ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/367455450

An Efficient Approach for Anonymising the Structure of Heterogeneous Graphs

Conference Paper · December 2022

DOI: 10.1109/BigData55660.2022.10020301

CITATIONS
0

READS
11

3 authors, including:

Rudolf Mayer
TU Wien
105 PUBLICATIONS **1,083** CITATIONS
SEE PROFILE

Andreas Ekelhart
SBA Research
72 PUBLICATIONS **1,413** CITATIONS
SEE PROFILE

All content following this page was uploaded by Rudolf Mayer on 01 February 2024.

The user has requested enhancement of the downloaded file.

IEEE Big Data 2022
This is a self-archived pre-print version of this article.
The final publication is available at IEEE via
http://dx.doi.org/10.1109/BigData55660.2022.10020301.

An Efficient Approach for Anonymising the Structure of Heterogeneous Graphs

Guillermo Alamán Requena
SBA Research, Vienna, Austria

Rudolf Mayer
SBA Research, Vienna, Austria
rmayer@sba-research.org

Andreas Ekelhart
SBA Research, Vienna, Austria
aekelhart@sba-research.org,

***Abstract*—Personal, sensitive information contained in data sets is often discouraging the exchange and sharing of data, or even rendering it impossible. To still enable data sharing, anonymisation is a strategy often employed to avoid possible record identification or inference. Anonymisation strategies are often data-type or modality dependent, as besides the actual attributes contained within a dataset, also certain other aspects might reveal information on the data subjects. For example in graph data, such as knowledge graphs, the structure within the graph, i.e. the connection between nodes, might allow to re-identify a specific person, e.g. by knowledge of the number of connections for some individuals within the dataset.**

Therefore, also the structure needs to undergo anonymisation to achieve privacy. In this paper, we optimise an algorithm that extended previous state of the art by considering multiple, different types of connections (relations) between nodes to achieve anonymity among each of these types. Our novel, open-source implementation scales to much larger graphs than previous work, which is important for efficiently anonymising ever-increasing volumes of big, linked data.

***Index Terms*—Graph Structure Anonymisation, Multiple Relational Types, Efficiency**

I. Introduction

With increasing hardware resources, and a growing number of collecting devices and applications, the amount of data collected is ever increasing. Among those are data on e.g. social networks, represented as graphs to store connections between individuals, organisations, and other entities. Several interesting data analysis tasks utilise such graphs, but as this data is highly personal, data protection becomes an important aspect. Besides social networks, many other domains use graphs to represent knowledge, also often including sensitive data. Anonymisation techniques try to address data protection concerns by e.g. reducing the detail and granularity of the data, often navigating the difficult task to preserve enough information for the downstream analysis tasks. Tabular data was among the first types for which anonymisation methods have been developed, for example to counter a re-identifcation attack by k-anonymity [1], or membership inference by differential privacy [2]. For a re-identification attack, **quasi-identifying** attributes [3] are of special concern. These are attributes that by themselves are not uniquely identifying an individual, but do so in combination with other quasi-identifiers, at least for a (large) portion of the participants in the dataset. Frequent examples are birth date, post (ZIP) codes, and sex.

These anonymisation concepts can be applied also to other types of data, but often, the peculiarities of these data have to be considered. In graphs, the values within nodes can often be treated in similar manner as tabular data, e.g. by applying k-anonymity to all quasi-identifying attributes. However, also the structural information encoded in the connections (edges) is of concern. An attacker with sufficient background knowledge might be able to re-identify an individual based on this graph structure alone, for example if the background knowledge includes detailed information on the number and types of connections of an individual. This is specially of concern for individuals that have highly unusual patterns of connections (e.g. unusually few or many connections, or infrequent types of connections in heterogenous graphs, etc.). As a consequence, there is a growing body of literature developing methods for anonymisation of the graph structure, for example adaptations of the concept of k-anonymity to graph connections, and combinations of node and structure anonymisation, in various types of graphs.

Most existing works consider homogeneous graphs, i.e. with only one type (e.g. the Friend of a Friend (FOAF) foaf:knows connection). However, in heterogeneous graphs, nodes are linked by potentially varying types of connections, e.g. because they have different business relations to each other. Heterogeneity provides additional information for an attacker, who could exploit all the structural combinations present in a heterogeneous graph to disclose individual's identities and sensitive information. Heterogeneity makes structure anonymisation more complex, as the search space for optimal, anonymous perturbations of the original graph also expands.

Based on the ideas from [4], we presented initial improvements on this method in [5], where we adapted the method to anonymise any type of heterogeneous RDF graph with multiple connection types. The idea in [4] is that the

This work was partially funded by the European Union's Horizon2020 research and innovation programme under grant agreement No 826078 (project FeatureCloud). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

SBA Research (SBA-K1) is a COMET Centre within the framework of COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna; COMET is managed by FFG.

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

so-called one-hop neighbourhood of any resource should be indistinguishable from the one-hop neighbourhood of at least k-1 other resources, thus achieving k-anonymity of the neighbourhoods. To that end, [4] developed a greedy heterogeneous graph modification algorithm for a simplified RDF graph that includes a limited number of at most four types of different semantic connections. There is however only a pseudo-code description of the algorithm, but no open-source or publicly available implementation available. Based on their approach, our contributions consist of

1) an extension to the approach of [4] to increase flexibility of the anonymisation method,
2) improved efficiency of the anonymisation algorithm to make it usable for larger graphs,
3) a freely available, open-source implementation in Python[1], and
4) an evaluation of the algorithm on synthetically generated graphs, several orders of magnitude larger than [4].

The remainder of this paper is organised as follows. Section II provides an overview on related work. In Section III, a detailed description of our method and extensions is provided, before Section IV evaluates the approach and analyses the efficiency of the algorithm. Finally, we provide conclusions and an outlook on future work in Section V.

II. Related Work

While the term "knowledge graph" has been used in literature for a long time, with the announcement of the Google Knowledge Graph in 2012 [6] major interest in industry and academia followed [7]. Other prominent companies that introduced knowledge graphs include eBay, Facebook, IBM, and Microsoft [8]. According to [7] a Knowledge Graph is defined as "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities". The graph-based data model may be a directed edge-labelled graph, a heterogeneous graph, a property graph, etc. We further distinguish open knowledge graphs, published online and accessible for the public (e.g., DBpedia, Freebase, Wikidata) and enterprise knowledge graphs which are typically company internal and support commercial use-cases [7]–[9]. Applications taking advantage of knowledge graphs include web search, social networks, recommendation engines, personal agents, risk assessment, and more besides.

With this proliferation of networked data, also graph data anonymisation has gained attention in the past years [10]. Attackers may use structural information e.g. in social networks to disclose personal information of individuals. In order to protect privacy, several graph anonymisation methods have been developed.

The most well-known approaches to combat sensitive information disclosure using the structure of networked data rely on the modification of the structure itself. Early works on this line were based on the idea of randomly deleting, adding or switching edges in order to prevent adversaries from using the network structure to identify targeted individuals in it [11], [12].

However, randomisation approaches do not take into account that privacy should be guaranteed for every node. k-anonymisation approaches on graphs address this issue by applying the idea of k-anonymity, a popular tabular data anonymisation method [1], [13]. *(k,l)* anonymity suggests that for each node, there should be at least other *k* nodes with which the node shares *l* neighbours. This is achieved by adding as few edges as possible [14]. Based on [15], [16] presented the idea of *(k_i, k_o)*-degree anonymity, which proposes that the in- and out degree sequences of the graph should be respectively k_i *and* k_o anonymous. This method relies on a combination of addition, switch and extension of edges to achieve this k-property. Several extensions and variants of k-anonymisation-like methods such as k-automorphism proposed by [17], k-RDF-neighbourhood by [4], or k-security and k-isomorphism by [18] have been developed.

Clustering based approaches are also popular for graph anonymisation. The idea behind those methods, firstly purposed by [19], is to cluster nodes together into super-nodes of size k, where k is an anonymity parameter. How to achieve this goal varies depending on the specificities of each method. The most popular algorithm for cluster anonymisation is SaNGreeA proposed in [20]. This method builds clusters of size k greedily according to a defined loss function. One of the main contributions of SaNGreeA is that it goes a step further by taking into account not only the structural information from the graph data, but also descriptive attributes linked to each of the nodes. [21] and [22] extended the ideas of [20] to improve SaNGreeA's performance through improvements on the loss function proposed in the original paper [20].

Similarly, researchers have tried to apply the intuition behind differential privacy to networked data. As described in [2], differential privacy "captures intuitively the increased risk to one's privacy incurred by participating in a database." The most popular approach on graphs is *edge differential privacy* which focuses on preventing an adversary from disclosing the existence of sensitive information (e.g., sexual relationships between individuals) from any single edge in a graph while preserving relevant network structural properties. Several work has been done on this field such as [23], which provides a solution for the publication of non-interactive networked data via differential privacy, [24], which presents a query-based differential privacy preserving graph generator, or [25] which proposes an edge differential privacy model for graph clustering.

Furthermore, probabilistic approaches assign edge probabilities to add uncertainty to graph data. For instance, (k, ϵ)-obfuscation proposed by [26] or random walk approaches like [27].

[1]https://github.com/sbaresearch/graph-anonymisation

III. METHOD

Here, we describe our method, and extensions and adaptions of the k-RDF-Neighbourhood anonymisation approach described by Heitmann et al. [4]. Since our method produces an anonymised version of a heterogeneous RDF graph, we first provide a formal definition of a heterogeneous graph, based on [28]:

Definition 1. *A heterogeneous graph is defined as a directed graph $G = (V, E, A, \Delta)$ where each node $\nu \in V$ and each edge $\epsilon \in E$ are associated with their type mapping functions $\theta(\nu) : V \rightarrow A$ and $\omega(\epsilon) : E \rightarrow \Delta$, respectively.*

Although there are many types of heterogeneous graphs, we focus on those using the Resource Description Framework (RDF)[2] and demonstrate our method on the Friend of a Friend (FOAF) [3] vocabulary. However, we are not restricted to FOAF – instead, our method can handle any RDF graph. Figure 1 shows an example of a heterogeneous RDF graph, where vertices of the type *foaf:Person* represent people, edges of the type *foaf:knows* represent relations between individuals, and edges of the type *foaf:CurrentProject* indicate projects an individual is working on. Other edges primarily serve to describe properties, such as *foaf:Age* or *foaf:Name*. The property *custom:has_disease* is an example of a custom property, defined outside the FOAF specification.

As our method addresses structural information, it operates on neighbourhoods of individual nodes; most specifically, we consider the so-called one-hop-neighbourhood.

Definition 2. *The one-hop-neighbourhood of a node ν is the subgraph $G_{1hop}(\nu) = (V_{1hop}, E_{1hop})$ where V_{1hop} is the set of all nodes ν_0 directly connected to ν (including ν itself) and E_{1hop} is the set of edges ϵ_0 connecting ν with $\nu_0 \in V_{1hop}$ and all the edges $epsilon_{V_{1hop}}$ which connect vertices in V_{1hop} among one another[4].*

Figure 1 shows a simplified one-hop-neighbourhood of the node *foaf:Person* "Alice" as all components within the red-dashed line. Following the methodology used in [29] and [4], we will demonstrate our anonymisation method on the one-hop neighbourhood of *foaf:Person* resources, which is a prime candidate for anonymisation, since protecting the privacy of individuals is the most common setting for anonymisation tasks.

The user needs to specify a list of attributes to consider for structure anonymisation, all other attributes will be removed by our method. Furthermore, certain node and edge types can be manually marked for removal or to be preserved as is, if sensitive attributes are meant to be published. The remaining edges form the so-called *target graph*. Note that node value anonymisation, if necessary, is a pre-requisite step to prevent re-identification, and not covered by our structure anonymisation method. However, it is possible to easily combine both.

Depending on the type of information they describe, edge connections within the one-hop-neighbourhood of a node ν of a target graph can be grouped into three different categories:

- **Attribute connections** are those edges that connect a node (e.g., *foaf:Person*) to a descriptive characteristic of this node; the value of which is stored in a Literal.
- **Unidirectional connections** are directed edges that connect a node (e.g., *foaf:Person*) to other entities, such as a document (*foaf:Document*) or a project (*foaf:CurrentProject*).
- **Bidirectional connections** are edges that symmetrically connect nodes (e.g., *foaf:Person*) with each other, e.g. the *foaf:knows* property.

With the list of edge types that exist in the one-hop-neighbourhood of each node ν of a target graph, we can define the anonymisation criteria in terms of the one-hop-neighbourhoods of $\nu \in N$ and the three types of connections described above.

Definition 3. *A **heterogeneous RDF graph is said to be k-anonymous** if there are at least k identical one-hop-neighbourhoods in the target graph for each node $\nu \in N$. Each group of nodes of size k with identical one-hop-neighbourhoods are called anonymised neighbourhoods. We consider that two attributes of the one-hop-neighbourhood of a pair of nodes x and y, are identical if they are generalised to the same level. We consider two unidirectional connections of the one-hop-neighbourhood of a pair of nodes x and y to be identical if they point exactly to the same resources. We consider the bidirectional connections of the one-hop neighbourhood of a pair of nodes x and y to be identical if their one-hop-neighbourhoods are isomorphic.*

We rely on three different steps or sub-algorithms (similar to [4]) to achieve the anonymisation criteria defined above. These algorithms are described below.

A. The Neighbourhood Code Extraction Algorithm

This step relies on comparing one-hop-neighbourhoods of target nodes (e.g., *foaf:Person*) across the target graph. Thus, we encode the node neighbourhood information into a more efficient data structure than the raw RDF graph. Due to its low indexing complexity ($O(n)$), a hashtable is well suited for this representation. The information contained in the one-hop-neighbourhood of a node ν is stored in different ways, depending on the type of edge connection, as follows:

- For **attribute connections**, the attributes of each node are stored as *key-value* pairs (e.g. *foaf:Age*="40").
- Regarding **unidirectional connections**, the resources to which each unidirectional connection of a node points to are stored in a list. The type of connection is the *key* and the list of resources is the *value* associated with it (for example: *foaf:CurrentProject*= ["Project1", "Project3", "Project7"]).

[2]https://www.w3.org/RDF/

[3]http://www.foaf-project.org/

[4]The one-hop-neighbourhood of a node is sometimes referred as the 1.5 degree network of a node, especially in the context of egocentric networks in social network analysis

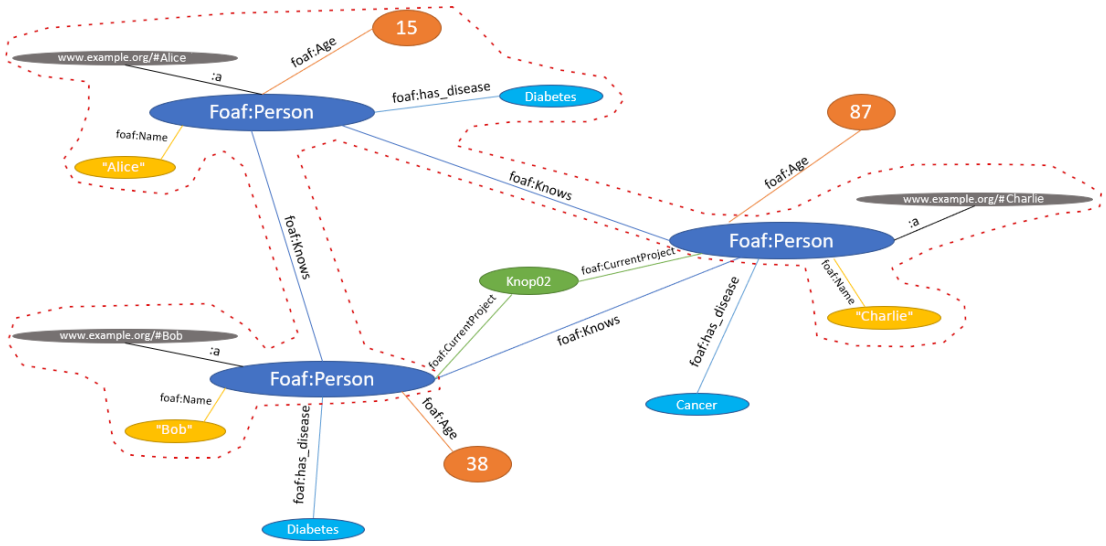


Fig. 1: Example of a heterogeneous RDF graph using FOAF vocabulary and custom properties

- **Bidirectional connections** are the most complex case in our structure anonymisation algorithm. This is due to the inter-connectivity of the targets through this type of connections. Multiple isomorphic tests have to be conducted for each bidirectional connection, to determine if two graphs are the same. At this time, no polynomial time algorithm for the general isomorphic problem [29] is known. For that reason, we need a better representation of the bidirectional connections of the one-hop-neighbourhood of the *foaf:Person* nodes. In our approach, we utilise the same string representation of the edges as proposed in [4] and based on [29]. We encode the information of each sub-graph G_{bidi_i} by considering only one type of bidirectional connection across the one-hop-neighbourhood of a node ν. Thus, the one-hop-neighbourhood of two *foaf:Person* nodes can be considered isomorphic in terms of that type of bidirectional connections if the generated codes are identical in structure. To constructed this encoding, we need to first find the *minimum* depth-first search (DFS) tree of each component, before we concatenate it in a list where all these minimum trees are stored [5].

  One of our key contributions in terms of implementation efficiency is the design of an algorithm that simplifies the search of the minimum DFS tree by dynamically discarding candidate paths. In the worst case scenario, which occurs if all the DFS trees in the subgraph fulfil the criteria, one of the paths is taken randomly, and the encoding algorithm then becomes $O(n!)$ – which is the same complexity as the original algorithm proposed by [4]. However, this particular case only occurs when all subgraphs are complete, and they then all have the same encoding; we can then reduce the process to return only one DFS. Figure 2 shows and example of an encoding of the bidirectional connections of the one-hop-neighbourhood of a given node ν. The first part of the list representation of each of the edges (first two numerical elements of the list) is the relevant one for isomorphic tests. In fact, after anonymising the graph, each node in the same neighbourhood as ν should have the same structure as ν itself (i.e., identical numerical representation within the encoded dictionary).

Following the terminology of [4], we refer to the dictionary encoding of the one-hop-neighbourhood of a node ν as the *Full Neighbourhood Code* of ν ($FNHC_v$).

B. Dissimilarity Computation Algorithm

To obtain the dissimilarity score between each of the nodes, we use the information stored in the Full Neighbourhood Code of each node (e.g. a *foaf:Person*). This measure will be used later on to build the neighbourhoods of size k. The dissimilarity between the one-hop-neighbourhood of two nodes x and y is computed as the weighted sum of the dissimilarity of each connection in that neighbourhood:

$$sim(FNHC_x, FNHC_y) = \sum_{i=0}^{N} \alpha_i * sim_i(FNHC_{x_i}, FNHC_{y_i}) \quad (1)$$

where N is the set of connection types present in the one-hop-neighbourhood, α_i is the weight of the dissimilarity of attribute i (sim_i) to the total dissimilarity between the nodes x and y. For each of the three edge types described above, we utilise a different type of dissimilarity function:

- The **dissimilarity of attribute connections** is computed as the normalised distance of two attributes x_i and y_i given a defined hierarchy tree. The dissimilarity ranges between 0 (identical) and 1 (reached highest level of hierarchy).

[5]The specific rules on what *minimum* means and how to concatenate the codes can be found in [4] and [29]

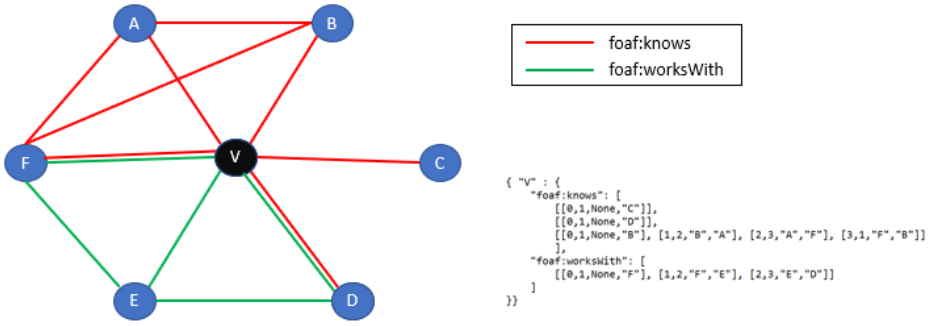


Fig. 2: Encoding Bidirectional Connections of a sample *foaf:Person* node ν

- The **dissimilarity of unidirectional connections** between two nodes x and y is based on the number of changes required for them to become equal. Given a set of Literals to which each connection points, the dissimilarity is thus defined as the number of connections of that type that have to be deleted, so that two nodes x and y are connected to exactly the same Literals or resources.
- The **dissimilarity of bidirectional connections** between two nodes x and y, given the one-hop-neighbourhood, is determined by the amount of edges one needs to delete so that one-hop-neighbourhoods of both nodes are identical (i.e. they become isomorphic).

By computing the similarity of each of the connections using the corresponding functions described above and applying the weighted sum provided in Equation (1), one can compute the complete dissimilarity between two nodes x and y.

C. The Graph Modification Algorithm

This is the third and final step of our method, which is heavily based on the ideas presented by [4], with several modifications to improve the effectiveness and efficiency. The main goal of this step is to transform the one-hop-neighbourhood of a group of k given nodes, so that the anonymisation criteria is fulfilled for all of them. In other words, these are modified to become identical one-hop-neighbourhoods as described in Definition 3. We call such a group of nodes *anonymised neighbourhoods* or *equivalence classes* (following the terminology of k-anonymity).

In the following, we describe how to transform each type of connection.

- To **generalise attribute connections**, the attributes of each of the k nodes are generalised to the value common to them on the lowest level in the hierarchy tree provided.
- For **generalising unidirectional connections**, one should remove edges so that each of the k nodes are connected exactly to the same Literals and resources (via those unidirectional connections). That is, the same edges as when calculating the dissimilarity between each of the unidirectional connections should be deleted. The motivation of only deleting edges, instead of adding new edges (or a combination of both approaches) is to not introduce false information (added edges) in the graph. As pointed out by [4], moreover, the approach of deleting edges complies with an open world assumption, which suggests that missing statements can also be true.
- The same reasoning will apply to the **generalisation of bidirectional connections**, which is the most complex aspect of this step. It relies on the same type of calculations used when computing the dissimilarity for this type of connections, as was the case for unidirectional connections. Therefore, for each node in the k-sized neighbourhood, one should delete all the necessary edges so that the one-hop-neighbourhood of each of them is isomorphic in terms of each of the bidirectional connections. In the case of dissimilarity, in order to calculate which edges to be deleted, we make use of the encoded full neighbourhood codes of each of the bidirectional connections. In order to decide which edges to delete, we make several pairwise comparisons between the one-hop-neighbourhoods of the nodes in the neighbourhood and update their codes accordingly at every step. In our method, it is sufficient to take one of the nodes as reference, and then perform this pairwise comparisons to every other node twice (i.e. a double-pass). At every comparison, the one-hop-neighbourhood of the reference node and the second node under comparison are updated via edge deletion, so that they are isomorphic. This way, after the first pass, the reference one-hop-neighbourhood takes the minimum isomorphic representation. In the second pass, this structure is then acquired by all the other nodes. This approach is a major improvement with respect to the method shown in [4], since they perform this comparisons $k!$ times. Figure 3 illustrates the intuition behind this idea.

We would like to point out two of the key challenges that arise when anonymising bidirectional connections.

1) First, when deleting edges during the described double pass, the edges of other one-hop-neighbourhoods that are also in the neighbourhood may be affected as well. This leads to more edges being deleted than necessary, and thus additional information loss and a lower utility of the resulting graph. To circumvent this issue, during

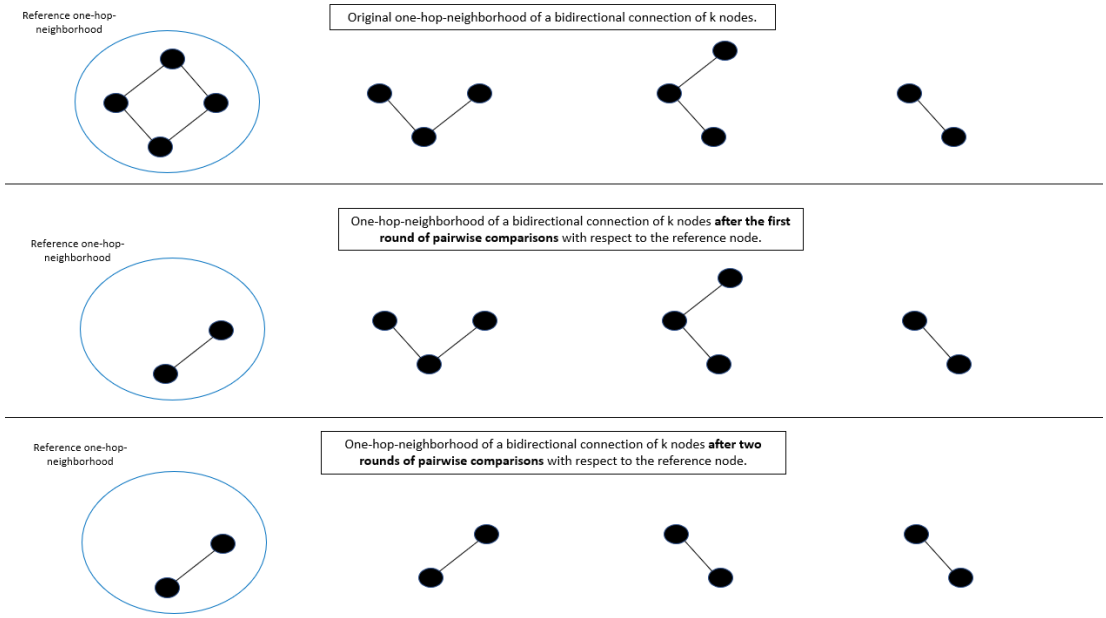


Fig. 3: Illustration of our *double pass* method in order to generalise the structure of a bidirectional connection of in a neighbourhood with $k = 4$ nodes

the double pass, we actually only *store* which edges to delete. They are then only actually deleted when the algorithm has finished, which reduces the number of deleted edges. Edge deletion may still cause some additional edges to be deleted in the neighbourhood, and therefore, these neighbourhoods might not be isomorphic anymore. However, since the calculation of which edges to delete ensures that they are actually isomorphic in first place, deleting additional edges of the structure of each of the one-hop-neighbourhoods does not reveal any additional information. Thus, we can still consider them isomorphic in terms of the anonymisation goal.

2) In the same way, deleting edges may affect the one-hop-neighbourhood of other nodes that are not in the same neighbourhood as the k target nodes being anonymised. There are two different scenarios for this:
   a) The one-hop-neighbourhood of *non-anonymised* nodes is affected. In this case, one needs to simply update the one-hop-neighbourhood of those nodes, but leave everything else as is.
   b) The one-hop-neighbourhood of *anonymised* nodes is affected. We can observer that this is the exact same situation as in a), and we can simply remove those edges and note that those one-hop-neighbourhoods are still anonymised – even if they are not exactly isomorphic anymore.

As we will show in Section IV, due to these efficiency improvements, our method is able to deal with larger and more complex graphs than the earlier approach. Note that the special cases described above were not considered by [4], and in turn, their method leads to larger information loss.

IV. Evaluation

TABLE I: Configuration to test overall scalability and runtime of our algorithm

Number of graphs	81
Number of different graph sizes	23
k	3
Average proportion of bidirectional connections per node	3

In this section, we focus on the evaluation of runtime performance of our proposed algorithm. Showing correctness of the anonymised graph, i.e. that it fulfils a certain k, is actually a rather trivial problem, and can be verified in polynomial time. However, as finding an optimal solution is generally considered an NP-hard problem (cf. e.g. [30] for k-anonymity of tabular data), runtime and scalability of heuristics to solve the anonymisation problem are important practical aspect to evaluate.

For evaluation purposes, we generated a dataset in the same manner as described by [4], that is, we fix certain properties of the graph, for example the overall number of nodes, and the average number of edges per node. The graph is then randomly generated to match this configuration [6].

We will start by analysing the overall **scalability and efficiency of the algorithm**. For this purpose, we run multiple anonymisation rounds in several graphs with increasing sizes (in terms of nodes), but the same characteristics as in [4],

[6]The generated graph data will be publicly available on Zenodo after the blind review process

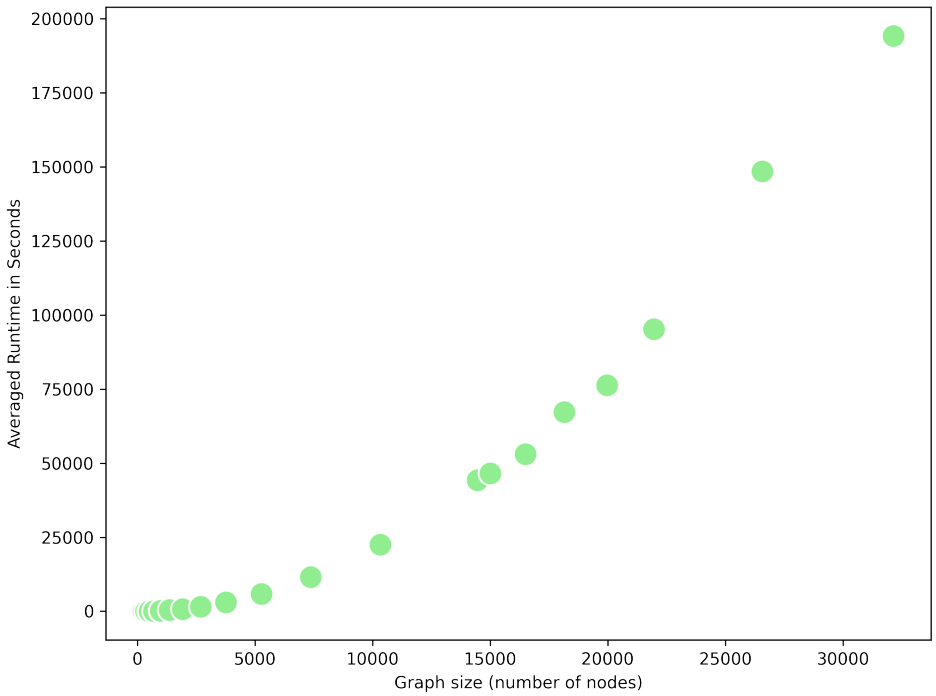


Fig. 4: Average runtime for different graph sizes, with $k = 3$ and on average three bidirectional connections per node

that is, on average three bidirectional connections per node, and a target value $k = 3$. Table I shows the parameters considered for this evaluation. Figure 4 depicts the runtime results averaged by the number of nodes (of type *foaf:Person* in this setting).

As expected, the runtime increases with the size of the graphs, since more neighbourhood structures need to be encoded, more similarity computations take place and more generalisations of nodes are performed, regardless of k and *the average of bidirectional connections* . We also observe a **significant improvement with respect to the limit described by [4]**. While they mention a limit for their algorithm for graphs with 256 nodes, 3 bidirectional connections per node on average and k equal to 3, we are able to successfully anonymise graphs with up to 32,151 nodes. However, this is not a hard upper limit for our solution, since due to our improvements on the neighbourhood encodings, we do not encounter memory issues as in [4]. The graph with 32,151 nodes on our hardware took around two days to anonymise, on a server-grade CPU from 2007[7] used in single thread mode. With more recent CPUs, we are confident that larger graphs could be anonymised with our algorithm in reasonable time.

In order to get a better insight on the influence of the properties of the graph and the anonymisation on the runtime, we extended our search over hyper-parameters: we varied the average number of connections, and k, for two graph sizes, namely with 256 and 512 nodes.The results for graphs of size 256 are shown in Table II – the results for graph size of 512 are almost identical, and thus omitted. We can observe two main patterns:

- For a fixed k, **the larger the average of bidirectional nodes per person, the longer the average runtime of our anonymisation algorithm**. The reason for this is the influence of the density of links for the anonymisation of the structure.
- In general, for a fixed average number of bidirectional connections per person, **the larger the k, the smaller the runtime.** This means that it is faster to compute few generalisations with large neighbourhoods (large k) than many generalisations with small neighbourhoods (small k).

Overall, the influence of the average bidirectional connections per person plays a more important role than the parameter k in terms of runtime, as we can also see from Figure 5, where the influence of the number of connections shows an exponential growth.

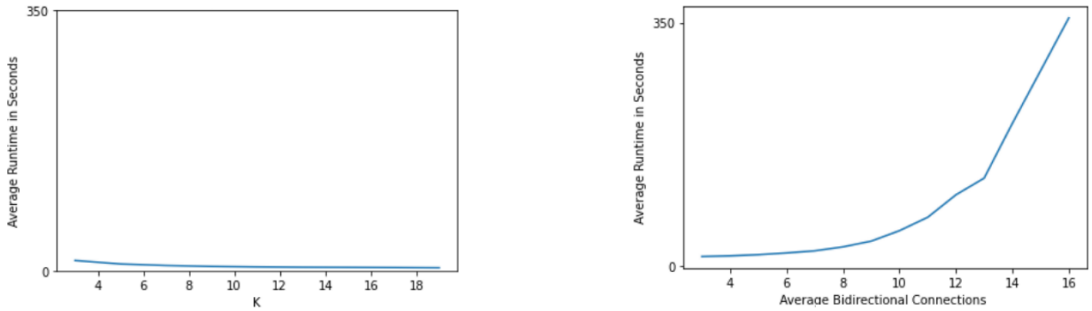
Since runtime may become an issue for very large graphs (as shown in Figure 4, a graph with 32,151 nodes takes around 2 days to be anonymised), we explored the possibility of paralellising parts of our algorithm. However, we empirically discovered that generalisation takes on average 85% of the total runtime, but this step is not parallelisable due to the influence that edge deletion has on neighbourhoods.

Finally, we also explored the influence of different settings on edge deletion during the generalisation phase. In general, bigger graphs have a higher likelihood of common structures within and hence, result in a smaller number of deleted bidirectional connections. A smaller k (Figure 6), as well as

[7]Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz

TABLE II: Average runtime in seconds for different combinations of k and average number of bidirectional connections in graphs of size 256 nodes. N.b.: seconds are rounded to integers for clarity

		K								
		3	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**
Average Bidirectional Connections per Person	**3**	14	9	7	6	5	5	4	4	4
	4	15	10	8	7	6	6	6	6	5
	5	16	12	10	9	9	8	8	7	7
	6	19	15	13	12	12	11	11	10	10
	7	22	19	18	16	16	15	15	14	14
	8	28	26	25	24	22	22	21	21	20
	9	36	36	36	34	34	32	31	30	29
	10	51	51	50	50	47	46	44	43	42
	11	70	74	74	72	69	66	65	62	58
	12	102	110	108	104	99	96	91	88	82
	13	126	132	129	125	120	114	107	100	96
	14	205	201	194	185	176	168	158	148	138
	15	357	352	322	308	291	270	258	238	219



(a) (b)

Fig. 5: Average runtime in seconds for graphs of size 256 with (a) different k and on average three bidirectional connections per node, and (b) different average connections per node, for a fixed $k = 3$

a higher density of bidirectional connections, also lead to a smaller number of deleted edges (Figure 7).

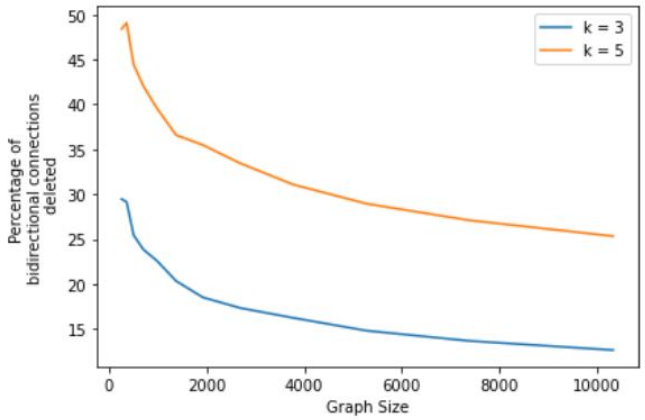


Fig. 6: Deletion of bidirectional connections for different graph sizes and k=3 (blue) and k=5 (orange)

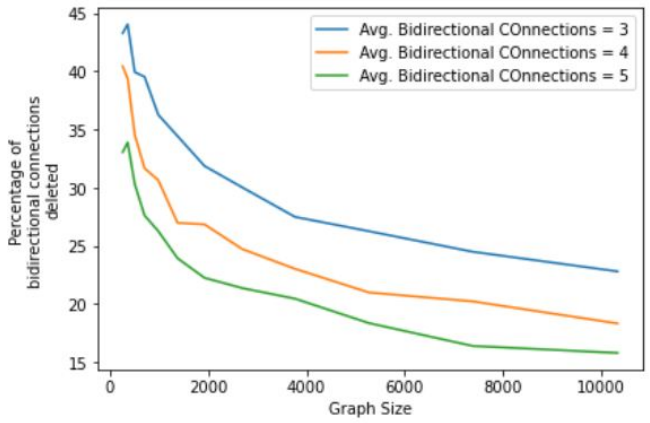


Fig. 7: Deletion of bidirectional connections for different graph sizes and average bidirectional connections equal to 3 (blue), 4 (orange) and 5 (green)

V. Conclusions

Anonymisation of graph data differs from relational data – besides the content of nodes (and potentially also the edges, e.g. labels attached to them), also the *structure* of graphs can become information that an attacker can utilise to perform. For example, in a re-identification attack, knowing how many connections of a certain type a certain person has to others, might allow to uniquely identify them.

In this paper, we have thus presented an improved algorithm for anonymising the structure of graphs. We extended previous work by (i) allowing heterogeneous graph structures with multiple types of nodes and edges, and (ii) providing an

efficient implementation with several modifications to the original algorithm, to scale up the size of problems that can be tackled. As we demonstrated on graphs that we generated analogous to the approaches in previous works, we are able to process graphs that are at least two orders of magnitude larger than earlier work.

Future work will focus on evaluating our approach in more diverse settings, including benchmark datasets, and measure the effects of the anonymisation on the utility.

REFERENCES

[1] P. Samarati, "Protecting respondents identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[2] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, vol. 4052. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[3] T. Dalenius, "Finding a needle in a haystack or identifying anonymous census records," *Journal of official statistics*, vol. 2, no. 3, 1986.

[4] B. Heitmann, F. Hermsen, and Decker, Stefan, "k - RDF-Neighbourhood Anonymity: Combining Structural and Attribute-based Anonymisation for Linked Data," in *Proceedings of the 5th Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn) co-located with 16th International Semantic Web Conference (ISWC)*, 2017. [Online]. Available: http://ceur-ws.org/Vol-1951/PrivOn2017_paper_3.pdf

[5] G. Alamán Requena, R. Mayer, and A. Ekelhart, "Anonymisation of Heterogeneous Graphs with Multiple Edge Types," in *Database and Expert Systems Applications*, vol. 13426. Cham: Springer International Publishing, 2022, pp. 130–135.

[6] A. Singhal. (2012) Introducing the knowledge graph: things, not strings. 2020-11-13. [Online]. Available: https://www.blog.google/products/search/introducing-knowledge-graph-things-not/

[7] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Computing Surveys*, vol. 54, no. 4, July 2021.

[8] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor, "Industry-Scale Knowledge Graphs: Lessons and Challenges," *Communications of the ACM*, vol. 62, no. 8, p. 36–43, July 2019.

[9] G. Hübscher, V. Geist, D. Auer, A. Ekelhart, R. Mayer, S. Nadschläger, and J. Küng, "Graph-based managing and mining of processes and data in the domain of intellectual property," *Information Systems*, May 2022.

[10] S. Ji, P. Mittal, and R. Beyah, "Graph Data Anonymization, De-Anonymization Attacks, and De-Anonymizability Quantification: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1305–1326, 2017.

[11] X. Ying and X. Wu, "Randomizing Social Networks: a Spectrum Preserving Approach," in *Proceedings of the SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2008, pp. 739–750.

[12] ——, "On link privacy in randomizing social networks," *Knowledge and Information Systems*, vol. 28, no. 3, pp. 645–663, 2011.

[13] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[14] T. Feder, S. U. Nabar, and E. Terzi, "Anonymizing Graphs," *arXiv:0810.5578 [cs]*, 2008. [Online]. Available: http://arxiv.org/abs/0810.5578

[15] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proceedings of the SIGMOD International Conference on Management of data (SIGMOD)*. Vancouver, Canada: ACM Press, 2008, p. 93.

[16] J. Casas-Roma, J. Salas, F. D. Malliaros, and M. Vazirgiannis, "k-Degree anonymity on directed networks," *Knowledge and Information Systems*, vol. 61, no. 3, pp. 1743–1768, 2019.

[17] L. Zou, L. Chen, and M. T. Özsu, "k-automorphism: a general framework for privacy preserving network publication," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 946–957, 2009.

[18] J. Cheng, A. W.-c. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. Indianapolis Indiana USA: ACM, 2010, pp. 459–470.

[19] E. Zheleva and L. Getoor, "Preserving the Privacy of Sensitive Relationships in Graph Data," in *Privacy, Security, and Trust in KDD*. Berlin, Heidelberg: Springer, 2008, pp. 153–171.

[20] A. Campan and T. M. Truta, "Data and structural k-anonymity in social networks," in *International Workshop on Privacy, Security, and Trust in KDD*, 2008, pp. 33–54.

[21] T. Tassa and D. J. Cohen, "Anonymization of Centralized and Distributed Social Networks by Sequential Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 2, pp. 311–324, 2013.

[22] D. Mohapatra and M. R. Patra, "Anonymization of attributed social graph using anatomy based clustering," *Multimedia Tools and Applications*, vol. 78, no. 18, pp. 25 455–25 486, 2019.

[23] R. Chen, B. C. M. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *The VLDB Journal*, vol. 23, no. 4, pp. 653–676, 2014.

[24] Y. Wang and X. Wu, "Preserving Differential Privacy in Degree-Correlation Based Graph Generation," *Trans. Data Privacy*, vol. 6, no. 2, pp. 127–145, 2013, place: Bellaterra, Catalonia, ESP Publisher: IIIA-CSIC.

[25] Y. Mülle, C. Clifton, and K. Böhm, "Privacy-Integrated Graph Clustering Through Differential Privacy," in *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference, Brussels, Belgium*, 2015, pp. 247–254. [Online]. Available: http://ceur-ws.org/Vol-1330/paper-39.pdf

[26] F. Bonchi, A. Gionis, and T. Tassa, "Identity obfuscation in graphs through the information theoretic lens," in *27th International Conference on Data Engineering*. Hannover, Germany: IEEE, 2011, pp. 924–935.

[27] P. Mittal, C. Papamanthou, and D. X. Song, "Preserving Link Privacy in Social Network Based Systems," in *20th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2013.

[28] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous Graph Transformer," in *Proceedings of The Web Conference 2020*. Taipei Taiwan: ACM, Apr. 2020, pp. 2704–2710.

[29] B. Zhou and J. Pei, "Preserving Privacy in Social Networks Against Neighborhood Attacks," in *24th International Conference on Data Engineering*. Cancun, Mexico: IEEE, 2008, pp. 506–515.

[30] A. Meyerson and R. Williams, "On the complexity of optimal K-anonymity," in *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*. Paris, France: ACM Press, 2004, p. 223.

View publication stats