

# Privacy-Integrated Graph Clustering Through Differential Privacy

Yvonne Mülle<sup>\*</sup>  
University of Zurich,  
Switzerland  
muelle@ifi.uzh.ch

Chris Clifton  
Purdue University, USA  
clifton@cs.purdue.edu

Klemens Böhm  
Karlsruhe Institute of  
Technology (KIT), Germany  
klemens.boehm@kit.edu

## ABSTRACT

Data mining tasks like graph clustering can automatically process a large amount of data and retrieve valuable information. However, publishing such graph clustering results also involves privacy risks. In particular, linking the result with available background knowledge can disclose private information of the data set. The strong privacy guarantees of the differential privacy model allow coping with the arbitrarily large background knowledge of a potential adversary. As current definitions of neighboring graphs do not fulfill the needs of graph clustering results, this paper proposes a new one. Furthermore, this paper proposes a graph clustering approach that guarantees 1-edge-differential privacy for its results. Besides giving strong privacy guarantees, our approach is able to calculate usable results. Those guarantees are ensured by perturbing the input graph. We have thoroughly evaluated our approach on synthetic data as well as on real-world graphs.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Clustering;  
H.2.8 [Database Applications]: Data Mining

## General Terms

Algorithms, Security

## Keywords

Clustering, Graph Mining, Differential Privacy

## 1. INTRODUCTION

For many types of data, the proper representation is a graph structure. There exist many approaches which automatically retrieve valuable information from graphs. One such approach is *graph clustering*. Clustering groups similar objects together and assigns dissimilar objects to different groups. In the context of social networks, graph clustering helps to better understand the structure of these networks.

<sup>\*</sup>Work originated while author was at Purdue University.

Being able to collect and process such graph data does not only have its opportunities, but also involves privacy risks. The *Gaydar* project [1] of two MIT students illustrates such a risk: by linking the *Facebook* friends of a person with the knowledge of the gender and sexuality of those friends, it has been possible to predict if the person is gay. Let us now assume that an operator of a social network wants to publish information on the community structure (i.e., cluster structure) of the network. The operator decides to publish the result of a graph clustering algorithm, i.e., the community structure of the network, without the actual graph structure. Such a clustering result can still put the users' privacy at risk. Dependent on the background knowledge, an outsider can retrieve actual connections between users from the community structure. In the worst case, it might even be possible to not only identify certain connections but to reconstruct the entire social network. In consequence, it is necessary to publish the community structure and thus the clustering result in a privacy preserving manner. As differential privacy gives strong privacy guarantees, it is desirable to publish a differentially private graph clustering result. For differentially private graph clustering, no approach exists so far. The paper focuses on proposing such an approach.

In order to apply differential privacy to graph clustering, the following challenges have to be solved. 1. It needs to be determined which parts of the graph information that the clustering result exposes are relevant to privacy: a node, that node's edges, or any edges in the graph. 2. A noise-adding mechanism must be developed to protect this information. As a consequence, only the graph parts to be protected should be perturbed. Choosing the appropriate perturbation is challenging as, for instance, minor changes in the edge structure of the graph can significantly alter the clustering result. 3. The ever present trade-off between privacy and usability must be solved so that both are preserved at a reasonable level. This also includes determining which properties a graph clustering algorithm and graphs must have in order to still produce a usable clustering result.

In this paper, we propose solutions to these challenges. An important category of graph clustering approaches calculates their results only based on the edge structure. The nodes are used to represent a cluster. Thus changes in the node set are directly visible in the clustering result. Therefore, we propose *m*-edge-differential privacy. It relies on a new neighboring graphs definition that guarantees privacy for the edges of a node, but does not require changing the node set. We also investigate if it is necessary to protect all edges of a node or if it is sufficient to only protect some

of them. Additionally, we propose *PIG*, an approach that combines the perturbation of the input graph with existing graph clustering approaches. *PIG* guarantees 1-edge-differential privacy, a notion that can be generalized to  $m$ -edge-differential privacy. The amount of noise created by the perturbation is configurable via a parameter that represents the trade-off between privacy and usability. We develop recommendations on how to set this parameter to achieve specific privacy goals.

## 2. RELATED WORK

Privacy for data mining tasks can be achieved by releasing sanitized data sets or by developing private data mining algorithms. The approaches not only differ in how data mining is performed, but also in the underlying privacy model.

*Privacy-preserving clustering* approaches perform clustering on tabular data where the data is distributed among several parties. They are based on the concept of secure multi-party computation. The goal is that each party only learns the final clustering result, but no intermediate values. Privacy preserving  $k$ -means clustering [26, 11, 10] and privacy-preserving DBSCAN [15] have been studied. All these approaches have in common that they perform clustering on tabular data and therefore are not able to deal with many instances of graph data. Furthermore, unlike differential privacy they provide no protection against the result inherently revealing individual information. A clustering result contains new, so far probably unknown correlations between entries or nodes and thus has the potential to reveal sensitive information.

*Differentially private data analysis* is the task of publishing a graph in a differentially private manner. [19] uses a Kronecker graph model in order to publish a differentially private maximum likelihood estimator for graphs. [24] proposes an approach that uses the degree correlations of the original graph to generate a differentially private  $dK$ -graph model. Thus, it aims to preserve as much structure of the unmodified graph as possible. Both approaches publish the synthetic graph for a general purpose. However, as data mining tasks often differ in what data they need, the accuracy of the result is expected to increase when the data is sanitized for a specific task.

*Differentially private data mining* approaches exist for tabular data as well as for graph data. Differentially private data mining techniques have been proposed for frequent itemsets [2, 17] and pattern mining [9, 25], but also for clustering and graph analysis. The challenge how to perform clustering in a differentially private manner has been addressed by [6], [22], and [7]. All approaches focus on  $k$ -median and  $k$ -means queries on tabular data. [6] realizes the task by publishing private coresets that are representative subsets of a database which preserve some geometric properties. [22] releases private clustering results by perturbing the coordinates of the center points that represent a cluster. [7] proposes an approach for differentially private  $k$ -median clustering. The approach uses the exponential mechanism to swap center points with non-center points.

The following approaches release individual graph properties in an edge-differentially private manner. Graph properties like the degree distribution [8], frequent graph patterns [25], counting queries for  $k$ -triangles and  $k$ -stars [12], and clustering coefficients [27] have been considered. Guaranteeing node-differential privacy has been taken into account

for the number of edges in a graph, counting queries like triangles,  $k$ -cycles,  $k$ -stars and certain estimators for power law graphs [13].

Differential privacy was successfully applied to all those data mining tasks. However, differentially private graph clustering has not yet been addressed. Thus, we focus on how graph clustering results can be released in a differentially private manner.

## 3. FOUNDATIONS

### 3.1 Graph Clustering

Clustering groups similar objects in so-called *clusters*. In contrast to clustering on tabular data, graph clustering also – or even exclusively – considers the structural data given by the edges of the graph. The following four criteria specify a graph clustering approach: the graph type, the cluster definition, the clustering realization and the representation of the graph clustering result.

The *graph type* contains the information on what sort of graph the clustering approach can be performed. Definition 1 formalizes the term *graph* as used in this paper.

*Definition 1.* Graph  $G = (V, E)$

A graph  $G = (V, E)$  consists of a set of nodes  $V$  and a set of edges  $E \subseteq V \times V$ . The graph is non-attributed and unweighted. The edges between nodes are undirected.

A *cluster definition* contains the properties that a set of nodes must fulfill in order to form a cluster. No universally accepted graph cluster definition exists. We consider cluster definitions that group nodes together based on their connectivity and other structural properties. Such properties are reachability, inter-cluster connectivity, and neighborhood.

The *realization of the cluster definition* consists of two steps: how many clusters a node can belong to and the concrete procedure of calculating the clustering result based on the given cluster definition. We do not restrict the realization of the cluster definition.

A *graph clustering result*  $Res = \{C_1, \dots, C_j\}$  is a set of clusters, each of which consists of a set of node IDs. This general solution is mostly used in state-of-the-art graph clustering algorithms. Furthermore, it contains all essential information and does not reveal additional structural information that could increase the privacy risk of a clustering result.

To sum up, this paper focuses on graph clustering approaches that fulfill the following requirements: (R1) They require an undirected, unweighted and non-attributed graph as input. (R2) Their cluster definition is based on connectivity and other structural properties of the graph. (R3) Their representation of the clustering result consists of node IDs.

### 3.2 Differential Privacy

*Differential privacy* [3] is a privacy model that gives strong privacy guarantees. It assumes a powerful adversary. The adversary has a broad, nearly unlimited background knowledge. He even is aware of all entries – except for a single one – of the data set. Despite this strong adversary, differential privacy protects against determining if the unknown individual is even in the data. This is the case as query results on the data set remain indistinguishable, independent of whether a single database entry has participated in the result calculation or not.

An algorithm  $\mathcal{A}$  is differentially private if the following holds true: For any possible result, it cannot be determined beyond a specific certainty if it was calculated on a graph  $G_1$  or on its neighboring graph  $G_2$ .

*Definition 2.  $\epsilon$ -Differential Privacy*

A graph clustering algorithm  $\mathcal{GCA}$  is  $\epsilon$ -differentially private if for all neighboring graphs  $G_1$  and  $G_2 \in \mathcal{N}(G_1)$ , and for all subsets  $\mathcal{S}$  of the set of all possible outputs  $\{Res_1, \dots, Res_i\}$ ,

$$\Pr[\mathcal{GCA}(G_1) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{GCA}(G_2) \in \mathcal{S}]$$

$\mathcal{N}(G_1)$  is the set of all neighboring graphs of graph  $G_1$ .

The term *neighboring graphs* for graph clustering is discussed in Subsection 4.1. In general, *deterministic algorithms* cannot achieve differential privacy (if the result is dependent on the input data set). But they can be converted into *non-deterministic algorithms* by adding non-deterministic noise. Noise-adding mechanisms for graph clustering approaches are discussed in Subsection 4.2.

## 4. GRAPH CLUSTERING MEETS DIFFERENTIAL PRIVACY

### 4.1 Neighboring Graphs

[8] has proposed two adaptations of differential privacy for graph data. *k-edge-differential privacy* assumes that the edges contain the sensitive information and thus up to any  $k$  edges in the graph should be protected. In contrast, *node-differential privacy* states that a node and all its adjacent edges contain the sensitive information. In the following, we analyze which requirements a neighboring graphs definition for graph clustering must fulfill. As a result we show that the neighboring graphs definition of node-differential privacy does not meet the needs of graph clustering and *k-edge-differential privacy* covers more neighboring graphs than required. Limiting the set of neighboring graphs to its sufficient set might simplify achieving differentially private graph clustering. Thus, we propose a new *neighboring graphs* definition based on the requirements we identify in the following.

**Requirement: Node Set.** A graph clustering result is represented by node IDs. Thus, adding or removing a node can be directly visible in the clustering result. Therefore, it is necessary that neighboring graphs consist of the same node set. This is why the neighboring graph definition of node-differential privacy is not applicable.

**Requirement: Edge Set.** We consider graph clustering approaches that calculate their result based on the edge structure of the graph. Thus, the clustering result provides information about structural similarities in the graph. Concealing private information encoded in the clustering result directly means concealing the existence of certain edges in the graph. When differential privacy is applied to tabular data, a certain row is both used to calculate the clustering result and at the same time should be concealed in the result. Analogously, the edges of a particular node are what determines the clustering – thus it stands to reason that this is the property that should be concealed in this case. This is why the neighboring graph definition of *k-edge-differential privacy* is too broad and thus not appropriate here.

**Requirement: Number of Protected Edges.** Is it sufficient to protect some of the edges of a node or is it necessary to protect all of them? Most real-world graphs are power law distributed, i.e., there exist many nodes with few edges and only few nodes with many edges. For instance, in a social network like Facebook, it is more likely that the highly connected nodes represent companies and public figures. Companies and public figures actively decided to reveal more information about their relationships to other participants than a private person would do. As a consequence, it is sufficient to protect the relationships (and thus edges) of private persons with few edges. Our neighboring graphs definition allows protecting  $m$  edges of a node. Setting  $m$  to a value that covers the number of edges a private person will have, allows protecting (almost) all edges of some nodes (if they have a low degree) and preserve the privacy of some edges for highly connected nodes.

**Neighboring Graphs Definition.** Definition 3 formalizes the term *neighboring graphs*. Its flexibility allows preserving all edges of a node or only a single edge in the whole graph. How many edges of a node are preserved is dependent on parameter  $m$ .

*Definition 3. Neighboring Graphs*

Let  $m \in \mathbb{N}$  be given. Two graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  (according to Definition 1) are neighboring graphs iff

1.  $V_2 = V_1$
2.  $\exists x \in V_1 : E_2 = (E_1 - E_a(x)) \cup E_b(x)$  with  
 $E_a(x) \subseteq \{(u, v) \in E_1 | u = x \vee v = x\}$  and  
 $E_b(x) = \{(u, v) \in E_2 | u = x \vee v = x \wedge (u, v) \notin E_1\}$
3.  $|E_1 \setminus E_2| + |E_2 \setminus E_1| \leq m$

$\mathcal{N}(G_1)$  is the set of all neighboring graphs of graph  $G_1$ .

Note that for the case  $m = k = 1$ , there is no difference between the neighboring graphs definition of *k-edge-differential privacy* and our definition. For  $m = k > 1$  *m-edge-differential privacy* covers a subset of neighboring graphs compared to *k-edge-differential privacy*.

**Impact of Definition 3 on Clustering Results.** Our neighboring graphs definition allows removing edges from and adding edges to a neighboring graph. Thus, comparing the clustering results of two neighboring graphs can result in the following differences: none, only a small number of nodes are clustered differently, or most nodes belong to different clusters, see Example 1.

*Example 1.* The original graph and two neighboring graphs are shown in Figure 1. Nodes within a circle represent a cluster. A cluster contains at least four nodes which are connected to at least three other nodes in the cluster. In graph  $G$  there exist three different clusters. Removing a single edge as in graph  $G_2$  results in the removal of Cluster 1 because its nodes no longer fulfill the cluster definition in  $G_2$ . Removing the edge between Node 1 and Node 6 as it is the case for graph  $G_3$  does not have an impact on the clustering result at all because the two nodes are also not clustered in  $G$ . However, adding two new edges to  $G_3$  (compared to  $G$ ) results in a new cluster in the clustering result of graph  $G_3$ : Cluster 4.

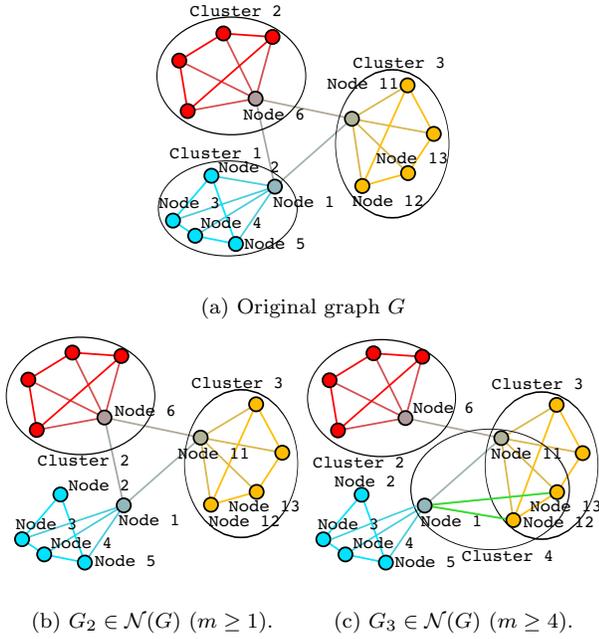


Figure 1: Illustration of the term *neighboring graphs*.

## 4.2 Perturbation Mechanisms

We are aware of three different types of perturbation in the literature: sampling, output perturbation and input perturbation. In the following, we discuss if they are suitable for our purposes.

**Sampling.** Sampling integrates the perturbation into the proceeding of the clustering approach. A possibility would be to only consider a sample of potential edges, which may change the *result set*. For instance, a clique clustering algorithm would have a non-zero probability of creating a cluster out of a set of nodes even though they are not fully connected to each other if it draws a sample of edges that make the nodes fully connected. An advantage of sampling could be that the required amount of noise can be adapted during the execution of the clustering algorithm. At this stage, more information on how to choose the sample could be available. This might reduce the influence of the perturbation on the clustering result. However, sampling has the following two drawbacks: (1) Each of the many existing graph clustering approaches must be separately adapted dependent on the individual cluster definitions and its realizations. (2) Integrating the perturbation into the clustering algorithm makes the analysis of privacy properties complex. In order to prove differential privacy, it is required to analyze the probabilities of clustering results for a specific algorithm. This is particularly challenging as the output space of a graph clustering algorithm is discrete, resulting in difficulty specifying a probability density function except by (intractable) enumeration. Additionally, finding a closed-form solution for the upper and lower bounds of the probabilities may become very difficult for complex clustering algorithms.

The aforementioned problems can be avoided if the perturbation is a separate step of the algorithm. This allows the perturbation to be analyzed individually.

**Output Perturbation.** Output perturbation means that the perturbation is performed on the graph clustering result. For instance, the cluster assignment of the nodes can

be changed. However, a clustering result only indirectly contains the information that should be protected – the edges of a node. Thus, there exists no intuitive mapping between how the cluster assignment has to be changed and the guarantee of protecting the existence of the edges of a node. The only possibility would be to additionally take the input data into account.

**Input Perturbation.** Input perturbation adds noise to the input graph. With this method, it is possible to perturb two neighboring graphs in a way that they are indistinguishable with a certain probability for the graph clustering algorithm. As a consequence, the probabilities of a graph clustering result do not need to be calculated; the output of an algorithm on differentially private input is differentially private. In the following, we discuss how the input perturbation must be realized to be appropriate for graph clustering. (1) According to Definition 3, perturbing the graph by changing its number of nodes is not an option. (2) Thus, it is only possible to perturb the graph by adding and removing edges. (2.1) Doing only one of the two (adding or removing edges) is not an option. The reason is that there always exist cases where such a procedure is not possible: If a graph is fully connected, no edges can be added. If a graph does not contain any edges, no edges can be removed. Thus, the clustering result probabilities would also not be affected, and only those results that are calculated for those graphs would be possible. (2.2) Determining the number of edge changes according to the global sensitivity of a clustering approach is not a possibility. The global sensitivity [4] contains the information about the maximum distance between two clustering results of neighboring graphs. There does not exist a correlation between the number of edge changes and the distance between graph clustering results: There is no guarantee that adding or removing  $l$  edges will result in a distance increased or decreased by  $l$  – yet the distance is what the global sensitivity is based on.

Thus, in order to achieve both differentially private results independent of the algorithm and allow for useful results, we propose an approach that combines edge sampling with edge perturbation in Section 5.

## 5. *PIG*

We propose *PIG* – **Privacy-Integrated Graph** clustering approach – as a general approach for guaranteeing 1-edge-differential privacy. It is independent of a concrete graph clustering algorithm as long as the clustering approach fulfills the requirements given in Subsection 3.1. Furthermore, *PIG* perturbs the input graph by perturbing the adjacency matrix of the graph dependent on differential privacy’s parameter  $\epsilon$ . As future work, *PIG* will be extended to  $m$ -edge-differential privacy.

*PIG* consists of two steps: the perturbation of the input graph, called  $PIG_{pert}$ , and the graph clustering algorithm applied to the perturbed graph. The idea behind *PIG* is as follows: If the perturbed versions of neighboring graphs are the same, the graph clustering approach will calculate the same clustering result.

The perturbation step of *PIG* is shown in Algorithm 1. The perturbation method is a combination of edge sampling and edge flipping, i.e., edge randomization. It operates on the adjacency matrix  $A$  of the input graph.  $a_{ij}$  refers to the entry of  $A$  in row  $i$  and column  $j$  and thus contains the information if an edge between node  $i$  and node  $j$  ex-

---

**Algorithm 1** Graph Perturbation Algorithm  $\mathcal{PIG}_{pert}$ 

---

```
1: function PERTURBGRAPH(Graph  $G = (V, E)$ , privacy
   parameter  $s$ )
2:   construct graph  $G_{pert} = (V', E')$  where  $V' = V$ 
3:   construct adjacency matrix  $A$  from  $E$  of  $G$ 
4:   initialize the adjacency matrix  $A'$  for  $E'$  of  $G_{pert}$ 
5:   for all  $a_{ij} \in A$  with  $i < j$  do
                                      $\triangleright$  Preservation
6:     if  $a_{ij}$  is chosen with probability  $1 - s$  then
7:       set  $a'_{ij} = a'_{ji} = a_{ij}$  in  $A'$  of  $G_{pert}$ 
8:     else
                                      $\triangleright$  Randomization
9:       if 0 is chosen with probability  $\frac{1}{2}$  then
10:        set  $a'_{ij} = a'_{ji} = 0$  in  $A'$  of  $G_{pert}$ 
11:       else
12:        set  $a'_{ij} = a'_{ji} = 1$  in  $A'$  of  $G_{pert}$ 
13:       end if
14:     end if
15:   end for
16:   return  $G_{pert}$ 
17: end function
```

---

ists. The existence of an edge is represented by a value of one, whereas an absence is represented as zero. As the input graph is undirected, the adjacency matrix is symmetric. This property is preserved in the perturbation step. As the definition of self-loops does not make sense in the case of undirected graphs, the corresponding entries  $a_{ii}$  are and also remain zero in the perturbed entries  $a'_{ii}$ .

The perturbation consists of the following mechanism: For each entry in the adjacency matrix, it is first determined if preservation or randomization should be performed. In order to make this choice, we introduce a privacy parameter  $s$  ( $s \in (0, 1]$ ). Preservation is chosen with probability  $(1 - s)$ , whereas randomization is chosen with probability  $s$ . The higher  $s$  is, the more entries in the matrix are randomized. In the case of preservation, the original entry of the adjacency matrix of the unperturbed graph is preserved in the perturbed version of the graph. With randomization, the entry in the perturbed version of the graph gets assigned 1 with probability  $\frac{1}{2}$  and 0 with the same probability. The assignment and thus the absence or presence of this particular edge in the graph is independent of its existence or absence in the original input graph.

**THEOREM 1.** *Edge-Differentially Private  $\mathcal{PIG}$* 

$\mathcal{PIG}$  guarantees 1-edge-differential privacy for  $\epsilon \geq \ln(\frac{2}{s} - 1)$  ( $s \in (0, 1]$ ).

The idea behind the proof of Theorem 1 is as follows: The proof is based on proving the basic definition of differential privacy (see Definition 2). The probability that  $\mathcal{PIG}$  calculates a certain clustering result consists of two terms: the probability that  $\mathcal{PIG}_{pert}$  returns a certain perturbed version of the original graph and the probability that the graph clustering algorithm used in  $\mathcal{PIG}$  calculates that result on the perturbed graph. In 1-edge-differential privacy neighboring graphs can only differ in one edge. Thus, the ratio of the probabilities that two neighboring graphs are perturbed to an equal graph is only dependent on the ratio of the probabilities that the differing edge gets assigned the same value in  $\mathcal{PIG}_{pert}$ . According to Algorithm 1, an entry preserves its

original value in the perturbed graph with a probability of  $1 - \frac{1}{2}s$ , and gets assigned its opposite value with a probability of  $\frac{1}{2}s$ . With this information, the probability ratio that two neighboring graphs are perturbed to the same graph can be expressed and Theorem 1 can be proven. For the proof of Theorem 1 please refer to [20].

**Choice of Privacy Parameter  $s$ .**

Due to its correlation with parameter  $\epsilon$  in  $\epsilon$ -differential privacy,  $\mathcal{PIG}$ 's privacy parameter directly influences the extent of privacy that  $\mathcal{PIG}$  is able to guarantee. But what is a sufficient value for  $s$ ? It is necessary to find a trade-off between the quality of  $\mathcal{PIG}$ 's clustering result and privacy guarantees in order to choose parameter  $s$ .

Ideally, an adversary cannot say for any edge in the perturbed graph whether or not it existed in the original graph with any greater confidence than if they were to flip a coin, i.e., a confidence greater than 50%. The idea to achieve this is to set  $s$  to such a value that the expected density of the perturbation result will be twice the original density. This implies that for each edge in the perturbed graph, an adversary can never say with more than 50% confidence that this edge was also present in the original graph. This is particularly relevant for very sparse graphs. For small  $s$  it occurs rarely that the perturbation will both be at an entry in the adjacency matrix where the edge is set to exist *and* the perturbation changes that entry.

**THEOREM 2.** *Choice of privacy parameter  $s$* 

If privacy parameter  $s$  has the following value, it then holds that the expected density in the perturbed graph is twice the original density  $d$  ( $d \leq 25\%$ ).

$$(1 - s) \cdot d + \frac{s}{2} \geq 2 \cdot d$$
$$\Leftrightarrow s \geq \frac{2 \cdot d}{1 - 2 \cdot d}$$

The derivation of the expected density is shown in Theorem 4. This way of choosing  $s$  is possible up to a density of 25% because the maximum value for  $s$  is reached at that point. However, this density limit covers virtually all real-world graphs. At higher densities, a new trade-off between privacy and usability would have to be found.

**Influence on Graph Structure.**

As  $\mathcal{PIG}_{pert}$  changes the edge structure of the input graph, we exemplarily analyze the following three important graph properties upon which cluster definitions are based: (1) connectivity that depends on the knowledge of the exact graph structure, and (2) the number of edges, and (3) the density that both are calculated based on the number of edges in the graph. We determine the expected changes of the graph properties in terms of the properties of the original graph. Given a graph  $G = (V, E)$ ,  $n = |V|$ ,  $m = |E|$ , we refer to  $P = (V_P, E_P)$  as a perturbed version of graph  $G$ .  $\mathcal{PIG}_{pert}$  preserves an entry in the adjacency matrix with probability  $1 - \frac{1}{2} \cdot s$ ; an entry gets assigned the opposite value in the perturbed graph with probability  $\frac{1}{2} \cdot s$ .

**Connectivity** As the perturbation mechanism operates on one edge at a time and does not take the other entries of the adjacency matrix into account, it is possible that the perturbed version of a connected input graph is unconnected.

**Number of Edges.** The expected number of edges  $E[|E_P|]$  in the perturbed graph  $P$  depends on the expected number

of preserved edges and that of added edges (i.e., flipped entries in the adjacency matrix of  $G$ ). The adjacency matrix of  $G$  contains  $(\frac{n \cdot (n-1)}{2} - m)$  changeable zero entries which can result in added edges in  $P$ . The main diagonal of the matrix contains non-changeable zero entries as self-loops are not allowed in the graph.

**THEOREM 3.** *Expected Number of Edges in  $P$*

$$\begin{aligned} E[|E_P|] &= (1 - \frac{1}{2} \cdot s) \cdot m + (\frac{n \cdot (n-1)}{2} - m) \cdot \frac{1}{2} \cdot s \\ &= (1 - s) \cdot m + \frac{n \cdot (n-1)}{4} \cdot s \end{aligned}$$

In a sparse graph, there exist only few edges and many changeable zero entries in its adjacency matrix. Thus, perturbing such a graph results in a high increase in the number of edges. The perturbed version of a graph with half of the maximum number of possible edges is expected to have the same edge count as before. If more than half of the possible edges are present, the number of edges decreases in the perturbed graph.

**Density.** The density of a graph  $G$  is defined as the ratio of the number of edges in  $G$  and its number of possible edges. The number of possible edges only depends on the number of nodes in the graph and the fact that the graph is undirected. Thus, it is not influenced by the graph perturbation, which only operates on the edge set of a graph.

**THEOREM 4.** *Expected Density  $d_P$  of  $P$*

$$E[d_P] = \frac{E[|E_P|]}{\frac{n \cdot (n-1)}{2}} = (1 - s) \cdot d + \frac{s}{2}$$

## 6. EVALUATION

The goal of our empirical evaluation is to analyze the impact of  $\mathcal{PIG}$  on cluster quality, i.e., its influence on the clustering result. Furthermore, we perform our evaluation on synthetic graphs that vary in their density. This is interesting as the behavior of  $\mathcal{PIG}_{pert}$  changes with increasing density. We use certain graphs and two graph clustering approaches, and from there we generalize our results to properties a graph and clustering approaches must have in order to cope with the perturbation introduced with  $\mathcal{PIG}$ .

### Setup.

**Graphs.** The synthetic graphs are generated based on [16]: both the community sizes and the number of edges per nodes are power law distributed. As real-world graphs, we use the Disney graph [21] that is a subgraph of the Amazon co-purchasing network with a small number of nodes and the Facebook graph [18] with more than ten times the nodes.

The cluster quality is measured by means of the F1 score. The non-perturbed clustering result is used as ground truth.

**Graph Clustering Approaches.** We use  $\mathcal{PIG}$  with two different graph clustering approaches:  $SCAN$  [28] and *Graph  $k$ -Medoids* [23].  $SCAN$  adapts the cluster definition of DB-SCAN [5] to graphs. The cluster definition is based on the density of the neighborhood ( $\epsilon$ -neighborhood), i.e., with how many neighboring nodes a node shares a certain number of neighboring nodes. If the size of the  $\epsilon$ -neighborhood exceeds a threshold  $\mu$ , the node becomes a core object. The graph perturbation of  $\mathcal{PIG}$  directly influences this property.

Adding and removing edges in a graph changes which nodes are in the neighborhood of a certain node. As the perturbation is done randomly, similar neighborhoods of neighboring nodes may become dissimilar when common nodes vanish and disjoint new node sets can be added to the neighborhood. *Graph  $k$ -Medoids* adapts the cluster definition of  $k$ -medoids [14] to graphs. On the one hand, its cluster definition is based on shortest paths which  $\mathcal{PIG}_{pert}$  can affect to a great extent. On the other hand, it has the two characteristics that the number of clusters is set to  $k$  and that all nodes are assigned to exactly one cluster. Thus, it is interesting whether these characteristics can reduce the influence of  $\mathcal{PIG}_{pert}$  on the clustering result.

### Cluster Quality.

Increasing parameter  $s$  means having more changes in the input graph. In Theorem 2, we have presented a heuristic that guarantees a sufficient amount of privacy while minimizing the required amount of perturbation. Thus, we evaluate the impact of the perturbation on cluster quality for a range of  $s$  around following minimum values resulting from the heuristic.

	Density	$s \geq$	$\epsilon \leq$
Synthetic graph (506 nodes)	1.29%	0.027	4.292
Disney graph	4.39%	0.096	2.988
Facebook graph	1.08%	0.022	4.499

As a general result, increasing privacy parameter  $s$  beyond a certain range around the minimum value and thus decreasing  $\epsilon$  increases the privacy guarantees at the cost of usable clustering results. Due to space limitations, the detailed evaluation of this aspect is in [20].

**$\mathcal{PIG}$ -SCAN.** Figure 2 shows the cluster quality of  $\mathcal{PIG}$ -SCAN for the real-world graphs, compared to the clustering results of  $SCAN$ . It also contains the corresponding cluster result statistics. The key *original* means that the same parameters as for the clustering on the non-perturbed graph are used. The key *optimal* means that we use those parameters that result in the highest F1 score. Thus, the comparison with the optimal parameter setting shows how close the clustering results on the perturbed and non-perturbed graphs are at best. However, it is difficult or impossible to determine the optimal parametrization without first performing the clustering on the non-perturbed graph and using the result as ground truth. Such a proceeding can result in a privacy risk, as the perturbed clustering result contains additional information on the non-perturbed one. This is not the case if the parametrization is independent of the non-private clustering result.

For the Disney as well as the Facebook graph the cluster qualities are very close to each other for the different parametrizations. First, we consider the *Disney* graph in Figure 2. The decrease in the cluster quality for the original parameter setting is the result of the decrease of the number of clustered nodes. The higher  $s$  is, the more random edges are added. This influences the neighborhood structure of the nodes and thus the similarity of the neighborhoods between neighboring nodes. Adding random neighbors implies that adding the same new neighbors to nodes of the former neighborhood is unlikely. As a result, the  $\epsilon$ -threshold of  $SCAN$  can no longer preserve the same set of core objects. If this set changes, the set of nodes which are candidates for a cluster also changes. Thus, the cluster structure itself

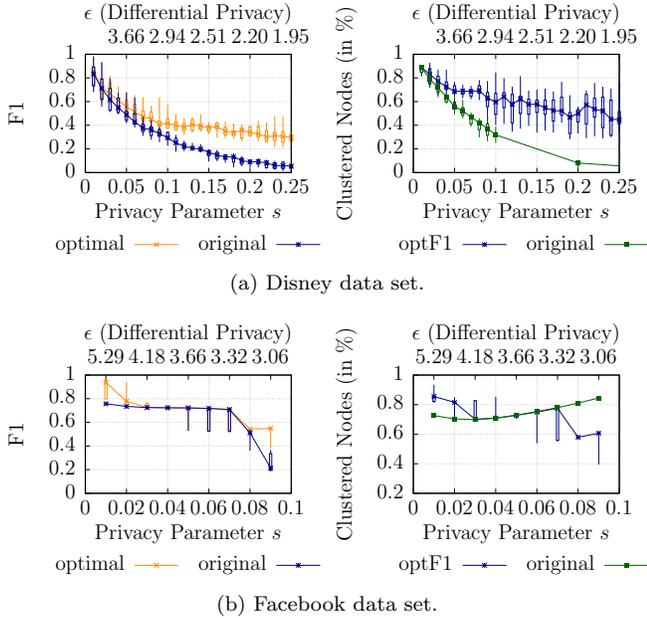


Figure 2: Cluster quality of *PIG-SCAN*.

greatly differs from the non-private clustering result. Thus, for  $s = 0.1$ , the F1 score is only about 0.3.

*Result 1.* Graphs with a high density like the Disney graph (4.39%) require a relatively high  $s$  value and thus a lot of perturbation. Additionally, if such a graph does not contain clearly separated clusters in its non-perturbed version – as it is the case for the Disney graph – those changes in the graph structure have an even higher impact. This makes it difficult for density-based approaches like *SCAN* to come up with a useful private clustering result.

For the quality results on the Facebook graph in Figure 2, the F1 score is almost constant for  $s \in [0.01, 0.07]$ . This behavior can be explained with *SCAN*'s parameter setting together with the structure of the Facebook graph. The Facebook graph is a graph with 4,039 nodes and a density of 1.08%. The non-private clustering result only contains six clusters with almost 90% clustered nodes. Using *SCAN* with  $\mu = 160$  requires that core objects must have many neighbors, and this results in a high intra-cluster connectivity. *SCAN*'s  $\epsilon$  is the minimum threshold of how similar the neighborhoods of neighboring nodes must be. Thus, setting *SCAN*'s  $\epsilon = 0.1$  reduces the required similarity. This particular parameter setting can cope very well with the additional noise in the graph, resulting in an F1-score of about 0.7 for the minimum required value  $s = 0.03$ . However, using such an extreme parametrization does not always result in stability against the perturbation. The number of clustered nodes for the original parametrization is about 20% less than the number for the non-private result. The fact that this results in a high F1 score and accuracy means that most of the nodes are true positives wrt. the non-perturbed clustering result.

*Result 2.* Sparse graphs like the Facebook graph which allows clustering almost all nodes in few, clearly separated clusters can preserve their strong intra-cluster connectivity upon perturbation. Thus, the impact of *PIG* on the cluster quality is small and results in useful clustering results.

*Result 3.* *SCAN*'s thresholds do not strictly bound the neighborhood size, but allow variations in the size that still pass them. Thus, graph clustering approaches which do not require an exact edge structure, but can cope with neighbor changes, and which bound their requirements by thresholds seem to be able to cope with *PIG*.

*PIG-Graph k-Medoids.* We evaluate the impact of *PIG-Graph k-Medoids* on the quality of the clustering result on the synthetic graphs with 506 nodes and the *Disney* graph. We use the same parametrization for the clustering on the perturbed graph as used on the non-perturbed one. The cluster quality of *PIG-Graph k-Medoids* is shown in Figure 3. For the synthetic graph, the F1 score is quite low. For the minimum recommended value of  $s = 0.03$ , there are only quality values under 0.4. Thus, the perturbed clustering result is no longer able to imitate the non-private clustering result very well. The reason is that its cluster definition uses shortest paths that are highly sensitive to edge changes. As a consequence, we have to negate the hypothesis that the two characteristics of *Graph k-Medoids* can reduce the impact of *PIG* on cluster quality. An interesting outcome is that *PIG* with *Graph k-Medoids* produces clustering results of higher quality on the Disney graph than with *SCAN*, especially as the result of *PIG-SCAN* are the worst on this graph. For the minimum recommended  $s = 0.1$ , the F1 score is almost 0.4. We hypothesize that the positive outcome is correlated with the special structure of this real-world graph and thus the perturbation can only influence the shortest path structure to a small extent.

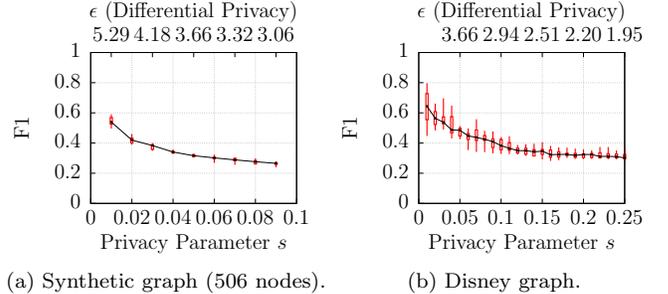


Figure 3: Cluster quality of *PIG-Graph k-Medoids*.

*Result 4.* Clustering algorithms whose cluster definition is based on metrics that are highly sensitive to edge changes do not perform well with *PIG*. For such approaches, the privacy guarantees dominate the usability.

*Graph Density Variation.* In the synthetic graphs, the nodes have many edges within their cluster and only few edges to nodes outside of it. Thus, increasing the density for those graphs means strengthening the structure within the communities and then adding more connections to nodes outside the community. With this behavior in mind, the cluster quality results shown in Figure 4 are as expected. For both node sizes, the densest graph has the highest cluster quality. We vary the density in 0.5% steps up to a density of 2.5%. Much higher densities are not possible with the graph generator used and 40 communities. The minimum recommended  $s$  values are as follows:

		1.0%	1.5%	2.0%	2.5%
506 nodes	$s \geq$	0.03	0.04	0.05	0.06

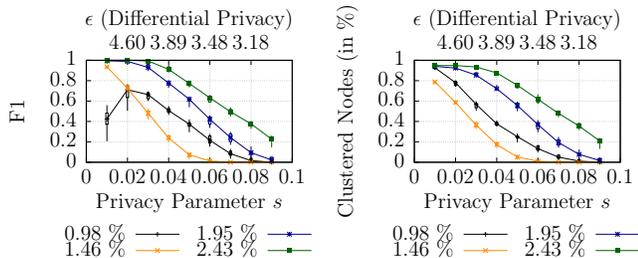


Figure 4: *PIG-SCAN* and density variation on synthetic graphs with 506 nodes.

The F1 score on the graph with density 0.98% is slightly better than the one on the graph with density 2.43%. An interesting outcome is the cluster quality on the graph with density 1.46%. For  $s > 0.02$ , the cluster quality is the worst compared to the quality on the other graphs. A reason is the percentage of clustered nodes: The higher  $s$  is, the fewer nodes are clustered. Compared to the results on the other graphs, it has the lowest clustered-nodes-rate. This has to do with the concrete graph structure and is not dependent on the density. The degree distribution can also not explain why the percentage of clustered nodes is the smallest. The graph with the lowest density has a similar degree distribution.

*Result 5.* The cluster quality highly depends on the  $s$  value chosen. The denser the graph is, the higher  $s$  must be. Thus, better cluster quality results on denser graphs are relativized compared to sparser graphs. This is because the denser ones require higher  $s$  values.

## 7. CONCLUSIONS

Publishing usable graph clustering results while giving strong privacy guarantees is an important and challenging task. It is important as a graph clustering result can reveal information that is not directly encoded in the result, e.g., parts of the original graph structure.

In this paper, we have adapted differential privacy to graph clustering. We have developed a neighboring graphs definition suitable for graph clustering results. Based on this definition, we have proposed  $m$ -edge-differential privacy. Furthermore, using graph perturbation as noise adding mechanism allows guaranteeing differential privacy for arbitrary graph clustering approaches. Based on those results, we proposed *PIG* that guarantees *1-edge-differential privacy*. It independently perturbs each entry of the adjacency matrix. In a thorough evaluation, we have analyzed the impact of *PIG* on the graph structure and cluster quality. We have shown that *PIG* can lead to good results and can preserve the cluster structure up to a certain extent.

This paper is a first stab at the problem of differentially private graph clustering. We see several topics worthy of further research. First, it might be worthwhile to classify the existing graph clustering approaches according to their ability to cope with a perturbation such as the one of *PIG*. Then, a user can better choose a clustering algorithm for *PIG* and can get a good and at the same time private clustering result. Second, we want to develop adaptation techniques for each category: Slightly adapting the graph clustering algorithm could result in more robustness against the perturbation of *PIG*.

## 8. REFERENCES

- [1] Gaydar Project at MIT, 2013. last accessed December 30, 2013.
- [2] R. Bhaskar et al. Discovering Frequent Patterns in Sensitive Data. In *KDD*, 2010.
- [3] C. Dwork. Differential Privacy. In *Automata, Languages and Programming*, volume 4052. 2006.
- [4] C. Dwork et al. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, volume 3876. 2006.
- [5] M. Ester et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, 1996.
- [6] D. Feldman et al. Private Coresets. In *STOC*, 2009.
- [7] A. Gupta et al. Differentially Private Approximation Algorithms. *CoRR*, abs/0903.4510, 2009.
- [8] M. Hay et al. Accurate Estimation of the Degree Distribution of Private Networks. In *ICDM*, 2009.
- [9] S.-S. Ho and S. Ruan. Differential Privacy for Location Pattern Mining. In *SPRINGL*, 2011.
- [10] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A New Privacy-Preserving Distributed k-Clustering Algorithm. In *SDM*, 2006.
- [11] S. Jha, L. Kruger, and P. McDaniel. Privacy Preserving Clustering. In *Computer Security (ESORICS)*, volume 3679. 2005.
- [12] V. Karwa et al. Private Analysis of Graph Structure. *Proceedings of the VLDB Endowment*, 4(11), 2011.
- [13] S. Kasiviswanathan et al. Analyzing Graphs with Node Differential Privacy. In A. Sahai, editor, *Theory of Cryptography*, volume 7785. 2013.
- [14] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. Delft University of Technology. 1987.
- [15] K. Kumar and C. Rangan. Privacy Preserving DBSCAN Algorithm for Clustering. In *Advanced Data Mining and Applications*, volume 4632. 2007.
- [16] A. Lancichinetti and S. Fortunato. Benchmarks for Testing Community Detection Algorithms on Directed and Weighted Graphs with Overlapping Communities. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 80(1), 2009.
- [17] N. Li et al. PrivBasis: Frequent Itemset Mining with Differential Privacy. *Proceedings of the VLDB Endowment*, 5(11), July 2012.
- [18] J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems 25*, 2012.
- [19] D. Mir and R. Wright. A Differentially Private Graph Estimator. In *ICDMW*, 2009.
- [20] Y. Mülle. Sanitizing Graph Clustering and Community Detection Results Through Differential Privacy. Master's thesis, Karlsruhe Institute of Technology (KIT), 2014.
- [21] E. Müller et al. Ranking Outlier Nodes in Subspaces of Attributed Graphs. In *ICDEW*, 2013.
- [22] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth Sensitivity and Sampling in Private Data Analysis. In *STOC*, 2007.
- [23] M. J. Rattigan, M. Maier, and D. Jensen. Graph clustering with network structure indices. In *ICML*, 2007.
- [24] A. Sala et al. Sharing Graphs Using Differentially Private Graph Models. In *IMC*, 2011.
- [25] E. Shen and T. Yu. Mining Frequent Graph Patterns with Differential Privacy. In *KDD*, 2013.
- [26] J. Vaidya and C. Clifton. Privacy-Preserving k-Means Clustering over Vertically Partitioned Data. In *KDD*, 2003.
- [27] Y. Wang et al. On Learning Cluster Coefficient of Private Networks. In *ASONAM*, 2012.
- [28] X. Xu et al. SCAN: A Structural Clustering Algorithm for Networks. In *KDD*, 2007.