pages 701–706, Pune, India, 2012. ACM Press. doi: 10.1145/2381716.2381850.

[131] Michael Sipser. *Introduction to the Theory of Computation, Second Edition, International Edition.* Thompson Course Technology, Boston, Massachusetts, 2006.

[132] S. Srivatsan and N. Maheswari. Privacy preservation in social network data using evolutionary model. *Materials Today: Proceedings*, 62:4732–4737, 2022. doi: 10.1016/j.matpr.2022.03.251. International Conference on Innovative Technology for Sustainable Development.

[133] Isabelle Stanton and Ali Pinar. Constructing and sampling graphs with a prescribed joint degree distribution. *ACM J Exp Algorithmics*, 17, 2012. doi: 10.1145/2133803.2330086.

[134] Statista. Number of social network users worldwide from 2017 to 2025 (in billions). https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/, 2020. Accessed: 2022-02-07.

[135] Statista. Most popular social networks worldwide as of January 2022, ranked by number of monthly active users (in millions). https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/, 2022. Accessed: 2023-01-26.

[136] Adam Stawowy. Evolutionary based heuristic for bin packing problem. *Comput Ind Eng*, 55(2):465–474, 2008. doi: 10.1016/j.cie.2008.01.007.

[137] Polisetty Sumanth and Rajeshwari K. Discovering top experts for trending domains on stack overflow. *Procedia Computer Science*, 143:333–340, 2018. doi: 10.1016/j.procs.2018.10.404.

[138] Yongjiao Sun, Ye Yuan, Guoren Wang, and Yurong Cheng. Splitting anonymization: a novel privacy-preserving approach of social network. *Knowl Inf Syst*, 47:595–623, 2016. doi: 10.1007/s10115-015-0855-2.

[139] Kaiyu Tang, Meng Han, Qinchen Gu, Anni Zhou, Raheem Beyah, and Shouling Ji. ShareSafe: An improved version of SecGraph. *KSII T Internet Inf*, 13(11):5731–5754, 2019. doi: 10.3837/tiis.2019.11.025.

[140] Tamir Tassa and Dror J. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. *IEEE T Knowl Data En*, 25(2):311–324, 2013. doi: 10.1109/TKDE.2011.232.

[141] Brian Thompson and Danfeng Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Sympo-*

sium on Information, Computer, and Communications Security, pages 218–227, New York, NY, USA, 2009. ACM Press. doi: 10.1145/1533057.1533088.

[142] UTD. UTD anonymizaton toolbox. `http://www.cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php`, 2012. Accessed: 2021-03-30.

[143] Vijay Verma and Rajesh Kumar Aggarwal. A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective. *Soc Netw Anal Min*, 10(43), 2020. doi: 10.1007/s13278-020-00660-9.

[144] Jan Vosecky, Dan Hong, and Vincent Y. Shen. User identification across social networks using the web profile and friend network. *Int J Web Applic*, 2(1):23–34, 2010.

[145] Dan J. Wang, Xiaolin Shi, Daniel A. McFarland, and Jure Leskovec. Measurement error in network data: A re-classification. *Social Networks*, 34(4):396–409, 2012. doi: 10.1016/j.socnet.2012.01.003.

[146] Yazhe Wang, Long Xie, Baihua Zheng, and Ken C. K. Lee. High utility k-anonymization for social network publishing. *Knowl Inf Syst*, 41(3):697–725, 2014. doi: 10.1007/s10115-013-0674-2.

[147] Stanley Wasserman, Katherine Faust, et al. *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[148] Chen Wen, Bernard C. Y. Tan, and Klarissa Ting-Ting Chang. Advertising effectiveness on social network sites: An investigation of tie strength, endorser expertise and product type on consumer purchase intention. In *ICIS 2009 Proceedings*, page 151. Association for Information Systems, 2009.

[149] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy*, pages 223–238. IEEE, 2010. doi: 10.1109/SP.2010.21.

[150] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 111–122, Lausanne, Switzerland, 2010. ACM Press. doi: 10.1145/1739041.1739058.

[151] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 911–920, New York, NY, USA, 2014. ACM Press. doi: 10.1145/2623330.2623642.

[152] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, page 587–596, New York, NY, USA, 2013. ACM Press. doi: 10.1145/2433396.2433471.

[153] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowl Inf Syst*, 42(1):181–213, 2015. doi: 10.1007/s10115-013-0693-z.

[154] Lyudmila Yartseva and Matthias Grossglauser. On the performance of percolation graph matching. In *Proceedings of the 1st ACM conference on Online social networks*, pages 119–130, New York, NY, USA, 2013. ACM Press. doi: 10.1145/2512938.2512952.

[155] Navid Yazdanjue, Mohammad Fathian, and Babak Amiri. Evolutionary algorithms for k-anonymity in social networks based on clustering approach. *Comput J*, 63(7):1039–1062, 2020. doi: 10.1093/comjnl/bxz069.

[156] Xiaowei Ying, Kai Pan, Xintao Wu, and Ling Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social net-

work publishing. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, pages 1–10, New York, NY, USA, 2009. ACM Press. doi: 10.1145/1731011.1731021.

[157] M. Yuan, L. Chen, P. S. Yu, and T. Yu. Protecting sensitive labels in social network data anonymization. *IEEE T Knowl Data En*, 25(3):633–647, 2013. doi: 10.1109/TKDE.2011.259.

[158] Cheng Zhang, Honglu Jiang, Xiuzhen Cheng, Feng Zhao, Zhipeng Cai, and Zhi Tian. Utility analysis on privacy-preservation algorithms for online social networks: an empirical study. *Pers and Ubiquit Comput*, pages 1–17, 2019.

[159] Yongheng Zhang, Yuliang Lu, Guozheng Yang, and Zijun Hang. Multi-attribute decision making method for node importance metric in complex network. *Applied Sciences*, 12(4), 2022. doi: 10.3390/app12041944.

[160] Dangzhi Zhao and Andreas Strotmann. Analysis and visualization of citation networks. *Synthesis lectures on information concepts, retrieval, and services*, 7 (1):1–207, 2015. doi: 10.2200/S00624ED1V01Y201501ICR039.

[161] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *2008 IEEE 24th International Conference on Data Engineering*, pages 506–515, Cancun, Mexico, 2008. IEEE. doi: 10.1109/icde.2008.4497459.

[162] Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl Inf Syst*, 28 (1):47–77, 2011. doi: 10.1007/s10115-010-0311-2.

[163] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proc VLDB Endow*, 2(1):946–957, 2009. doi: 10.14778/1687627.1687734.

List of Figures

List of Tables

Author's publications related to the dissertation topic

[1] Jana Medková. Composition attack against social network data. *Computers & Security*, 74:115–129, 2018. doi: 10.1016/j.cose.2018.01.002.

[2] Jana Medková. Anonymization of geosocial network data by the (k, l)-degree method with location entropy edge selection. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–8, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3407023.3409184.

[3] Jana Medková. High-degree noise addition method for the k-degree anonymization algorithm. In *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems*, pages 1–6, Hachijo Island, Japan, 2020. IEEE. doi: 10.1109/scisisis50064.2020.9322670.

[4] Jana Medková. The impact of anonymization on the geosocial network metrics used in socio-economic analysis. In *Proceedings of the international scientific conference Hradec Economic Days 2020*, pages 543–550. University of Hradec Kralove, 2020. doi: 10.36689/uhk/hed/2020-01-062.

[5] Jana Medková and Josef Hynek. HAkAu: Hybrid algorithm for effective k-automorphism anonymization of social networks. *Social Network Analysis & Mining*, 13(63), 2023. doi: 10.1007/s13278-023-01064-1.

[6] Jana Medková and Josef Hynek. Application-oriented framework for social network datasets and IoT environments. *Accepted for the publication in Proceedings of the 15th International Conference, SecITC 2022*, Bucharest, Romania. Springer.

Author's other publications

[7] Josef Hynek and Jana Medková. Logic programming and reasoning puzzles. In *2022 31st Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*, pages 1–6, 2022. doi: 10.1109/EAEEIE54893.2022.9820010.

[8] Sabrina Friedl, Jana Medková, and Gunther Pernul. Privacy-preserving IoT monitoring to enhance forensic readiness. *Submitted to ARES 2023: the 18th International Conference on Availability, Reliability and Security.*

Research activities

Participation in research projects

- **2019** - Information and knowledge management and cognitive science in tourism 3 (SPEV project No. 2108)
  - responsible researcher: prof. RNDr. Josef Zelenka, CSc.
  - my role: research team member as PhD student
  - my contribution: research in anonymization in geosocial networks and the preparation of the conference article that was published in 2020
- **2020** - Information and knowledge management and cognitive science in tourism 4 (SPEV project No. 2108)
  - responsible researcher: prof. RNDr. Josef Zelenka, CSc.
  - my role: research team member as PhD student
  - my contribution: two conference articles [4, 2]
- **2020** - ICT as Support Tool for Cognitive Processes (SPEV project No. 2105)
  - responsible researcher: doc. RNDr. Petra Poulová, Ph.D.
  - my role: research team member as PhD student
  - my contribution: the conference article [3]
- **2021** - IT as Support Tool for Cognitive Processes (SPEV project No. 2105)
  - responsible researcher: doc. RNDr. Petra Poulová, Ph.D.
  - my role: research team member as PhD student
  - my contribution: research in k-automorphism anonymization and the preparation of the impact journal article
- **2022** - ICT as Support Tool for Cognitive Processes (SPEV project No. 2105)
  - responsible researcher: doc. RNDr. Petra Poulová, Ph.D.
  - my role: research team member as PhD student
  - my contribution: the completion and the submission of the impact journal article [5] and publishing two conference articles [6, 7]
- **2023** - IT as Support Tool for Cognitive Processes (SPEV project proposal)
  - responsible researcher: doc. RNDr. Petra Poulová, Ph.D.
  - my role: research team member as PhD student
  - my planned contribution: writing two conference articles

International cooperation and internship abroad

- **2021** - The 4th International Summer School for PhD students (virtual event)
  - duration: 24th - 26th August 2021 (3 days)
  - organizer: Turiba University, Riga, Latvia
  - the contribution of the summer school: develop doctoral students' scientific communication and writing competence, improve knowledge and skills in quantitative and qualitative methodology
- **2022** - The internship abroad at University of Regensburg, Germany
  - Faculty of Business, Economics and Management Information Systems; Faculty of Informatics and Data Science
  - internship at the chair of information systems Prof. Dr. Günther Pernul
  - duration: 17th January - 13th February 2022 (4 weeks)
  - the contribution of the internship: discussing findings in diferrent topics of the information security with PhD students participating in the research group of prof. Pernul, learning about working processes in other academic environment
- **2022** - Hosting visiting researcher from University of Regensburg at FIM UHK, Czech Republic
  - visitor: Sabrina Friedl - PhD student at the research group of the chair of information systems Prof. Dr. Günther Pernul in University of Regensburg
  - duration: 19th November - 2nd December 2022 (2 weeks)
  - founding: The visit was supported by the mobility grant of Czech-Bavarian agency.
  - the contribution of the visit: The visit was the result of the continuing cooperation with Sabrina that began during my internship in Regensburg. We worked together on project of digital forensic analysis on anonymized data. We prepared a concept of conference article that has been submitted to the international conference in March 2023 [8].

A. Additional experimental results

This attachment contains tables describing the results of data utility measurement with the SecGraph evaluation tool. This measurement was made for heu-kDA and HAkAu algorithms in *Chapter 6* and *Chapter 7* respectively. Because of the extent of the complete results, only the tables describing the results for the anonymization parameter $k = 15$ were included in the text, and the full results are presented here.

A.1 Data utility measurement for heu-kDA

	Polblogs		Wiki-Vote		Caida	
k=5	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.872	**0.998**	0.847	**0.996**	0.563	**0.913**
BC	**0.973**	0.972	0.951	**0.970**	**0.944**	0.943
CC	**1.000**	**1.000**	**1.000**	**1.000**	0.992	1.000
CD	**0.999**	0.915	0.917	**1.000**	0.308	**0.676**
Deg.	0.997	**0.999**	0.993	**0.998**	0.978	**0.981**
ED	0.896	**1.042**	1.012	**1.010**	0.873	**0.982**
EV	0.982	**0.999**	0.993	**0.998**	0.914	**0.950**
HS	0.849	**1.000**	0.891	**0.999**	0.593	**0.899**
Infe.	0.901	**0.913**	**0.827**	0.800	0.714	**0.774**
JD	0.520	**0.538**	**0.653**	0.652	**0.628**	0.258
LCC	0.988	**0.998**	**0.998**	**0.998**	0.938	**0.955**
NC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
PR	0.481	**1.000**	0.648	**0.995**	0.342	**0.951**
RX	**0.337**	0.000	**0.867**	0.000	**0.259**	0.000

Table A.1: Heu-kDA: utility measurement for $k = 5$. (Source: author's work.)

	Polblogs		Wiki-Vote		Caida	
k=10	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.856	**0.987**	0.839	**0.993**	0.495	**0.869**
BC	**0.966**	0.934	0.902	**0.933**	**0.883**	0.849
CC	0.999	**1.000**	0.999	**1.000**	0.997	**0.999**
CD	**0.991**	0.898	0.915	**0.948**	0.232	**0.583**
Deg.	0.989	**0.995**	0.983	**0.995**	0.920	**0.936**
ED	**0.979**	0.976	**1.014**	1.025	0.872	**1.093**
EV	0.965	**0.995**	0.987	**0.995**	0.830	**0.924**
HS	0.827	**0.998**	0.888	**0.997**	0.530	**0.816**
Infe.	**0.893**	0.861	0.768	**0.797**	**0.696**	0.575
JD	**0.286**	0.273	**0.456**	0.445	**0.514**	0.197
LCC	0.965	**0.982**	0.993	**0.999**	0.915	**0.964**
NC	**1.000**	0.998	0.999	**1.000**	0.999	**1.000**
PR	0.487	**0.999**	0.628	**0.990**	0.334	**0.902**
RX	**0.339**	0.000	**0.883**	0.000	**0.250**	0.000

Table A.2: Heu-kDA: utility measurement for $k = 10$. (Source: author's work.)

	Polblogs		Wiki-Vote		Caida	
k=15	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.849	**0.960**	0.834	**0.987**	0.491	**0.789**
BC	0.904	**0.908**	0.885	**0.910**	**0.838**	0.734
CC	0.999	**1.000**	0.999	**1.000**	0.996	**0.999**
CD	0.857	**0.888**	0.838	**0.868**	0.067	**0.549**
Deg.	0.977	**0.990**	0.965	**0.991**	0.847	**0.887**
ED	0.955	**0.956**	0.982	**1.004**	0.817	**0.985**
EV	0.957	**0.984**	0.977	**0.991**	0.807	**0.867**
HS	0.814	**0.984**	0.876	**0.993**	0.470	**0.676**
Infe.	**0.914**	0.886	**0.848**	0.805	0.700	**0.719**
JD	**0.232**	0.110	**0.318**	0.304	**0.487**	0.141
LCC	0.955	**0.984**	0.988	**0.998**	0.887	**0.938**
NC	0.998	**1.000**	**1.000**	**1.000**	0.999	**1.000**
PR	0.518	**0.996**	0.607	**0.945**	0.309	**0.803**
RX	**0.335**	0.000	**0.757**	0.000	**0.245**	0.000

Table A.3: Heu-kDA: utility measurement for $k = 15$. (Source: author's work.)

	Polblogs		Wiki-Vote		Caida	
k=15	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.843	**0.953**	0.821	**0.983**	0.473	**0.713**
BC	**0.928**	0.879	0.855	**0.885**	**0.764**	0.733
CC	0.999	**1.000**	0.999	**1.000**	0.995	**0.999**
CD	0.848	**0.922**	0.650	**0.878**	0.047	**0.544**
Deg.	**0.969**	0.985	0.951	**0.989**	0.801	**0.854**
ED	0.951	**1.040**	0.970	**0.990**	0.718	**0.963**
EV	0.945	**0.977**	0.971	**0.987**	0.789	**0.823**
HS	0.796	**0.980**	0.875	**0.988**	0.456	**0.570**
Infe.	**0.893**	0.865	**0.846**	0.795	0.776	**0.804**
JD	**0.131**	0.120	0.228	**0.262**	**0.467**	0.147
LCC	0.958	**0.984**	0.984	**0.998**	0.859	**0.941**
NC	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**
PR	0.493	**0.997**	0.584	**0.972**	0.351	**0.874**
RX	**0.341**	0.000	**0.730**	0.000	**0.224**	0.000

Table A.4: Heu-kDA: utility measurement for $k = 20$. (Source: author's work.)

	Polblogs		Wiki-Vote		Caida	
k=15	k-DA	heu-kDA	k-DA	heu-kDA	k-DA	heu-kDA
AS	0.821	**0.916**	0.798	**0.948**	0.442	**0.574**
BC	**0.854**	0.797	**0.826**	0.824	0.507	**0.627**
CC	**0.999**	**0.999**	0.998	**1.000**	0.994	**0.998**
CD	0.386	**0.753**	0.684	**0.910**	0.054	**0.482**
Deg.	0.932	**0.966**	0.905	**0.976**	0.575	**0.727**
ED	0.887	**1.049**	**0.999**	0.990	0.776	**0.994**
EV	0.925	**0.960**	0.942	**0.967**	0.691	**0.746**
HS	0.784	**0.972**	0.843	**0.960**	0.356	**0.457**
Infe.	0.904	**0.909**	**0.867**	0.817	**0.743**	0.591
JD	**0.071**	0.021	**0.105**	**0.105**	**0.444**	0.096
LCC	0.916	**0.939**	0.959	**0.997**	0.824	**0.927**
NC	**0.994**	0.988	**1.000**	**1.000**	0.998	**1.000**
PR	0.454	**0.994**	0.528	**0.928**	0.266	**0.809**
RX	**0.337**	0.000	**0.572**	0.000	**0.209**	0.000

Table A.5: Heu-kDA: utility measurement for $k = 50$. (Source: author's work.)

A.2 Data utility measurement for HAkAu

	Prefuse					Polblogs					Wiki-Vote				
k=5	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
AS	0.112	0.199	**0.412**	0.316	0.284	0.872	0.998	**0.900**	0.899	0.708	0.847	0.996	**0.877**	0.875	0.846
BC	0.956	0.969	**0.998**	**0.998**	0.741	0.973	0.972	0.988	**0.997**	0.430	0.951	0.970	0.997	**0.999**	0.565
CC	0.997	0.997	0.999	**1.000**	0.991	**1.000**	1.000	**1.000**	**1.000**	0.993	**1.000**	1.000	0.998	**1.000**	0.981
CD	0.408	0.688	0.817	**1.000**	0.127	0.999	0.915	0.888	**1.000**	0.050	0.917	1.000	0.720	**0.980**	0.015
Deg.	0.936	0.930	0.993	**0.997**	0.840	0.997	0.999	0.998	**1.000**	0.874	0.993	0.998	0.999	**1.000**	0.851
ED	0.918	1.001	0.994	**0.997**	1.049	0.896	1.042	0.950	**0.966**	0.891	1.012	1.010	**0.995**	1.009	0.980
EV	0.934	0.931	0.953	**0.988**	0.895	0.982	0.999	0.998	**0.999**	0.860	0.993	0.998	0.999	**1.000**	0.851
HS	0.156	0.046	**0.430**	0.360	0.186	0.849	1.000	0.877	**0.885**	0.851	**0.891**	0.999	0.871	0.875	0.763
Infe.	0.591	0.637	0.595	**0.757**	0.676	0.901	0.913	0.880	0.912	**0.934**	0.827	0.800	0.835	0.797	**0.867**
JD	0.763	0.041	0.211	**0.835**	0.044	0.520	0.538	0.418	**0.450**	0.139	**0.653**	0.652	0.602	0.647	0.133
LCC	0.987	0.992	0.983	**0.994**	0.775	0.988	0.998	0.956	**0.992**	0.840	0.998	0.998	0.993	**1.000**	0.804
NC	0.998	0.999	0.993	**1.000**	0.920	**1.000**	1.000	0.983	**1.000**	0.813	**1.000**	1.000	0.998	**1.000**	0.675
PR	0.497	0.981	0.534	0.531	**0.908**	0.481	1.000	0.494	0.485	**0.874**	0.648	0.995	**0.676**	0.672	0.626
RX	0.348	0.000	0.868	**0.952**	0.412	0.337	0.000	0.355	**0.564**	0.452	0.867	0.000	0.974	**0.976**	0.426

Table A.6: HAkAu: utility measurement for $k = 5$. (Source: author's work.)

	Prefuse					Polblogs					Wiki-Vote				
k=10	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
AS	0.241	0.141	**0.630**	0.382	0.283	0.856	0.987	0.892	**0.902**	0.452	0.839	0.993	**0.883**	0.880	0.806
BC	0.817	0.541	**0.967**	0.811	0.440	0.966	0.934	0.977	**0.995**	0.178	0.902	0.933	0.982	**0.993**	0.463
CC	0.994	0.973	0.927	0.943	**0.996**	0.999	1.000	**1.000**	**1.000**	0.987	0.999	1.000	0.998	**1.000**	0.981
CD	0.302	0.519	0.632	**0.984**	0.144	0.991	0.898	0.808	**1.000**	0.066	**0.915**	0.948	0.698	0.861	0.015
Deg.	0.853	0.853	0.973	**0.985**	0.702	0.989	0.995	0.995	**0.999**	0.599	0.983	0.995	0.997	**0.999**	0.831
ED	0.835	1.041	**1.018**	1.369	0.966	**0.979**	0.976	1.189	1.083	0.910	1.014	1.025	1.018	**1.006**	0.948
EV	**0.881**	0.814	0.504	0.621	0.880	0.965	0.995	0.993	**0.998**	0.652	0.987	0.995	0.996	**0.998**	0.820
HS	0.238	0.026	**0.620**	0.388	0.125	0.827	0.998	**0.879**	**0.879**	0.740	**0.888**	0.997	0.858	0.867	0.722
Infe.	0.655	0.644	0.488	0.585	**0.694**	0.893	0.861	0.917	0.887	**0.935**	0.768	0.797	0.822	0.820	**0.870**
JD	**0.643**	0.216	0.038	0.035	0.150	0.286	0.273	**0.362**	0.327	0.095	0.456	0.445	0.456	**0.505**	0.132
LCC	0.980	0.978	0.982	**0.990**	0.936	0.965	0.982	0.949	**0.982**	0.808	0.993	0.999	0.989	**0.999**	0.805
NC	0.965	0.992	0.988	**0.999**	0.970	**1.000**	0.998	0.990	**1.000**	0.639	0.999	1.000	0.997	**1.000**	0.616
PR	0.523	0.972	0.563	0.548	**0.955**	0.487	0.999	**0.494**	0.487	0.269	0.628	0.990	**0.683**	0.676	0.608
RX	0.398	0.542	0.450	**0.672**	0.489	0.339	0.000	**0.529**	0.340	0.287	0.883	0.000	0.924	**0.978**	0.342

Table A.7: HAkAu: utility measurement for $k = 10$. (Source: author's work.)

	Prefuse					Polblogs					Wiki-Vote				
k=15	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
AS	0.157	0.155	0.199	**0.383**	0.219	0.849	0.960	0.858	**0.901**	0.608	0.834	0.987	**0.886**	0.881	0.799
BC	0.528	0.526	**0.834**	**0.834**	0.615	0.904	0.908	0.971	**0.994**	0.420	0.885	0.910	0.966	**0.991**	0.477
CC	**0.995**	0.961	0.913	0.943	**0.995**	0.999	1.000	**1.000**	**1.000**	0.993	0.999	1.000	**1.000**	**1.000**	0.981
CD	0.214	0.509	**0.810**	0.715	0.102	0.857	0.888	0.756	**1.000**	0.061	0.838	0.868	0.755	**0.960**	0.018
Deg.	0.795	0.784	0.973	**0.983**	0.779	0.977	0.990	0.991	**0.998**	0.826	0.965	0.991	0.992	**0.999**	0.817
ED	0.802	1.164	1.374	1.381	**0.826**	0.955	0.956	1.132	**0.987**	0.844	0.982	1.004	**1.013**	0.984	0.911
EV	0.904	0.718	0.411	0.692	**0.905**	0.957	0.984	0.966	**0.997**	0.792	0.977	0.991	0.989	**0.998**	0.800
HS	0.276	0.022	0.201	**0.381**	0.041	0.814	0.984	0.816	**0.881**	0.806	**0.876**	0.993	0.846	0.865	0.674
Infe.	**0.694**	0.644	0.617	0.602	0.646	0.914	0.886	0.896	0.895	**0.928**	0.848	0.805	0.835	0.850	**0.870**
JD	**0.660**	0.215	0.029	0.124	0.005	**0.232**	0.110	0.230	0.224	0.080	0.318	0.304	0.353	**0.397**	0.087
LCC	0.972	0.974	0.970	**0.996**	0.880	0.955	0.984	0.905	**0.983**	0.827	0.988	0.998	0.988	**1.000**	0.774
NC	0.982	0.986	0.982	**0.993**	0.957	0.998	1.000	0.947	**1.000**	0.781	**1.000**	1.000	0.996	**1.000**	0.712
PR	0.572	0.976	0.537	0.544	**0.949**	0.518	0.996	0.498	0.487	**0.666**	0.607	0.945	**0.690**	0.678	0.588
RX	0.460	0.000	0.937	**0.933**	0.490	0.335	0.000	0.332	0.346	**0.398**	0.757	0.000	0.889	**0.971**	0.379

Table A.8: HAkAu: utility measurement for $k = 15$. (Source: author's work.)

	Prefuse					Polblogs					Wiki-Vote				
k=20	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu	k-DA	heu-kDA	tMean	Union	HAkAu
AS	**0.381**	0.145	0.360	0.290	0.177	0.843	0.953	0.660	**0.902**	0.371	0.821	0.983	**0.888**	0.881	0.788
BC	0.721	0.646	0.720	**0.771**	0.530	0.928	0.879	0.972	**0.987**	0.201	0.855	0.885	0.970	**0.985**	0.485
CC	0.990	0.957	0.949	**0.996**	0.993	0.999	1.000	0.999	**1.000**	0.988	0.999	1.000	0.997	**1.000**	0.981
CD	0.033	0.420	**0.823**	0.771	0.102	0.848	0.922	0.830	**1.000**	0.057	0.650	0.878	0.860	**0.966**	0.018
Deg.	0.672	0.729	0.966	**0.984**	0.720	0.969	0.985	0.980	**0.997**	0.558	0.951	0.989	0.994	**0.998**	0.801
ED	0.782	1.030	1.201	1.366	**0.961**	0.951	1.040	1.099	**1.034**	0.797	0.970	0.990	1.019	**0.989**	0.841
EV	0.715	0.697	0.476	0.706	**0.860**	0.945	0.977	0.768	**0.996**	0.590	0.971	0.987	0.992	**0.997**	0.791
HS	0.156	0.018	**0.362**	0.300	0.031	0.796	0.980	0.578	**0.882**	0.697	**0.875**	0.988	0.849	0.863	0.665
Infe.	**0.753**	0.702	0.621	0.545	0.633	0.893	0.865	0.900	0.899	**0.933**	0.846	0.795	0.813	0.873	**0.899**
JD	0.044	0.018	0.016	**0.083**	0.011	0.131	0.120	**0.252**	0.171	0.050	0.228	0.262	**0.378**	0.324	0.066
LCC	0.873	0.982	0.947	**0.986**	0.947	0.958	0.984	0.920	**0.968**	0.801	0.984	0.998	0.969	**0.999**	0.755
NC	0.858	0.990	0.983	**0.995**	0.958	**1.000**	1.000	0.981	**1.000**	0.614	**1.000**	1.000	0.991	**1.000**	0.779
PR	0.661	0.966	0.561	0.539	**0.954**	0.493	0.997	**0.496**	0.492	0.295	0.584	0.972	**0.691**	0.680	0.598
RX	0.355	0.000	0.700	**0.914**	0.465	0.341	0.000	0.345	**0.591**	0.290	0.730	0.000	**0.913**	0.906	0.379

Table A.9: HAkAu: utility measurement for $k = 20$. (Source: author's work.)

B. Supplementary material

The disc attached to this work contains the full text of the doctoral thesis and the MATLAB code of the proposed composition attack and anonymization methods heu-kDA and HAkAu. Used external tools and input data for testing are also included. The SecGraph evaluation tool is not included with reference to [59]. The content of the disc has the following structure:

Thesis - includes the pdf file with the thesis text.

SNdatasets - includes three text files with SN datasets for testing

- *prefuse_edges.txt* - the edge set of the Prefuse dataset [51]
- *polblogs_edges.txt* - the edge set of the Polblogs dataset [2]
- *Wikivote_edges.txt* - the edge set of the WikiVote dataset [80]

CompAttack - includes output files, **.m* files with the MATLAB code of the composition attack, UTD anonymization toolbox [142], MATLAB implementation of BA model [38] and auxiliary files

- **input** - excel files with attribute tables U_A for input dataset that will be generated with BA model
- **output** - output files with results
- **pom** - auxiliary files
- *workflow.m* - the main file that should be run to perform the attack. At the beginning of this file, the user sets the input parameters.
- *A_graph_gen.m* - runs SFNG and anonymization functions from the UTD toolbox to generate two graphs representing SN datasets. The attribute tables *aNT1, aNT2* have to be set in this file.
- *BB_processing.m* - the preprocessing stage
- *C_composition.m* - the composition attack
- *D_reducing.m* - reducing the cardinality of the output set
- *E_postreducing.m* - the postprocessing stage
- *F_results.m* - producing results
- *G_writeresults.m* - exporting results
- *H_rocg.m* - ROC analysis
- *SFNG.m* - the main function from B-A model [38]
- *CNet.m* - the auxiliary function from B-A model [38]

- *PLplot.m* - the auxiliary function from B-A model [38]
- *anonymization.bat* - the executable file starting the anonymization with UTD toolbox [142]
- **.jar, *.bat, sqlite.dll, config.xml, original_config* - UTD toolbox [142]

HeukDA - includes output files, **.m* files with the MATLAB code of the heu-kDA and auxiliary files. Before the procedures are executed, **SNdatasets** folder has to be added to the path in MATLAB.

- **output** - output files with results
- **output_datasets** - output files with anonymized graphs
- *heu_worklow.m* - the main file that should be run to anonymize the input dataset with heu-kDA. At the beginning of this file, the user sets the input parameters.
- *heu_anonym.m* - the anonymization procedure
- *heu_edgeedit.m* - the edge editing procedure
- *heu_edgeswitch.m* - the edge switching procedure
- *heu_getgreedy.m* - finding the degree sequence of the input graph
- *heu_greedyedit.m* - the high-degree node noise addition procedure
- *heu_GVmeasures.m* - data utility measurement in both input and output graphs
- *heu_smallGVmeasures.m* - data utility measurement only in the output graph
- *resultstoexcel.m* - exporting results
- *avg_cc.m* - the computation of the average clustering coefficient
- *cc.m* - the computation of the clustering coefficient
- *distG.m* - the computation of the average distance
- *transG.m* - the computation of the transitivity
- *uploadGV.m* - importing the input dataset
- *GVrandomswitch.m* - auxiliary file
- *rungreedy2.m* - auxiliary file
- *getoutput.m* - auxiliary file
- *setoutput.m* - auxiliary file
- *greedy2.txt* - auxiliary file

HAkAu - includes output files, **.m* files with the MATLAB code of the HAkAu, the implementation of GraMi algorithm [30] and auxiliary files. Before the procedures are executed, **SNdatasets** and **HeukDA** folders must be added to the path in MATLAB. The MATLAB Global Optimization Toolbox [92] is required for the computation.

- **input** - input to the GraMi algorithm generated with MATLAB code
- **output** - output files with results and resulting anonymized datasets
- **temp** - temporary files
- **GraMi-master** - the implementation of GraMi algorithm [30] customized to the usage in HAkAu
- *test.m* - starting the repeated run of HAkAu
- *testing.m* - one run of HAkAu including preprocessing and postprocessing stages
- *iworkflow_ga.m* - the workflow of one run of HAkAu
- *iworkflow_part0.m* - setting parameters of HAkAu. At the beginning of this file, the user sets the input parameters.
- *iworkflow_part1.m* - setting input parameters of GraMi and running it
- *iworkflow_part2.m* - processing the GraMi output and setting the input to GA
- *iworkflow_part3.m* - running GA. At the beginning of this file, the user sets the input parameters of GA.
- *buildgraphs.m* - building graphs from chromosomes
- *input2grami.m* - preparing the input to GraMi
- *makeadjacency.m* - making the adjacency matrix from GraMi output
- *remfalse.m* - removing incorrect rows
- *finduniquerows2.m* - finding the subset of vertex-disjoint graphs
- *doubleselection.m* - the selection function in GA
- *nonscaling.m* - the scaling function. No scaling.
- *fitscalingrank2.m* - the scaling function. Rank scaling.
- *makeTPNodes.m* - preparing the list of subgraphs $P_{i1}, \ldots, P_{im}$
- *icreate_popCH.m* - creating the initial population
- *varlength.m* - computing the length of *varCH*

- *varvaluescoulette.m* - the roulette selection for selecting new elements of *varCH*
- *mymutation.m* - the mutation procedure of GA
- *mutationCH.m* - the mutation of *CH*
- *mutationvarCH.m* - the mutation of *varCH*
- *icrossrandomOX.m* - the crossover procedure of GA
- *inside_crossovertwopoint.m* - two-point crossover function of *CH*
- *irandomvvOX.m* - two-point crossover function of *varCH*. The crossover is applied on the whole *varCH*.
- *irandomvvpartOX.m*- two-point crossover function of *varCH*. The crossover is applied on the parts of *varCH* representing particular subgraphs separately.
- *otocpole.m* - the auxiliary function
- *removemiddle.m* - the auxiliary function
- *iff.m* - the fitness function of GA
- *countcosts_mat.m* - computing $ExCost(G, G^*)$ and $CECost$ from the chromosome
- *changeG.m* - postprocessing after GA. The subgraphs $P'_{i1}, \ldots, P'_{im}$ are removed from the graph H.
- *prepare2nextGA.m* - postprocessing after GA. Crossing edges and subgraphs $P'_{i1}, \ldots, P'_{im}$ are exported.
- *checkvarCH.m* - the auxiliary function
- *makematrix.m* - building $P'_{i1}, \ldots, P'_{im}$ from $P_{i1}, \ldots, P_{im}$ and the chromosome
- *results2excel.m* - exporting results after one run of GA
- *write2file.m* - the auxiliary function
- *restnodes2izo.m* - modifying $V(H)$ after all runs of HAkAu to isomorphisms
- *crossing.m* - adding crossing edges into the resultant graph
- *crossing_degree.m* - adding crossing edges into the resultant graph. This version is based on the nodes' degree.
- *fullresults2excel.m* - exporting results at the end of HAkAu
- *heu_measures.m* - computing network metrics of the output graph
- *clusteringcoef.m* - the computation of the clustering coefficient

- *makeauxiliary.m* - making the auxiliary graph for testing the resistance against deanonymization attacks
- *setGlobalx.m* - the auxiliary file
- *getGlobalx.m* - the auxiliary file
- *setfreq.pl* - setting input parameters to GraMi according to the input parameters of HAkAu
- *uploadrelax.pl* - postprocessing after the run of GraMi. The user should change the paths in this file.