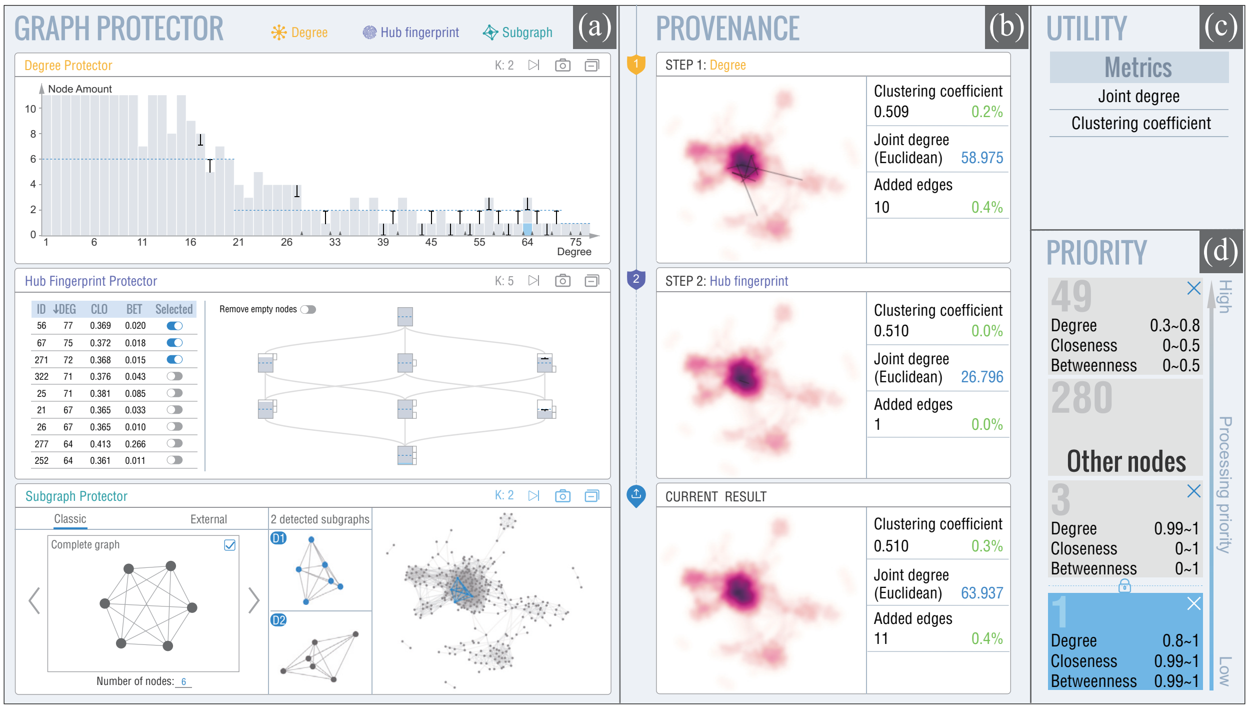
GraphProtector: A Visual Interface for Employing and Assessing Multiple Privacy Preserving Graph Algorithms

Xumeng Wang, Wei Chen, Jia-Kai Chou, Chris Bryan, Huihua Guan, Wenlong Chen, Rusheng Pan and Kwan-Liu Ma



ID	↓DEG	CLO	BET	Selected
56	77	0.369	0.020	✓
67	75	0.372	0.018	✓
271	72	0.368	0.015	✓
322	71	0.376	0.043	
25	71	0.381	0.085	
21	67	0.365	0.033	
26	67	0.365	0.010	
277	64	0.413	0.266	
252	64	0.361	0.011	

STEP 1: Degree
Clustering coefficient 0.509 0.2%
Joint degree (Euclidean) 58.975
Added edges 10 0.4%

STEP 2: Hub fingerprint
Clustering coefficient 0.510 0.0%
Joint degree (Euclidean) 26.796
Added edges 1 0.0%

CURRENT RESULT
Clustering coefficient 0.510 0.3%
Joint degree (Euclidean) 63.937
Added edges 11 0.4%

49
Degree 0.3~0.8
Closeness 0~0.5
Betweenness 0~0.5

280
Other nodes

3
Degree 0.99~1
Closeness 0~1
Betweenness 0~1

1
Degree 0.8~1
Closeness 0.99~1
Betweenness 0.99~1

Fig. 1. The visual privacy preservation stage of *GraphProtector*. (a) The graph protector view integrates multiple privacy-preserving schemes. (b) The provenance view illustrates the effect caused by each process. (c) The utility view lists user-selected utility metrics. (d) The priority view depicts the processing priority for nodes/identities specified by users.

Abstract— Analyzing social networks reveals the relationships between individuals and groups in the data. However, such analysis can also lead to privacy exposure (whether intentionally or inadvertently): leaking the real-world identity of ostensibly anonymous individuals. Most sanitization strategies modify the graph's structure based on hypothesized tactics that an adversary would employ. While combining multiple anonymization schemes provides a more comprehensive privacy protection, deciding the appropriate set of techniques—along with evaluating how applying the strategies will affect the utility of the anonymized results—remains a significant challenge. To address this problem, we introduce *GraphProtector*, a visual interface that guides a user through a privacy preservation pipeline. *GraphProtector* enables multiple privacy protection schemes which can be simultaneously combined together as a hybrid approach. To demonstrate the effectiveness of *GraphProtector*, we report several case studies and feedback collected from interviews with expert users in various scenarios.

Index Terms—Graph privacy; *k–anonymity*; structural features; privacy preservation

1 Introduction

- *Xumeng Wang, Wei Chen, Wenlong Chen and Rusheng Pan are with Zhejiang University. E-mail: wangxumeng@zju.edu.cn; chenwei@cad.zju.edu.cn; {chenwenlong, panrusheng}@zju.edu.cn. Wei Chen is corresponding author.*
- *Jia-Kai Chou, Chris Bryan and Kwan-Liu Ma are with University of California, Davis. E-mail: {jkchou, cjbryan}@ucdavis.edu, ma@cs.ucdavis.edu.*
- *Huihua Guan is with Zhejiang University and Alibaba Group. E-mail: higtonic@gmail.com.*

Manuscript received 31 Mar. 2018; accepted 1 Aug. 2018. Date of publication 16 Aug. 2018; date of current version 21 Oct. 2018. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2018.2865021

In today's world, intra-personal information is regularly collected and modeled using social networks. Analyzing the structures of these networks provides insights in the relationships of individuals within populations. Among the social sciences, network research serves many purposes, such as detecting communities [52], identifying lead influencers [40], and examining the spread of information [35]. In corporate environments, network analysis provides support for business-critical operations [36], including targeted marketing [7, 20], crime and fraud detection [7, 16], and behavioral analysis of customers [26, 51].

Datasets built using individuals often contain *sensitive* information. By analyzing the network, the specific identities and attributes of ostensibly anonymous individuals can be (either intentionally or inadvertently) compromised [50]. We refer to this as privacy *exposure* or *leaking*. It is important to perform proper privacy preserving operations before the data can be shared or released.

Privacy preservation schemes are anonymization approaches that counteract hypothesized adversarial attack models. A common approach for network sanitization is to extend the well-known *k–anonymity* model [45]: assuming a closed-world, in which no external dataset/information is considered known by the attackers, privacy is exposed when a localized structure, property, or group of individuals can be uniquely identified in a given network dataset.

Most anonymization models, including *k–anonymity*, focus on specific subsets of privacy specification. As a mechanism for sanitizing a dataset, this means they are only able to handle the corresponding type of attack(s). Considering network privacy in a more comprehensive manner by simultaneously combining multiple anonymization models is a trending research direction [27]. A significant challenge in incorporating multiple models is comparing the effectiveness of different sanitization schemes. There are two reasons such comparison is helpful: (1) It provides understanding for how privacy preservation is being accomplished as an event-driven pipeline of anonymization actions. (2) It makes visible the change in utility in transitioning from a raw, leaking dataset to one that is processed and sanitized. Privacy requirements and utility measurements are highly dependent on dataset context and user semantics. Thus, providing flexibility and transparency in the privacy preservation process is a crucial but non-trivial task.

To help accommodate this challenge, we present *GraphProtector*, a novel system for preserving privacy in network datasets via the interactive synthesis and application of multiple anonymization models. Figure 2 shows the workflow for *GraphProtector*, which employs several linked views with data-driven, backend heuristics in a multi-stage, user-in-the-loop, sanitization pipeline.

GraphProtector allows a user to load a network dataset (in this paper, we focus on simple graphs) and then employ one or more privacy preserving mechanisms (which we refer to as *protectors*) to conduct targeted dataset anonymization. Protectors act based on a customizable set of *identity priorities* and *utility metrics*. That is, when a protector finds privacy leaks in the dataset, its removal scheme is based on a user-defined set of specifications. Protector recommendations can be customized and compared against each other, allowing for flexible implementation of dataset sanitization. A provenance component to *GraphProtector* provides a historical view of dataset change over the course of applying multiple protectors, allowing a user to see exactly how the dataset's utility and privacy exposure is changing.

GraphProtector is designed around a set of task requirements from discussion with domain researchers. We present two case studies of real-world datasets that demonstrate how the system can be used. We additionally interview several domain experts in privacy and related research areas and discuss collected feedback. Their comments shed light on how tools that accommodate visualization-based, user-in-the-loop workflows like *GraphProtector* can be integrated into their normal analysis routines. As an initial attempt at visually establishing complex, custom privacy protections for graph-based datasets. Our results open up opportunities for expanded research in the domain, including open-world considerations, more complex anonymization models, and multiple privacy preserving operations.

2 Related Work

Privacy Preservation for Graphs. Node-link diagrams, also called graphs, are visual representations of network datasets. For an overview of general graph visualization, see [34, 44, 47].

With the boosting of data collection techniques, researchers now extend classical models for privacy preservation to graph data [56]. Based on differential privacy models, some studies chose to make appropriate perturbations to the graph structure [41, 49, 53]. For instance, Xiao et al. [53] abstracted a dendrogram from a simple graph according to hierarchical random graph (HRG) model. In reconstructing the original graph based on the dendrogram, they applied differential privacy methods to add noise so that the privacy issues could be removed effectively.

Differential privacy approaches are vulnerable to many variants of de-anonymization attacks, while models based on *k–anonymity* may conditionally resist them [27]. *K–anonymity* [45] anonymizes individuals by making at least k individuals indistinguishable, where k is an adjustable, user-defined parameter: the higher the k, the higher the enabled privacy preservation. The original model, designed for tabular data, achieves anonymization by obscuring the attribute values. Later studies regard structural features as a kind of attribute, and have extended this model to deal with privacy issues in graph data, e.g. k-degree [33], k-automorphism [57], and k-isomorphism [11]. These models anonymize structural features in the graph by constructing equivalent features. Attackers are thus unable to identify which one is the target. Results generated by these models have an adequate ability to resist partial attacks, but not all kinds of attacks.

In addition to differential and anonymity approaches, other models anonymize based on features of the graph itself. Ying and Wu [55] edit the graph by randomly adding, deleting, and switching edges. Thompson et al. [46] present two algorithms, bounded t-means clustering and union-split, that leverage clustering to achieve privacy preservation by graph clustering methods.

Privacy-Awareness for Other Visualization Techniques. For general visualization research that integrates privacy awareness (both sanitization and detection), common approaches fall into two categories: (1) preserving privacy in the visualization itself (similar to the aforementioned graph-based work), and (2) using visualization to facilitate a privacy preservation pipeline.

For the first approach, privacy sanitization is applied directly to the visualization, creating an updated chart that is sanitized. As an example, Dasgupta and Kosara [15] extend parallel coordinates by employing a screen-space sanitization technique. Only an approximate interval of data values for polylines can be recognized, thus meeting a *k–anonymity* threshold. Likewise for summary histograms, Archambault et al. [3] aggregate components. Considering user diversity, Oksanen et al. [39] visualize trajectory data by privacy preserving heatmaps.

In other instances, privacy preservation must be accomplished in a more complex or analytic-specific manner. Thus, interactive and iterative processing is necessary to conduct an action-based pipeline for updating the dataset and visualization. Visualization exists within a system both to analyze and review the state of privacy preservation. This approach has recently been used by Chou et al. to handle event sequence datasets and [13] and ontological social network data [12]. The systems provide users with suggested obfuscations for sanitizing data based on syntactic anonymity models. Similarly, work by Wang et al. [48] allows a data owner to share sanitized data directly based on a desired level of privacy preservation.

Like these latter systems, *GraphProtector* embeds visualization as a way to provide exploration, understanding, and provenance into a custom sanitization pipeline.

Evaluating Privacy Preservation. No matter how sanitization is accomplished, its effectiveness can be evaluated mainly by measuring the resultant amount of privacy preservation and the change in visualization utility.

To verify the effect of privacy preservation, de-anonymization algorithms can be used to simulate attacks. For example, in *k–anonymity* approaches, the parameter k reflects the minimum amount of privacy preservation directly. For simple graphs (which we focus on in this paper), attacks usually identify sub-graphs and other structures within the network by querying for specific features [24, 28, 30, 37, 38]. The proportion of individual nodes that can be de-anonymized measures the amount of privacy preservation.

Unfortunately, privacy protection always comes at the cost of utility loss. For graphs, there are a variety of widely used ways to measure utility, including node degree, centrality, graph diameter, and average path length. These features can be represented in different forms: a value generalizes the status of the entire graph [42], a vector illustrates the specialty of each node [6, 8, 18, 19] and a matrix reflects the relationships between each pair of nodes [23]. For more specialized scenarios, utility metrics can be directed for specific tasks, like community detection [54] and role extraction [25].

3 Task Requirements

We focus on *simple graphs* because they lay the foundation for complex graphs. Also called strict graph, simple graphs are unweighted and undirected graphs that do no contain graph loops or multiple edges [21]. Attacks against simple graphs may come from the attribute values of the individuals represented by nodes or the relation links indicated by the graph structure. Actually, the relationships in some graph data may involves more information, which lead to weighted, directed or multiple edges. We will leave this as future work.

Our targeted users are those who work with sensitive network datasets, require flexible and personalized privacy preservation, and are knowledgeable about anonymization approaches for graphs. This means supporting customized solutions that provide sufficient protection while minimizing loss of overall dataset utility. Following the definition of Ji et al. [27], we define "utility" as the similarity of structural properties.

To drive our design and help define goals, we initially consulted over a series of meetings with a domain researcher who regularly works with network datasets and studies privacy preserving algorithms for graph data. Based on this, we define a set of four task requirements (TR) for building a visual privacy preserving system:

TR1: Learn about the characteristics of the graph. Making sense of a graph that contains hundreds or thousands of nodes and edges is challenging, even when leveraging advanced visualization techniques like optimized layouts and interaction. Augmenting the base node-link diagram with additional statistical views (node distributions by property, etc.) provides a better holistic overview of the dataset and its properties.

TR2: Set priority before applying privacy preserving algorithms. Before modifying the graph, users should have the flexibility to set the priorities or constraints with respect to which nodes are "touched" first. While different sets of anonymization operations, i.e., modifying different parts of a graph, may fulfill the same privacy requirement, some of them can introduce a significant and unwanted change to the characteristic of the graph (or a specific set of nodes). Depending on a user's needs, a region of the graph that meets certain structural criteria or contains important information should be kept as intact as possible.

TR3: Evaluate and compare schemes. There are many potential graph modifications that can fix a privacy leak, depending on which algorithm(s) is used, which nodes or edges are touched, and how they are updated. Researchers may choose one of several different schemes, either using one single or multiple privacy preserving algorithms. Being able to compare different schemes allows the user to determine the most appropriate action. This also includes understanding what level of protection each scheme provides, what steps the scheme takes, and what is the utility cost of achieving the protection.

TR4: Record the provenance of graph modifications. Privacy preserving operations modify the topology of a graph, introducing uncertainty and invariably degrading the graph's resultant utility. It is important to allow users to keep track of the changes that have been made, allowing them to go back and try new schemes if the current result is deemed unsatisfactory.

4 Privacy Preservation Approach and Workflow

We outline the privacy preservation approach employed in *GraphProtector* and describe at a high-level its multi-stage, user-in-the-loop workflow (shown in Figure 2) to identify and sanitize leaks. In Section 5, we describe in detail how *GraphProtector*'s interfaces and interactions facilitate this workflow.

4.1 Structural Features that Expose Privacy

We consider three types of structural features—introduced by Hay et al. [24]—that can be queried to compromise the privacy of individuals in simple graphs: node degree, hub fingerprint, and subgraph. With the development of technology, graph privacy may face more threats. In this work (as a beginning study), we focus on the three basic issues.

Node degree. The degree (or valency) of a node is the number of edges connected to that node. Node degree distribution in networks is usually uneven and skewed, tailing from similarly weighted nodes to outliers. By submitting queries that return nodes with a specific degree, attackers may be able to identify individuals with a unique node degree value in the network.

Hub Fingerprint. A hub refers to a node that contains extreme or unique feature values. Examples of hubs include nodes with high degree or nodes that exist on many shortest paths throughout the graph (i.e., they bridge clusters in the graph or are centrally located).

In many real-world social networks, hubs are high-profile individuals like celebrities or government officials. It is challenging to mask the identity of hubs since their "fingerprints" are anomalous compared to other nodes. Attackers can leverage a node's connection to one or more hubs to expose its identity. This is known as a hub fingerprint attack.

Subgraph. A subgraph is a subset of connected nodes extracted from the network. A common attack involving subgraphs is to first embed a subgraph with a specifically designed structural connection to a target graph, then utilize that knowledge to initiate other privacy attacks [5]. If not enough instances of a "sensitive" subgraph exist in the network, it constitutes a privacy leak.

As a part of *GraphProtector*'s workflow, we design "protectors" (Figure 2(c)) for each of these privacy issues to identify leaks—see Section 5.2.1 for descriptions.

4.2 Defining a Privacy Preserving Model

When structural features in a graph are identified as exposing privacy, there are various ways to achieve anonymization. Common methods include adding/swapping nodes and edges (introducing uncertainty), merging nodes (generalization), and removing nodes and edges (suppression) [33].

In our case, we are focusing on protecting the identities of persons (nodes). To avoid exerting excessive effects on individuals by introducing "dummy" nodes, we only allow edges to be modified—specifically, edge addition. By considering only one type of operation, we reduce computational complexity and avoid potential operational conflicts. Compared to edge deletion, adding edges provides an additional benefit: the effect of deleting edges can only be uni-directional, i.e., it always removes information. In contrast, adding edges not only introduces previously non-existed information, in some cases it also removes some information. For example, adding edges to all the nodes that have a specific node degree can remove such information in the resultant graph. As a potential future work, we note that combining multiple sanitization techniques can and should certainly be explored. However, this is a non-trivial task requiring careful design and evaluation.

Two approaches can be used to provide privacy protection. Given a set of privacy leaking nodes: (1) make structural changes to some of the non-privacy leaking nodes so that at least k sets of nodes obtain the same features, or (2) make structural changes to the set of privacy leaking nodes so that their structural features are equivalent to another set of nodes in the graph. *GraphProtector* supports both approaches. Before a protection action is invoked, the system checks to guarantee that no additional privacy issue is introduced upon making structural changes.

As an example for the first approach, assume 10–anonymity is required for a set of 7 privacy leaking nodes with degree $= 9$. To ensure protection, 3 non-privacy leaking nodes must be updated to also have degree $= 9$. Since protection is achieved by adding edges, nodes with lower degree have processing priority. To minimize the total necessary modifications, non-privacy leaking nodes with degree $= 8$ are first checked, followed by nodes with degree $= 7$, and so on. This process continues until enough candidate nodes are found and edges are added between them.

The second method adds edges directly to the set of privacy leaking nodes. Using the above scenario, we progressively add edges to the 7 privacy leaking nodes with degree $= 9$ (making their degrees $= 10, 11$, or more) until the privacy requirement is met.

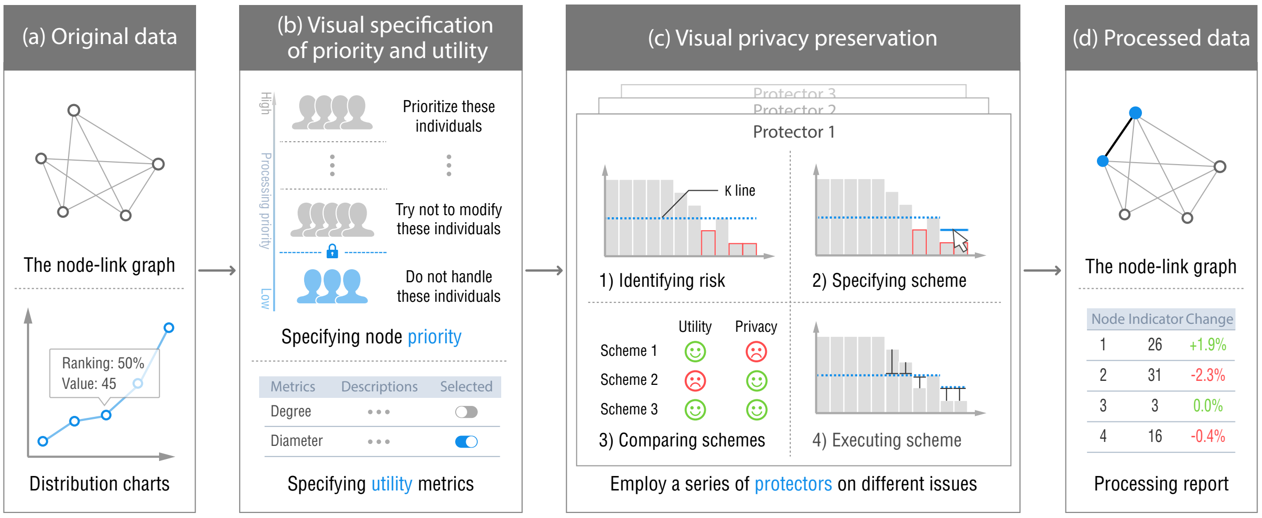


Fig. 2. The *GraphProtector* workflow: (a) A network dataset is loaded and its relationships and properties are visualized. (b) Desired *node priority* and *utility* metrics are specified to direct the subsequent anonymization actions. (c) One or more *protectors* are employed which modify the dataset's topology to preserve privacy. (d) The updated, processed dataset with an accompanying processing report.

Since *GraphProtector* supports both schemes, to choose an approach the method that requires the least number of edge additions is selected. The pseudocode of this process is provided in Appendix I.

4.3 Workflow

At a high-level, *GraphProtector*'s workflow links together backend algorithms for detecting, evaluating, and sanitizing privacy leaks in structural features via several linked, interactive views. Figure 2 shows the major steps and interactions in the workflow. We also denote the specific tasks from Section 3 that each step in the workflow accommodates.

Original Data. First, a network dataset is loaded into *GraphProtector*. The node-link diagram is shown at the beginning to facilitate users in selecting the metrics they need. Combined with the statistical distribution charts of selected metrics, users can quickly grasp the characteristic of the input network **(TR1)**.

Visual Specification of Priority and Utility. To set the priority of operations prior to applying privacy preserving algorithms (TR2), sets of nodes can be grouped into bins. These bins are manually ordered according to their processing *priority*, which denotes the order that they will be first considered as candidate nodes applicable for privacy preserving operations. Setting lower processing priority to important nodes reduces the possibility that they will be directly affected by topology modifications. Particularly significant nodes can even be locked to prevent from modification.

Prior to applying privacy preserving algorithms, users can set processing priorities to the nodes in the network **(TR2)**. Nodes can be grouped into bins, while nodes in the same bin are considered having the same priority. Setting a lower processing priority to a set of important nodes can reduce the possibility of those nodes being affected by the privacy preserving operations. Particularly significant nodes can even be locked to prevent from modification.

For example, a user might wish to “prioritize the 30% of nodes with the lowest degree,” or “if possible, do not touch the 1% of nodes with the highest betweenness.” When the number of feasible operations for handling a leak is more than one (which is usually the case), decisions are made according to this user-defined priority. In order to guarantee the solution, a user should not lock an excessive number of nodes, but instead define a detailed list of processing priorities.

In addition to priority, the user can specify one or more utility metrics to evaluate the effects of potential sanitization actions. “Number of added edges” is the default metric, but there are many ways to quantify topology modifications (see Section 5.1.3). Topology modifications induced by privacy preserving operations are expressed as vectors; the updated graph is compared to the prior via a high-dimensional similarity calculation.

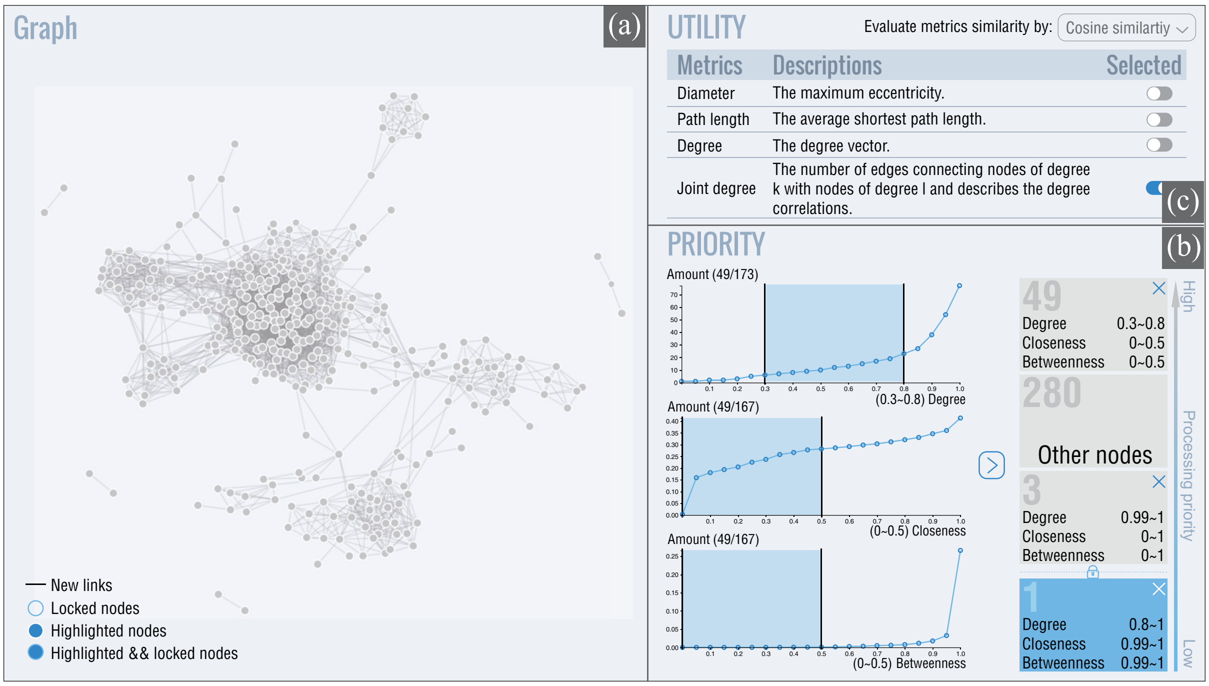
Visual Privacy Preservation. At this point, the user employs one or more protectors to sanitize the dataset. This process uses interactive decision-making based on automatic identification of leaking structural features (degree, hub fingerprint, subgraph) with application of our privacy preserving model to modify the graph's topology and fix specific leaks.

Since each protector type applies to different structural features, it disassembles the process of graph modifications into steps. A user may also employ more than one of each type of protector, for example, sanitizing multiple subgraphs, or invoke them in any desired order.

Protectors follow a standard framework, shown in Figure 2(c)). After the protector identifies the risk(s), the user customizes a scheme to guide the privacy preserving algorithm. Evaluating and comparing schemes allows for the “best” one (according to the user's semantics) to be picked (TR3). For example, an unrealistic scheme may force sequential processing to achieve the goal with a high price. If the user requires that “all the nodes have the same degree,” the graph's original degree distribution will be completely destroyed and all utility lost. Therefore, the user should select and execute a more targeted scheme that only updates the graph such that the affected structural feature no longer exposes privacy. As schemes are executed, the graph is updated to provide a provenance view of prior executed protectors (TR4).

Application of protectors follows a unified flow, shown in Figure 2(c)). First, privacy risks are identified based on the user-defined k value(s). Then, the user customizes a scheme or multiple schemes to guide the privacy preserving algorithm. Next, performing comparison and evaluation to the schemes allows for the “best” one (according to the user's semantics) to be picked **(TR3)**. For example, an unrealistic scheme may force sequential processing to achieve the goal with a high price. If the user requires that “all the nodes have the same degree,” the graph's original degree distribution will be completely destroyed. Therefore, the user should select and execute a more targeted scheme that only updates the graph such that the affected structural feature no longer exposes privacy. As schemes are executed, the graph is updated to provide a provenance view of prior executed protectors **(TR4)**.

Processed Data. Once the user is satisfied that the network is sanitized, the “final version” can be exported and reviewed (TR3). This includes the updated dataset (the node-link graph) as well as a processing report of changes.



Evaluate metrics similarity by: Cosine similartiy

Metrics	Descriptions	Selected
Diameter	The maximum eccentricity.	○
Path length	The average shortest path length.	○
Degree	The degree vector.	○
Joint degree	The number of edges connecting nodes of degree k with nodes of degree l and describes the degree correlations.	●

Fig. 3. The interface for visual specification of priority and utility is composed of three views: (a) the node-link view, rendering the overview of the graph, (b) the utility view and (c) the priority view for specifying node priority.

5 System Design

We now describe in detail the primary views and interactions in *GraphProtector*. The system is based around two interfaces (Figures 1 and 3) and five linked views. Throughout this section, we use a Facebook friendship dataset that contains 333 nodes and 2,519 edges [32].

5.1 Interface for Specification of Priority and Utility

The graph interface contains three views: a visualization of the loaded graph and two panels for specifying priority and utility metrics.

5.1.1 Graph View

The first view visualizes the network dataset using a node-link diagram and force-directed layout, as shown in Figure 3(a) (**TR1**). Nodes and edges in the graph are highlighted based on interactions and operations made during *GraphProtector*'s workflow. In addition, users are allowed to check the clustering results [22] when comparing the processed data and original data.

5.1.2 Priority View

The priority view initializes by showing a list of derived features for nodes: degree, closeness, betweenness, eigenvector, and constraint. Selecting a set of these metrics creates a set of line charts showing the statistical distributions of nodes for each selection (**TR1**). For example, in Figure 3(b), the user has selected node degree, closeness, and betweenness centrality. Each chart displays the relationship between the metric's values (y–axis) and the percentile rankings of the nodes (x–axis).

By brushing on the line charts, the user can specify a set of nodes (based on the intersection of brushes). This set can be created as a bin for establishing node priority (**TR2**), which is placed in the vertical stack to the right of the line charts. (By default, all nodes start out in the same box and the stack only contains this one box.) As new boxes are created and added to the stack, the user can drag and drop boxes to switch their priority ordering. Boxes below the blue "lock 🔒" line are locked—nodes in the boxes will not be changed by any privacy preservation operation.

5.1.3 Utility View

The utility view, shown in Figure 3(c), lists out the utility metrics that can be employed to evaluate the change in utility after privacy preserving operations are applied to the dataset (**TR3**). The user can select one or multiple metrics from the following: joint degree, diameter, path length, clustering coefficient, and node degree.

While structural modifications to the graph (i.e., adding edges) are stored as vectors, a similarity measure is required to evaluate the change in utility (with respect to the prior state of the graph). We provide four common metrics for this: Euclidean distance, Manhattan distance, cosine similarity and Jaccard similarity.

5.2 Interface for Privacy Preservation

After a dataset is loaded and the user has specified priority and utility settings, the Protector Interface can be opened. Here, visual privacy preservation is enacted and the user can examine the series of changes to the graph's provenance. At any point, the user may also navigate back to the graph interface to view the current, updated view of the dataset, as changes are reflected in the graph view. The protector interface contains two panels: the graph protector view and the provenance view.

5.2.1 Graph Protector View

Protectors are selected, configured, and executed using the graph protector view (Figure 1(c)). To start, the user creates a desired protector by clicking the corresponding label at the top of the panel. Adding multiple protectors of the same type is possible by clicking on the same button multiple times. Though applying the protectors follows the same sequence of actions (Section 4.3), each type of the protectors is individually designed according to the structural feature(s) it sanitizes.

Degree Protector. The degree protector shows the degree distribution of nodes as a bar chart (as in Figure 1(a) and Figure 4). As degree distribution is usually skewed in graph datasets, this means that some degree bins will contain no nodes, introducing gaps in the chart. To address this, we remove gaps and add a "degree gaps" tick mark to denote jumps, as shown in Figure 4.

To set up a privacy protection scheme, with respect to node degrees in this case, the user first needs to assign the desired k value. To do that, we define two types of settings. First, a "global" k_G setting enforces the *k–anonymity* for the entire node degree distribution (unless otherwise specified). Brushing along the bars sets "local" k_L values for the nodes with selected ranges of degrees. The desired k_L values for each of these localized regions can be adjusted independently. In Figure 4, the global k_G is set to 2 and there are two local k_L values set for nodes with relatively low and high degrees (set to 6 and 1, respectively).

When interacting with the degree protector, the system highlights degree bins based on their status. Bins that do not meet *k–anonymity* (they are below their k_G or k_L line) are given a red border; locked nodes colored blue.

After specifying a scheme, the user can click on the "preview ▶|" icon to see how the graph will be updated *as if the scheme is accepted and executed*. Modifications that will happen to the graph are noted on the bar chart using T-marks: ⊤ to indicate an increase in the number of nodes to a bin and ⊥ to indicate a decrease. A more detailed example can be seen on the top of Figure 1(a).

Multiple schemes can be previewed and compared by clicking on the "camera 📷" icon. At this point, one of the schemes can be accepted and executed.

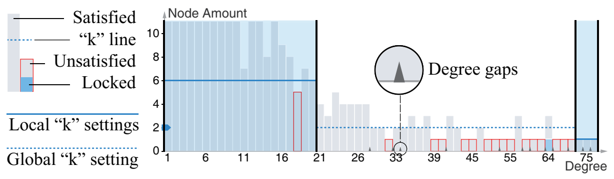


Fig. 4. The module design for the degree protector.

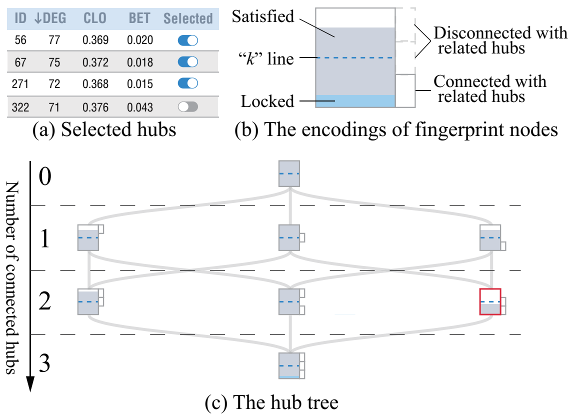
Hub Fingerprint Protector. The hub fingerprint protector is shown below the degree protector in Figure 1(a). Opening this protector initially shows a table of nodes (left side of the protector). The user must first identify a set of nodes that are to be considered as hubs. Usually, nodes with high degree or centrality are strong candidates.

Rows can be sorted by the metrics selected in the priority view, which are the metrics that the user considers as important to perform the analysis tasks. When nodes are toggled as hubs, they are pinned to the top of the list (see Figure 5(a)). After each hub selection, *GraphProtector* re-generates the fingerprints of the other nodes in the graph. Since a fingerprint for a node is determined by its relationship with hub nodes (i.e., is it connected or not?), we generate fingerprints based on whether a node is an incident to each of the hub nodes.

Based on the selection of hub nodes, we categorize incident (non-hub) nodes based on their relationships to the hubs. For Figure 5(a), since there are three selected hubs, nodes are categorized into four groups: (1) no connection to any hubs, (2) connection to one hub, (3) connection to two hubs, or (4) connection to all three hubs.

This categorization is used to construct a hub tree, which shows how likely a hub fingerprint attack could occur for the selected hubs. For Figure 5(a), its corresponding hub tree is shown in Figure 5(c)). This tree has four levels (based on the four categories), where each level corresponds to the relationship category (top level contains nodes with no relationship, etc.). At each row, squared bins are created to represent all possible combinations of selected hubs at that level. For example, at the third row (level 2) of Figure 5(c), each squared bin corresponds to a possible combination of two hub nodes out of the three selected hubs. Edges between bins in neighboring levels indicate that nodes in a fingerprint node of upper level (connected to lesser hub nodes) can be moved to a fingerprint node of lower level (connected to more hub nodes) by adding edges to them.

The encoding of a fingerprint node is shown in Figure 5(b). The encoding can be interpreted as a metaphor for filling water into a bottle. Encodings are similar to the degree protector: the dashed k line is the desired water level, indicating the privacy level to be achieved. The accumulated height of the blue bar (the number of locked nodes) and the gray bar (the number of nodes that are not locked) shows the total number of nodes that have the same corresponding fingerprint. The gray boxes on the right-hand side of the fingerprint node describe the type of fingerprint. In Figure 5(b), we demonstrate a fingerprint node that contains nodes linked to only the third hub node.



ID	↓DEG	CLO	BET	Selected
56	77	0.369	0.020	●
67	75	0.372	0.018	●
271	72	0.368	0.015	●
322	71	0.376	0.043	○

(a) Selected hubs
Satisfied
"k" line
Locked
Disconnected with related hubs
Connected with related hubs
(b) The encodings of fingerprint nodes
Number of connected hubs
0
1
2
3
(c) The hub tree

Fig. 5. (a) The three nodes with highest degree are selected as hubs. (b) For each fingerprint nodes, the feature statistic is shown on the left and the feature (connection with the hubs) is explained on the right. (c) The hub tree constructed based on the three hubs shown in (a).

In some cases, many fingerprints are empty—that is, no nodes link to the specific sets of hubs. Since it is redundant and can be misleading at the time to display those empty fingerprint nodes, we provide a mechanism to allow users to collapse those nodes to show a more compact layout. Figure 6 demonstrates an example.

Like the degree protector, the hub fingerprint protector includes functionality to preview and compare schemes. Upon deciding on a scheme, the user can execute the scheme and update the graph.

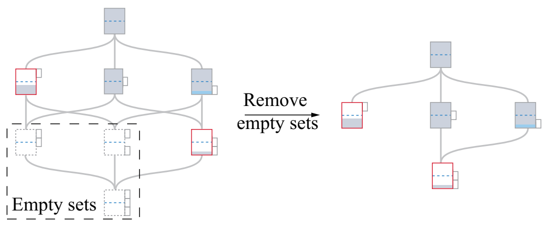


Fig. 6. The hub tree before and after removing empty sets.

Subgraph Protector. Subgraph support is fundamentally different than degree and hub fingerprint, since subgraphs are based on semantic knowledge about unique substructures within the dataset. Since the potential number of subgraphs in a network is almost combinatorial, we require the user to know what subgraph structures to sanitize.

Upon opening this protector, the user initially specifies an exemplar subgraph. This queries the graph and shows instances of matched subgraphs. Matched subgraphs can be inspected individually to check whether they contain any privacy issues. Exemplars can be derived from classic types of subgraphs by modulating their parameters [10] or loaded from prepared files.

Instead of requiring exact matches, we employ an error-tolerant [17] approach for matching subgraphs. Tolerance is a parameter between 0 and 1. For example, a subgraph of 4 nodes and 5 edges is considered matched to a complete graph with 4 nodes (and 6 edges) given a tolerance value of $1/6$, as shown in Figure 7. The method of VF2 [14] is employed to identify all subgraphs within the tolerance. Identified subgraphs within the set tolerance value are called "detected subgraphs". If the number of detected subgraphs is less than user-defined k, we then keep searching for similar subgraphs using a larger tolerance (which are called "similar subgraphs"). Similar subgraphs are the candidate subgraphs that can be used to protect the privacy of detected subgraphs.

For similar subgraphs, we use dashed edges to indicate where edges will be added to make them identical to the detected subgraphs. The user can additionally review node information and highlight their positions, as shown in the subgraph protector in Figure 1(a).

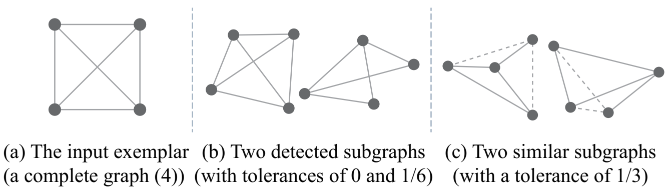

(a) The input exemplar (a complete graph (4)) (b) Two detected subgraphs (with tolerances of 0 and 1/6) (c) Two similar subgraphs (with a tolerance of 1/3)

Fig. 7. Inputting the exemplar (a) with a matching tolerance as 1/6, the subgraphs in (b) are qualified, while the subgraphs in (c) are not. The relative positions of the nodes in the subgraphs are preserved.

5.2.2 Provenance View

The provenance view provides a summary of all prior executed privacy preserving operations **(TR4)**. It consists of two parts. The first part is a density-estimated plot which provides a quick overview of the graph. The second part shows the change of utility caused by each of the privacy preserving schemes. On top of the density-estimated plot, we rendered the edges added to the graph after applying the corresponding privacy preserving scheme. We originally considered two approaches for the density-estimated plot: kernel density estimation (KDE) [43] and curve density estimates (CDE) [31]. For our context, CDE is more suitable as it displays edge distribution through a linear kernel and thus better retains the edges' finer visual appearance (see comparison in Figure 8).

As shown in Figure 1(b), the step histories in the provenance view show the changes made by prior protector operations. The graph is rendered using CDE with added edges overlaid at each step. Changes for utility metrics are shown to the right of each history view. Increases or decreases in values are highlighted in green and red, respectively, while if the metric outputs a similarity score it is colored in blue. The current state of the graph is shown at the end of the provenance view. The user can see the specific changes made by a protector in the graph view by clicking on its corresponding provenance view. If a user is dissatisfied with the current result, the most recent operation can be cancelled from the provenance view. If a user is satisfied with the state of the processed dataset, it can be exported into a documentary report, containing the sanitized data and a post-processing report of changes made.

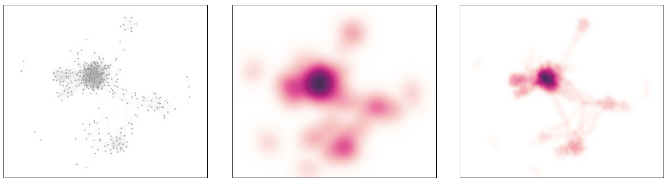

(a) The node-link view (b) KDE (c) CDE

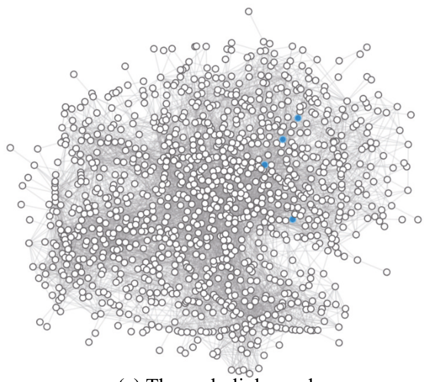
Fig. 8. Visualizing a graph with the graph view (a), KDE (b) and CDE (c).

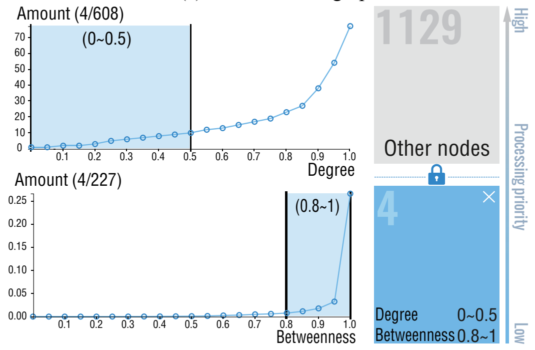
6 Case Studies

We present two case studies that demonstrate using *GraphProtector*. The first compares protector schemes against each other, while the second deals with setting processing priorities and utility preservation metrics.

6.1 Email Communication Dataset

The email communication dataset in this first case study records 5,451 email contacts among 1,133 users in a university [2], and is shown as a node-link diagram in Figure 9(a).


(a) The node-link graph



(b) The priority setting

Fig. 9. The email communication dataset. (a) In the graph view, the locked nodes are highlighted in blue. (b) In the priority view, the nodes that have "degree less than 50th percentile" and "betweenness greater than 80th percentile" are selected by brushing.

To start, node degree and betweenness are chosen for specifying priority. Figure 9(b) shows the distributions of these two metrics. By brushing the two charts, we quickly find that only 4 nodes have a low degree and a high betweenness (Figure 9)). We then add these nodes to the processing priority list and lock them to ensure they are not touched by protector operations. We also select the following metrics for utility analysis: path length, joint degree, clustering coefficient and betweenness.

After going to the interface for privacy preservation, we first create a node degree protector with two schemes. For the first scheme (S_1), to simulate a purely automatic model, we set the global k (the k_G value) to 5, as seen in Figure 10(a). For the second scheme (S_2), we set $k_G = 3$, but then brushes to assign a localized $k_L = 7$ value for nodes with lower degree (below 30), as shown in Figure 10(b).

By comparing the utility changes required to execute each scheme (shown on the bar charts), we notice that S_1 requires not only adding nearly double amount of edges, it also results in larger impact to the other utility measures. We therefore choose to perform S_2.

Next, we open a subgraph protector and check if there are privacy issues caused by the following five types of classic subgraphs: complete, circle, path, star and complete bipartite. We input several generalized exemplar subgraphs for each type and set the tolerance to 0. As a result, we find a subgraph, Figure 11(b), that matches a pre-defined bipartite graph exemplar of size $(6,3)$, as seen in Figure 11(a). To achieve 2-anonymity for this case, we continue to search for matching subgraphs with a relaxed tolerance value. When the tolerance is set to $1/9$, one additional matching subgraph, Figure 11(c), can be found. Adding two edges to the matched subgraph, as presented as the dashed lines in Figure 11(c), fixes the privacy leak.

6.2 Face-to-Face Contacts Dataset

For the second case study, we use a dataset describing face-to-face contacts during an exhibition [1]. Edges represent conversations

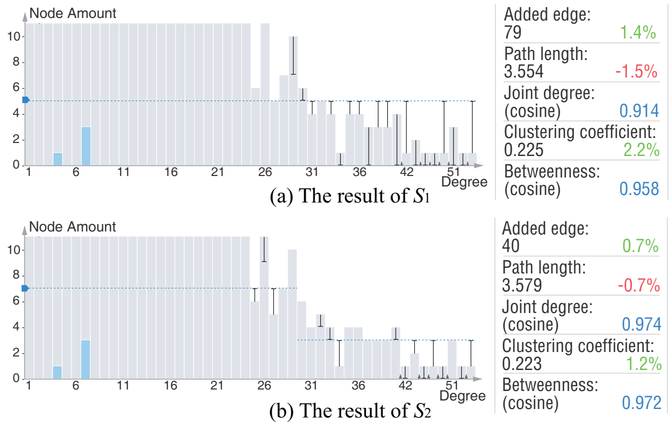


Fig. 10. Applying two schemes to the Email communication dataset. The changes in utility metrics show that S_1 yield nearly double loss than S_2.

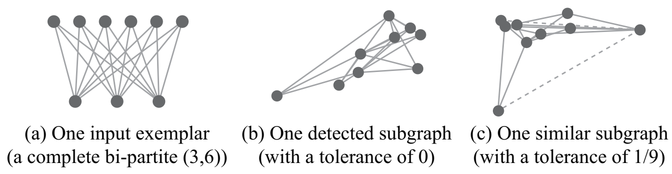

(a) One input exemplar (a complete bi-partite (3,6))
(b) One detected subgraph (with a tolerance of 0)
(c) One similar subgraph (with a tolerance of 1/9)

Fig. 11. An input subgraph and two matching subgraphs in the email communication dataset. Dashed lines represent the edges to be added to achieve 2-anonymity.

between visitors that lasted longer than 20 seconds. This social network has 410 nodes and 2,765 edges. To illustrate the importance of the processing priority, in this case, we compare the changes in utility metrics when using different priority schemes.

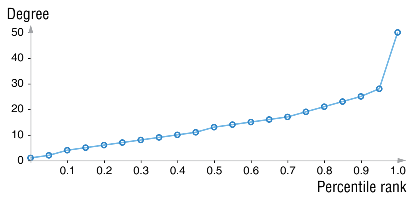


Fig. 12. The line chart of node degree over percentile ranking.

Figure 12 indicates that the degree distribution of the dataset is uniform except for nodes with the highest degree. To further study this distribution, degree is selected as the utility evaluation metric. We then create two priority schemes: (1) nodes with “degree less than the 2nd percentile” are locked (S_3), and (2) nodes with “degree more than the 98th percentile” are locked (S_4). For each priority scheme, the same privacy operations are applied:

Step 1: Using a degree protector, k_G is set as 2.

Step 2: Using a hub fingerprint protector, we choose the four nodes with the highest closeness as hub nodes and sets $k_G = 5$.

After Step 1, the results produced from the two schemes differ greatly, as shown in Figure 13(a). To perform privacy protection, S_3 needs to add 3 edges while S_4 adds 33 edges. This is because some of the high-degree nodes have a unique degree. S_3 allows *GraphProtector* to add edges to those high-degree nodes, thus nodes with very high degrees can be made to have the same degree with each other. In contrast, S_4 locks those high-degree nodes, meaning they cannot be touched. As a result, other nodes must be used to resolve privacy issues, which require the addition of more overall edges. As shown in the second CDE of Figure 13(a), a large number of edges are added to two nodes, in order to increase their degrees. This significantly affects the degree distribution of those low-degree nodes, as shown in the second bar chart of Figure 13(a).

The two schemes add similar numbers of edges in Step 2. However, results of CDEs (Figure 13(b)) show that S_4 adds 9 edges to the same hub node. This modification would significantly undermine the characteristics of the hub with high closeness.

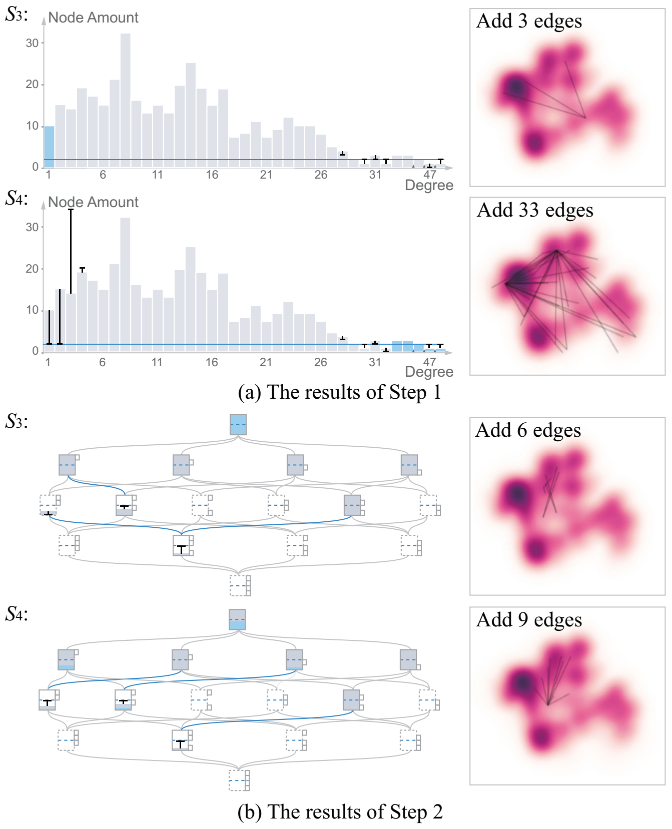


Fig. 13. Two steps with two schemes applied to the Face-to-Face contact dataset. In each row, changes in the number of structural features are visualized on the left and the CDE with the new edges is displayed on the right. Besides, edges indicating the feature transformations in Step 2 are highlighted in blue.

In this case, the effectiveness of processing priority is verified. To get satisfactory results, it is necessary to start with an appropriate priority. If users want to preserve the features consisting of a group of nodes, the decisions need to be made carefully. The locking function not only limits the modification of the node but also limits the means to protect the node. As mentioned in Section 4.2, one of two privacy preserving approaches must modify the node for anonymization. If a locked node has privacy issue, automatic algorithms can only tackle it with one approach, preventing the consideration of the utility. In summary, locking nodes can only protect the features of nodes, but not the features of the entire graph.

7 Discussion

To solicit further feedback about our system, we interviewed domain experts in data privacy about how tools like *GraphProtector* can provide a detailed, guided approach for enabling privacy preservation. In addition, we discuss additional considerations for tools like this: steering sanitization pipelines, dealing with more complex network datasets, and current system limitations.

7.1 Expert Reviews

During the design process, we kept in touch with the domain expert who helped formalize a set of task requirements (Section 3). During development, he continually provided suggestions for improving the system's functionality and design—for example, selecting nodes in the priority view by ranking percentage is more common in their field of research as opposed to specific attribute values.

After the system was fully implemented, we interviewed an additional 4 experts who research privacy preservation and/or computer security. Specifically, each was familiar with network privacy, especially the use of algorithms to preserve privacy and sanitize datasets. Normally this is done in a data-only context. Visualization is not leveraged and the application of algorithms is done as a "one-size-fits-all" approach; there is no customized sanitization for dataset subsets. This means that the resultant utility of a dataset is a major concern of theirs.

In discussing their normal preservation workflows, two main problems were commonly encountered: (1) reviewing data in tabular form is time-consuming and tedious, and (2) it is intractable to formulate and compare detailed solutions to specific issues.

For each expert, we first introduced the *GraphProtector* system and conducted a live, hands-on demo (taking approximately 30 minutes). We then discussed the feasibility of our approach and asked how feasible they considered tools like *GraphProtector* for their dataset sanitization needs.

Being able to measure utility on-the-fly was immediately and widely seen as a major benefit by several of the experts. Integrating multiple techniques with custom schemes was also seen as beneficial. Though not all experts used our specific protectors modules in their normal workflows, they did not encounter difficulty in learning their design and understanding their functionality.

A slight surprise to us was that some experts had trouble with the provenance view. As one noted, his workflow did not involve looking at historical snapshots or any exploratory, user-in-the-loop components. Instead, his research heavily rely on developing advanced algorithms that can automatically identify the optimal parameters for privacy preservation. In contrast, another expert praised that our system provided a "*fine-grained data processing*" pipeline. In particular, this expert appreciated being able to locally set k-values as well as processing priority—he saw this as an improvement for accurate privacy preservation. For his research group, such a design exactly met their requirements: seeking detailed sanitization solutions for network datasets.

An additional expert commented that visualizing the data helps interpretation, especially when presenting results for a processed dataset. He expressed interest in using our design in the future when presenting and publishing his results at conferences/journals or when there is a need for explanation to shareholders and non-experts.

7.2 Detailed Guidance

By integrating automatic models, a visual analysis system allows users to make adjustments in a steering fashion [29]. By using visual analytics, we "uncover the black box" of the privacy models and meet the needs for specific, customized privacy preservation settings.

In *GraphProtector*, users guide automated algorithms in two aspects: terminals (privacy preserving goals) and directions (processing priorities). For individual or localized issues, k_L values can be set. This is significant because requirement differences in data distribution and practical applications make global settings hard to transfer between data contexts. Automatic algorithms cannot easily fulfill this task in multiple scenarios.

In addition, setting reasonable priority orderings can guide directions for automatic algorithms and simplify the process of balancing privacy and utility. It is verified that finding an optimal solution to privacy preservation is NP-hard [4]. Based on the user interactions, the exploration space is greatly limited in a way that focuses graph sanitization according to *GraphProtector*'s workflow.

7.3 Defining Graph Utility

In Section 3, we defined utility as the similarity of structural properties within a graph [27] . It is important to note that, similar to how privacy algorithms depend on task and context, a definition of utility also depends on task and context. As an example, instead of measuring the amount that the graph's structure has changed, utility could be measured as how accurate (or precise) the answers to a set of rule-based questions will be. After sanitization, if the number of correct responses remains high, then utility could be considered successfully preserved.

For our purposes, our collaborating expert users were solely interested in measuring utility via structural graph properties. Adding in new metrics for evaluating utility in new scenarios is certainly possible, but is likewise beyond this paper's scope.

7.4 Scalability and Performance

Scalability is almost always a point of concern when designing visual interfaces. Given limited screen space, how to best visualize and facilitate the understanding of large and dense graph datasets is a challenging and open research topic. A common practice for addressing such an issue is to obtain different global and local statistical features that describe the input graph in multiple aspects (such as the functionality provided in our *GraphProtector* system). Another possible approach is to first apply clustering or aggregation algorithms to reveal the high-level structure of the graph and then allow the user to acquire more detailed information about different parts of the graph through exploration. Further detailed discussion along this direction, however, is beyond the scope of this paper.

Computational efficiency is important for interactive system as users typically expect a short response time. Unfortunately, querying structures and statistical information in large graphs is time– and memory–intensive. For instance, querying 20% of all nodes has a time complexity of $\theta(N^{3.5})$ [9], even when specific information like labels is known.

Pre–computation is a possible solution to improve the performance of tools like *GraphProtector*. However, during sanitization the graph is dynamically changing—making pre–computation infeasible. In addition, some complicated structural features may be manually identified based on the user's current semantics. For instance, users may need to specify subgraphs with arbitrary structures and fulfill subgraph-matching tasks with respect to the background of attackers.

To improve efficiency, we can adjust optimization goals by the setting k_G and k_L values and performing lazy searches. When querying for a structure, the search stops once adequate quantified subgraphs are found; users can be informed that this issue is not a risk. This approach is feasible because small structures tend to frequently appear in large-scale networks.

8 Conclusion

Privacy preservation is a challenging topic due to the conflict between privacy and utility. Diverse features cause massive perspectives for cracking the graph, which increases the difficulty of defense. We design and implement a novel visual interface, called *GraphProtector*, to resist potential attacks on structural features. After dividing questions into issues, users can work on an issue each time and customize accurate strategies. Feedback about the utility is presented to assist users in comparing strategies.

We demonstrate the effectiveness of our approach through two case studies with real-world datasets and expert interviews. With the development of security technology, privacy protection may face increasing multi-facet threats. Integrating more comprehensive techniques can provide stronger defenses but can also lead to conflicts. In the future, we plan to find a more effective way to control conicts generated from hybrid approaches.

Acknowledgements

This research has been sponsored in part by National 973 Program of China (2015CB352503), National Natural Science Foundation of China (61772456, 61761136020), Alibaba-Zhejiang University Joint Institute of Frontier Technologies. This research is also supported in part by the U.S. National Science Foundation through grant IIS-1320229 and IIS-1741536.

REFERENCES

[1] Infectious. `http://konect.uni-koblenz.de/networks/sociopatterns-infectious`.

[2] U. rovira i virgili. `http://konect.uni-koblenz.de/networks/arenas-email`.

[3] D. Archambault and N. Hurley. Visualization of trends in subscriber attributes of communities on mobile telecommunications networks. *Social Network Analysis and Mining*, 4(1):205, Jun 2014.

[4] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Workshop on Knowledge and Data Engineering Exchange*, pp. 45–52. IEEE, 1999.

[5] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou R3579X?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pp. 181–190. ACM, 2007.

[6] P. Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

[7] M. Boss, H. Elsinger, M. Summer, and S. Thurner 4. Network topology of the interbank market. *Quantitative Finance*, 4(6):677–684, 2004.

[8] R. S. Burt. Structural holes and good ideas. *American journal of sociology*, 110(2):349–399, 2004.

[9] V. Carletti, P. Foggia, A. Saggese, and M. Vento. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[10] G. Chartrand, P. Erdos, and O. R. Oellermann. How to define an irregular graph. *College Math. J*, 19(1):36–42, 1988.

[11] 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*, pp. 459–470.

[12] J.-K. Chou, C. Bryan, and K.-L. Ma. Privacy preserving visualization for social network data with ontology information. In *Pacific Visualization Symposium*, pp. 11–20. IEEE, 2017.

[13] J.-K. Chou, Y. Wang, and K.-L. Ma. Privacy preserving event sequence data visualization using a Sankey diagram-like representation. In *SIGGRAPH ASIA 2016 Symposium on Visualization*, p. 1.

[14] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004.

[15] A. Dasgupta and R. Kosara. Adaptive privacy-preserving visualization using parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2241–2248, 2011.

[16] W. Didimo, G. Liotta, F. Montecchiani, and P. Palladino. An advanced network visualization system for financial crime detection. In *IEEE Pacific Visualization Symposium*, pp. 203–210, 2011.

[17] S. P. Dwivedi and R. S. Singh. Error-tolerant graph matching using homeomorphism. In *International Conference on Advances in Computing, Communications and Informatics*, pp. 1762–1766, 2017.

[18] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[19] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[20] A. Garas, P. Argyrakis, and S. Havlin. The structural role of weak and strong links in a financial market network. *The European Physical Journal B*, 63(2):265–271, 2008.

[21] A. Gibbons. *Algorithmic graph theory*. Cambridge University Press, 1985.

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

[23] M. Gjoka, B. Tillman, and A. Markopoulou. Construction of simple graphs with a target joint degree matrix and beyond. In *IEEE Conference on Computer Communications*, pp. 1553–1561, 2015.

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

[25] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1231–1239, 2012.

[26] N.-C. Hsieh. An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems with Applications*, 27(4):623–633, 2004.

[27] S. Ji, W. Li, P. Mittal, X. Hu, and R. A. Beyah. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *USENIX Security Symposium*, pp. 303–318, 2015.

[28] S. Ji, W. Li, M. Srivatsa, J. S. He, and R. Beyah. Structure based data de-anonymization of social networks and mobility traces. In *International Conference on Information Security*, pp. 237–254. Springer, 2014.

[29] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, eds., *Information Visualization*, pp. 154–175. Springer, 2008.

[30] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *Proceedings of the VLDB Endowment*, 7(5):377–388, 2014.

[31] O. D. Lampe and H. Hauser. Curve density estimates. *Computer Graphics Forum*, 30(3):633–642, 2011.

[32] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*, pp. 539–547, 2012.

[33] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 93–106.

[34] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[35] Q. Luo and D. Zhong. Using social network analysis to explain communication characteristics of travel-related electronic word-of-mouth on social networking sites. *Tourism Management*, 46:274–282, 2015.

[36] K. K. Möller and A. Halinen. Business relationships and networks:: Managerial challenge of network era. *Industrial Marketing Management*, 28(5):413–427, 1999.

[37] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pp. 173–187, 2009.

[38] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn. Community-enhanced de-anonymization of online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 537–548.

[39] J. Oksanen, C. Bergman, J. Sainio, and J. Westerholm. Methods for deriving and calibrating privacy-preserving heat maps from mobile sports tracking application data. *Journal of Transport Geography*, 48:135–144, 2015.

[40] S. A. Ríos, F. Aguilera, J. D. Nuñez-Gonzalez, and M. Graña. Semantically enhanced network analysis for influencer identification in online social networks. *Neurocomputing*, 2017.

[41] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, pp. 81–98.

[42] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.

[43] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[44] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867, 2013.

[45] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[46] B. Thompson and D. Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 218–227. ACM, 2009.

[47] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J. Fekete, and D. W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[48] X. Wang, J.-K. Chou, W. Chen, H. Guan, W. Chen, T. Lao, and K.-L. Ma. A utility-aware visual approach for anonymizing multi-attribute tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):351–360, 2018.

[49] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation

based graph generation. *Transactions on data privacy*, 6(2):127–145, 2013.

[50] X. Wu, X. Ying, K. Liu, and L. Chen. A survey of privacy-preservation of graphs and social networks. In C. C. Aggarwal and H. Wang, eds., *Managing and mining graph data*, pp. 421–453. Springer US, Boston, MA, 2010.

[51] Y. Wu, S. Liu, K. Yan, M. Liu, and F. Wu. Opinionflow: Visual analysis of opinion diffusion on social media. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1763–1772, 2014.

[52] M. Xiao, J. Wu, and L. Huang. Community-aware opportunistic routing in mobile social networks. *IEEE Transactions on Computers*, 63(7):1682–1695, 2014.

[53] Q. Xiao, R. Chen, and K.-L. Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 911–920. ACM, 2014.

[54] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*, pp. 587–596, 2013.

[55] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 739–750.

[56] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pp. 506–515, 2008.

[57] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.