# Edge Deletion based Subgraph Hiding

LEYLA TEKIN, BELGIN ERGENC BOSTANOGLU
Department of Computer Engineering,
Izmir Institute of Technology,
Izmir,
TURKEY

*Abstract:* - Extracting subgraphs from graph data is a challenging and important subgraph mining task since they reveal valuable insights in many domains. However, in the data sharing scenario, some of the subgraphs might be considered as sensitive by the data owner and require hiding before publishing the data. Therefore, subgraph hiding is applied to the data so that when subgraph mining algorithms, such as frequent subgraph mining, subgraph counting, or subgraph matching, are executed on this published data, sensitive subgraphs will not appear. While protecting the privacy of the sensitive subgraphs through hiding, the side effects should be kept at a minimum. In this paper, we address the problem of hiding sensitive subgraphs on graph data and propose an Edge deletion-based heuristic (EDH) algorithm. We evaluate our algorithm using three graph datasets and compare the results with the previous vertex masking heuristic algorithms in terms of execution time and side effects in the context of frequent subgraph hiding. The experimental results demonstrate that the EDH is competitive concerning execution time and outperforms the existing masking heuristic algorithms in terms of side effects by reducing information loss of non-sensitive patterns significantly and not creating fake patterns.

*Key-Words:* - Graph data, subgraph privacy, knowledge hiding, privacy preserving graph mining, sensitive subgraph hiding, sharing graph data, subgraph mining, disclosure threshold.

Received: July 27, 2023. Revised: May 13, 2024. Accepted: June 26, 2024. Published: July 17, 2024.

## 1 Introduction

Graphs are effective tools for modeling complex structures in distinct domains, such as molecules in chemo informatics [1], protein networks in bioinformatics [2], [3], and social networks [4]. Subgraphs can be used in a variety of applications that are anomaly detection [5], cluster analysis [6], graph classification [7], [8], and community detection [9], [10].

Although subgraphs allow for identifying patterns and extracting useful insights, publishing graph data may raise privacy concerns. Privacy-preserving publishing of graph data is mostly related to anonymization, and perturbation, generalization, and cryptographic-based anonymization techniques have been proposed [11], [12], [13], [14], [15]. All of these studies focus on data hiding. Few works concentrate on knowledge hiding [16]. The study [16] dealt with subgraph hiding and developed hiding techniques to protect the privacy of sensitive subgraph patterns in data-sharing scenario.

The existing algorithms to hide sensitive subgraphs while publishing graph data mask the vertices of the graph using different heuristics, [16]. They follow the blocking-based approach that may bring out privacy breaches, [17]. The output graph data shows the touched vertices and produces fake subgraphs with masked vertex labels after the hiding process. Our motivation is (i) to develop an algorithm that can be used to hide sensitive subgraphs as a precaution to subgraph mining algorithms like frequent subgraph mining [18], subgraph counting [19], [20], and subgraph matching [21], [22], etc., (ii) to concentrate on the edges rather than the vertices of the graph because modifying an edge has less effect on the graph structure than affecting all the edges of a vertex, and (iii) to delete edges instead of updating the labels with a symbol, and (iv) not to create fake subgraphs on the shared data. With this motivation, the algorithm we designed and implemented differs from the existing subgraph hiding algorithms by dealing with the edges of the sensitive graphs rather than the vertices, performing deletions instead of masking, and not generating new fake patterns as a masking symbol is not used.

Our study focuses on subgraph hiding on graph data to preserve the privacy of sensitive subgraphs. We offer a new Edge deletion-based heuristic (EDH) algorithm to hide sensitive subgraphs by decreasing their frequencies below a disclosure

threshold defined by the data owner. We conduct experiments to compare EDH with the available heuristic algorithms [16], and to be consistent, explanations and experiments are given in the frequent subgraph hiding context. We measure the execution times and side effects of the algorithms. The results point out that EDH achieves significant improvement on the side effects with quite less information loss and not causing fake patterns and results in similar execution times.

The organization of the paper is as follows. Section 2 provides a background to help in understanding the key concepts, and explains the problem and measures. Section 3 describes the proposed EDH algorithm. Section 4 presents the performance evaluation of the EDH and existing algorithms on three datasets. Section 5 contains the related work, and Section 6 gives the conclusion and future work.

# 2 Background

This section provides the background information with the basic definitions of terms and concepts, states the subgraph hiding problem in the context of frequent subgraph hiding (FSH) and introduces the performance measures used in this context.

## 2.1 Basic Definitions

**Graph concepts:** A graph is composed of a nonempty vertex (or node) set V and an edge set E, in which the vertices are linked by the edges. When no direction is associated with the edges of a graph, the graph is called an undirected graph. Otherwise, it is a directed graph.

In a labeled graph, labels are assigned to the vertices or edges from a set of symbols. An unlabeled graph has no such labeling. Figure 1a and Figure 1b display vertex and edge-labeled graph examples.

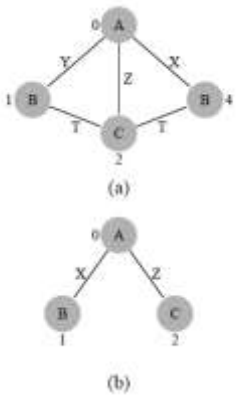


Fig. 1: A subgraph isomorphism example

**Subgraph isomorphism:** Let H and G be graphs. These graphs are isomorphic if there is a bijection *f* between the vertex sets of *H* and *G* (indicated as *f: V(H) → V(G))* such that any two vertices (*u* and *v*) are adjacent in graph *H* if and only if *f(u)* and *f(v)* are adjacent in *G*. If the graphs have vertex and edge labels, the vertex labels of *u* and *f(u)*, and the edge labels of *(u,v)* and *(f(u),f(v))* must also be equal. The subgraph isomorphism problem determines whether a graph includes a subgraph that is isomorphic to a given graph. Figure 1 shows a subgraph isomorphism example.

A subgraph is an induced subgraph if it has the correspondence of all the edges connecting the corresponding vertices in the graph. Figure 1 is not an example of induced subgraphs since the edge (2,4) in the graph is missing in the subgraph.

**Frequent subgraph:** A graph dataset *D* is a collection of graphs (called graph transactions). The *frequency (freq)* of a subgraph is the number of graphs containing the subgraph, and the support of a subgraph is a ratio of *freq* to $|D|$, where $|D|$ is the total number of graphs in the dataset.

If the support (*supp*) of a subgraph *H* is greater than or equal to a specified threshold (σ, minimum support threshold), that is $supp(H) \geq \sigma$, then it is a *frequent subgraph*. A frequent subgraph mining algorithm obtains the set of frequent subgraphs.

**Sensitive pattern:** Let *D* be a transactional graph database. A set of sensitive patterns, indicated by *SP*, is defined by the data owner and should be hidden. In [16], $SP \subset FP$, where σ is the minimum support threshold, and *FP* is a set of all frequent subgraph patterns that can be mined from *D* based on σ.

**Disclosure threshold:** A disclosure threshold, denoted by ψ, is a hiding threshold; and used to hide all the sensitive patterns in *SP*. A sensitive pattern *P* is hidden by decreasing its frequency below the given disclosure threshold, hence $freq(P) < \psi$.

**Sensitive graph:** Let *G* be a graph in a transactional graph database *D*, and *SP* be a set of sensitive patterns. If there exists a sensitive pattern $P \in SP$ and *P* is a subgraph of *G*, then *G* is called a sensitive graph. That is, if a graph in *D* supports a sensitive pattern in *SP*, then it is a *sensitive graph*.

## 2.2 Frequent Subgraph Hiding

The frequent subgraph hiding (FSH) problem transforms a graph database *D* into a sanitized graph database *D′* such that no sensitive pattern is disclosed when a frequent subgraph mining algorithm is applied to the *D′*, and it has as little impact as possible on the original data and non-sensitive patterns.

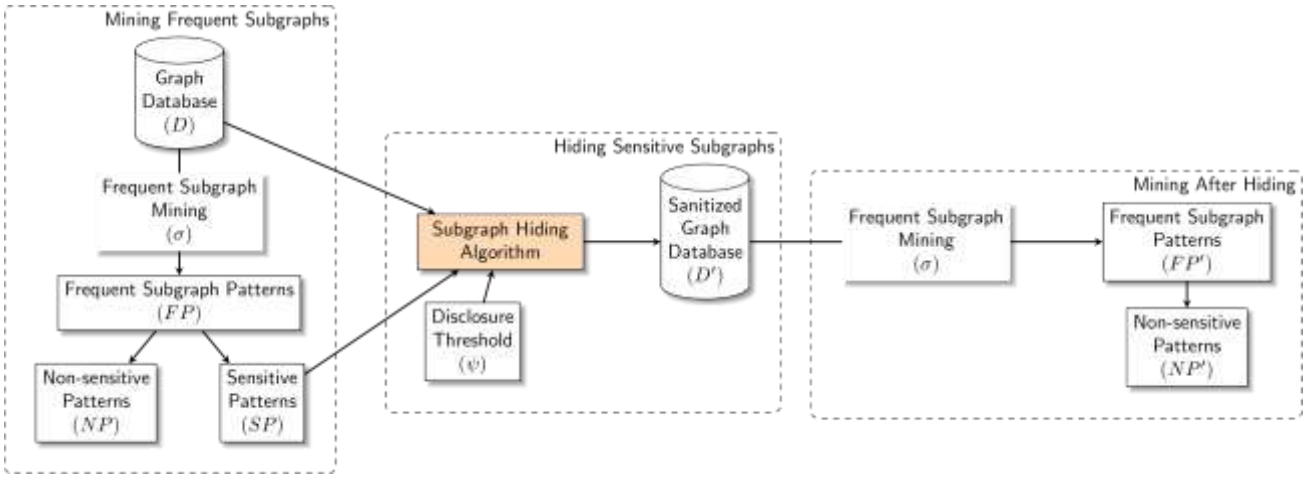


Fig. 2: Frequent subgraph hiding process

The FSH process is illustrated in Figure 2. It takes a graph database *D*, a sensitive pattern set *SP*, and a disclosure threshold ψ as the inputs, and produces the sanitized database *D′* as the output. Now, *D′* is ready to be shared since if a subgraph mining algorithm is applied, only non-sensitive patterns can be extracted.

There are two subproblems to solve the FSH problem. The first subproblem is to find the sensitive graphs for sanitization in the database for each sensitive pattern. The second one is the sanitization of an identified sensitive graph so that the graph no longer supports the sensitive pattern.

The vertex masking-based heuristic algorithms for the FSH problem are proposed by following the blocking-based approach, [16]. The study selects the sensitive graphs with fewer matches to sanitize and introduces three heuristics for the sanitization of a sensitive graph. The heuristics are Heuristic 1, Heuristic 2, and Heuristic 3.

- Heuristic 1: It finds the distinct vertex labels in the pattern, and for each label, counts the number of vertices in the graph. It masks all the vertices having the label with the least occurrence in the graph.
- Heuristic 2: The most frequent graph vertex in all the matches of the sensitive pattern is detected. The label of this vertex is masked in the graph, and the matches containing the vertex are deleted from the match set. It repeats until the match set is empty.
- Heuristic 3: The frequency of every distinct graph vertex in all the matches of the sensitive pattern is calculated. The frequencies are sorted in decreasing order. The first i vertices are chosen for masking such that the sum of frequencies of these i vertices is not less than the match set size. But, these vertices may be within the same matches, and the procedure, including the subgraph matching, is repeated until the match set is empty. It requires running a subgraph-matching algorithm multiple times.

Figure 3 gives an illustrating example of sanitization of a sensitive graph with Heuristic 2. Suppose the subgraph in Figure 1b is the sensitive pattern and the sensitive graph in Figure 3a is given. There are 3 matches of the pattern in this graph, that are *{2:0, 0:1, 3:2}*, *{2:0, 1:1, 3:2}* and *{5:0, 0:1, 4:2}*. The idea of Heuristic 2 is to choose the most frequent vertex in the matches. If there is more than one such vertex, it selects randomly. One of the most frequent vertices of the graph in the matches is vertex 2 which is included in 2 matches. Heuristic 2 first masks it on the graph. Then, 1 match is left and Heuristic 2 masks the vertex 5. The graph sanitized with Heuristic 2 can be seen in Figure 3b. This hiding may produce artifact patterns with the missing vertex label as shown in Figure 3c and Figure 3d.

## 2.3 Performance Measures

To evaluate the performance of an FSH algorithm, the following measures are used, which are adapted from [23].

**Hiding Failure (HF).** The ratio of sensitive patterns that are discovered after sanitization. It is computed as; $HF = |SP'|/|SP|$, where $|SP'|$ and $|SP|$ are the number of sensitive patterns that are discovered from the sanitized database *D′* and the original database *D*, respectively.

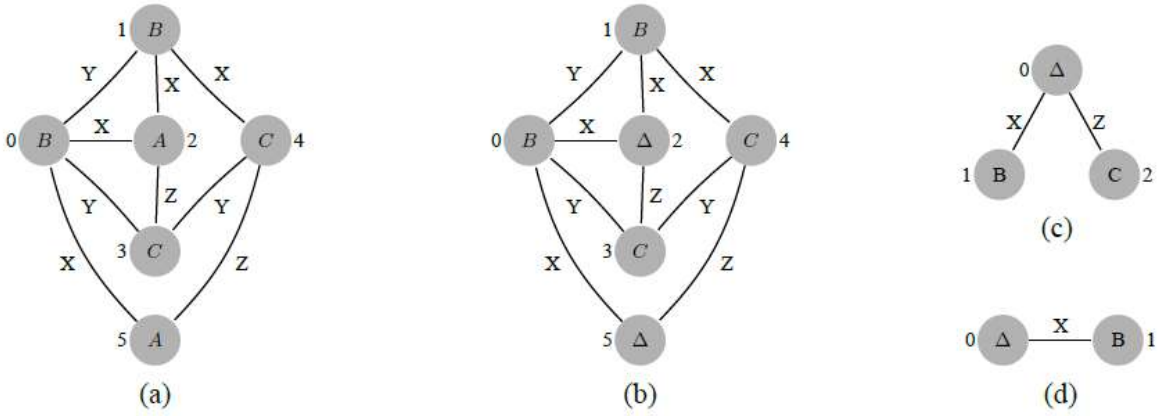


Fig. 3: An illustrating example of sanitization; (a) a sensitive graph with the matches of the sensitive pattern in Figure 1b, (b) the graph after masking based sanitizing with Heuristic 2, and Figure (c)-(d) possible artifact patterns

**Artifact patterns (AP).** The number of the discovered patterns that are artifacts and have missing vertex labels due to masking with a symbol. It is calculated as; $AP = |FP'|-|FP \cap FP'|$, where $|FP'|$ and $|FP|$ are the number of frequent patterns that are discovered from the sanitized database $D'$ and the original database $D$, respectively.

**Information Loss (IL).** The ratio of non-sensitive patterns that are hidden accidentally during sanitization. It is computed as; $IL = ((|FP| - |SP|) - (|FP \cap FP'|-|SP'|))/ (|FP|-|SP|)$, where it excludes frequent patterns containing the masking symbol with the $|FP \cap FP'|$ to find how many of the frequent patterns in the original database are present in the sanitized database. The equation is obtained according to the side effects calculation in [24]. The loss of a frequent pattern that includes a sensitive pattern is not considered since it will be inevitably hidden when the sensitive one is hidden, [24].

**Distance (Dist).** The total number of vertices and edges that are changed (masked or deleted) during the sanitization, and calculated as; $Dist = |V| - |V'_U| + |E| - |E'_U|$, where $V'_U$ is the set of the vertices unchanged in the sanitized database, and $E'_U$ is the set of the edges unchanged in the sanitized database. It is defined to compare the distance of subgraph hiding algorithms based on distinct approaches using graph concepts.

# 3 Edge Deletion-Based Heuristic (EDH) Algorithm

A frequent subgraph-hiding algorithm hides a set of sensitive patterns by reducing the frequency of every sensitive subgraph pattern below a predefined disclosure threshold. The frequency of a sensitive pattern decreases one if all of its matches are removed from a sensitive graph in the database. Following this idea, the Edge deletion-based heuristic (EDH) algorithm selects the most frequent edge in all the matches of the sensitive subgraph pattern in the input graph. Removing the most frequent common edge in the matches of the sensitive pattern from the graph will handle most of the matches of the sensitive pattern at the same time. With single and most effective removal of common victim edge will have little impact on the input graph. In other words, while hiding the sensitive patterns, the non-sensitive patterns can be preserved in the database to the maximum possible extent with the removal of a minimum number of edges.

EDH is primarily composed of five major steps as follows:

Step1: Compute the relevance of the data graphs and the relevance of the sensitive patterns. To find the relevance of each data graph to the sensitive pattern set, the number of matches of all the sensitive patterns in each data graph is calculated. Likewise, the relevance of each sensitive pattern to the graph dataset is found. Thus, for each sensitive pattern, the number of matches of the pattern in all the graphs is calculated.

Step2: Sort the data graphs. The data graphs are sorted in increasing order of their relevance numbers computed in Step 1.

Step3: Sort the sensitive patterns. The sensitive patterns are sorted in increasing order of their relevance numbers computed in Step 1.

Step4: Compute the number of sensitive graphs to be sanitized. According to the given disclosure threshold $\psi$, the number of graphs that need to be sanitized to hide each sensitive pattern P is computed by $freq(P)-\psi+1$.

**Algorithm 1:** EDH Algorithm.

**Input:** An original database $D$, A sensitive pattern set $SP$, A disclosure threshold $\psi$
**Output:** The database $D$ modified so that the patterns in $SP$ cannot be mined.

1: **for** each $P \in SP$ **do** ▷ $G.matches$ and $P.matches$ are 0.
2: $mCount \leftarrow 0$
3: **for** each $G \in D$ **do**
4: $matchNum \leftarrow$ Count the number of the matches $M(G, P)$
5: $mCount \leftarrow mCount + matchNum$
6: $G.matches \leftarrow G.matches + matchNum$
7: **end for**
8: $P.matches \leftarrow mCount$
9: **end for**
10: Sort $D$ in increasing order of $G.matches$
11: Sort $SP$ in increasing order of $P.matches$
12: **for** each $P \in SP$ **do**
13: $NumGraphsToSanitize \leftarrow freq(P) - \psi + 1$
14: **for** $i \leftarrow 1$ to $NumGraphsToSanitize$ **do**
15: Get next $G \in D$ such that $G$ contains $P$
16: $M \leftarrow$ Find the matches $M(G, P)$
17: Replace values of the matches in $M$ with keys
18: Sort $M$ based on values
19: **while** $M \neq \varnothing$ **do**
20: $E \leftarrow$ empty list ▷ $E$ keeps all the $G$ edges in $M$.
21: **for** each $m \in M$ **do**
22: **for** $(v1, v2) \in P.edges$ **do**
23: $(gv1, gv2) \leftarrow (m[v1], m[v2])$
24: Add $(gv1, gv2)$ into $E$
25: **end for**
26: **end for**
27: $e \leftarrow$ The most frequent edge of $G$ in $E$
28: Delete the edge $e$ from $G$
29: $M \leftarrow M - M_e$ where $M_e$ includes the matches containing $e$
30: **end while**
31: **end for**
32: **end for**

The disclosure threshold here is a direct measure, not a percent to balance between the disclosure and privacy which means no sensitive pattern is disclosed.

Step5: Sanitize a sensitive graph. This step takes a sensitive graph and a sensitive pattern as inputs and performs sanitization on the graph by removing all the matches of the pattern. The edge distortion-based approach that we propose relies on edge deletions from the graph.

Our edge deletion-based subgraph hiding algorithm EDH is illustrated in Algorithm 1. In step 1, the relevance of the data graphs and the relevance of the sensitive patterns are calculated, which can be seen in lines 1 to 9. The *matchNum* is the number of matches of a sensitive pattern in a graph of the database. For each pattern, the *matchNum* value is summed through all the graphs in the dataset, and assigned to *P.macthes*. Also, for each data graph, the *G.matches* are accumulated across all the sensitive patterns. In step 2, the data graphs are sorted in increasing order of *G.macthes* (line 10). In step 3, the sensitive patterns are sorted in increasing order of *P.macthes* (line 11). Then, for each sensitive pattern $P \in SP$, in Step 4, *NumGraphsToSanitize* is found at line 13 as the number of graphs to be sanitized. *NumGraphsToSanitize* sensitive graphs are sanitized for each pattern by removing all of its matches. For this, in Step 5, local sanitization of a sensitive graph is performed in lines 16-30:

1. All the matches of the sensitive pattern in the graph (*M*) are found. A match in *M* consists of the pairs of graph vertex id and its corresponding pattern vertex id. The keys of each match are replaced with their values, and *M* is sorted by the values.
2. In lines 20-26, all the graph edges in *M* are determined.
3. The most frequent edge of the graph in *M* is determined. If there is more than one edge having the same frequency, the first one in the list *E* is selected.
4. The edge is deleted from the graph.
5. The matches containing the edge are removed from *M*, and it is iterated from 2 until no match is left.

Let us give an illustrating example of sanitizing the sensitive graph in Figure 3a by using our edge distortion-based approach. All the matches of the sensitive pattern in Figure 1b are removed, as described in Step 5. First, EDH finds the three matches of the pattern in the graph. All the edges of the graph in these matches are detected. EDH finds (2, 3) as the most frequent edge of the graph in the matches, and deletes it. The matches including the edge (2, 3) are removed from the match set. Then, EDH finds (0, 5) as one of the most frequent ones in the remaining one match, and deletes it. The graph sanitized with an edge distortion-based approach can be seen in Figure 4. This distortion approach does not generate new artifact patterns.

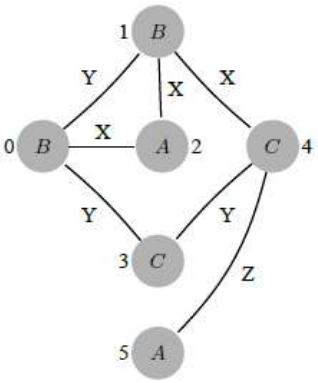


Fig. 4: The graph after sanitizing with the edge distortion based approach

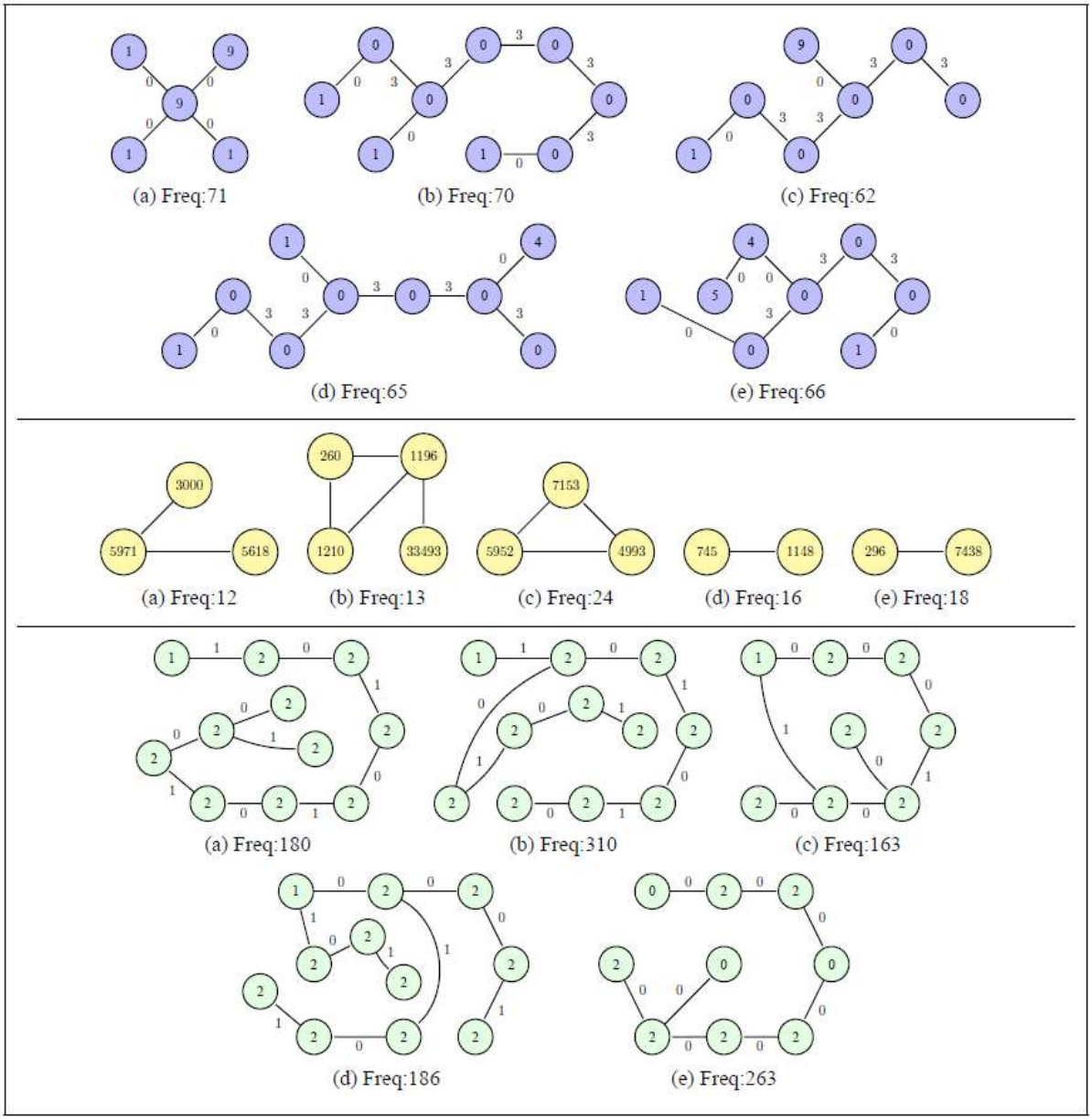


Fig. 5: Sensitive patterns chosen from Chemical, Movielens and NCI109 datasets (from top to bottom)

# 4 Performance Evaluation

We compare our edge deletion-based algorithm EDH with three heuristics of the vertex-masking algorithm [16], which are described in section 2.2. We name Heuristic 1, Heuristic 2 and Heuristic 3 as VMask1, VMask2 and VMask3, respectively. We implement the algorithms in Python. All the experiments are performed on a PC with an Intel Core i7 8750 2.2 GHz CPU and 16GB RAM, running Windows 10.

## 4.1 Datasets

We carry out our experiments on three real datasets that are Chemical [25], Movielens [26] and NCI109 [27].

The characteristics of the datasets used in this study can be seen in Table 1 where Chem, Movie, and NCI represent Chemical, Movielens, and NCI109, respectively.

In Chemical dataset, a graph corresponds to a chemical compound. Vertices represent atoms and edges represent bonds between atoms. A vertex label represents the atom type, and an edge label represents the bond type. It is a sparse dataset. Movielens dataset includes the movie tags assigned by users and times when the tags were assigned. It is a single graph with 47,957 tag assignments between two specific dates. From the single graph, a graph is created for each day, where a vertex is a tagged movie on the day and an edge is placed if two movies have the same tag. A vertex label corresponds to the movie id, and the edges do not have labels, that is only '0' is used as the label and the number is specified as 1 in the table. After deleting graphs with no edges, the dataset contains 802 graphs in total. NCI109 is a commonly used cancer screen dataset.

## 4.2 Experimental Results

In the study [16], three measures (execution time, number of masking symbols, and ratio of preserved frequent patterns) were used. To evaluate the performance of blocking and distortion-based FSH algorithms, we use the measures explained in Section 2.3. In addition, the execution times of the algorithms are tested. The recorded execution times are CPU times. Since all the algorithms hide all the sensitive patterns, hiding failure is 0% for all of them. That is why, the algorithms are compared in terms of (i) execution time, (ii) information loss, (iii) artifact patterns, and (iv) distance.

We run the gspan [28] algorithm to find the frequent patterns from the three datasets. Five sensitive subgraph patterns are selected randomly for each dataset. Figure 5 shows the selected sensitive patterns from Chemical, Movielens, and NCI109 datasets (from top to bottom). The frequency of each sensitive pattern is written below it. In VMask2, VMask3, and EDH, to obtain the matchings of a subgraph in a graph, VF2 [29] algorithm is used. The experiments are done to see how the varying disclosure thresholds affect the performance of the algorithms for the datasets.

Table 1. Characteristics of the datasets

| | Chem | Movie | NCI |
|---|---|---|---|
| # of graphs | 340 | 802 | 4127 |
| # of vertices | 9189 | 29001 | 122494 |
| # of edges | 9317 | 202263 | 132603 |
| Avg # of vertices | 27 | 36.2 | 29.7 |
| Avg # of edges | 27.4 | 252.2 | 32.1 |
| # of dist. vertex labels | 66 | 5800 | 38 |
| # of dist. edge labels | 4 | 1 | 3 |

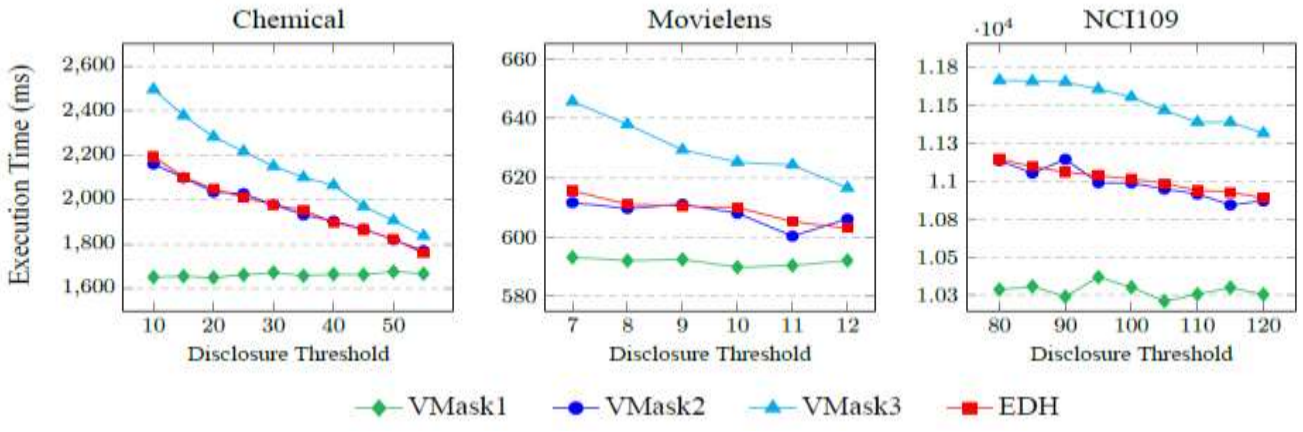


Fig. 6: Execution time (ms) vs disclosure threshold

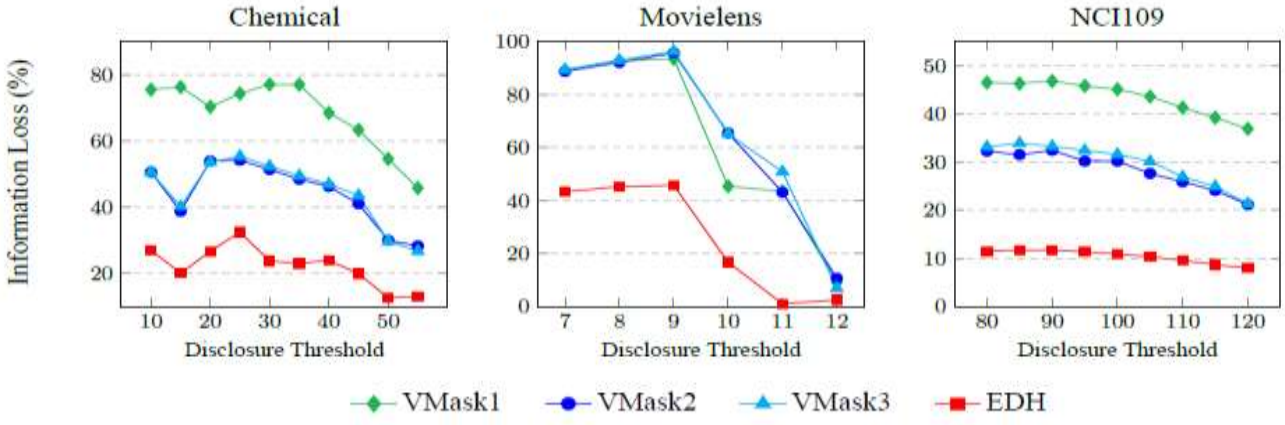


Fig. 7: Information loss (%) vs disclosure threshold

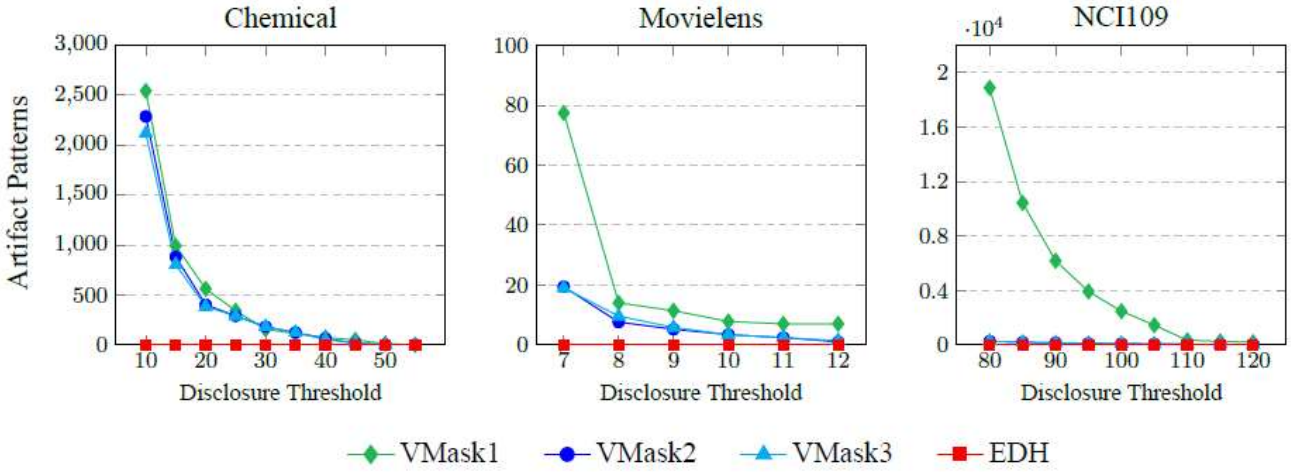


Fig. 8: Artifact patterns vs disclosure threshold

Every result is obtained by an average of five executions. Support and disclosure thresholds are kept the same while measuring the information loss and artifact patterns. The experiments of Movielens are done for the disclosure thresholds between 7 and 12 because frequent patterns within the dataset cannot be found for threshold 6 in our computer, and the frequency of the pattern with the least frequency is 12.

Figure 6 demonstrates the execution times (in ms) of VMask1, VMask2, VMask3, and EDH algorithms on Chemical, Movielens, and NCI109 datasets for different threshold values. It can be seen that VMask1 has the best execution time since it does not consider the matchings of a sensitive pattern in a sensitive graph so it does not execute a subgraph-matching algorithm. VMask3 has the worst execution time since it may need to run a subgraph-matching algorithm multiple times. VMask2 and EDH show similar execution time results. We also see that the execution times of the algorithms decrease while the disclosure threshold increases because sanitization is applied to less number of graphs. Additionally, the difference between the results is more on the Chemical and NCI109 datasets than that on the Movielens dataset because a vertex in a graph in that dataset is a movie, and each movie ID (vertex label) can be included at most once in each graph, resulting in at most one match for a sensitive pattern. Moreover, the Movielens dataset has very low support values for patterns.

The information loss measure quantifies the ratio of non-sensitive frequent patterns hidden unintentionally during sanitization. Figure 7 illustrates the information loss (%) of the algorithms on Chemical, Movielens, and NCI109 datasets for different threshold values.

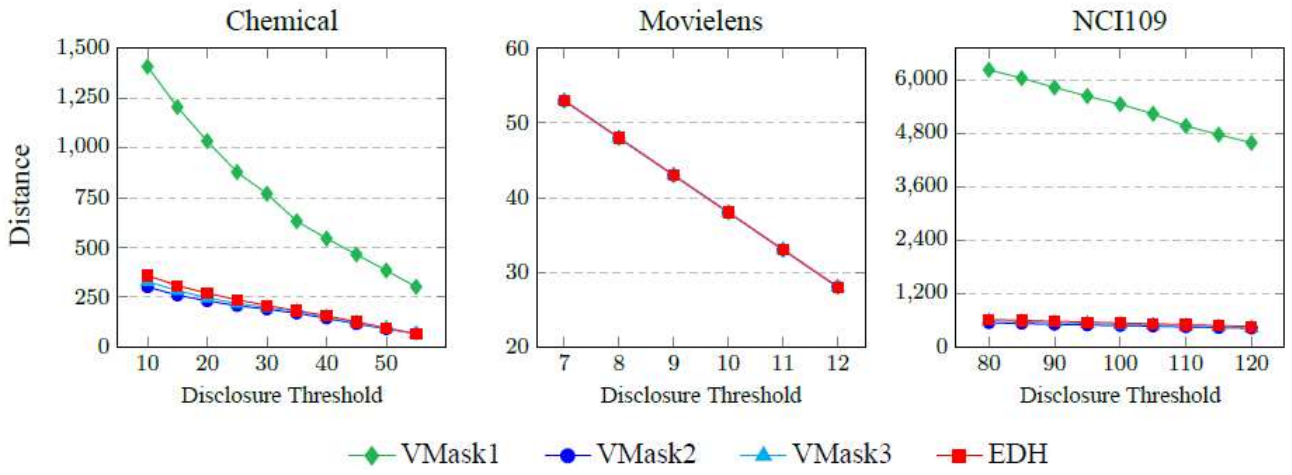


Fig. 9: Distance vs disclosure threshold

According to the figures, the EDH outperforms the other algorithms on all the datasets since it takes into account deleting edges rather than masking the labels of the vertices. If a vertex is masked, the edges associated with that vertex are also affected since non-sensitive frequent patterns in the original data containing that vertex cannot be extracted so its edges will be indirectly impacted as well. However, when EDH removes an edge, most of the non-sensitive patterns can be discovered with the other edges. That is why, it shows the best performance with regard to the preservation of the non-sensitive patterns. The ratio changes almost from 1.5 to 2.5 for VMask2 and VMask3, and from 2 to 4 for VMask1 on the Chemical dataset. The ratio is approximately 2 for the disclosure thresholds between 7 and 9 and decreases as the disclosure threshold gets closer to 12 on the Movielens dataset. Next, the ratio is roughly 3 for VMask2 and VMask3 and increases to almost 4 for VMask1 on the NCI109 dataset. Moreover, as shown in the figures, when the other heuristics are compared, VMask2 and VMask3 have similar information loss, and VMask1 has the worst result.

Blocking the labels of the vertices in the sensitive graphs with a symbol that is not included in the vertex and edge labels results in many vertices having this symbol, and frequent patterns containing this symbol appear after sanitization. Figure 8 depicts these artifact patterns of the algorithms for different disclosure thresholds on the datasets. It can be viewed from the figures that EDH does not produce artifact patterns, whereas the masking-based heuristics cause them since EDH hides sensitive subgraphs by deleting the chosen edges so it does not use a masking symbol. Therefore, no new patterns appear after the hiding operation. For the other algorithms, the number of artifact patterns decreases as the disclosure threshold increases because a smaller number of masking symbols is inserted due to sanitizing less number of graphs.

The distance between the original database and the sanitized database should be minimal and they should be similar. The distance is found by counting the total number of altered vertices and edges. That is, in blocking-based approaches, it is the total number of masked vertices, and in our approach, it is the total number of deleted edges. Figure 9 presents the distance of VMask1, VMask2, VMask3, and EDH for the datasets, which are less in the higher disclosure thresholds because of sanitizing fewer graphs. Although the values are close to each other, deleting/masking an edge affects the graph structure to a smaller extent than that of a vertex. Additionally, the values for the Movielens dataset are the same for all the algorithms.

## 5 Related work

This section examines the related work. The focus of the study in our paper is somewhat related to (i) anonymization on graphs, (ii) community hiding, and (iii) frequent pattern hiding.

Social networks have attracted a great number of users with advanced features. The privacy of the users of social network data is an important concern. The threats for the users are classified into three categories [30]: (1) identity disclosure, (2) link disclosure, and (3) attribute disclosure. To protect the users' information, anonymization is done before publishing the data. Only changing the users' identifiable information like name, email address or phone numbers with random information does not guarantee privacy, which is known as naive

anonymization, since an adversary can utilize structural information of the graph for re-identifying the users from the anonymized graph [31]. An adversary can use background knowledge about the users to perform an attack on the released graph, such as structural information (degree, neighborhood), external auxiliary information, subgraphs, etc. [32].

Anonymization on social network data problem can be defined as: Given a social network graph and adversary background knowledge, create an anonymized form of the graph, which allows the data analysis while preserving the privacy of the users. There are various privacy-preserving techniques for graph anonymization. They can be primarily classified into four approaches: (i) clustering (generalization) based approaches [33], [34], (ii) differential privacy-based approaches [35], (iii) randomization (random graph editing) approaches [36], [37], and (iv) constrained approaches [38], [39], [40], [41]. Further, the last class, called k-anonymity, can also be categorized as degree, neighborhood, and subgraph anonymization.

Community detection helps us understand the network structure and discover meaningful communities. But, it raises privacy issues such as revealing of sensitive community information, such as undercover police community. Hence, hiding a target community in other communities of a network to avoid it from being extracted by a community detection algorithm is critical, [42]. According to [43], the indicators of good hiding of a target community C are: (1) reachability, the members of C are still in the same connected component, (2) spreading, the members of C spread into as many communities as possible, and (3) hiding the members of C in the largest communities.

The study [44], offers a heuristic algorithm DICE (randomly disconnecting d links internally and connecting b − d links externally with a budget b) and uses a measure of concealment to understand how well a community is concealed. [43], introduces community safety to determine the least safe members and update their links. This approach does not need global knowledge of communities. They propose a greedy algorithm based on community safety that enables hiding by rewiring links with a budget. [42], devises a new safeness function, and their algorithm Hs hides a target community through a certain number of link perturbations. It deletes intra-C links (links within the community) and adds inter-C links (links between communities). To choose the most proper links, the safeness gain function is employed. Besides, methods based on graph neural networks have been proposed. [45], generates adversarial graphs by attacking the deep graph-based model in order to hide targeted individuals in a community. A graph auto-encoder is proposed to hide the global community structure in [46]. Further, hiding against overlapping community detection is considered and an algorithm to hide a node in an overlapping area is developed in [47].

Before moving to subgraph hiding, we investigate the studies on association rule hiding and frequent itemset hiding to comprehend also the tabular data solutions. The approaches can be divided into five classes: (1) Border-based Approaches, [48], [49], [50], [51] focus on modifying the borders in the lattice of frequent and non-frequent itemsets of the dataset. (2) Exact Approaches, [52], [53], [54] express the hiding problem as a constraint satisfaction problem that is solved by integer programming. In spite of ensuring optimal solutions, if exist, they need high computations. (3) Reconstruction Based Approaches [55], [56] re-generate the sanitized database from the non-sensitive frequent itemsets. (4) Evolutionary Approaches [57], [58], [59] have been proposed for sensitive pattern-hiding problems using genetic algorithms, particle swarm optimization-based, and ant colony system-based algorithms. (5) Heuristic Approaches contain the fast and efficient algorithms. They have two particular subgoals during the hiding process that are hiding as many sensitive rules as possible, and minimizing the side effects, but they do not ensure optimality. Two types of heuristic approaches are distortion-based [23], [60], [61], [62], [63], [64] and blocking-based approaches [65], [66]. Blocking-based approaches replace the real values (0 or 1) of items with the unknowns (i.e., question marks) in the chosen transactions, but they do not insert false information into the database. Distortion-based approaches place 0s to 1s or vice versa in the chosen transactions.

When we turn our attention to subgraph hiding, the problem is introduced in [16]. The authors develop blocking based heuristic algorithms that mask the vertices of the selected graphs to hide sensitive subgraphs. These algorithms can hide all the sensitive subgraphs, but they produce fake subgraphs, they affect all the edges of masked vertices, and they might cause privacy breaches.

# 6 Conclusion

The privacy of sensitive subgraphs is an important issue when the data is published. Thus, they need to be hidden before releasing the data. The most related study is based on the blocking approach that

may cause side effects and privacy breaches. In this study, we offer an edge deletion-based subgraph hiding algorithm called EDH. Then, we compare our algorithm with the blocking-based algorithms on three real datasets for varying disclosure thresholds. The measures used for comparison are execution time, information loss, artifact patterns, and distance. According to the experimental results, while giving similar execution results, the EDH algorithm achieves fewer side effects.

In the future work, we plan to extend the EDH for hiding multiple sensitive subgraphs by deleting overlapping edges of sensitive subgraphs in order to minimize side effects. Another axis of future work can be to demonstrate the impact of subgraph hiding in a specific problem like a community detection attack.

*References:*

[1] X. Kong, W. Huang, Z. Tan, and Y. Liu, "Molecule generation by principal subgraph mining and assembling," *Advances in Neural Information Processing Systems*, vol. 35, New Orleans, Louisiana, USA, pp. 2550–2563, 2022, https://doi.org/10.48550/arXiv.2106.15098.

[2] F. C. Queiroz, A. M. Vargas, M. G. Oliveira, G. V. Comarela, and S. A. Silveira, "ppigremlin: a graph mining based detection of conserved structural arrangements in protein-protein interfaces," *BMC bioinformatics*, vol. 21, pp. 1–25, 2020, https://doi.org/10.1186/s12859-020-3474-1.

[3] A. Mrzic, P. Meysman, W. Bittremieux, P. Moris, B. Cule, B. Goethals, and K. Laukens, "Grasping frequent subgraph mining for bioinformatics applications," *BioData mining*, vol. 11, pp. 1–24, 2018, https://doi.org/10.1186/s13040-018-0181-9.

[4] L. Li, P. Ding, H. Chen, and X. Wu, "Frequent pattern mining in big social graphs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 3, pp. 638–648, 2021. 10.1109/TETCI.2021.3067017.

[5] L. Potin, R. Figueiredo, V. Labatut, and C. Largeron, "Pattern mining for anomaly detection in graphs: Application to fraud in public procurement," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Turin, Italy, 2023, pp. 69–87. https://doi.org/10.1007/978-3-031-43427-3_5.

[6] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, Halifax, NS, Canada, 2017, pp. 555–564. https://doi.org/10.1145/3097983.3098069.

[7] T. Ma, Q. Pan, H. Wang, W. Shao, Y. Tian, and N. Al- Nabhan, "Graph classification algorithm based on graph structure embedding," *Expert Systems with Applications*, vol. 161, p. 113715, 2020. https://doi.org/10.1016/j.eswa.2020.113715.

[8] L. Wang, F. V. Lin, M. Cole, and Z. Zhang, "Learning clique subgraphs in structural brain network classification with application to crystallized cognition," *NeuroImage*, vol. 225, p. 117493, 2021. https://doi.org/10.1016/j.neuroimage.2020.117493.

[9] P.-Z. Li, L. Huang, C.-D. Wang, and J.-H. Lai, "Edmot: An edge enhancement approach for motif-aware community detection," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, Anchorage, AK, USA, 2019, pp. 479–487. https://doi.org/10.1145/3292500.3330882.

[10] J. Ma and J. Fan, "Local optimization for clique-based overlapping community detection in complex networks," *IEEE Access*, vol. 8, pp. 5091–5103, 2019. 10.1109/ACCESS.2019.2962751.

[11] M. Kiabod, M. N. Dehkordi, and B. Barekatain, "A fast graph modification method for social network anonymization," *Expert Systems with Applications*, vol. 180, p. 115148, 2021. https://doi.org/10.1016/j.eswa.2021.115148.

[12] H. Zhang, L. Lin, L. Xu, and X. Wang, "Graph partition based privacy-preserving scheme in social networks," *Journal of Network and Computer Applications*, vol. 195, p. 103214, 2021. https://doi.org/10.1016/j.jnca.2021.103214.

[13] J. Medková and J. Hynek, "Hakau: hybrid algorithm for effective k-automorphism anonymization of social networks," *Social Network Analysis and Mining*, vol. 13, no. 1, p. 63, 2023. https://doi.org/10.1007/s13278-023-01064-1.

[14] H. Kaur, N. Hooda, and H. Singh, "k-anonymization of social network data using neural network and svm: K-neurosvm," *Journal of Information Security and*

*Applications*, vol. 72, p. 103382, 2023. https://doi.org/10.1016/j.jisa.2022.103382.

[15] G. Stromire and I. Potoczny-Jones, "Empowering smart cities with strong cryptography for data privacy," in *Proceedings of the 1st ACM/EIGSCC Symposium on Smart Cities and Communities*, Portland, OR, USA, 2018, pp. 1–7. https://doi.org/10.1145/3236461.3241975.

[16] O. Abul and H. Gökçe, "Knowledge hiding from tree and graph databases," *Data & Knowledge Engineering*, vol. 72, pp. 148–171, 2012. https://doi.org/10.1016/j.datak.2011.10.002.

[17] V. S. Verykios, E. D. Pontikakis, Y. Theodoridis, and L. Chang, "Efficient algorithms for distortion and blocking techniques in association rule hiding," *Distributed and Parallel Databases*, vol. 22, pp. 85–104, 2007. https://doi.org/10.1007/s10619-007-7013-0.

[18] L. Yuan, D. Yan, W. Qu, S. Adhikari, J. Khalil, C. Long, and X. Wang, "T-fsm: A task-based system for massively parallel frequent subgraph pattern mining from a big graph," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–26, 2023. https://doi.org/10.1145/3588928.

[19] T. Fu, C. Wei, Y. Wang, and R. Ying, "Desco: Towards generalizable and scalable deep subgraph counting," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, Merida, Mexico, 2024, pp. 218–227. https://doi.org/10.1145/3616855.3635788.

[20] K. Nakamura, M. Nishino, N. Yasuda, and S.-i. Minato, "Compdp: A framework for simultaneous subgraph counting under connectivity constraints," in *21st International Symposium on Experimental Algorithms (SEA 2023)*, Barcelona, 2023. https://doi.org/10.4230/LIPIcs.SEA.2023.11.

[21] X. Jian, Z. Li, and L. Chen, "Suff: accelerating subgraph matching with historical data," *Proceedings of the VLDB Endowment*, vol. 16, no. 7, pp. 1699–1711, 2023. https://doi.org/10.14778/3587136.3587144.

[22] T. Jin, B. Li, Y. Li, Q. Zhou, Q. Ma, Y. Zhao, H. Chen, and J. Cheng, "Circinus: Fast redundancy-reduced subgraph matching," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–26, 2023. https://doi.org/10.1145/3588692.

[23] S. R. Oliveira and O. R. Zaiane, "Privacy preserving frequent itemset mining," in *Proceedings of the IEEE international conference on Privacy, security and data mining*, Maebashi City, Japan, vol. 14, 2002, pp. 43–54.

[24] A. Gkoulalas-Divanis and G. Loukides, "Revisiting sequential pattern hiding to enhance utility," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, San Diego, California, USA, 2011, pp. 1316–1324. https://doi.org/10.1145/2020408.2020605.

[25] A. Srinivasan, R. D. King, S. H. Muggleton, and M. J. E. Sternberg, "The predictive toxicology evaluation challenge," in *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, vol. 1, 1997, pp. 4–9.

[26] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011)," in *Proceedings of the fifth ACM conference on Recommender Systems*, Chicago, Illinois, USA, 2011, pp. 387–388, https://doi.org/10.1145/2043932.2044016.

[27] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, and D. Phung, "Learning graph representation via frequent subgraphs," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, San Diego, California, USA, 2018, pp. 306–314, https://doi.org/10.1137/1.9781611975321.35.

[28] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings*, Maebashi City, Japan, 2002, pp. 721– 724, DOI: 10.1109/ICDM.2002.1184038.

[29] 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*, vol. 26, no. 10, pp. 1367–1372, 2004, DOI: 10.1109/TPAMI.2004.75

[30] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, Vancouver, Canada, 2008, pp. 93-106, https://doi.org/10.1145/1376616.1376629.

[31] 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*, Banff, Alberta, Canada, 2007, pp. 181–190, https://doi.org/10.1145/1242572.1242598.

[32] J. H. Abawajy, M. I. H. Ninggal, and T. Herawan, "Privacy preserving social network data publication," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 1974–1997, 2016, 10.1109/COMST.2016.2533668.

[33] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 102–114, 2008. https://doi.org/10.14778/1453856.1453873.

[34] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Class-based graph anonymization for social network data," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 766–777, 2009, https://doi.org/10.14778/1687627.1687714.

[35] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *Theory of Cryptography Conference*, Tokyo, Japan, 2013, pp. 457–476. https://doi.org/10.1007/978-3-642-36594-2_26.

[36] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing social networks," *Computer science department faculty publication series*, p. 180, 2007.

[37] J. Casas-Roma, "Privacy-preserving on graphs using randomization and edge-relevance," in *International Conference on Modeling Decisions for Artificial Intelligence*, Tokyo, Japan, 2014, pp. 204–216, https://doi.org/10.1007/978-3-319-12054-6_18.

[38] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra, "k-degree anonymity and edge selection: improving data utility in large networks," *Knowledge and Information Systems*, vol. 50, no. 2, pp. 447–474, 2017, https://doi.org/10.1007/s10115-016-0947-7.

[39] M. Kiabod, M. N. Dehkordi, and B. Barekatain, "Tsram: A time-saving k-degree anonymization method in social network," *Expert Systems with Applications*, vol. 125, pp. 378–396, 2019, https://doi.org/10.1016/j.eswa.2019.01.059.

[40] X. Ding, C. Wang, K.-K. R. Choo, and H. Jin, "A novel privacy preserving framework for large scale graph data publishing," *IEEE transactions on knowledge and data engineering*, 2019. DOI: 10.1109/TKDE.2019.2931903

[41] K. Huang, H. Hu, S. Zhou, J. Guan, Q. Ye, and X. Zhou, "Privacy and efficiency guaranteed social subgraph matching," *The VLDB Journal*, pp. 1–22, 2022, https://doi.org/10.1007/s00778-021-00706-0.

[42] X. Chen, Z. Jiang, H. Li, J. Ma, and S. Y. Philip, "Community hiding by link perturbation in social networks," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 3, pp. 704–715, 2021, DOI: 10.1109/TCSS.2021.3054115.

[43] V. Fionda and G. Pirro, "Community deception or: How to stop fearing community detection algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 660–673, 2017. DOI: 10.1109/TKDE.2017.2776133.

[44] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 2, pp. 139– 147, 2018, https://doi.org/10.1038/s41562-017-0290-3.

[45] J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, and J. Huang, "Adversarial attack on community detection by hiding individuals," in *Proceedings of The Web Conference 2020*, Taipei, Taiwan, 2020, pp. 917–927, https://doi.org/10.1145/3366423.3380171.

[46] D. Liu, Z. Chang, G. Yang, and E. Chen, "Community hiding using a graph autoencoder," *Knowledge-Based Systems*, vol. 253, p. 109495, 2022, https://doi.org/10.1016/j.knosys.2022.109495.

[47] D. Liu, G. Yang, Y. Wang, H. Jin, and E. Chen, "How to protect ourselves from overlapping community detection in social networks," *IEEE Transactions on Big Data*, vol. 8, no. 4, pp. 894–904, 2022, DOI: 10.1109/TBDATA.2022.3152431.

[48] X. Sun and P. S. Yu, "A border-based approach for hiding sensitive frequent itemsets," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, Houston, TX, USA, 2005, pp. 8–pp. 10.1109/ICDM.2005.2.

[49] G. V. Moustakides and V. S. Verykios, "A maxmin approach for hiding frequent itemsets," *Data & Knowledge Engineering*, vol. 65, no. 1, pp. 75–89, 2008, https://doi.org/10.1016/j.datak.2007.06.012.

[50] S. Sharma and D. Toshniwal, "Mr-i maxmin-scalable twophase border based knowledge hiding technique using mapreduce," *Future Generation Computer Systems*, vol. 109, pp. 538–550, 2020, https://doi.org/10.1016/j.future.2018.05.063.

[51] P. Krasadakis, G. Futia, V. S. Verykios, and E. Sakkopoulos, "Graph based hiding of sensitive knowledge," in *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*. Atlanta, GA, USA, 2023, pp. 199–203.10.1109/ICTAI59109.2023.00037.

[52] S. Menon, S. Sarkar, and S. Mukherjee, "Maximizing accuracy of shared databases when concealing sensitive patterns," *Information Systems Research*, vol. 16, no. 3, pp. 256–270, 2005, https://doi.org/10.1287/isre.1050.0056.

[53] A. Gkoulalas-Divanis and V. S. Verykios, "Hiding sensitive knowledge without side effects," *Knowledge and Information Systems*, vol. 20, no. 3, pp. 263–299, 2009, https://doi.org/10.1007/s10115-008-0178-7.

[54] V. S. Verykios, E. C. Stavropoulos, P. Krasadakis, and E. Sakkopoulos, "Frequent itemset hiding revisited: pushing hiding constraints into mining," *Applied Intelligence*, vol. 52, no. 3, pp. 2539–2555, 2022, https://doi.org/10.1007/s10489-021-02490-4.

[55] Y. Guo, "Reconstruction-based association rule hiding," in *Proceedings of SIGMOD2007 Ph.D. Workshop on Innovative Database Research*, Beijing, 2007, pp. 51–56.

[56] S. Li, N. Mu, J. Le, and X. Liao, "Privacy preserving frequent itemset mining: Maximizing data utility based on database reconstruction," *Computers & Security*, vol. 84, pp. 17–34, 2019, https://doi.org/10.1016/j.cose.2019.03.008.

[57] C.-W. Lin, T.-P. Hong, K.-T. Yang, and S.-L. Wang, "The ga-based algorithms for optimizing hiding sensitive itemsets through transaction deletion," *Applied Intelligence*, vol. 42, pp. 210–230, 2015, https://doi.org/10.1007/s10489-014-0590-5.

[58] J. C.-W. Lin, Q. Liu, P. Fournier-Viger, T.-P. Hong, M. Voznak, and J. Zhan, "A sanitization approach for hiding sensitive itemsets based on particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 53, pp. 1–18, 2016, https://doi.org/10.1016/j.engappai.2016.03.007.

[59] A. Telikani, A. H. Gandomi, A. Shahbahrami, and M. N. Dehkordi, "Privacy-preserving in association rule mining using an improved discrete binary artificial bee colony," *Expert Systems with Applications*, vol. 144, p. 113097, 2020, https://doi.org/10.1016/j.eswa.2019.113097.

[60] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino, "Hiding association rules by using confidence and support," in *International Workshop on Information Hiding*, Pittsburgh, PA, USA, 2001, pp. 369–383,M https://doi.org/10.1007/3-540-45496-9_27.

[61] S. R. Oliveira and O. R. Zaiane, "Protecting sensitive knowledge by data sanitization," in *Third IEEE International conference on data mining*, Melbourne, FL, USA, 2003, pp. 613–616. 10.1109/ICDM.2003.1250990.

[62] A. Amiri, "Dare to share: Protecting sensitive knowledge with data sanitization," Decision Support Systems, vol. 43, no. 1, pp. 181–191, 2007, https://doi.org/10.1016/j.dss.2006.08.007.

[63] T.-P. Hong, C.-W. Lin, K.-T. Yang, and S.-L. Wang, "Using tf-idf to hide sensitive itemsets," *Applied intelligence*, vol. 38, pp. 502–510, 2013, https://doi.org/10.1007/s10489-012-0377-5.

[64] P. Cheng, J. F. Roddick, S.-C. Chu, and C.-W. Lin, "Privacy preservation through a greedy, distortion-based rule-hiding method," *Applied Intelligence*, vol. 44, pp. 295–306, 2016, https://doi.org/10.1007/s10489-015-0671-0.

[65] Y. Saygin, V. S. Verykios, and C. Clifton, "Using unknowns to prevent discovery of association rules," *ACM Sigmod Record*, vol. 30, no. 4, pp. 45–54, 2001, https://doi.org/10.1145/604264.604271.

[66] S.-L. Wang and A. Jafari, "Using unknowns for hiding sensitive predictive association rules," in *IRI-2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005*. IEEE, Las Vegas, NV, USA, 2005, pp. 223–228, DOI:10.1109/IRI-05.2005.1506477.

**Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)**

- Leyla Tekin: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization.
- Belgin Ergenç Bostanoğlu: Conceptualization, Formal analysis, Writing - Review & Editing, Supervision.

**Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself**

No funding was received for conducting this study.

**Conflict of Interest**

The authors have no conflicts of interest to declare.