Universiteit Leiden

# Master Computer Science

Anonymization Algorithms for Privacy-Sensitive Networks


| | |
|---|---|
| Name: | Xinyue Xie |
| Student ID: | s3284778 |
| Date: | 25/08/2023 |
| Specialisation: | Data Science |
| Supervisors: | Dr. Frank W. Takes<br>Rachel G. de Jong MSc<br>Dr. Mark P. J. van der Loo |


Master's Thesis in Computer Science


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

The widespread utilization of graph data across various domains has led to increased data sharing. Consequently, as these data may contain sensitive information, preserving privacy in graph data has become an important concern. In this thesis, we explore how to preserve the privacy of individuals by maximizing anonymity within a given "budget", i.e., allowed number of modifications to the graph. We measure node anonymity by ego network size and ego network structure and propose various so-called anonymization algorithms to increase overall node anonymity. We introduce and apply the proposed methods to five datasets and compare node anonymity, algorithm runtime, and effect on network structure based on various metrics.

# 1 Introduction

In modern times, graph data are used for various fields, including social network analysis, disease spread, and trade. There has been an important increase in interest in graph data, leading to increasing demand for sharing graph data for research purposes. However, some graph data may contain sensitive information, such as social connections on platforms like Facebook [16, 22], resulting in the need to prioritize privacy preservation in graphs. There are two major types of disclosure [15]: identity disclosure, which involves identifying individuals, and attribute disclosure, which infers specific individual traits. In this thesis, we focus on the former. One approach to avoid identity disclosure is naive user identity (ID) removal. Unfortunately, the work of Backstrom et al. [2] points out that simply removing the ID of the users before publishing the data does not guarantee privacy because individuals can be re-identified based on structural information of the graph.

Anonymization can prevent the identification of individuals under an assumed certain attack scenario by removing or modifying data. Simultaneously, it should preserve the structure and enable meaningful analysis. Research on anonymization can be divided into two main aspects: anonymity measures and anonymization methods. Based on different anonymity measures, current anonymization methods can be primarily categorized into three types: *k-anonymity*-based methods [5, 18, 31, 32], clustering-based methods [13, 24], differential-privacy-based methods [9, 10]. The anonymity method *k-anonymity* is commonly used in graph data, which ensures each node in a dataset is indistinguishable from at least $k-1$ other nodes with respect to a certain amount of knowledge. The knowledge corresponds to attack scenarios and to some extent influences anonymization methods.

The work of Liu and Terzi [18] proposed *k-Degree Anonymity* ($k$-DA) to measure the anonymity of the graph. In $k$-DA, nodes are considered anonymous when there are at least $k-1$ nodes with the same degree. They provided an anonymization method for this anonymity measure by generating a degree sequence and then adding edges according to the degree sequence. Also based on $k$-DA, Lu et al. [19] proposed an anonymization method that simultaneously adds edges to the graph and updates the degree sequence. Chester et al. [6] also consider this anonymity measure. After generating the degree sequence, they anonymize the graph by adding fake nodes and adding edges between real nodes and fake nodes.

Zhou and Pei [31] focused on the structure of nodes' neighborhoods and introduced a novel anonymity measure known as *k-Neighborhood Anonymity* ($k$-NA). They also developed a greedy method to resist the attack. Zou et al. [32] took all structural information into account and proposed *k-automorphism* to measure anonymity. They also presented the *k-Match* algorithm to ensure the anonymity of nodes. The $k$-Match algorithm first partitions the graph into blocks and divides similar blocks into groups, and then adds edges to anonymize the blocks in the group. Similarly, Cheng et al. proposed *k-isomorphism* [5] as an anonymity measure and explored

various methods to achieve *k-isomorphism* based on graph embeddings. Based on $k$-NA, Romanini et al. [21] proposed using the uniqueness of neighborhoods as a measure, giving a more macro-level perspective. They also explored the influence of density in random networks on uniqueness, and proposed to anonymize the graph by randomly deleting edges from the original graph. More recently, de Jong et al. [7, 8] provided algorithms that efficiently measure *d-k-anonymity*, offering the flexibility to adjust the scope of the neighborhood through the parameter $d$.

Most of the aforementioned anonymization methods focus on ensuring anonymity for each node in the graph. However, in some scenarios, such methods may not be efficient enough, especially when dealing with graphs that contain numerous unique nodes. These methods could lead to a large amount of changes, affecting the network structure and consequently influencing various analytical results. Therefore, we attempt to introduce anonymization methods that anonymize the graph while limiting the number of changes to the graph. In this thesis, our main research question is:

*How can we efficiently anonymize a given network as much as possible, while making as few changes to the graph structure as possible?*

We present five methods for this main research question, addressing on the following sub-questions:

- Which graph modification operations are most useful for anonymizing graphs?
- Given a particular anonymity measure, which method of anonymization is most effective at anonymizing a given network?
- How does the runtime of various network anonymization methods depend on the structural properties of the considered networks?
- Deleting which types of edges is most useful to increase overall network anonymity?

A brief overview of privacy-preserving techniques in social networks is presented in Section 2. In Section 3, we introduce definitions of some commonly used terms in graph analysis and anonymity in networks. Section 4 discusses the methods we used. Section 5 introduce the datasets we used. The experimental setup and the results are included in Section 6. In the last section, a conclusion is provided.

# 2 Related Work

There are several graph anonymization techniques that are commonly used to protect sensitive data, such as *k-anonymity*, differential privacy, and data aggregation. In this section, we give a brief overview of current techniques for graph anonymization.

Clustering-based methods can also be considered generalization-based methods or aggregation-based methods. It combines data at a higher level of granularity by combining some similar nodes and edges to *supernodes* and *superedges* [4, 13, 29], to protect privacy. Differential privacy is also used in graph anonymization. This approach is usually applied by allowing users to query data without sharing the full graph. Differential-privacy-based methods add noise in the query results or in actual data [23]. They can be further divided into node differential privacy [20, 23] and edge differential privacy [1, 12]. Another branch is *k-anonymity*-based methods. A graph satisfies *k-anonymity* when each node in this graph is indistinguishable from at least $k-1$ other nodes. The *k-anonymity*-based methods modify the original graph to ensure *k-anonymity*, and release the modified graph.

A comparison between a *k-anonymity*-based method [19] and a clustering method [4] was conducted by Campan et al. [3]. According to the results, the *k-anonymity*-based method can better preserve the local structure of the graph. Moreover, since the differential-privacy-based methods do not release the full graph, analysis of the data will be limited. Therefore, in this thesis, we focus on *k-anonymity*-based methods.

Two aspects need to be discussed for *k-anonymity*-based methods: the anonymity measure and the anonymization algorithm. The level of attacker knowledge determines what kind of nodes are indistinguishable, thus, affecting the measure of anonymity. Liu and Pei [18] introduced $k$-DA, measuring anonymity by degree. Hay et al. [14, 13] extended it to *k-candidate*, using the degree set of neighbors to model the adversary's knowledge. Zhou and Pei [31] introduced $k$-NA, which takes the structure of the neighborhood of nodes into account. Zou et al. [32] not only consider the structure of the neighborhood, but take all the graph structure into account and presented *k-automorphism*. Similarly, Cheng et al. [5] and Wu et al. [26] proposed *k-isomorphism*, which describes graphs containing $k$ disjoint isomorphic subgraph, and *k-symmetry*, which requires automorphism of node partition, respectively. Then, van der Loo [25] proposed *d-k-anonymity*, which enlarges the size of the neighborhood of $k$-NA. The radius of the neighborhood can be adjusted by parameter $d$.

Many methods were developed for graph anonymization. Some of them are measure-agnostic, like randomly deleting [21] from the original graph, or deleting edges and adding edges at the same time [14, 27, 28]. Many other anonymization methods are measure-specific. For $k$-DA, Liu and Pei [18]'s method first construct a degree sequence that ensures $k$-DA. Then, they calculate the difference between the original degree sequence and the constructed degree sequence and add edges according to the calculated results. Then, Zhang et al. [30] proposed an edge plausibility

metric and developed an improved edge-adding approach that extends Liu and Pei's method to some extent. Chester et al. [6] also focus on $k$-DA, and anonymized the graph by adding fake nodes according to the degree sequence, and connecting the fake nodes with real nodes. For $k$-NA, Zhou and Pei [31] first greedily group the nodes with similar neighborhoods together. Then, they make nodes in the same group equivalent.

In this thesis, we focus on *k-anonymity*-based edge deletion methods. The measures we considered are *d-k-anonymity* [7, 8, 25] and *(n,m)-anonymity*, introduced in Section 4.1. We focus on graph uniqueness as used in the work of Romanini et al. [21]. For the anonymization method, we propose to use four methods that are measure-agnostic and one greedy method specific to *(n,m)-anonymity*.

# 3 Preliminaries

In this section, we summarize notation and definitions that are commonly used to analyze graph data and describe anonymity in networks.

## 3.1 Graphs

Let a graph $G = (V, E)$ be used to model the network, where $V$ is the set of *nodes* and $E$ is the set of *edges*. Here, $v \in V$ is used to represent the individuals in the network, and $\{u, v\} \in E$ is a connection between two individuals, with $u, v \in V$. The number of nodes is denoted as $|V|$, and $|E| \leq \binom{|V|}{2}$ denotes the number of edges.

For two nodes $u, v \in V$, the *distance* between $u$ and $v$, denoted as $dist(u, v)$, is the minimum number of edges that need to be traversed to get from one node to the other. Since our graphs are undirected, it follows that $dist(u, v) = dist(v, u)$. If there is no path between $u$ and $v$, $dist(u, v)$ is defined as $\infty$. For the node itself, it follows that $dist(v, v) = 0$. The number of nodes for which the distance to $v$ equals 1 is called the *degree* of $v$, denoted as $deg(v)$.

When looking at the $d$-neighborhood of a node $v$, we look at the subgraph containing all the nodes whose distance from it is not more than $d$ and all edges between these nodes, as defined below. When $d = 1$ we look at the 1-neighborhood, commonly known as the *ego network*.

**Definition 3.1** (*d-Neighborhood*)**.** Given a graph $G = (V, E)$ and a node $v \in V$, the $d$-neighborhood of $v$, denoted as $N_d(v)$, is defined as $N_d(v) = (V_d(v), E_d(v))$, where $V_d(v) = \{u \in V | dist(u, v) \leq d\}$, $E_d(v) = \{\{u, w\} \in E | u, w \in V_d(v)\}$.

For $u, v, w \in V$, we call the triplet $\{u, v, w\}$ a *triangle* if $\{u, v\}, \{v, w\}, \{u, w\} \in E$. The *transitivity* and *average clustering coefficient (ACC)* quantify the tendency of nodes in a graph to form triangles. Transitivity is the ratio of actual triangles to the potential triangles within the graph.

$$Transitivity(G) = \frac{3 \cdot |\{\{u, v, w\} | \{u, v\}, \{u, w\}, \{v, w\} \in E\}|}{|\{u \in V | \{u, v\}, \{u, w\} \in E\}|} \tag{1}$$

Given a node $v \in V$, the clustering coefficient $CC(v)$ is defined as:

$$CC(v) = \frac{2 \cdot |\{\{u, w\} \in E | \{u, v\}, \{v, w\} \in E\}|}{deg(v) \cdot (deg(v) - 1)} \tag{2}$$

The $ACC$ of a graph is defined as:

$$ACC(G) = \frac{1}{|V|} \sum_{v \in V} CC(v) \tag{3}$$

Another important term in graph analysis is *ismorphism*, which determines whether two graphs are structurally indistinguishable.

**Definition 3.2** (Graph isomorphism)**.** Given two graphs $G = (V, E)$ and $G' = (V', E')$, $G$ and $G'$ are isomorphic if there is a bijection $\phi : V \to V'$, such that $\{u, v\} \in E$, precisely when $\{\phi(v), \phi(u)\} \in E'$.

## 3.2 Uniqueness of a graph

We say that a node is *k-anonymous* when it is indistinguishable from at least $k - 1$ other nodes using some measure for equivalence. The measure for equivalence used corresponds to a scenario, where we assume an attacker has this information. When two nodes are indistinguishable given a measure, denoted as $v \simeq u$ for $u, v \in V$, they are in the same *equivalence class*. A node is *unique* when no other nodes are in the same equivalence class as this node. For $v \in V$, uniqueness of a node $\delta_m(v)$ is defined as

$$\delta_m(v) = \begin{cases} 1, & \{u \in V | u \simeq_m v\} = v \\ 0, & otherwise \end{cases} \tag{4}$$

The uniqueness of a graph is defined as the fraction of unique nodes.

$$U_a(G) = \frac{\Sigma_{v \in V} \delta_m(v)}{|V|} \tag{5}$$

where $a$ denotes the chosen anonymity measure (see Section 4.1).

# 4 Approach

In this section, we introduce our approach in detail, which includes the two anonymity measures and the five anonymization methods. The two anonymity measures we use are *(n,m)-anonymity* and *d-k-anonymity*, which are introduced in Section 4.1. The modification operations are presented in Section 4.2, consisting of random edge deletion as baseline in Section 4.2.1, two heuristics combined with degree and uniqueness properties of nodes in Section 4.2.2 and Section 4.2.3 respectively, a logistic-regression-based method in Section 4.2.4 and a greedy method aimed at *(n,m)-anonymity* in Section 4.2.5.

Table 1 gives a brief overview of these anonymization methods. The results of random methods and the two heuristics based on random edge deletion are nondeterministic and the *(n,m)*-greedy method is specific to *(n,m)-anonymity*.

Table 1: The anonymization methods used in this thesis. The middle row indicates whether this anonymization method is deterministic. The last column shows whether the methods are measure-specific.

| Method | Deterministic | Measure-specific |
|---|---|---|
| Random deletion/addition/rewiring | ✗ | ✗ |
| Degree-based deletion | ✗ | ✗ |
| UA-based deletion | ✗ | ✗ |
| LR-based deletion | ✓ | ✗ |
| *(n,m)*-greedy deletion | ✓ | ✓ |

## 4.1 Anonymity measures

We introduce anonymity measures based on two different levels of background knowledge of the attackers.

### 4.1.1 *(n,m)-Anonymity*

Here we assume that in a social network, individuals can see the list of their own friends and how many common friends they have. Suppose the attacker obtained this information, then he or she can recognize a specific individual in the graph by comparing the number of nodes and edges in the person's ego network.

We define *(n,m)-anonymity* to describe the scenario that an attacker knows the number of nodes $n$ and the number of edges $m$ in a node's 1-neighborhood. The measure of *(n,m)-anonymity* can be extended to different distances of the neighborhood. Here, we focus on ego networks and therefore refer to $(n, m)$ as the *ego state*.

**Definition 4.1** (Ego State)**.** Given a graph $G = (V, E)$ and a node $v \in V$, the ego state of $v$, denoted $S(v)$, is defined as $S(v) = (n, m)$, where $n = |V_1(v)|, m = |E_1(v)|$.

Then, *(n,m)-anonymity* is defined as below.

**Definition 4.2** (*(n,m)-Anonymity*)**.** Given a graph $G = (V, E)$, two nodes $u, v \in V$ are in the same equivalence class, denoted as $u \simeq_{nm} v$, when $S(u) = S(v)$. The size of the equivalence class of a node is called its *(n,m)-anonymity*.

For example, in Figure 1a, $V_1(B) = \{A, B\}$, $E_1(B) = \{\{A, B\}\}$, thus, $S(B) = (n_B, m_B) = (2, 1)$. $A \simeq_{nm} G$, $E \simeq_{nm} F \simeq_{nm} H$ because they have the same number of nodes and edges in their ego networks.

### 4.1.2 *d-k-Anonymity*

The anonymity measure of *d-k-Anonymity* [25] extends $k$-NA to a larger neighborhood size. Other than *(n,m)-anonymity*, *d-k-anonymity* considers the full structural information of the neighbourhood. *d-k-Anonymity* is a stricter measure, as it assumes complete attacker knowledge of the neighbourhood structure up to and including distance $d$. The definition of *d-k-anonymity* is shown below. For nodes $u, v \in V$, if $u \simeq_{dk} v$ it follows that $u \simeq_{nm} v$.

**Definition 4.3** (*d-k-Anonymity*)**.** Given a graph $G = (V, E)$ and two nodes $u, v \in V$, $u$ and $v$ are $d$-equivalent, denoted as $u \simeq_{dk} v$; there is an isomorphism $\phi : N_d(u) \to N_d(v)$, such that $\phi(u) = v$. The size of the equivalence class of a node is called its *d-k-anonymity*.

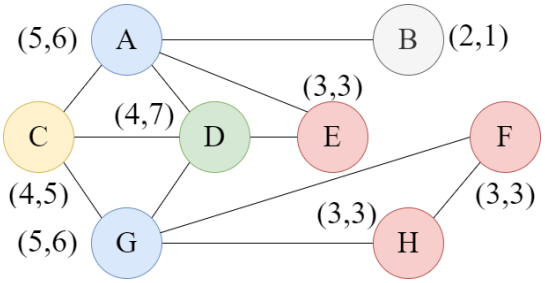
The parameter $d$ can be used to define the knowledge of the attacker. In this thesis, we assume the attacker only has information on the ego networks, therefore, $d$ is set to 1.

Figure 1b gives an example of a graph where nodes are partitioned based on *d-k-anonymity*. Compared to Figure 1a, nodes $A$ and $G$ are not equivalent anymore, as there is a degree 1 node $B$ in $N_1(A)$ that no node in $N_1(G)$ can map to.

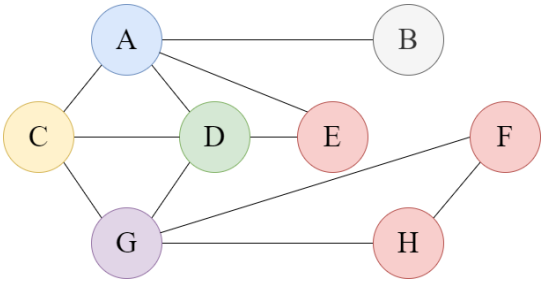
We use the two anonymity measures to determine which nodes are unique and to calculate the uniqueness of the graph as a whole.

## 4.2 Edge deletion methods

In this section, we introduce the anonymization methods we used in this thesis. We start with introducing three modification operations, addition, rewiring and deletion, and the corresponding random algorithm in Section 4.2.1. Then, we focus on algorithms using the edge deletion modification, and the random edge deletion as our baseline. We propose two heuristics (see Section 4.2.2 and Section 4.2.3), an approach




(a) Equivalence classes based on *(n,m)-anonymity*.




(b) Equivalence classes based on *d-k-anonymity*.

Figure 1: Figure 1a and 1b show the equivalence nodes based on *(n,m)-anonymity* (left) and *d-k-anonymity* (right), respectively. Nodes in the same color are in the same equivalence class. The 2-tuple node labels in 1a denote the ego states of each node.

using an LR model (see Section 4.2.4), and a greedy algorithm for *(n,m)-anonymity* (see Section 4.2.5).

Algorithm 1 presents the overview of the deletion process. Unless stated otherwise, the methods in the following section follow the same steps, with the only difference being the algorithm for selecting the edges to be deleted, shown in line 5. Variable $G$ refers to the original graph, $N$ refers to the number of edges modified in total, $M$ refers to the anonymization method discussed in the following section, and $a$ refers to the anonymity measure discussed in Section 4.1. Besides, all the methods, except the baseline, include calculating values corresponding to the anonymity measure. Therefore, we need to recompute the anonymity of all the nodes during deletion, which is time-consuming. At the same time, deleting a single edge or a few edges often has little effect on the remaining edges, and hence on the selection procedure, especially when the networks are large. Therefore, we compute the uniqueness after $g$ edge deletions. Parameter $g$, called the recompute gap, is used to balance the performance and the running time. It describes how many edges are deleted before we update the unique node set.

### 4.2.1 Random edge addition/deletion/rewiring

We perform graph modification by the three following random operations, where the probability of deletion for all edges is uniform:

- Random edge addition. Given a graph $G = (V, E)$, randomly add non-existing edges to $E$.
- Random edge deletion. Given a graph $G = (V, E)$, randomly delete existing edges from $E$.

**Algorithm 1** Graph anonymization by edge deletion

**Input:** original graph $G = (V, E)$, budget $N$, method $M$, measure $a$, recompute gap $g$

$G' \leftarrow G$;
$t \leftarrow initialize(N, g)$; ▷ Initialize the number of iterations
$U_a, V_{unique} \leftarrow unique(G', a)$; ▷ Compute the uniqueness and the unique nodes
**while** $t > 0$ **do**
    $E_{del} \leftarrow select(G', g, M, V_{unique})$; ▷ Determine the $g$ edges to be deleted
    $delete(G', E_{del})$;
    $t \leftarrow t - 1$;
    $U_a, V_{unique} \leftarrow unique(G', a)$;
**end while**

**Output:** $G'$, $U_a$

- Random edge rewiring. Given a graph $G = (V, E)$, randomly choose several edge pairs, and replace each pair $\{u, v\}, \{u', v'\}$ by $\{u, v'\}, \{u', v\}$.

Based on Romanini et al.'s [21] results and the preliminary experiment in Section 6.2, which shows that edge deletion is by far the most efficient, we decided to use random edge deletion as a baseline and explore how to make the edge deletion process more effective.

The recompute gap $g$ and the anonymity measure $a$ have no impact on the edges chosen by random edge deletion.

### 4.2.2 Degree-based deletion

Given our aim to anonymize the graph with a limited number of modifications, it is important to note that not all nodes can be anonymized. To achieve the greatest possible overall anonymity for the graph, it becomes necessary to disregard certain nodes. In a social network, degree distributions typically follow a power law, which means the majority of nodes have few edges, while a few nodes have a huge number of edges connected to them. Thus, the degree can be very different if we choose two high-degree nodes. Anonymizing these nodes can hence be difficult, as numerous deletions are likely required. For example, given that nodes $u, v$ are nodes with the highest degree in a graph, e.g., $deg(v) = 200$, $deg(u) = 150$, if we want to anonymize $v$, at least 50 edges need to be deleted, by which various analytical results of the graph will likely be affected. However, low-degree nodes are usually more numerous in social networks, with less various structures of ego networks. Hence, these low-degree nodes are easier to anonymize.

Therefore, we want deletions to occur between low-degree nodes, as we expect that these have more effect on anonymity. However, anonymous nodes are also more common in nodes with lower degrees. Simply selecting edges to delete based on

degrees could significantly perturb nodes that are already anonymous, which is not our expected outcome. Hence, we set a weight $W_{deg} = 0$ here to make sure the deletion will not touch the anonymous nodes.

The set of edges between unique nodes is denoted $E_{unique} = \{\{u, v\} \in E | u, v \in V_{unique}\}$. We introduce a heuristic based on the degree. For each edge, we assign a weight according to Equation 6.

$$W_{deg}(u, v) = \begin{cases} 0, & \{u, v\} \notin E_{unique}, |E_{unique}| \geq g \\ \frac{1}{\max(deg(u), deg(v))}, & otherwise \end{cases} \tag{6}$$

As mentioned in Algorithm 1, we choose $g$ edges to delete before updating the uniqueness, hence, we need at least $g$ positive weights. To avoid edges with positive weights being insufficient, before calculating weights, we first determine $|E_{unique}|$ is not less than $g$.

With the weights of all edges, the probability of deleting an edge would be:

$$P_{deg}(u, v) = \frac{W_{deg}(u, v)}{\sum_{i,j \in V} W_{deg}(i, j)} \tag{7}$$

#### 4.2.3 UA-based deletion

Another perspective is to consider what kind of nodes will be influenced when we delete an edge. We use the number of affected unique nodes (U) and anonymous nodes (A) to define the weight of edges.

As we focus on the ego network, deleting an edge will affect the two nodes that are connected by it and the nodes that include the edge in a triangle. For example, in Figure 1, if we delete edge $\{C, G\}$, the ego network of node $C, G$ and $D$ would be changed. So we can get the set of the affected nodes $V_{eff}(\{u, v\})$.

$$V_{eff}(\{u, v\}) = \{V_1(u) \cap V_1(v)\} \tag{8}$$

This heuristic follows the intuition that the best edge to delete should affect as many unique nodes as possible and as few anonymous nodes as possible. We use $U_{eff}$ and $A_{eff}$ to denote the set of affected unique nodes and anonymous nodes.

$$U_{eff} = \{v \in V_{eff} | \delta_m(v) = 1\} \tag{9}$$

$$A_{eff} = \{v \in V_{eff} | \delta_m(v) = 0\} \tag{10}$$

Then, the weight of the edges is defined as

$$W_{ua}(u, v) = \frac{|U_{eff}| + 0.01}{|A_{eff}| + 0.01} \tag{11}$$

Here, we added 0.01 to the denominator and numerator respectively to prevent division by zero in Equation 11 and Equation 12. We summarize the probability of deleting and edge in Equation 12.

$$P_{ua}(u,v) = \frac{W_{ua}(u,v)}{\sum_{i,j \in V} W_{ua}(i,j)} \tag{12}$$

#### 4.2.4 Logistic regression-based deletion

After having the degree and uniqueness properties of the edges as defined above, we ask the question of how to combine them together to better determine which edges should be deleted. This problem can also be considered a binary classification problem. We have a set of edges, and the task is to classify them to either "delete" or "not to delete".

*Logistic Regression* (LR) is a commonly used technique for solving binary classification problems. It is a discriminative model based on probability theory and maximum likelihood estimation.

Figure 2 presents the process of using the LR model for edge deletion. We start with preprocessing the networks to generate training data. Then, we use these data to train the LR model. Next, we utilize the trained model to predict which edges to delete to anonymize the graph. Finally, deleting edges based on the predicted result, we have a modified graph.

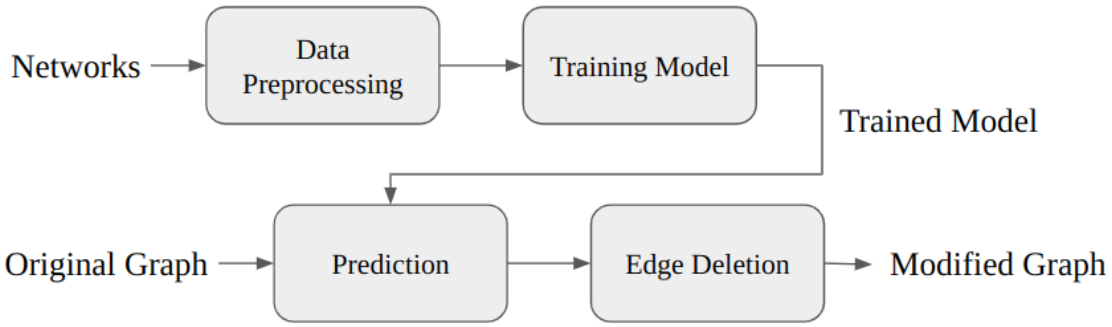



Figure 2: Steps of LR-based deletion.

For the data preprocessing step, we first identify the feature and the label. For each edge $\{u, v\} \in E$, we use eight features to predict the effect of deleting it:

- $|V|$: the number of nodes;
- $|E|$: the number of edges;
- $T$: the number of triangles it is part of;

- $\max(deg(u), deg(v))$;
- $\min(deg(u), deg(v))$;
- $|U_{\mathit{eff}}|$: the number of affected unique nodes;
- $|A_{\mathit{eff}}|$: the number of affected anonymous nodes;
- $\delta_m(u) + \delta_m(v)$: the number of unique nodes directed connected by $\{u, v\}$.

Then, we assign a label for each edge to indicate whether deleting this edge will make the graph more anonymous. We compare the uniqueness before and after deleting each edge from the original graph. If deleting it makes the graph more anonymous, it would be included in the positive class (the label is $True$), otherwise, it would be the negative class (the label is $False$).

Ideally, we prefer training a general model that could be applied to all possible graphs. So the dataset should be diverse, as otherwise it would overfit to some specific situation. We do not want training data to contain data from all the networks in Section 5 because we cannot observe whether it can be applied to new networks. Considering that *d-k-anonymity* is stricter than *(n,m)-anonymity*, we labeled the edges based on *d-k-anonymity*. The final training data contains the 8-dimensional feature vectors and labels of edges from the Copnet SMS, Copnet FB, and ca-GrQc network (see Section 5) based on *d-k-anonymity*. The model is trained on three different networks, with a total of 21,622 edges. We choose a small dataset (Copnet SMS), a large dataset (ca-GrQc), and a dense dataset (Copnet FB).

The model was evaluated by the accuracy of the training dataset, i.e., the ratio of correctly predicted labels to all labels. For the optimizer, we tried stochastic average gradient (SAG), conjugate gradient (CG), limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [17] and liblinear [11]. We use a penalized LR, and experimented L2 (ridge regression) norm penalty with all optimizer and L1 (lasso regression) norm penalty with liblinear. The L1 norm penalty, combined with liblinear was chosen with the highest accuracy of 0.902.

Given the trained model and the feature vectors of edges in the original graph, we predict the effects of deleting an edge. Since our goal is to choose the most efficient $g$ edges to be deleted, we do not use the predicted labels but the probability of each class instead. $\mathcal{P}_L$ denotes the estimated probabilities of edges being in the positive class. For example, $\mathcal{P}_L = (0.1, 0.5, 0.9)$ means that in this 3-edge graph, the probability that these three edges belong to the positive category are 0.1, 0.5, and 0.9, respectively. Generally, a higher probability indicates deleting the edge is more likely to lower the uniqueness of the graph. Then the edges would be evaluated based on the predicted probabilities. The process is shown below. lines 5–8 consist of a specialized selection function for LR-based deletion (see Algorithm 1, line 5).

**Algorithm 2** LR-based edge deletion

**Input:** original graph $G = (V, E)$, budget $N$, recompute gap $g$, trained classifier $L$, measure $a$

$G' \leftarrow G$;
$t \leftarrow initialize(N, g)$; ▷ Initialize the number of iterations
$U_a, V_{unique} \leftarrow unique(G', a)$; ▷ Compute the uniqueness and the unique nodes
**while** $t > 0$ **do**
    $\mathcal{F}_G \leftarrow feature(G', V_{unique})$; ▷ Calculate feature vectors of the edges
    $\mathcal{P}_L \leftarrow predict(\mathcal{F}_G, L)$; ▷ Predict probabilities
    $sort(\mathcal{P}_L)$;
    $E_{del} \leftarrow top(\mathcal{P}_L, g)$;
    $delete(G', E_{del})$;
    $t \leftarrow t - 1$;
    $U_a, V_{unique} \leftarrow unique(G', a)$;
**end while**

**Output:** $G'$, $U_a$

### 4.2.5 *(n,m)*-Greedy Deletion

In this section, we propose a greedy approach for obtaining *(n,m)-anonymity*, with the goal of iteratively obtaining the largest decrease in uniqueness.

First, we explore the precise effect of deleting an edge on ego states, i.e., the number of nodes and edges in the ego network of a node. As we mentioned in Section 4.2.3, deleting an edge will influence the anonymity of nodes included in any triangle the edge is a part of. Given a graph $G = (V, E)$, a node $v \in V$ and an edge $\{u, w\} \in E$, the exact effect of deleting $\{u, w\}$ on ego state $S(v)$, denoted $(\Delta n(v), \Delta m(v))$, can be defined as follows

$$\Delta n(v) = \begin{cases} -1, & v \in \{u, w\} \\ 0, & otherwise \end{cases} \tag{13}$$

$$\Delta m(v) = \begin{cases} 1 - |V_1(u) \cap V_1(w)|, & v \in \{u, w\} \\ -1, & \{u, v\}, \{v, w\} \in E \\ 0, & otherwise \end{cases} \tag{14}$$

Using the equations, we can determine the next ego state of nodes without actually deleting an edge, thus, finding how many nodes remain in this equivalence class. Only nodes in $V_{\mathit{eff}}$, i.e., nodes directly connected by this edge or nodes that contain this edge in their triangles, would be affected cf. Equation 13 and 14. Hence, we check the number of nodes in the current and the new equivalence class for each affected node, shown in Algorithm 3.

The variable *eff* denotes the change in the number of anonymous nodes in the graph $G$ after deleting an edge $\{u, v\}$. For each node in $w \in V_{\mathit{eff}}$, we first move it out

**Algorithm 3** Evaluate edge in *(n,m)*-greedy deletion

**Input:** original graph $G = (V, E)$, edge $e = \{u, v\}$

1: Initialize the ego states $S$ and equivalence classes $C_S$;
2: $\mathit{eff} \leftarrow 0$;
3: $V_{\mathit{eff}} \leftarrow V_1(u) \cap V_1(v)$; ▷ cf. Equation 8
4: **for** $w \in V_{\mathit{eff}}$ **do**
5: Remove $w$ from its previous equivalence class $C_S(w)$;
6: **if** $|C_S(w)| = 0$ **then**
7: $\mathit{eff} \leftarrow \mathit{eff} + 1$;
8: **else**
9: **if** $|C_S(w)| = 1$ **then**
10: $\mathit{eff} \leftarrow \mathit{eff} - 1$;
11: **end if**
12: **end if**
13: $S'(w) \leftarrow \mathit{update}(S(w), \{u, v\})$; ▷ cf. Equation 13, 14
14: Add $w$ to new the equivalence class $C_{S'}(w)$;
15: **if** $|C_{S'}(w)| = 1$ **then**
16: $\mathit{eff} \leftarrow \mathit{eff} - 1$;
17: **else**
18: **if** $|C_{S'}(w)| = 2$ **then**
19: $\mathit{eff} \leftarrow \mathit{eff} + 1$;
20: **end if**
21: **end if**
22: **end for**

**Output:** *eff*

from the previous equivalence class $C_S(w)$ because its ego state would be changed. Then, check the number of remaining nodes in the previous $C_S(w)$. If no node remains in $C_S(w)$ (line 6), i.e., $w$ is originally unique, removing $\{u, v\}$ would make the number of unique nodes minus 1, thus *eff* plus 1. If exactly one node remains in $C_S(w)$ (line 9), i.e., removing $\{u, v\}$ makes the remaining node unique, *eff* minus 1. Similarly, count the nodes in the new equivalence class $C_{S'}(w)$ (lines 14–18), and update *eff*.

After checking all the affected nodes, we get the exact effect for each edge deletion. By sorting the effects in descending order, the edge with the largest impact on uniqueness is chosen. The pseudocode is shown in Algorithm 4. Array $\mathcal{U}_{G\Delta}$ is used to evaluate the impacts of deleting each edge, which is the output by Algorithm 3. Lines 5–9 consist of a specialized selection function for *(n,m)*-greedy deletion (see Algorithm 1, line 5).

**Algorithm 4** Greedy Edge Deletion

**Input:** original graph $G = (V, E)$, budget $N$, recompute gap $g$, measure $a$

$G' \leftarrow G$;
$t \leftarrow initialize(N, g)$; ▷ Initialize the number of iterations
$U_a, V_{unique} \leftarrow unique(G', a)$; ▷ Compute the uniqueness and the unique nodes
**while** $t > 0$ **do**
    **for** $e \in E$ **do**
        $\mathcal{U}_{G\Delta}[e] \leftarrow evaluate(G', e)$; ▷ see Algorithm 3
    **end for**
    $sort(\mathcal{U}_{G\Delta})$;
    $E_{del} \leftarrow top(\mathcal{U}_{G\Delta}, g)$;
    $delete(G', E_{del})$;
    $t \leftarrow t - 1$;
    $U_a, V_{unique} \leftarrow unique(G', a)$;
**end while**

**Output:** $G'$, $U_a$

# 5 Datasets

This section includes an introduction of the dataset used in this thesis. Considering a desire for diversity in size, density, level of clustering, and the original overall uniqueness, we choose five datasets. A brief description of datasets is given below.

- *Copnet SMS, Copnet FB* [22]: The first two networks are based on the Copnethagen Networks Study [22]. The data was collected from more than 700 university students over a period of four weeks, including information on Bluetooth signal strength, phone calls, text messages, and Facebook friendships.

  The information of text messages are used for the Copnet SMS dataset, where the nodes are the senders and the recipients, and edges mean sent text messages. We ignore edge direction and timestamps. The Facebook friendships are used for Copnet FB, in which nodes are users and edges indicate friends on Facebook.

- *CollegeMsg* [16]: This dataset comprises private message exchange information from an online social platform at the University of California, Irvine. Within this platform, users can search for others, view their profiles, and initiate conversations. In this network, nodes are users, and an edge exists if two users had a conversation.

- *ca-GrQc* [16]: This network covers the co-authorship of papers submitted to the General Relativity and Quantum Cosmology (GR-QC) category on the e-printer arXiV. The dataset includes papers published between January 1993 and April 2003, spanning a period of 124 months. Nodes in this network are authors, and edges indicate the authors had a co-author experience.

- *ego Facebook* [16]: This dataset consists of Facebook friend lists, with the data collected from survey participants utilizing the Facebook app. It includes users profiles, connections, and some features of the ego networks in 10 separated subnetworks. We used all connections in the 10 subnetworks to generate the graph. Nodes in this graph are the users, and edges are the connections on Facebook.

All the graphs are undirected after processing. Some general properties for these networks are shown in Table 2. The datasets have varying numbers of nodes and edges. The graph of ca-GrQc is denser than CollegeMsg, as they have a similar number of edges, but ca-GrQc has significantly more nodes than CollegeMsg. However, the density does not lead to a higher clustering level. In this table, the ACC and transitivity of ca-GrQc and ego Facebook are high, whereas those of Copnet SMS and CollegeMsg are low. Additionally, the initial uniqueness of Copnet SMS and ca-GrQc is relatively low.

Table 2: Information of the original graphs. The second column to the fourth column present the number of nodes, the number of edges, and the number of triangles in the graphs, respectively. The fifth and sixth columns describe the ACC and transitivity of the graph, and the last two columns are the uniqueness of the original graph.

| Dataset | $\|V\|$ | $\|E\|$ | Triangles | ACC | Transitivity | $U_{nm}$ | $U_{dk}$ |
|---|---|---|---|---|---|---|---|
| Copnet SMS | 568 | 697 | 97 | 0.139 | 0.154 | 0.026 | 0.044 |
| Copnet FB | 800 | 6,429 | 13,698 | 0.315 | 0.244 | 0.472 | 0.817 |
| CollegeMsg | 1,899 | 13,838 | 14,319 | 0.109 | 0.057 | 0.239 | 0.401 |
| ca-GrQc | 5,242 | 14,496 | 48,260 | 0.530 | 0.630 | 0.055 | 0.135 |
| ego Facebook | 4,039 | 88,234 | 1,612,010 | 0.606 | 0.519 | 0.587 | 0.812 |

# 6 Experiments

In this section, we present the experimental setup and results. First, we present our experimental settings in Section 6.1. Then, we applied the aforementioned methods to the chosen datasets, reporting the outcomes across subsequent sections. In Section 6.2, we discuss which modification operation should be used. In Section 6.3 we explore how these methods affect the uniqueness during the whole deletion process. In Section 6.4, we explore how the runtime varies depending on the structural properties of networks. In Section 6.5 we delete edges with a fixed budget and discuss which anonymization method is most effective. In Section 6.6, we analyze the features for the LR model, to find out what kind of edges contribute most to graph anonymity.

## 6.1 Experimental setup

The five methods in Section 4 were applied to the datasets mentioned in Section 5. In Section 4.2, we mentioned the recompute gap $g$, which is dependent on the number of edges $|E|$ of the network. In each iteration, $g$ edges will be deleted, and the uniqueness will be recomputed. We set $g = \lfloor 0.01|E| \rfloor$. The used $g$ for each network can be found in Table 3.

The approaches are implemented in Python, using packages such as iGraph and Scikit-learn. Code can be found at: `https://github.com/christine99x/networkAnonymization`. For computing *d-k-anonymity*, we use code from de Jong et al. [7]: `https://github.com/RacheldeJong/dkAnonymity`.

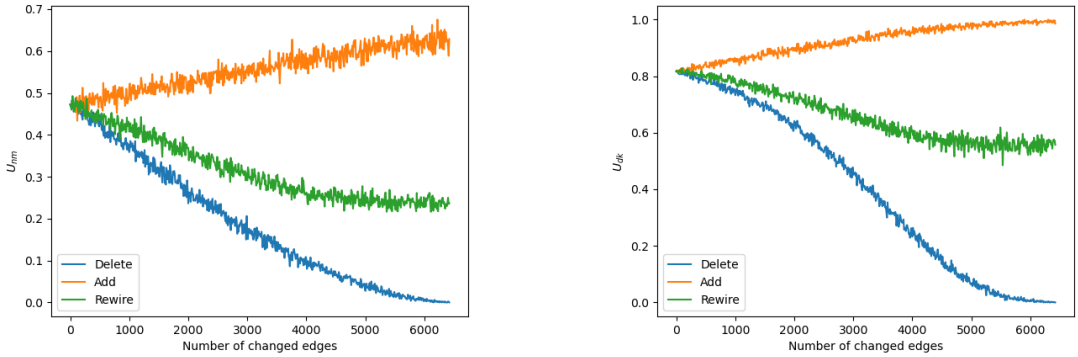
To account for randomness in methods, and to get a better estimation of the runtime, the experiments were repeated 10 times and the average values were taken. All the experiments were run on a Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz.

## 6.2 Anonymization operations

Before diving into specific algorithms, we ran a preliminary experiment to find out which modification operation we should focus on. We compared the results of randomly deleting, adding, and rewiring edges on the Copnet FB dataset with two anonymity measures, shown in Figure 3.

These results show that random deletion is by far the most effective operation. The reason is that nodes with lower degree are more likely to be equivalent to each other because the possible structures of their ego network are limited. Edge deletion lowers the average degree of nodes, resulting in simple ego networks, and thus fewer unique nodes in the graph. On the contrary, adding edges will increase the average degree of nodes, and therefore, can increase the uniqueness of the graph.

For random rewiring, the uniqueness first drops and then stabilizes. We believe this happens because after some rewiring operation, the graph becomes similar to a




(a) *(n,m)-Anonymity* (b) *d-k-Anonymity*

Figure 3: Overall uniqueness (vertical axis) of different edge perburbation methods on Copnet FB for *(n,m)-anonymity* (left) and *d-k-anonymity* (right). The horizontal axis is the number of deleted edges.

random graph generated by the original graph's degree distribution, so the uniqueness will remain stable around a certain value.

Because of the results above, we focus on edge deletion.

## 6.3 Comparison of anonymization algorithms

Figure 4 shows the uniqueness during the deletion process for each of the five methods. We analyze the results by comparing their performance on the two discussed anonymity measures, and throughout the deletion process.

First, for the whole deletion process, we discuss whether the methods perform consistently on both of the measures. Comparing the trends of the curves in the left and right columns of Figure 4, the LR model (red) and *(n,m)*-greedy deletion (purple) perform differently on the two anonymity measures, while the other methods behave similarly for both of the measures. The method of *(n,m)*-greedy deletion does not show any advantages over *d-k-anonymity*, decreasing steadily, and performs similarly to random, or worse after a large number of edge deletions. An explanation is that the *(n,m)*-greedy deletion is specific to *(n,m)-anonymity*, so it would not be effective when we measure the graph using *d-k-anonymity*. Similarly, since our training data for our LR model is created based on *d-k-anonymity*, the LR-based deletion does not have that much advantage in *(n,m)-anonymity*. Therefore, although, as we mentioned, when a network satisfies *d-k-anonymity*, it must satisfy *(n,m)-anonymity*, it does not mean that a method performs well on *d-k-anonymity* must perform well on *(n,m)-anonymity*.

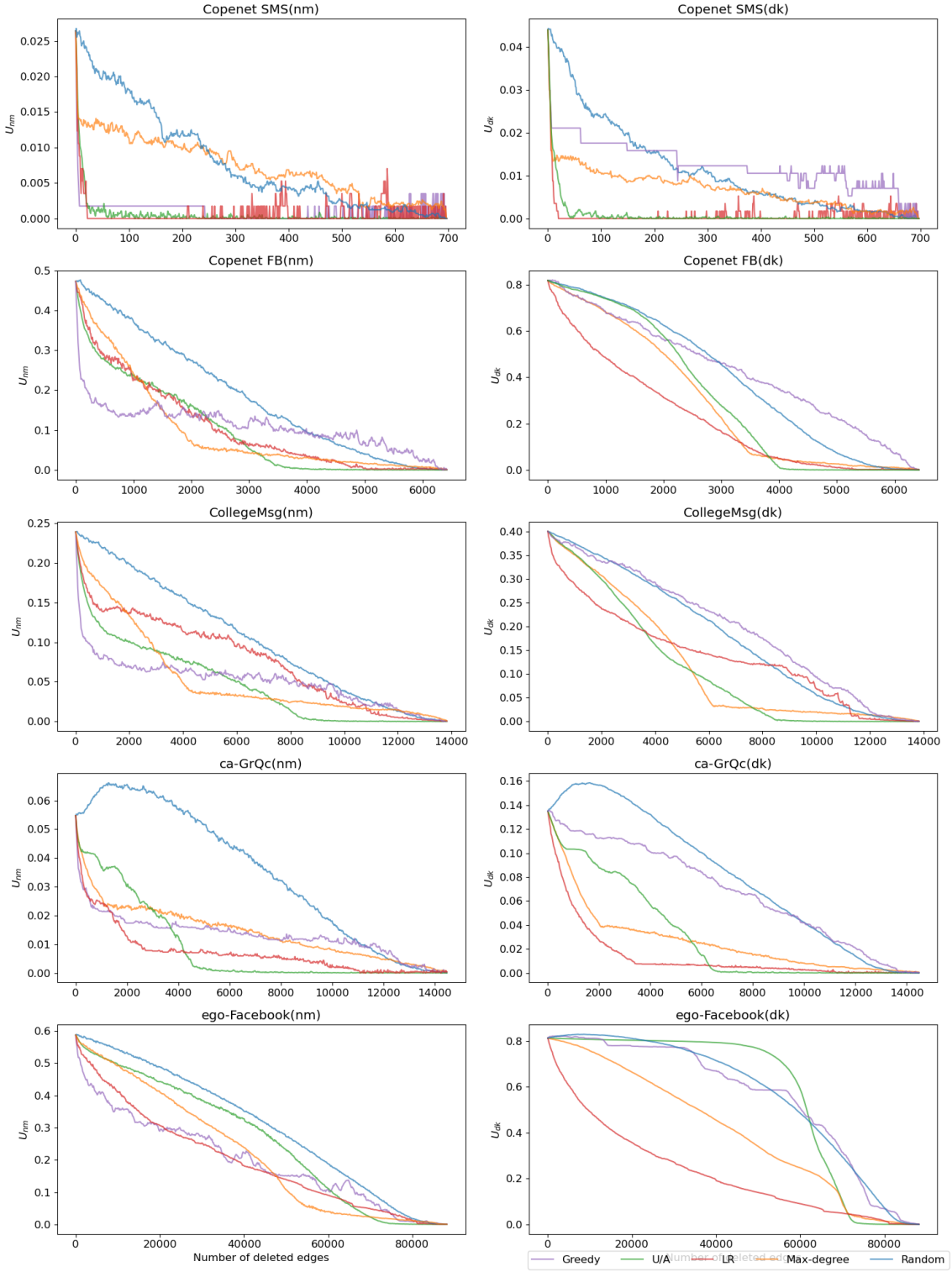



Figure 4: Overall uniqueness (vertical axis) of different anonymization methods on five datasets for *(n,m)-anonymity* (left) and *d-k-anonymity* (right). The horizontal axis is the number of deleted edges.

Then, considering the different stages of deletion, at the start of the process, almost all the methods are better than the baseline (blue). For *(n,m)-anonymity* (left), by evaluating the effects precisely, *(n,m)*-greedy deletion (purple) becomes the most effective method. While, for *d-k-anonymity* (right), the most effective method is the LR-based deletion (red), because it combines the information of degree and uniqueness of nodes, and utilizes the prior knowledge from the training data. So if we only have a small budget of edges to be deleted, it is better to choose *(n,m)*-greedy deletion for *(n,m)-anonymity* and LR-based deletion for *d-k-anonymity*.

At the end of the deletion process, when many edges are deleted, the UA-based deletion method has a stable performance and outperforms all other measures. Contrary to the start of the process, *(n,m)*-greedy deletion (purple) performs badly for most of the networks. The main reason is that the *(n,m)*-greedy deletion is always looking for a locally optimal choice, anonymizing the network by deleting one edge. However, anonymization is a continuous process, and edges are dependent. The optimal choice in the case of deleting one edge may not be the optimal option throughout the global deletion process. In addition, we have the recompute gap $g$, which would affect the accuracy of evaluating the edges, as the unique nodes set would not be updated immediately after deleting an edge. This affects *(n,m)*-greedy deletion the most, since a small misevaluation can lead to a large difference in edge selection. As a consequence, if our goal is to choose a method that can make the uniqueness of the graph lower than an extremely low predefined threshold with the fewest deletions, the best choice is UA-based deletion (green). The method of *(n,m)*-greedy deletion (purple) is the worst of these methods for most of the networks.

## 6.4 Anonymization runtime

In this section, we compare the runtime of the five methods under two measures to gain a rough understanding of their time consumption. Additionally, we explore how the network's structural features, like size and clustering, influence the runtime of these methods by contrasting the runtime on different networks. The results are shown in Table 3.

According to Table 3, the three non-deterministic methods run much faster than LR-based deletion and *(n,m)*-greedy deletion. For these non-deterministic methods, the UA-based deletion is more time-consuming than the degree-based deletion, which is in turn more expensive than random deletion. In most of the networks, LR-based deletion costs more time than the *(n,m)*-greedy deletion.

For the runtime of the LR-based deletion and *(n,m)*-greedy deletion, ca-GrQc is an exception for the aforementioned conclusion. This might be caused by the relatively high ACC and transitivity, which means many triangles are contained in this graph. It greatly increases the runtime of the evaluation process in Algorithm 3. Therefore, the *(n,m)*-greedy deletion method would be more time-consuming on this dataset.

Table 3: Average runtime of the whole deletion process for different datasets. The second column presents the methods. The third column is the recompute gap $g = \lfloor 0.01|E| \rfloor$. The last two columns are average runtimes for *(n,m)-anonymity* and *(d-k-anonymity)*.

| Dataset | Method | Recompute gap $g$ | Time(s) | |
|---|---|---|---|---|
| | | | nm | dk |
| Copnet SMS | Random | 6 | 0.487 | 1.125 |
| | Max-degree | | 0.530 | 1.167 |
| | U/A | | 0.573 | 1.240 |
| | LR | | 9.079 | 10.205 |
| | Greedy | | 4.010 | 4.909 |
| Copnet FB | Random | 64 | 1.022 | 2.283 |
| | Max-degree | | 1.499 | 2.871 |
| | U/A | | 2.813 | 4.049 |
| | LR | | 98.143 | 99.465 |
| | Greedy | | 39.672 | 41.472 |
| CollegeMsg | Random | 138 | 2.299 | 3.955 |
| | Max-degree | | 4.022 | 6.723 |
| | U/A | | 8.616 | 10.424 |
| | LR | | 280.606 | 289.227 |
| | Greedy | | 184.870 | 186.642 |
| ca-GrQc | Random | 144 | 5.471 | 3.866 |
| | Max-degree | | 8.098 | 6.999 |
| | U/A | | 10.957 | 10.125 |
| | LR | | 346.113 | 308.993 |
| | Greedy | | 662.962 | 638.086 |
| ego-Facebook | Random | 882 | 13.023 | 24.731 |
| | Max-degree | | 218.198 | 282.679 |
| | U/A | | 277.080 | 291.480 |
| | LR | | 7908.942 | 7932.515 |
| | Greedy | | 3010.817 | 3033.524 |

For the measures, generally, measuring *d-k-anonymity* is more expensive than *(n,m)-anonymity*. However, compared to the difference in running time among the methods, the difference between measures is not obvious, because, in each iteration, the selecting process is more time-consuming than computing uniqueness.

## 6.5 Anonymization with a fixed budget

In the experiments above, we deleted up to all edges for each network. As our goal is to explore how to anonymize a given network with fewer changes, in this section,

we present more experiments to explore the performances of the methods with a 1% fixed budget. In addition, we present the ACC and transitivity of the graphs after edge deletion, providing a reference to the impact of the methods on the overall network structure.

The results of *(n,m)-anonymity* and *d-k-anonymity* are presented in Table 4 and Table 5, respectively. In the tables, $G_O$ is the original graph, and for its corresponding time, we record the running time of measuring the uniqueness. We show the lowest uniqueness values of each network in bold.

Table 4: Uniqueness $U_{nm}$, ACC, and transitivity of different datasets after deleting 1% of the edges, and the corresponding runtimes. The third presents the data for the original graph ($G_O$), followed by values for random edge deletion ($G_{random}$), degree-based deletion ($G_{degree}$), UA-based deletion ($G_{UA}$), LR-based deletion ($G_{LR}$), and *(n,m)*-greedy method ($G_{greedy}$).

| Dataset | Metrics | $G_O$ | $G_{random}$ | $G_{degree}$ | $G_{UA}$ | $G_{LR}$ | $G_{greedy}$ |
|---|---|---|---|---|---|---|---|
| Copnet SMS | $U_{nm}$ | 0.026 | 0.026 | 0.016 | 0.012 | 0.016 | **0.004** |
| | ACC | 0.139 | 0.137 | 0.136 | 0.138 | 0.136 | 0.131 |
| | Transitivity | 0.154 | 0.152 | 0.150 | 0.154 | 0.143 | 0.144 |
| | Time (s) | 0.007 | 0.035 | 0.038 | 0.045 | 1.159 | 0.039 |
| Copnet FB | $U_{nm}$ | 0.472 | 0.469 | 0.443 | 0.419 | 0.444 | **0.274** |
| | ACC | 0.315 | 0.312 | 0.312 | 0.314 | 0.310 | 0.313 |
| | Transitivity | 0.244 | 0.242 | 0.241 | 0.240 | 0.241 | 0.242 |
| | Time (s) | 0.018 | 0.103 | 0.133 | 0.374 | 12.978 | 4.770 |
| CollegeMsg | $U_{nm}$ | 0.239 | 0.239 | 0.216 | 0.201 | 0.207 | **0.147** |
| | ACC | 0.109 | 0.108 | 0.108 | 0.109 | 0.107 | 0.107 |
| | Transitivity | 0.057 | 0.056 | 0.056 | 0.056 | 0.156 | 0.154 |
| | Time (s) | 0.042 | 0.187 | 0.271 | 0.892 | 34.139 | 17.675 |
| ca-GrQc | $U_{nm}$ | 0.055 | 0.054 | 0.044 | 0.044 | 0.040 | **0.036** |
| | ACC | 0.530 | 0.521 | 0.526 | 0.529 | 0.523 | 0.525 |
| | Transitivity | 0.630 | 0.625 | 0.624 | 0.638 | 0.628 | 0.627 |
| | Time (s) | 0.081 | 0.356 | 0.376 | 0.832 | 39.537 | 62.584 |
| ego-Facebook | $U_{nm}$ | 0.587 | 0.586 | 0.569 | 0.569 | 0.548 | **0.518** |
| | ACC | 0.606 | 0.600 | 0.601 | 0.603 | 0.596 | 0.592 |
| | Transitivity | 0.519 | 0.515 | 0.515 | 0.506 | 0.518 | 0.520 |
| | Time (s) | 0.359 | 1.701 | 6.753 | 15.876 | 1157.716 | 334.088 |

For *(n,m)-anonymity*, all the newly proposed methods are more effective than random edge deletion in the uniqueness value. The most effective method is always *(n,m)*-greedy deletion for the included networks. However, it is also the most destructive method to ACC and transitivity for most of the datasets. Especially for ACC,

in three datasets, the impact on ACC is greater than 0.005. For the two datasets with relatively low transitivity, Copnet SMS and CollegeMsg, their transitivity is also affected.

Table 5: Uniqueness $U_{dk}$, ACC, and transitivity of different datasets after deleting 1% of the edges, and the corresponding runtimes. The third presents the data for the original graph ($G_O$), followed by values for random edge deletion ($G_{random}$), degree-based deletion ($G_{degree}$), UA-based deletion ($G_{UA}$), LR-based deletion ($G_{LR}$), and *(n,m)*-greedy method ($G_{greedy}$).

| Dataset | Metrics | $G_O$ | $G_{random}$ | $G_{degree}$ | $G_{UA}$ | $G_{LR}$ | $G_{greedy}$ |
|---|---|---|---|---|---|---|---|
| Copnet SMS | $U_{dk}$ | 0.044 | 0.043 | 0.022 | 0.022 | **0.016** | 0.025 |
| | ACC | 0.139 | 0.137 | 0.137 | 0.139 | 0.136 | 0.131 |
| | Transitivity | 0.154 | 0.154 | 0.148 | 0.154 | 0.143 | 0.144 |
| | Time (s) | 0.018 | 0.112 | 0.112 | 0.121 | 1.208 | 0.518 |
| Copnet FB | $U_{dk}$ | 0.817 | 0.816 | 0.805 | 0.812 | **0.775** | 0.819 |
| | ACC | 0.315 | 0.311 | 0.313 | 0.311 | 0.304 | 0.313 |
| | Transitivity | 0.244 | 0.242 | 0.242 | 0.239 | 0.243 | 0.242 |
| | Time (s) | 0.037 | 0.233 | 0.280 | 0.543 | 16.861 | 5.396 |
| CollegeMsg | $U_{dk}$ | 0.401 | 0.397 | 0.393 | 0.393 | **0.365** | 0.391 |
| | ACC | 0.109 | 0.108 | 0.109 | 0.108 | 0.106 | 0.107 |
| | Transitivity | 0.057 | 0.056 | 0.056 | 0.055 | 0.056 | 0.054 |
| | Time (s) | 0.086 | 0.413 | 0.527 | 1.232 | 43.077 | 23.250 |
| ca-GrQc | $U_{dk}$ | 0.135 | 0.138 | 0.127 | 0.128 | **0.116** | 0.134 |
| | ACC | 0.530 | 0.522 | 0.524 | 0.529 | 0.520 | 0.525 |
| | Transitivity | 0.630 | 0.624 | 0.630 | 0.638 | 0.629 | 0.627 |
| | Time (s) | 0.100 | 0.466 | 0.522 | 0.996 | 35.625 | 67.425 |
| ego-Facebook | $U_{dk}$ | 0.812 | 0.816 | 0.808 | 0.812 | **0.765** | 0.819 |
| | ACC | 0.606 | 0.600 | 0.601 | 0.602 | 0.596 | 0.592 |
| | Transitivity | 0.519 | 0.515 | 0.516 | 0.508 | 0.515 | 0.520 |
| | Time (s) | 0.632 | 2.904 | 9.159 | 17.907 | 1172.720 | 324.253 |

Then, for *d-k-anonymity*, presented in Table 5, the LR-based deletion is the most effective approach to lower the uniqueness. It is more harmful to ACC compared to deletion under *(n,m)*-anonymity with the same method, but it performs better on transitivity. The degree-based deletion and UA-based deletion are still better than random edge deletion in lowering the uniqueness, and degree-based deletion is better for the involved networks.

For ACC and transitivity, although we did not consider them when designing the approaches, all the methods have no obvious disadvantages compared to the baseline.

In general, we find that the best method for *(n,m)-anonymity* is *(n,m)*-greedy

deletion, and the best method for *d-k-anonymity* is LR-based deletion. Both methods are harmful to ACC. Additionally, we also provide the data of ACC, transitivity, and runtime as references in case one is interested in specific criteria. For example, if we want to choose a method with an impact on ACC less than 0.005, according to Table 4, the best choice would be the UA-based deletion, because graphs anonymized by the UA-based deletion has the lowest $U_{nm}$ among the methods meeting this criterion.

## 6.6 Analysis of the results of logistic regression

From the trained LR model, we can look at the coefficient for each feature to understand which types of edges are effective to delete for anonymization. When the p-value is more than 0.05, the feature is considered to be significant. The coefficients of the LR model and the significance are presented in Table 6. Values $|U_{\mathit{eff}}|$, $|A_{\mathit{eff}}|$ and $T$ are related, as the number of triangles influences which nodes are affected. According to the results, they all do not significantly relate to the labels (contribute to anonymization or not) of edges, while the remaining features are all significantly correlated with labels. The coefficient shows that $|V|$ and $\delta(u) + \delta(v)$ are positively correlated with the label and other features are negatively correlated. Therefore, in a more sparse graph, deleting edges seems more likely to make the graph anonymous. Edges connecting nodes with lower degrees are more likely to contribute to graph anonymization. The greater the number of unique nodes directly connected by an edge, the more likely it is that deleting it will make the graph anonymous.

Table 6: Coefficients of the trained LR model. The middle column presents the coefficient for each feature. The last column presents whether the feature is significant.

| Features | Coefficient | Significant |
|---|---|---|
| $\delta_m(u) + \delta_m(v)$ | 1.862e+00 | ✓ |
| $\min\{deg(u), deg(v)\}$ | -1.793e-01 | ✓ |
| $T$ | 1.658e-01 | ✗ |
| $\lvert A_{\mathit{eff}}\rvert$ | -6.459e-02 | ✗ |
| $\lvert U_{\mathit{eff}}\rvert$ | -3.532e-02 | ✗ |
| $\max\{deg(u), deg(v)\}$ | -3.436e-02 | ✓ |
| $\lvert V\rvert$ | 7.853e-04 | ✓ |
| $\lvert E\rvert$ | -3.143e-04 | ✓ |

# 7 Conclusion

In this thesis, we explored methods to anonymize a given graph as much as possible with as few changes as possible. To evaluate anonymity, we used the previously proposed measure *d-k-anonymity* and newly proposed *(n,m)-anonymity*, accounting for the ego network size rather than precise structure. We first investigated which modification operation is most useful. Preliminary experiments showed that random deletion is the most effective among random edge addition, deletion, and rewiring. It was chosen as our baseline. Additionally, we proposed four anonymization methods: degree-based deletion, UA-based deletion, based on the number of unique nodes of the affected nodes, LR-based deletion, based on logistic regression, and *(n,m)*-greedy deletion. We evaluated the uniqueness after deleting 1% of the nodes. Our results show that LR-based deletion is the most efficient anonymization method for *d-k-anonymity* and *(n,m)*-greedy method is the best method for *(n,m)-anonymity*.

Another question we discussed is the runtime of the algorithms. In most cases, the time consumption of these methods is as follows: random edge deletion < degree-based deletion < UA-based deletion < *(n,m)*-greedy deletion < LR-based deletion. However, for a graph with a high clustering level, *(n,m)*-greedy deletion might take more time than LR-based deletion. The results of LR show that deleting edges with the following features is more likely to increase the overall anonymity of the graph: 1) edges connecting two low-degree nodes; 2) edges connecting one or more unique nodes. Additionally, edge deletion appears more efficient in sparse networks.

Future work might explore innovative approaches that integrate the preservation of network structure and anonymization more effectively. Moreover, newly designed approaches should consider the dependencies among edges when modifying the network. Additionally, steps can be made to improve the LR-based deletion, including using more effective features or moving towards more advanced machine learning models, like random forest and neural networks.

# References


[1] Abdullah Al-Rabeeah and Mohammed Hashim. Social network privacy models. *Cihan University-Erbil Scientific Journal*, 3(2):92–101, 2019.

[2] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x? Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, pages 181–190, 2007.

[3] Alina Campan, Yasmeen Alufaisan, and Traian Marius Truta. Preserving communities in anonymized social networks. *ACM Transactions on Data Privacy*, 8(1):55–87, 2015.

[4] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *Proceedings of the 2nd International Workshop on Privacy, Security, and Trust in Knowledge Discovery in Databases*, pages 33–54, 2009.

[5] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proceedings of the ACM Special Interest Group on Management of Data International Conference on Management of Data*, page 459–470, 2010.

[6] Sean Chester, Bruce Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and Venkatesh Srinivasan. K-anonymization of social networks by vertex addition. In *Proceedings of the 15th East-European Conference on Advances in Databases and Information Systems*, pages 107–116, 01 2011.

[7] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. Algorithms for efficiently computing structural anonymity in complex networks. *ACM Journal of Experimental Algorithmics*, 2023.

[8] Rachel G. de Jong, Mark P.J. van der Loo, and Frank W. Takes. Beyond the ego network: The effect of distant connections on node anonymity. *arXiv preprint*, arXiv:2306.13508, 2023.

[9] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages, and Programming*, pages 1–12, 2006.

[10] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, pages 486–503, 2006.

[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, jun 2008.

[12] Yumeng Fu, Wei Wang, Hao Fu, Wu Yang, and Dan Yin. Privacy preserving social network against Dopv attacks. In *Proceedings of the 19th International Conference on Web Information Systems Engineering*, pages 178–188, 2018.

[13] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the Very Large Data Base Endowment*, 1(1):102–114, 2008.

[14] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

[15] Diane Lambert. Measures of disclosure risk and harm. *Journal of Official Statistics*, 9:313–313, 1993.

[16] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014. Accessed Date: 22 Nov, 2023.

[17] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[18] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM Special Interest Group on Management of Data International Conference on Management of data*, pages 93–106, 2008.

[19] Xuesong Lu, Yi Song, and Stéphane Bressan. Fast identity anonymization on graphs. In *Proceedings of the 23rd International Conference on Database and Expert Systems Applications*, pages 281–295, 2012.

[20] Youyang Qu, Shui Yu, Wanlei Zhou, Shiping Chen, and Jun Wu. Customizable reliable privacy-preserving data sharing in cyber-physical social networks. *IEEE Transactions on Network Science and Engineering*, 8(1):269–281, 2020.

[21] Daniele Romanini, Sune Lehmann, and Mikko Kivelä. Privacy and uniqueness of neighborhoods in social networks. *Scientific reports*, 11(1):20104, 2021.

[22] Piotr Sapiezynski, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann. Interaction data from the copenhagen networks study. *Scientific Data*, 6(1):315, 2019.

[23] Shafaq Shakeel, Adeel Anjum, Alia Asheralieva, and Masoom Alam. k-NDDP: An efficient anonymization model for social network data release. *Electronics*, 10(19), 2021.

[24] Brian Thompson and Danfeng Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, page 218–227, 2009.

[25] Mark van der Loo. Topological anonymity in networks. Technical report, Statistics Netherlands, 2022.

[26] 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, 2010.

[27] Xiaowei Ying, Kai Pan, Xintao Wu, and Ling Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, pages 1–10, 2009.

[28] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *Proceedings of the 8th Society for Industrial and Applied Mathematics International Conference on Data Mining*, pages 739–750. SIAM, 2008.

[29] Jinquan Zhang, Xiao Wang, Yanfeng Yuan, and Lina Ni. RcDT: Privacy preservation based on R-constrained dummy trajectory in mobile social networks. *IEEE Access*, 7:90476–90486, 2019.

[30] Yang Zhang, Mathias Humbert, Bartlomiej Surma, Praveen Manoharan, Jilles Vreeken, and Michael Backes. Towards plausible graph anonymization. *arXiv preprint*, arXiv:1711.05441, 2019.

[31] 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. IEEE, 2008.

[32] Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the Very Large Data Base Endowment*, 2(1):946–957, 2009.