Network Anonymisation for Science

Improving (n, m)-greedy edge deletion anonymisation using global heuristic

Jakub Matyja[1]

Supervisor & Responsible Professor: Dr. Anna L.D. Latour[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

Name of the student: Jakub Matyja
Final project course: CSE3000 Research Project
Thesis committee: Dr. Anna L.D. Latour, Dr. Carolin Brandt

Abstract

Network anonymisation is an essential procedure in processing data structured as graphs to achieve non-identifiability of participating entities. This quality is particularly desirable among networks representing stakeholders whose identity should not be compromised for ethical or legal reasons or when such structures are publicly disclosed. Extensive use of sensitive graph-like data contributed to the development of anonymisation methods that vary in terms of performance given the network topology or imposed constraints. We present modifications to an existing greedy algorithm using equivalence classes and discuss the rationale behind them. The experimental results obtained on networks from various scientific fields indicate that algorithms taking into account the size of equivalence classes in edge evaluation can consistently outperform the existing greedy algorithm in terms of the solution quality for larger number of available deletions. The proposed improvements also lead to comparable running time.

1 Introduction

Preserving the anonymity of networks holding sensitive information lead to the development of various measures that determine the graph anonymity or strategies of anonymising such graphs. One of the widely adopted anonymity measures is k-anonymity [Samarati and Sweeney, 1998]. It stipulates that all entities participating in the network cannot be distinguished from $k-1$ other entities based on the chosen criterion, according to which these entities are classified. Several extensions to k-anonymity were proposed in the literature to address specific types of attacks [Machanavajjhala *et al.*, 2006; Li *et al.*, 2007]. (n, m)-k-anonymity is one of the measures that stems from k-anonymity and determines if nodes are indistinguishable according to their signature (n, m).

The (n, m)-greedy edge deletion method is a highly performant algorithm for network anonymisation with respect to (n, m)-k-anonymity. It iteratively removes edges for which the estimate of the change in network's *uniqueness* is the highest. Uniqueness is a measure according to which the anonymity of the network is determined. In particular, the work of [Xie, 2023] has shown that the greedy approach provides outstanding results in anonymising graphs for a limited number of iterations, usually outperforming other algorithms presented in contemporary literature, such as *logarithmic regression-based* (LR-based) and *affected unique and anonymous nodes-based* (UA-based) methods [de Jong *et al.*, 2025]. However, its performance deteriorates for a greater number of edge deletions in the network, eventually producing suboptimal results.

We propose a modification to the (n, m)-greedy deletion algorithm with the aim of improving its effectiveness for large number of iterations. We introduce a heuristic based on the size of *equivalence classes* — sets of nodes that are indistinguishable *w.r.t.* the chosen anonymity measure. The notion of equivalence classes is central to many anonymisation methods, including the (n, m)-greedy edge deletion. Another reason behind choosing equivalence classes as a suitable heuristic is that several methods outperforming the aforementioned greedy approach, such as LR-based or UA-based strategies, are reported to perform exceptionally well for large number of deletions due to prioritising low-degree and unique nodes [Xie, 2023; de Jong *et al.*, 2025]. This observation motivates the choice of the size of equivalence classes as a possible indicator for more accurate edge deletions with performance in the long run in mind.

This report provides an overview of the modifications we introduce to the (n, m)-greedy edge deletion method using the size of equivalence classes and discusses their effect on the performance of that anonymisation method from the perspective of three characteristics of anonymisation algorithms. Each of them discusses a distinct feature that defines the modification's applicability in practice: solution quality described by the final uniqueness of the graph, available budget representing the maximum number of available deletions, and solving speed in terms of physical time of execution.

To achieve this goal, we construct a theoretical framework to support the choice of the size of equivalence classes as a meaningful heuristic capable of yielding insightful results in Section 3. Furthermore, we propose several edge evaluation functions in this section and inspect them in terms of chosen benchmarks to assess their performance in Section 4. Finally, the results obtained through the constructed experimental setup are refined and discussed in Section 5.

This report provides an insight into the contribution of the size of equivalence classes in the greedy anonymisation algorithm. More precisely, we propose that the (n, m)-greedy algorithm is modified by prioritising edges that connect nodes belonging to larger equivalence classes. Most importantly, we show that the use of this heuristic improves the performance of the algorithm in terms of solution quality for a large number of deletions without compromising its robustness. Moreover, we argue that such behaviour could be owed to the fact that deletion-based anonymisation methods inherently lead to an increase in the average size of the equivalence classes in the network.

2 Preliminaries

In this section, we present concepts and definitions relevant to graph processing and network anonymisation, as well as related mathematical notation.

2.1 Graphs

Let $G = (V, E)$ denote a graph with nodes (also called vertices) V and edges E, $\{u, v\} \in E$ given that $u, v \in V$. For the purpose of this research, graphs are simple and undirected.

2.2 Network anonymity

Network anonymity is typically defined in literature as a quality of the network that ensures the non-uniqueness of its nodes *w.r.t.* a desirable anonymity measure, in our case (n, m)-k-*anonymity*. This measure serves as an oracle for the greedy algorithm, determining which nodes of the network are considered anonymous, and is defined as follows:

Definition 2.1 ((n,m)-k-anonymity [Latour, 2024])**.** *Given a simple, undirected graph $G=(V,E)$, it is said to be k-anonymous w.r.t. (n,m) for $k\in\mathbb{Z}^+; n,m\in\mathbb{N}$ if for each combination of n neighboring nodes and m incident triangles there are either zero nodes with that (n,m) combination, or at least k.*

This anonymity measure counts the number of *incident triangles* of each node, where triangle of a graph $G=(V,E)$ is a set of three pairwise unique nodes $u,v,w\in V$, such that $\{u,v\},\{u,w\},\{v,w\}\in E$. Such triangle is *incident* to each of the nodes u, v and w.

In network processing, achieving network anonymity is a desirable goal that ensures its comprising components not to be uniquely identifiable under the chosen anonymity measure — a trait particularly sought-after for publicly accessible networks that guarantees privacy of participating entities. *Equivalence classes* provide framework for determining which nodes are considered anonymous. They are extensively used by the (n,m)-greedy algorithm and the proposed modification to it, and are defined as:

Definition 2.2 (Equivalence class [de Jong *et al.*, 2024])**.** *Equivalence class EC_M of a given graph and anonymity measure M is a set of nodes, such that $\forall v,w\in EC_M : v\cong_M w$, where $v\cong_M w$ describes equivalence of two nodes v and w using anonymity measure M.*

The anonymity of the entire network is assessed based on another measure, in our case graph *uniquenesss*:

Definition 2.3 (Graph uniqueness [de Jong *et al.*, 2024])**.** *The uniqueness $U_M(G)$ of a graph $G=(V,E)$ using anonymity measure M is the fraction of unique nodes in the graph given M, i.e., the fraction of nodes $v\in V$ for which $|EC_M(v)|=1$.*

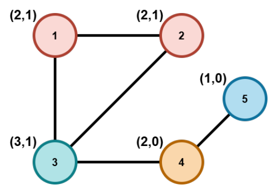
$$U_M(G)=\frac{|\{v\mid v\in V\wedge|EC_M(v)|=1\}|}{|V|}\tag{1}$$

2.3 Motivating Example

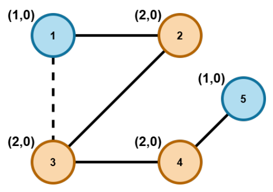
Consider a network consisting of five nodes and five edges, as depicted in Figure 1a. According to (n,m)-2-anonymity, the anonymity measure used in the (n,m)-greedy algorithm, there are initially two anonymous nodes: 1 and 2, and three uniquely identifiable nodes: 3, 4 and 5. Equivalence classes are represented in the figure using distinct colours. All nodes with the same signature, as is the case with nodes 1 and 2, share the same colour. All nodes with unique signatures have colours that are not assigned to any other node.

The (n,m)-2-anonymity takes into account both the degree of the node, represented by n, and the number of incident triangles, represented by m. For that reason, node 4 is distinguishable from nodes 1 and 2 because it has no incident triangles, despite the identical degree. On the other hand, nodes 1 and 2 have the same signature $(2,1)$, making them anonymous. In the figure, each node is annotated with its signature next to it.

To anonymise the network, the edge connecting nodes 1 and 3 is deleted, leading to the situation in Figure 1b. The dashed line connecting these nodes indicates the removed edge. As a result, nodes 1 and 5 have degree 1, whereas nodes 2, 3 and 4 have degree 2. However, in comparison to the situation before deleting the edge, there are no triangles, meaning that nodes 1 and 5 have signature $(1,0)$ and all the other nodes have signature $(2,0)$. This is also reflected by the updated colours of nodes — nodes 1 and 5 share one colour, whereas nodes 2, 3 and 4 share the other. Since there are no nodes that have a unique signature in this example, the network became fully anonymised.



(a) Network before edge deletion.



(b) Network after edge deletion.

Figure 1: Influence of edge deletion on anonymity of the network.

3 Approach

In this section, we present a detailed approach to modifying the (n,m)-greedy deletion algorithm using the sizes of equivalence classes. We first outline the structure of the algorithm and rationale behind it, and identify how it can be modified efficiently to incorporate the chosen heuristic. We propose several variants of the modification based on a common set of requirements and motivate our choice.

3.1 (n,m)-greedy edge deletion

The (n,m)-greedy edge deletion method described in [Xie, 2023] aims to anonymise the graph by repeatedly removing edges that result in the largest decrease in the uniqueness of the network ΔU. For any node that is connected by the deleted edge $\{u,w\}$ or belongs to a triangle this edge is a part of, such an operation leads to a change in the node's signature (n,m). More specifically, the degree of nodes connected by the removed edge is affected. Furthermore, the number of incident triangles will change for all nodes belonging to triangles with the deleted edge. These changes are defined in the following manner:

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

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

where $V_1(v)$ is a set of nodes consisting of v and its direct neighbours.

3.2 Edge Deletion Algorithm

The codebase used in this research is an adaptation of work presented in [Xie, 2023]. The presented code is functionally equivalent and includes minor performance improvements and generalisations. Any of the modifications to the original algorithm proposed in Section 3.4 is provided as an argument to Algorithm 1.

Algorithm 1 Edge Deletion Algorithm

```
Input: original graph $G = (V, E)$, evaluation function $f$
 1: $U[0] \leftarrow uniqueness(G)$
 2: $G' \leftarrow G$
 3: $i \leftarrow 1$
 4: while $|E| > i \wedge U[i-1] \neq 0$ do
 5:     $(V', E') \leftarrow G'$
 6:     for $e \in E'$ do
 7:         $\mathit{eff}, size_u, size_v \leftarrow evaluate(G, e)$
 8:         $E_{eval}[e] \leftarrow f(\mathit{eff}, size_u, size_v)$
 9:     end for
10:     $e_{del} \leftarrow argmax(E_{eval})$
11:     $delete(G', e_{del})$
12:     $U[i] \leftarrow uniqueness(G')$
13:     $i \leftarrow i + 1$
14: end while
Output: $G', U$
```

The presented algorithm takes as arguments the graph we intend to anonymise G, as well as a function evaluating the priority of the deletion of edges f. It starts by computing the initial uniqueness of the input graph and stores it in the output array U with graph uniqueness values, where index i in that array corresponds to the uniqueness value of the modified graph after i deletions. The algorithm then proceeds to remove edges until the network is fully anonymised. For each iteration of the algorithm, all remaining edges are evaluated according to the initial measure returned by Algorithm 2 that is used in the (n, m)-greedy algorithm. The returned evaluation value *eff* and sizes of equivalence classes of nodes connected by the evaluated edge, $size_u$ and $size_v$, become arguments for the provided evaluation function f. All edges are ranked according to the output provided by function f and the edge with the largest value e_{del} is removed using the *delete* function. The iteration is concluded by recomputing the uniqueness of the graph and updating the counter for the iteration number i. Finally, the algorithm returns the anonymised graph G' and the array of uniqueness values after each deletion U.

As a result, edges of the graph can be ranked according to an arbitrary measure that takes three numerical arguments. In addition to the estimated change in uniqueness *eff*, *evaluate* function described in Algorithm 2 also returns the sizes of equivalence classes of nodes u and v connected by the inspected edge $e = \{u, v\}$ to avoid computational overhead.

This function estimates the change in the uniqueness of the graph by calculating the change in the number of anonymous nodes as a result of edge deletion. It counts the number of anonymous edges before the removal, recomputes the signatures of nodes after an edge deletion according to the method outlined in Section 3.1 and counts the number of anonymous edges again. The returned value *eff* is the difference between the number of anonymous edges after and before the deletion.

Algorithm 2 *evaluate* function

```
Input: 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: $V_{\mathit{eff}} \leftarrow V_1(u) \cap V_1(v)$
 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:     $S'(w) \leftarrow update(S(w), \{u, v\})$
14:     Add $w$ to the new 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
23: $size_u \leftarrow |C_S(u)|$
24: $size_v \leftarrow |C_S(v)|$
Output: $\mathit{eff}, size_u, size_v$
```

3.3 Motivation Behind the Size of Equivalence Classes

In this subsection, we motivate the choice of the size of equivalence classes as a suitable heuristic to take into account when determining the edge to be removed in an iteration. We observe that neither $\Delta n(v)$ nor $\Delta m(v)$ can be positive, hence each deletion leads to a decrease in n, m or both values of the signature (n, m) for some of the nodes. m describes the number of incident triangles in the 1-neighbourhood of the node v, *i.e.* in a subgraph consisting of node v, all of its neighbours and all edges that connect any two neighbours of v, as well as v with any of its neighbours. Any triangle incident to v in such a scenario will include exactly two edges going out of v: $\{v, u\}$ and $\{v, w\}$, and exactly one edge connecting nodes u and w. For that reason, every triangle incident to v in 1-neighbourhood of node v can be represented by an edge that is not incident to that node. Using this observation, we note that for a given n, the maximum number m in the signature (n, m) is:

$$m_{max}(n) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

It means that there are fewer low-degree signatures, *i.e.* those with low values of n, making it more likely for multiple nodes of degree n to share a signature. As a result, the expected size of an equivalence class increases as the value of n decreases. Methods that outperform (n, m)-greedy algorithm in the long run tend to prioritise edges connecting two low-degree nodes [Xie, 2023], which means that ranking higher the edges that connect nodes belonging to larger equivalence classes could yield promising results. However, unlike a heuristic based explicitly on the degree of nodes, the proposed alternative takes into account the global structure of the network. This potentially addresses the issue of the (n, m)-greedy algorithm searching for locally optimal modifications to the network at the expense of long-term performance.

3.4 Proposed Modifications

Due to the nature of the (n, m)-greedy deletion method, a convenient and computationally efficient modification consists of a function that takes three arguments: expected change in uniqueness ΔU used to originally rank the edges, and the sizes of equivalence classes $|EC(u)|$ and $|EC(w)|$ for an edge $\{u, w\}$. We also specify that this function must provide the same output if the sizes of equivalence classes are interchanged, because the graphs are undirected. From now on, we will refer to this property as *symmetry*.

In each iteration of the (n, m)-greedy deletion algorithm, the remaining edges are now ordered according to the output of the proposed function $f(\Delta U, |EC(u)|, |EC(w)|)$, where higher value indicates a more desirable deletion. Importantly, equivalence classes do not need to be computed for each evaluation of the function or even the iteration again — an efficient implementation in Algorithm 2 reuses the equivalence classes used to determine ΔU.

Multiplication and Addition

One of the most straightforward approaches to include additional variables, here $|EC(u)|$ and $|EC(w)|$, in the formula, is to incorporate a commutative operation combining these variables, ensuring the symmetry. Examples of such operations include addition and multiplication. Such an approach would also prioritise edges connecting nodes that belong to larger equivalence classes. Therefore, we suggest the following formulas:

$$(1)\ f(\Delta U, x, y) = \Delta U \cdot x \cdot y$$
$$(2)\ f(\Delta U, x, y) = \Delta U \cdot (x + y)$$

Algorithms using functions (1) and (2) will hereafter be named *Multiplication* and *Addition* algorithms respectively.

Machine learning activation functions

LR-based anonymisation methods present exceptional performance for a large number of deletions. For these methods, the edges to be removed are decided by a trained model using an activation function. Therefore, activation functions that are widely used in logistic regression and other machine learning applications are noteworthy candidates for the proposed function. One of them is the *softmax* function, which is defined as:

$$\sigma(E)_i = \frac{e^{e_i}}{\sum_{e_x \in E} e^{e_x}}$$

where i is the index of a specific edge.

We observe that for every i, the denominator of $\sigma(E)_i$ is the same. Hence it can be omitted, or in other words $\sigma(E)_i \propto e^{e_i}$. Such a simplification allows us to avoid linear time complexity with respect to the number of remaining edges E for a single evaluation of the function. As a result, we present two variants of this function:

$$(3)\ f(\Delta U, x, y) = \Delta U \cdot e^x \cdot e^y$$
$$(4)\ f(\Delta U, x, y) = \Delta U \cdot (e^x + e^y)$$

Additionally, one of the regression methods that takes multiple input values and is widely used in machine learning is Multiple Linear Regression (MLR). This function is defined as:

$$P(X) = \frac{1}{1 + e^{-(\beta_0 + \Sigma_{x_i \in X} \beta_i \cdot x_i)}}$$

Consider $X = \begin{pmatrix} x \\ y \end{pmatrix}$, which gives rise to three variables for $P(X)$: β_0, β_1 and β_2. To ensure symmetry, $\beta_1 = \beta_2$ must hold. Moreover, these variables can be equal to the multiplicative identity of 1, whereas β_0 can be equal to the additive identity of 0 for simplicity. These considerations give rise to the following suggestion for the formula:

$$(5)\ f(\Delta U, x, y) = \frac{\Delta U}{1 + e^{-(x+y)}}$$

Hereafter, algorithms using functions (3) and (4) will be called *Softmax Multiplication* and *Softmax Addition* algorithms, whereas the algorithm using function (5) will be called *MLR* algorithm.

Formalism of Proposed Functions

We define a set of constraints that all proposed functions in the form $f(\Delta U, |EC(u)|, |EC(w)|)$ must respect. These requirements guarantee that the proposed functions are compatible with undirected graphs and that ordering of the output values can be established for specific inputs:

$$(6)\ f(\Delta U, x, y) = f(\Delta U, y, x)$$
$$(7)\ x_1 > x_2 \Rightarrow f(\Delta U, x_1, y) > f(\Delta U, x_2, y)$$
$$(8)\ x_1 > x_2 \land y_1 > y_2 \Rightarrow f(\Delta U, x_1, y_1) > f(\Delta U, x_2, y_2)$$
$$(9)\ \Delta U_1 > \Delta U_2 \Rightarrow f(\Delta U_1, x, y) > f(\Delta U_2, x, y)$$

Equation (6) describes the symmetry mentioned in the preceding subsections. This property ensures that the output of the function does not depend on the order of the provided sizes of equivalence classes. Inequalities (7), (8) and (9) establish a trivial ordering that guarantees that if only a subset of input arguments differs, the ordering of output values will depend solely on the ordering of that subset of arguments.

4 Experiments

In this section, we present a detailed research question and associated subquestions, define relevant criteria to evaluate the algorithms against, and describe the experimental setup used to assess the proposed improvements to the (n, m)-greedy deletion algorithm. We also compare the performance of each modified algorithm against its unmodified counterpart, hereby also referred to as the original algorithm.

4.1 Research Questions

Main Question: How does including the size of equivalence classes in edge evaluation influence the performance of (n, m)-greedy edge deletion anonymisation?

- **Solution Quality**: How does the final uniqueness of the network change?
  Evaluation criterion: Lower final uniqueness for some network topologies, number of deletions, or both achieved by modified algorithms in comparison to the original algorithm.
- **Available Budget**: How does the number of available deletions impact the change in network uniqueness?
  Evaluation criterion: Identification of thresholds or intervals over which the original (n, m)-greedy algorithm is outperformed by its modified versions with reference to the number of edge deletions.
- **Solving Speed**: How does the time required to complete the algorithm change?
  Evaluation criterion: Comparable physical runtime and reasonable additional time complexity of modified algorithms in comparison to the original one.

4.2 Experimental Setup

Software.

We implement the original (n, m)-greedy algorithm and all of the presented modifications[1] with Python 3.9.8. We also use the following libraries: igraph 0.11.9 and NetworkX 3.2.1 for graph manipulation, NumPy 2.0.2 for mathematical operations, SciPy 1.13.1 for reading *Matrix Market* format graphs and Matplotlib 3.10.0 for graph plotting.

Hardware.

We run our experiments on DelftBlue cluster [Delft High Performance Computing Centre (DHPC), 2024], where nodes use Intel(R) Xeon(R) Gold 6226R CPU, running at 2.90GHz. The DelftBlue's operating system is Red Hat Enterprise Linux 8.10.

Experimental parameters.

We restrict our algorithms to 5 CPUs per task, 3600 CPU s, and 4 GB RAM per processing unit to encode the network and execute an algorithm on it. The reported runtime is measured in CPU s by DelftBlue nodes.

Problem instances.

The dataset used in our experiments consists of five networks retrieved from *NetworkRepository*[2], *Copenhagen Networks Study*[3], *SuiteSparse Matrix Collection*[4] and *Matrix Market*[5]. We created this set with the aim of including networks from different fields that have different topologies, like social networks and grid-like graphs. We also wanted to include densely and sparsely connected networks, as well as graphs with varying initial uniqueness. We also limited the number of edges in the network to at most 10^5 to ensure reasonable runtime given the experimental setup. The networks used in the experiments are listed below:

Western US Power Grid (power-net): Network representing the power stations in the Western US and connections between them. There are several variants of these graphs available in the repository and we chose the variant named bcspwr08.

University Emails (email-univ): Network of university emails resembling a social network.

COPNET Network of Facebook Friends (fb-friends): Social network representing Facebook friendship connections between students.

DIMACS Benchmark Graph C125-9 (C125-9): Artificial grid-like network created as part of DIMACS benchmark set.

Economic Model of Victoria, Australia, 1880 (mahindas): Economic model of the Australian state — Victoria. For the purpose of this research, this asymmetric network is transformed into an undirected graph using basic functionality of the NetworkX library.

Network		$\lvert V \rvert$	$\lvert E \rvert$	Avg. Degree	Avg. Num. of Triangles	Initial Uniqueness
power-net	[4]	1624	3837	5	0	0.011
email-univ	[2]	1133	5451	9	14	0.230
fb-friends	[3]	800	6429	16	51	0.472
C125-9	[2]	125	6963	111	5535	0.792
mahindas	[5]	1258	7619	12	8	0.102

Table 1: Networks used in the experiments.

4.3 Experimental Results

In this section, we present the experimental results and address the posed research questions.

Solution Quality

We compare the quality of solutions produced by the original algorithm and all proposed modified algorithms for all five networks. Since all of the algorithms are deterministic, it is sufficient to run each of them on every network from the dataset once. In the experiments, we found that the (n, m)-greedy algorithm is consistently outperformed by the proposed algorithms in terms of solution quality for social networks. One example of such network is email-univ, presented in Figure 2.

From the suggested algorithms, we observed that both the *Softmax Multiplication* and the *Softmax Addition* rank among the best-performing for the aforementioned social networks but tend to produce suboptimal results against the *Multiplication*, *Addition* and *MLR* algorithms for the most sparsely connected networks in our dataset — mahindas and power-net. Despite keeping comparable graph uniqueness to the original algorithm throughout the execution, they also anonymise the network fully, *i.e.* lead to the graph uniqueness of 0, after a considerably larger number of deletions for these networks. This observation is depicted in Figure 3.

We found that for densely connected networks, as is the case with C125-9, for which the experimental results are presented in Figure 4, none of the proposed algorithms provide considerable improvement for a large number of deletions.

[1]Available at `https://github.com/Leightox/researchproject`

[2][Rossi and Ahmed, 2014]

[3][Sapiezynski *et al.*, 2019]

[4][Davis and Hu, 2011]

[5][Boisvert *et al.*, 1997]

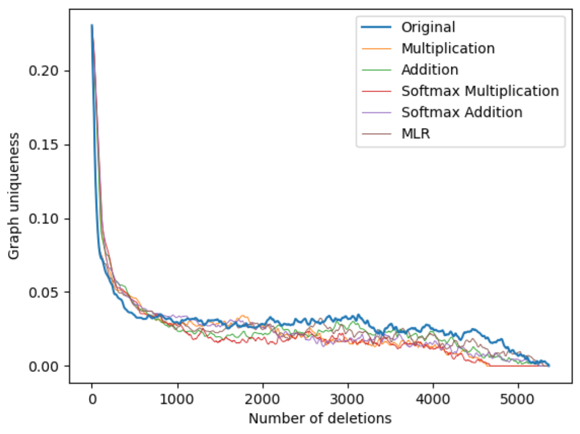


Figure 2: Performance in terms of graph uniqueness of the investigated algorithms on the email-univ network.

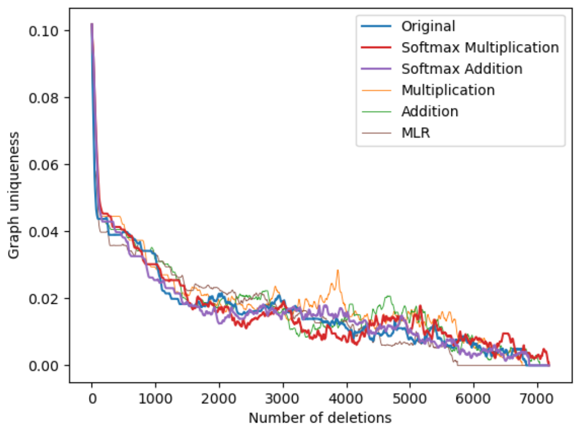


Figure 3: Performance in terms of graph uniqueness of the *Softmax* algorithms against other algorithms on the mahindas network.

Moreover, we observed that this is the only network from our dataset for which the majority of suggested modifications are performing suboptimally to the original algorithm for any number of removals. The *MLR* algorithm performs comparably well to other presented algorithms given a large number of operations on the network. However, it offers an observable improvement for the C125-9 network up to the number of deletions after which the graph uniqueness decreases for all of the proposed implementations. In the case of social graphs, this algorithm required more deletions than all of the proposed alternatives and typically had higher graph uniqueness throughout the execution of the algorithm. Another exception to that is the mahindas network, for which the *MLR* fully anonymised the network using almost 20% fewer deletions than the next algorithm — the *Multiplication* algorithm.

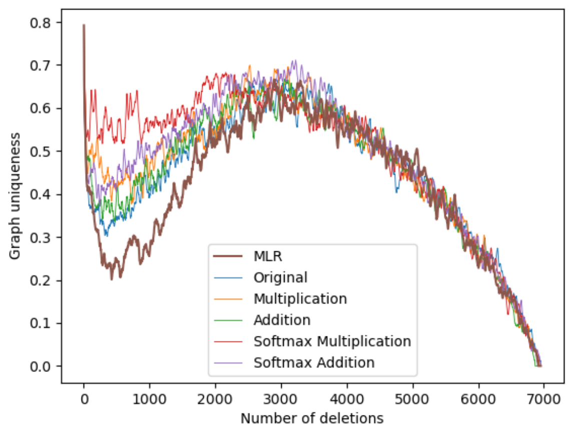


Figure 4: Performance in terms of graph uniqueness of the *MLR* algorithm against other algorithms on the C125-9 network.

Available Budget

Alongside the solution quality, we also investigate under what circumstances the proposed algorithms yield lower uniqueness of the modified graph than the original algorithm. Our experiments confirm that the (n, m)-greedy algorithm leads to lower graph uniqueness for small number of deletions, ranging from 1% of the total number of edges for sparse networks like mahindas and power-net to 5–10% of the total number of edges for social networks, which are fb-friends and email-univ in our dataset. However, as more deletions are performed, the proposed modified algorithms produce solutions of better quality in terms of the uniqueness than the original algorithm.

In the experimental results, we can oftentimes identify approximate thresholds of the number of deletions, above which a certain subset of the suggested algorithms or all of them tend to perform better than the original algorithm in terms of the graph uniqueness. Importantly, these algorithms and thresholds differ across the networks in our dataset. For social graphs, the example of which is email-univ presented in Figure 2, all of the suggested algorithms typically return more anonymised networks than the original algorithm if the available budget, *i.e.* the maximum number of allowed edge removals, exceeds the previously identified threshold. In the case of sparse networks, some of the proposed solutions, most notably the *MLR*, *Multiplication* and *Addition* algorithms, yield lower graph uniqueness only over certain intervals of the number of deletions. Even though these algorithms fully anonymise the graph using smaller number of edge removals, there are some values of the uniqueness for which the original algorithm requires fewer iterations. Such a situation takes place in the case of the power-net network, depicted in Figure 5.

We also observed that for the C125-9 network illustrated in Figure 4, only the *MLR* algorithm provides noticeable improvement in terms of the solution quality over the original algorithm, although for smaller available budget, unlike for the other graphs in our dataset. On the other hand, the perfor-

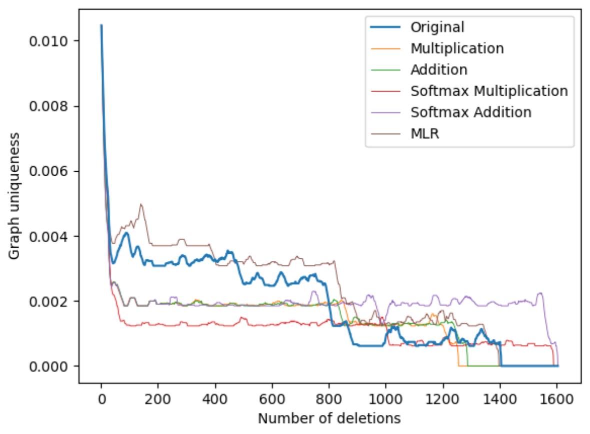


Figure 5: Performance in terms of graph uniqueness of the investigated algorithms on the power-net network.

mance with regard to that metric of other proposed algorithms was either comparable to the original algorithm, as was the case for the *Multiplication* and the *Addition*, or considerably worse than the original algorithm, like with the *Softmax* algorithms. Moreover, none of the suggested algorithms provided noticeable improvement for larger number of deletions.

Solving Speed

For the original (n, m)-greedy algorithm and all of the proposed modifications, we execute the algorithms five times and measure their runtime. Due to the limited timeframe of the research, as well as considerable computational complexity of the algorithms, the solving speed is investigated only on one network that represents the entire dataset the most accurately. We choose fb-friends for that purpose because of comparable performance in terms of graph uniqueness and comparable number of deletions required to fully anonymise the network for all of the inspected algorithms.

Table 2 compares the runtime measurements obtained for all considered algorithms. All of the presented algorithms, including the original (n, m)-greedy algorithm, complete the execution in comparable time ranging from 50 minutes to 53 minutes consistently. We observe that algorithms requiring more deletions to fully anonymise the network have marginally higher running times. However, both their average total execution time $\overline{T}$ and the average execution time per iteration $\frac{\overline{T}}{N}$ are comparable across all of the algorithms, considering the size of the network. For the algorithms performing more deletions N, we note that their average total running time is higher but the average running time per iteration is slightly lower, likely due to later iterations being less time consuming.

5 Conclusion

This work focused on improving the (n, m)-greedy edge deletion algorithm using a global heuristic. First, we conducted a literature review that helped identify the shortcomings of the original algorithm and what characteristics of edges are typically favoured implicitly by algorithms that outperform greedy in the long run. One of the largest drawbacks of (n, m)-greedy method originates from the fact that this strategy prioritises locally optimal deletions in the network, initially providing outstanding performance but deteriorating drastically the more edges are removed from the network. We address this issue by specifying that aside from the estimate in the uniqueness of the graph used by the original algorithm, additionally incorporating the size of equivalence classes of nodes connected by the deleted edge may yield promising improvements in the quality of solutions in the long run. To implement and verify this idea, we formulated several algorithms that modify how the original greedy framework ranks edges with the aim of prioritising larger equivalence classes, inspired by the contemporary literature and ideas present in other anonymisation methods.

Our experimental results indicate that the size of equivalence classes holds considerable promise as a suitable global heuristic, oftentimes helping the proposed algorithms produce solutions of improved quality in terms of the graph uniqueness in comparison to the (n, m)-greedy algorithm. For suggested algorithms exhibiting noticeable advantage over the greedy algorithm, we reported that there exists a number of edge deletions above which the solution quality was usually at least as good as for the original algorithm.

The algorithms proposed by us vary in terms of performance depending on the network they are run on, especially in the context of the solution quality. We observed that the *Softmax* algorithms give rise to the lowest graph uniqueness out of the suggested algorithms for social networks but underperform with regard to that measure for sparse and dense graphs, sometimes even against the original algorithm. On the other hand, the *MLR* algorithm performs well in terms of the produced graph uniqueness on sparse networks, especially the mahindas network. It is also the only algorithm to provide noticeable improvement in that regard for the dense C125-9 network and, unlike any other algorithm, for a limited number of iterations. The *Multiplication* and *Addition* algorithms provide a solid middle ground between other proposed solutions — while their performance in the context of solution quality is usually outmatched by one of the other suggested algorithms, they typically produce satisfactory results that are superior to the graph uniqueness obtained by using the original algorithm. Between these two algorithms, the *Multiplication* algorithm finishes in considerably lower or comparable number of iterations for all networks from our dataset.

Importantly, the presented enhanced algorithms introduce negligible computational and runtime overhead, making these solutions practically applicable in place of the (n, m)-greedy edge deletion. Moreover, the measured total runtime was lower for proposed algorithms that required fewer edge deletions, while their average runtime per iteration was comparable with the measurements obtained for the original algorithm. This can also be interpreted as an indirect improvement in terms of solving speed with respect to the (n, m)-greedy edge deletion.

Algorithm	N	$\overline{T}$	$\min(T)$	$\max(T)$	σ_T	$\frac{\overline{T}}{N}$	$\frac{\min(T)}{N}$	$\frac{\max(T)}{N}$	$\frac{\sigma_T}{N}$
Original	6344	3144	3142	3147	1.78	0.496	0.495	0.496	$2.81 \cdot 10^{-4}$
Multiplication	5968	3027	3020	3031	4.07	0.507	0.506	0.508	$6.83 \cdot 10^{-4}$
Addition	6220	3030	3024	3041	6.51	0.487	0.486	0.489	$10.47 \cdot 10^{-4}$
Softmax Multiplication	5911	3033	3028	3038	3.79	0.513	0.512	0.514	$6.41 \cdot 10^{-4}$
Softmax Addition	6026	3041	3033	3051	5.72	0.505	0.503	0.506	$9.50 \cdot 10^{-4}$
MLR	6274	3057	3045	3083	13.76	0.487	0.485	0.491	$21.92 \cdot 10^{-4}$

Table 2: Runtime measurement results of investigated algorithms on the fb-friends network. N represents the number of deletions to fully anonymise the network, and T stands for the set of running times for the corresponding algorithm. The results in all rows other than the one corresponding to the number of deletions N are reported in seconds. The results are presented with respect to the total running time and the running time per iteration using four measures: mean, minimum and maximum values, and standard deviation.

5.1 Future Work

In this paper, we investigated a number of functions in the form of $f(\Delta U, |EC(a)|, |EC(b)|)$ that augmented the evaluation of edges in the proposed modifications of the (n, m)-greedy algorithm. Future extensions to our work could include inspecting other functions or the suggested functions using different networks. In particular, all investigated instances consisting of a network and an introduced algorithm showed striking resemblance to the original algorithm in the general pattern of network uniqueness throughout the algorithm's execution. It is worth examining whether this pattern is identifiable because of the estimate of the change in uniqueness and its significance in ranking the edges for all of the considered versions of the anonymisation method.

Only one of the suggested algorithms — the *MLR* algorithm — lead to a noticeable improvement in the performance on C125-9 network, and only up to a certain budget. Interestingly, this algorithm usually underperformed against its counterparts for networks exhibiting different characteristics, especially social networks like email-univ and fb-friends but also power-net, which is another sparse network. For that reason, we could pay special attention to investigating what kind of functions could produce more promising results for dense networks like C125-9 and if different, not identified in this report, properties might govern such lack of improvement for that specific type of graphs in the case of other proposed algorithms.

In order to confirm the experimental results of solving speed for the fb-friends network, mainly that the proposed algorithms introduce negligible computational overhead in terms of running time, measurements can be performed on other networks belonging to our dataset or completely different graphs. Furthermore, the existing experiment can be repeated to obtain additional runtime measurements for the inspected network.

We also inspected how prioritising edges connecting nodes that belong to larger equivalence classes influences the performance of the algorithms. We argued that such enhancement of the evaluation mechanism might exploit an underlying mathematical property of deletion-based methods. However, further investigation could be conducted to identify this phenomenon more rigorously.

6 Acknowledgements

This research was conducted as part of CSE3000 Research Project course at Delft University of Technology. The computational resources for this work were provided by the university's DelftBlue supercomputer. We thank our supervisor, Anna L.D. Latour, and the research team — Rachel G. de Jong and Frank W. Takes — for their continued support and guidance throughout the project, and providing valuable insight and resources. We also thank the entire Network Anonymity for Science group for giving constructive feedback and engaging collaboration during the project.

Responsible Research

Ethical Considerations

Ethical considerations are some of the core motivations behind network anonymisation as a whole. Whenever the entities participating in a network cannot be uniquely identifiable for ethical but also legal reasons, reliable and efficient anonymisation methods are necessary. For example, privacy plays a key role in healthcare, security or business — important aspects of everyday life. However, one of the most considerable drawbacks of network anonymisation is how resource-demanding and computationally-expensive it is currently. For that reason, economically or technologically challenged individuals or societies may be excluded from benefiting from advancements in privacy preservation of data.

Our work presents modifications to an existing algorithm that aim to improve its overall performance. We anticipate that our results or recommendations for future work could guide the scientific community to further advancements that would bridge the aforementioned gap and make anonymisation of graph-like structures more accessible and affordable on a larger scale. We acknowledge that advancements in the field of network anonymisation may also be misused by malicious parties. Aside from ensuring anonymity in the positive context, such applications can be used to purposefully obscure vital features of individual entities participating in a network or the graph as a whole. This may raise concerns about the reproducibility or applicability of future research on datasets modified using anonymisation methods.

Research Integrity

Throughout conducting this research, we upheld standards outlined in the TU Delft Code of Conduct [Roeser and Copeland, 2020] and the Netherlands Code of Conduct for Research Integrity [KNAW *et al.*, 2018], and we followed conventions and best practices existing in the field of network anonymisation and algorithmics. We also devised internal author instructions that we respected in reporting the experimental setup and obtained results in accordance with the resolutions of the aforementioned codes of conduct. The produced results were reported accurately and discussed thoroughly, providing objective and balanced insight with respect to the posed research questions. We credited all external sources that were cited in this paper and datasets used in the experiments.

Reproducibility

Ensuring reproducibility of results obtained and presented in this paper was a vital aspect of our work. We provide a reproducible pseudocode, as well as a repository containing the source code used in the experiments. We also include detailed specification of the used software, hardware and their setup to minimise the possibility of discrepancies in the produced output. Furthermore, we ensured that all of the presented modifications of the original algorithm are deterministic. However, the runtime of the algorithms might vary depending on the node they are run on or the parameters specified for Delft-Blue jobs. Depending on the permissions granted to a Delft-Blue account, the available resources may also differ.

Appendix A Additional Graphs for Solution Quality

This section includes graphs presenting the performance in terms of graph uniqueness of the investigated algorithms on networks, for which such graphs weren't included in the report's body.

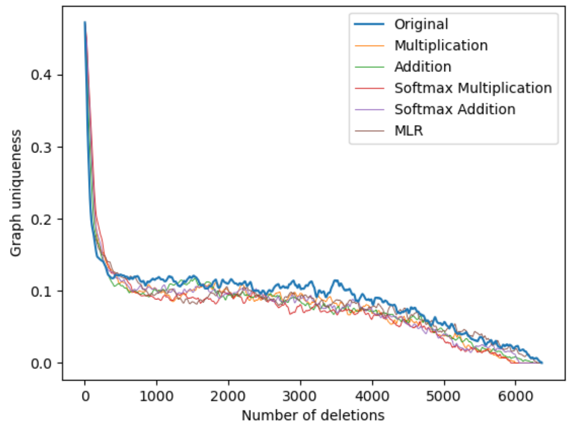


Figure 6: Performance in terms of graph uniqueness of the investigated algorithms on the fb-friends network.

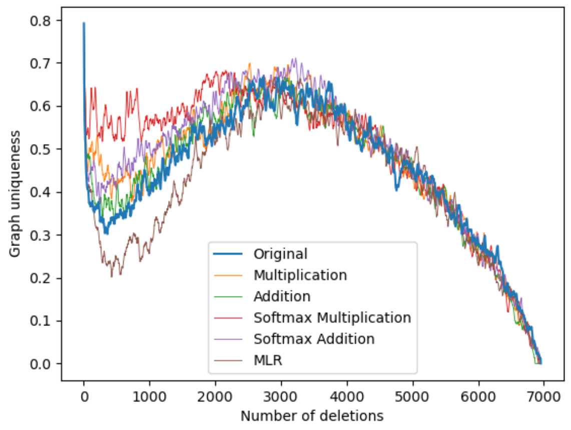


Figure 7: Performance in terms of graph uniqueness of the investigated algorithms on the C125-9 network.

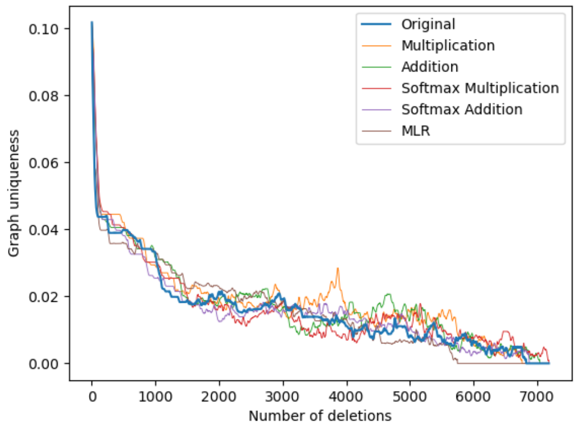


Figure 8: Performance in terms of graph uniqueness of the investigated algorithms on the mahindas network.

References

[Boisvert *et al.*, 1997] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard F. Barrett, and Jack J. Dongarra. *Matrix Market: a web resource for test matrix collections*, pages 125–137. Springer US, Boston, MA, 1997.

[Davis and Hu, 2011] Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1), December 2011.

[de Jong *et al.*, 2024] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. A systematic comparison of measures for k-anonymity in networks, 2024.

[de Jong *et al.*, 2025] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. The anonymization problem in social networks, 2025.

[Delft High Performance Computing Centre (DHPC), 2024] Delft High Performance Computing Centre

(DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[KNAW *et al.*, 2018] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU. Nederlandse gedragscode wetenschappelijke integriteit, 2018.

[Latour, 2024] Anna L.D. Latour. Research Note - Anonymisation - ILP encoding. 2024.

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

[Machanavajjhala *et al.*, 2006] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, 2006.

[Roeser and Copeland, 2020] S. Roeser and S.M. Copeland. Tu delft code of conduct: Why what who how. 2020.

[Rossi and Ahmed, 2014] Ryan Rossi and Nesreen Ahmed. Networkrepository: A graph data repository with visual interactive analytics. 10 2014.

[Samarati and Sweeney, 1998] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

[Sapiezynski *et al.*, 2019] Piotr Sapiezynski, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann Jørgensen. The Copenhagen Networks Study interaction data. 11 2019.

[Xie, 2023] Xinyue Xie. Anonymization algorithms for privacy-sensitive networks. Master's thesis in computer science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, August 2023. Available at https://theses.liacs.nl/pdf/2022-2023-XieXinyue.pdf.