Noname manuscript No.
(will be inserted by the editor)

A Practical Approximation Algorithm for Optimal k-Anonymity

Batya Kenig · **Tamir Tassa**

Received: date / Accepted: date

Abstract k-Anonymity is a privacy preserving method for limiting disclosure of private information in data mining. The process of anonymizing a database table typically involves generalizing table entries and, consequently, it incurs loss of relevant information. This motivates the search for anonymization algorithms that achieve the required level of anonymization while incurring a minimal loss of information. The problem of k-anonymization with minimal loss of information is NP-hard. We present a practical approximation algorithm that enables solving the k-anonymization problem with an approximation guarantee of $O(\ln k)$. That algorithm improves an algorithm due to Aggarwal et al. [1] that offers an approximation guarantee of $O(k)$, and generalizes that of Park and Shim [19] that was limited to the case of generalization by suppression. Our algorithm uses techniques that we introduce herein for mining closed frequent generalized records. Our experiments show that the significance of our algorithm is not limited only to the theory of k-anonymization. The proposed algorithm achieves lower information losses than the leading approximation algorithm, as well as the leading heuristic algorithms. A modified version of our algorithm that issues ℓ-diverse k-anonymizations also achieves lower information losses than the corresponding modified versions of the leading algorithms.

Keywords privacy-preserving data mining · k-anonymity · ℓ-diversity · approximation algorithms for NP-hard problems · frequent generalized itemsets

Batya Kenig
Division of Computer Science
The Open University
Ra'anana, Israel

Tamir Tassa
Division of Computer Science
The Open University
Ra'anana, Israel
Tel.: +972-9-7781272
Fax: +972-9-7780605
E-mail: tamirta@openu.ac.il

1 Introduction

In recent years, there has been a tremendous growth in the amount of personal data that can be collected and analyzed. Data mining tools are increasingly being used to infer trends and patterns. Of particular interest are data containing structured information on individuals. However, the use of data containing personal information has to be restricted in order to protect individual privacy. Although identifying attributes like ID numbers and names are never released for data mining purposes, sensitive information might still leak due to linking attacks that are based on the public attributes, a.k.a *quasi-identifiers*. Such attacks may join the quasi-identifiers of a published table with a publicly accessible table like the voters registry, and thus disclose private information of specific individuals. In fact, it was shown in [24] that 87% of the U.S. population may be uniquely identified by the combination of the three quasi-identifiers: birthdate, gender and zipcode. Privacy preserving data mining [3] has been proposed as a paradigm of exercising data mining while protecting the privacy of individuals.

One of the most well-studied methods of privacy preserving data mining is k-anonymization, that was proposed by Samarati and Sweeney [21,22,25]. This method suggests to generalize the values of the public attributes, so that each of the released records becomes indistinguishable from at least $k-1$ other records, when projected on the subset of public attributes. As a consequence, each individual may be linked to sets of records of size at least k in the released anonymized table, whence privacy is protected to some extent.

The values of the database are modified via the operation of generalization, while keeping them consistent with the original ones. A cost function is used to measure the amount of information that is lost by the generalization process. The objective is to modify the table entries so that the table becomes k-anonymous and the information loss (or cost function) is minimized.

Meyerson and Williams [17] introduced this problem and studied it under the assumption that the table entries may be either left unchanged or totally suppressed. In that setting, the cost function to be minimized is the total number of suppressed entries in the table. They proved that the problem is NP-hard by showing a reduction from the k-dimensional perfect matching problem. They devised two approximation algorithms: One that runs in time $O(n^{2k})$ and achieves an approximation ratio of $O(k \ln k)$; and another that has a fully polynomial running time (namely, it depends polynomially on both n and k) and guarantees an approximation ratio of $O(k \ln n)$.

Aggarwal et al. [1] extended the setting of suppressions-only to generalizations by hierarchical clustering trees. In that setting, each attribute A_j has a corresponding balanced tree, $\mathcal{T}(A_j)$, that describes a hierarchical clustering (or taxonomy) of A_j. Each node of $\mathcal{T}(A_j)$ represents a subset of A_j, the root of the tree is the entire set A_j, the descendants of each node represent a partition of the subset that corresponds to the ancestor node, and the leaves correspond to the singleton subsets. Given such a balanced tree, they considered generalization operators that may replace an entry in the jth column with any of its ancestors in $\mathcal{T}(A_j)$. Generalization by suppression is a special case of generalization by clustering trees where all trees are of height 2. They suggested a simple cost function for that setting and then they devised an approximation algorithm with an approximation ratio of $O(k)$.

Gionis and Tassa [8] improved the first algorithm of [17] by offering an approximation ratio of $O(\ln k)$, rather than $O(k \ln k)$, and applying it to a wider class of "proper" generalization operators (See Definition 3 herein). That class extends the framework of hierarchical clustering trees [1] as it allows also unbalanced clustering trees. The results of [8] also apply to a wider class of measures of loss of information. However, the runtime of their algorithm remains $O(n^{2k})$.

Finally, Park and Shim [19] devised, independently of [8], a similar algorithm that also provides an approximation ratio of $O(\ln k)$. Their algorithm, as opposed to the one in [8], has a practical runtime, but it applies only to generalizations by suppression, a case which is too restricted for practical applications.

Another approach to the problem is using heuristical algorithms. There is a multitude of such algorithms, e.g. the Mondrian algorithm [14] or k-member clustering [5], but the most prominent heuristical algorithms that were shown to produce the best results in terms of information loss are the agglomerative algorithm [7,18], sequential clustering [9], and algorithms that are based on space-filling curves [6].

1.1 Our contributions

The main contribution of this study is a practical anonymization algorithm that guarantees an approximation ratio of $O(\ln k)$ and applies to all proper generalizations and to any monotone measure of loss of information. Our algorithm is based on the algorithm of Park and Shim which was restricted to suppressions only. The extension from suppressions only to any proper generalization is based on techniques that we devise herein for mining generalized frequent records. Hence, the algorithm that is proposed here is the last step (for now) in the above described theoretical journey that was made in [17,1,8,19]. An interesting research problem that naturally arises in wake of this study, is whether this is indeed the last step, or is it possible to improve the logarithmic approximation ratio.

The significance of our algorithm is not limited to theory only. Our experiments on several databases using several measures of information loss show that the proposed approximation algorithm constantly achieves lower information losses than the currently best known approximation algorithm (the algorithm of [1]) and the above-mentioned heuristical algorithms. Hence, in terms of achieving minimal information losses, our algorithm appears to offer the best performance. However, the runtime of the algorithm is larger than that of the other algorithms, at least for smaller values of k, and it scales badly with respect to the dimension (the number of attributes). Hence, this disadvantage should be taken into consideration when choosing an anonymization algorithm. We believe that runtime is a secondary factor in this context, since in typical applications of k-anonymity, the input data is a result of collection efforts that span a very long time (months and even years). Hence, an anonymization algorithm that runs few seconds does not offer a meaningful advantage with respect to an algorithm that runs several hours or even several days. If the output of the slower algorithm provides anonymized tables with better utility, then that is the algorithm of choice. It should be noted that while the theoretical time complexity of our algorithm remains $O(n^{2k})$, like the algorithms of [17,8], its runtime in practice is much shorter; see more on that point in Section 4.

As k-anonymity must be enforced together with ℓ-diversity, we propose a general post-processing procedure that can be applied on the output of any k-anonymization algorithm in order to convert that output to an ℓ-diverse k-anonymization. Our experiments indicate that also after the application of that post-processing procedure of ℓ-diversification, the ℓ-diverse k-anonymizations issued by our algorithm are characterized by lower information losses than the corresponding anonymizations issued by other leading algorithms.

1.2 Organization of the paper

In Section 2 we provide the basic definitions and describe the problem of optimal k-anonymity. In Section 3 we describe the relevant related work; specifically, we discuss the three approximation algorithms of [1,8,19]. In the subsequent two sections we describe our main contribution: Section 4 is devoted to our general $O(\log k)$-approximation algorithm for the problem of k-anonymity; and in Section 5 we describe an algorithm for mining closed frequent generalized records, which is a key ingredient in the approximation algorithm. In Section 6 we discuss the notion of ℓ-diversity and the importance to enforce it in conjunction with k-anonymity; we then describe a post-processing procedure that can convert any k-anonymization to one that respects also ℓ-diversity. In Section 7 we describe our experiments that demonstrate the advantages of the proposed approximation algorithm. We conclude in Section 8 with suggested future research directions.

2 Preliminaries

We consider databases that hold information on individuals in some population U. Each individual is described by r public attributes (a.k.a quasi-identifiers), $A_1, \ldots, A_r$, and s private attributes, $Z_1, \ldots, Z_s$ (usually it is assumed that $s = 1$). Each of the attributes consists of several possible values:

$$A_j = \{a_{j,l} : 1 \le l \le m_j\}, \quad 1 \le j \le r,$$

and

$$Z_j = \{z_{j,l} : 1 \le l \le n_j\}, \quad 1 \le j \le s.$$

For example, if A_j is gender, then $A_j = \{M, F\}$, while if it is the age of the individual, it is a bounded non-negative natural number. The public database holds all publicly available information on the individuals in U; it takes the form

$$D = \{R_1, \ldots, R_n\}, \tag{1}$$

where $R_i \in A_1 \times \cdots \times A_r,\ 1 \le i \le n$. The corresponding private database holds the private information

$$D' = \{S_1, \ldots, S_n\}, \tag{2}$$

where $S_i \in Z_1 \times \cdots \times Z_s,\ 1 \le i \le n$. The complete database is the concatenation of those two databases, $D||D' = \{R_1||S_1, \ldots R_n||S_n\}$. We refer to the records of R_i and S_i, $1 \le i \le n$, as public and private records, respectively. The jth component of the record R_i (the (i, j)th entry in the database D) will be denoted $R_i(j)$.

2.1 Generalization

One of the means to anonymize a database is generalization; i.e., replacing the values that appear in the database with subsets of values, so that each entry $R_i(j)$, $1 \le i \le n$, $1 \le j \le r$, which is an element of A_j, is replaced by a subset of A_j that includes that element.

Definition 1 Let A_j, $1 \le j \le r$, be finite sets and let $\overline{A}_j \subseteq \mathcal{P}(A_j)$ be a collection of subsets of A_j. Let $D = \{R_1, \ldots, R_n\}$ be a table where each record R_i, $1 \le i \le n$, is taken from $A_1 \times \cdots \times A_r$. A table $\overline{D} = \{\overline{R}_1, \ldots, \overline{R}_n\}$ is a generalization of D, if $\overline{R}_i \in \overline{A}_1 \times \cdots \times \overline{A}_r$, and $R_i(j) \in \overline{R}_i(j)$, for all $1 \le i \le n$ and $1 \le j \le r$.

A special kind of generalization is generalization by suppression, where $\overline{A}_j = A_j \cup \{A_j\}$ for all $1 \leq j \leq r$. Namely, each entry is either left unchanged or is totally suppressed.

There are three main models of generalization. In *single-dimensional global recoding,* each collection of subsets $\overline{A}_j$ is a clustering of the set A_j (in the sense that it consists of disjoint subsets that cover A_j), and then every entry in the jth column of the database is mapped to the unique subset in $\overline{A}_j$ that contains it. As a consequence, every single value $a \in A_j$ is always generalized in the same manner. In *local recoding,* the collection of subsets $\overline{A}_j$ covers the set A_j but it is not a clustering. In that case, each entry in the table's jth column is generalized independently to one of the subsets in $\overline{A}_j$ that includes it. In such a model, if the age 34 appears in the table in several records, it may be left unchanged in some, generalized to 30–39, or totally suppressed in other records. Clearly, local recoding is more flexible and might enable k-anonymity with a smaller loss of information. The third model is an intermediate one and is called *multi-dimensional global recoding*. In that model, like in local recoding, the collection of subsets $\overline{A}_j$ is a cover of the set A_j (namely, each value of A_j may be contained in more than one subset in $\overline{A}_j$). However, it is a global recoding in the sense that there exists a global mapping function $g : A_1 \times \cdots \times A_r \to \overline{A}_1 \times \cdots \times \overline{A}_r$ and $\overline{D} = g(D)$.

In this study we consider the case of local recoding that allows greater flexibility and, hence, enables achieving k-anonymity with (possibly) smaller information losses. As mentioned before, the problem of k-anonymization with minimal loss of information is NP-hard in the case of local recoding. In the case of single-dimensional global recoding the search space is much smaller and the problem may be solved optimally [4,13].

Definition 2 A relation $\sqsubseteq$ is defined on $\overline{A}_1 \times \cdots \times \overline{A}_r$ as follows: If $R, R' \in \overline{A}_1 \times \cdots \times \overline{A}_r$, then $R \sqsubseteq R'$ if and only if $R(j) \subseteq R'(j)$ for all $1 \leq j \leq r$. In that case, we say that R narrows R', or equivalently, that R' generalizes R. Furthermore, $R' \sqsubset R$ means that $R' \sqsubseteq R$ and $R' \neq R$.

We will assume hereinafter that the collections of subsets used for generalization, $\overline{A}_j$, $1 \leq j \leq r$, satisfy the following property [8].

Definition 3 Given an attribute $A = \{a_1, \ldots, a_m\}$, a corresponding collection of subsets $\overline{A}$ is called proper if it includes all singleton subsets $\{a_i\}$, $1 \leq i \leq m$, it includes the entire set A, and it is laminar in the sense that $B_1 \cap B_2 \in \{\emptyset, B_1, B_2\}$ for all $B_1, B_2 \in \overline{A}$.

As shown in [8, Lemma 3.3], the class of proper generalizations coincides with the class of generalizations by possibly unbalanced hierarchical clustering trees. (Such a clustering tree, or a taxonomy, is illustrated in Figure 1.) Hence, our framework in this study extends the framework that was considered in [1] (i.e., balanced hierarchical clustering trees) and, in particular, the framework of generalization by suppression [17,19].

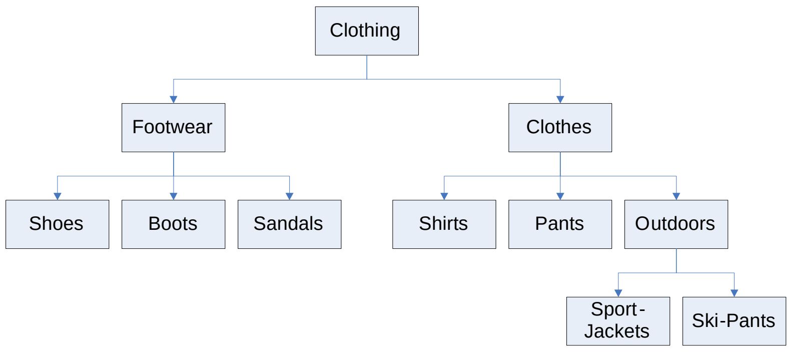


Fig. 1 A hierarchical clustering tree (taxonomy)

2.2 Measures of information loss

The generalization of a record R into some generalized record $\overline{R}$ has an associated cost in terms of information loss, denoted $d(\overline{R})$. There are several ways of defining that cost; e.g., the tree measure [1], the Loss Metric (LM) [12,18], the Ambiguity Metric (AM) [18], or the entropy measure [8]. For example, the LM measure is defined as follows:

$$d(\overline{R}) = \frac{1}{r} \cdot \sum_{j=1}^{r} \frac{|\overline{R}(j)| - 1}{|A_j| - 1} \,. \tag{3}$$

If $g(D) = \{\overline{R}_1, \ldots, \overline{R}_n\}$ is a generalization of $D = \{R_1, \ldots, R_n\}$, then the overall generalization cost is

$$\Pi(D, g(D)) := \frac{1}{n} \cdot \sum_{i=1}^{n} d(\overline{R}_i) \,.$$

In this study we do not focus on any particular measure. We only assume that the measure takes the form $d(\overline{R}_i) = \sum_{j=1}^{r} d(\overline{R}_i(j))$ (namely, it associates a generalization cost in each dimension and then adds them up), and that it is monotone:

Definition 4 A measure of information loss, $d(\cdot)$, is called monotone if for any two generalized records $\overline{R}, \overline{R}' \in \overline{A}_1 \times \cdots \times \overline{A}_r$, where $\overline{R}'$ generalizes $\overline{R}$, it holds that $d(\overline{R}') \geq d(\overline{R})$.

2.3 k-Anonymization

A k-anonymization of a database $D = \{R_1, \ldots, R_n\}$ is a generalization $g(D) = \{\overline{R}_1, \ldots, \overline{R}_n\}$ where for all $1 \leq i \leq n$ there exist indices $1 \leq i_1 < i_2 < \cdots < i_{k-1} \leq n$, all of which are different from i, such that $\overline{R}_i = \overline{R}_{i_1} = \cdots = \overline{R}_{i_{k-1}}$. The k-anonymization optimization problem is defined as follows [1,8,17].

Definition 5 Let $D = \{R_1, \ldots, R_n\}$ be a database with public attributes A_j, $1 \leq j \leq r$. Given collections $\overline{A}_j \subseteq \mathcal{P}(A_j)$, and a measure of information loss Π, find a corresponding k-anonymization, $g(D) = \{\overline{R}_1, \ldots, \overline{R}_n\}$, where $\overline{R}_i \in \overline{A}_1 \times \cdots \times \overline{A}_r$, that minimizes $\Pi(D, g(D))$.

3 Previous Approximation Algorithms for k-Anonymization

3.1 Overview

There are three leading approximation algorithms for the problem of k-anonymization with minimal loss of information.

– Algorithm A: The algorithm by Gionis and Tassa [8] achieves an approximation ratio of $O(\ln k)$. It is currently the best available approximation algorithm in terms of its approximation guarantee. However, it is impractical since its runtime is $O(n^{2k})$.
– Algorithm B: Park and Shim [19] proposed a similar algorithm to that of [8]. It offers the same approximation guarantee, and it has a practical runtime. However, it is restricted to the case of generalization by suppression.
– Algorithm C: The Forest algorithm of Aggarwal et al. [1] runs in fully polynomial time and guarantees an approximation ratio of $O(k)$. It is the best approximation algorithm that is both practical and general (i.e., not restricted to generalization by suppression).

Our contribution in this study is an approximation algorithm that:

– guarantees an approximation ratio of $O(\ln k)$ (like Algorithms A and B above);
– is practical (like Algorithms B and C);
– applies to all proper generalization operators and monotone measures of information loss (like Algorithm A).

Our proposed algorithm relies on the same approach as Algorithms A and B. In this section we describe those two algorithms (Algorithm A is described in Section 3.4 and Algorithm B is described in Section 3.5). Our description somewhat differs from that in [8, 19] as our goal here is not only to describe those algorithms but also to set the grounds for presenting our algorithm in the next section.

3.2 Generalization cost of sets of records

Let $M \subseteq D$ be a subset of records. The *closure* of M, denoted $\overline{M}$, is the minimal generalized record that generalizes all records in M,

$$\overline{M} = \min_{\sqsubseteq} \left\{ Q \in \overline{A}_1 \times \ldots \times \overline{A}_r : R \sqsubseteq Q \ \text{ for all } R \in M \right\} .$$

The corresponding generalization cost of M, denoted $d(M)$, is defined as the generalization cost of the closure, i.e., $d(M) = d(\overline{M})$.

The following lemma introduces the property of sub-additivity, which will be used later in our analysis of the algorithms. (That lemma was proved in [8] for two special cases of generalization cost measures.)

Lemma 1 *Assume that all collections of subsets, $\overline{A}_j$, $1 \leq j \leq r$, are proper. Then the generalization cost is sub-additive in the sense that for all $S, T \subseteq A_1 \times \ldots \times A_r$,*

$$S \cap T \neq \emptyset \Longrightarrow d(S \cup T) \leq d(S) + d(T) .$$

Proof Denote $U = S \cup T$ and let

$$S_j = \{s(j) : s \in S\}, \quad T_j = \{t(j) : t \in T\}, \quad U_j = \{u(j) : u \in U\}$$

denote the set of values of the jth attribute, $1 \leq j \leq r$, that appear in S, T, and U, respectively. Let $\overline{S}_j$, $\overline{T}_j$ and $\overline{U}_j$ be the minimal sets in $\overline{A}_j$ that include S_j, T_j and U_j, respectively. Since $S \cap T \neq \emptyset$, we conclude that $S_j \cap T_j \neq \emptyset$. Hence $\overline{S}_j \cap \overline{T}_j \neq \emptyset$. But since $\overline{A}_j$ is proper, we have that $\overline{S}_j \subseteq \overline{T}_j$ or $\overline{T}_j \subseteq \overline{S}_j$. Therefore, $\overline{U}_j = \overline{S}_j$ or $\overline{U}_j = \overline{T}_j$. Hence, since d is monotone, whence it attains only non-negative values, we infer that $d(\overline{U}_j) \leq d(\overline{S}_j) + d(\overline{T}_j)$ for all $1 \leq j \leq r$. Adding up the last inequalities, we arrive at the conclusion that $d(S \cup T) \leq d(S) + d(T)$. $\square$

3.3 Anonymizations, covers, and clusterings

Any k-anonymization of D defines a clustering of D where each cluster consists of all records that were replaced by the same generalized record. Given a k-anonymization of D, we can assume that all of its induced clusters are of size between k and $2k-1$. Indeed, if there exists a cluster of size greater than $2k-1$, we may arbitrarily split it into two clusters of size at least k without increasing the overall information loss, as implied by the monotonicity of the cost measure.

Definition 6 An $[\ell, m]$-cover is a cover γ of D by subsets $S \subset D$ of size $\ell \leq |S| \leq m$. An $[\ell, m]$-clustering is an $[\ell, m]$-cover where all subsets are disjoint.

Let $\Gamma_{[k,2k-1]}$ be the set of all $[k, 2k-1]$-covers of D and let $P_{[k,2k-1]}$ be the subset of all $[k, 2k-1]$-clusterings. As discussed above, any optimal k-anonymization corresponds to a clustering in $P_{[k,2k-1]}$.

Given a $[k, 2k-1]$-cover $\gamma \in \Gamma_{[k,2k-1]}$ of the database D, we define its generalization cost as:

$$d(\gamma) = \sum_{S \in \gamma} d(S)\,. \tag{4}$$

Furthermore, if $\gamma \in P_{[k,2k-1]}$ we define its anonymization cost as

$$ANON(\gamma) = \sum_{S \in \gamma} |S| \cdot d(S)\,. \tag{5}$$

If $g(D)$ is the k-anonymization that corresponds to the $[k, 2k-1]$-clustering γ, then $\Pi(D, g(D)) = \frac{1}{n} ANON(\gamma)$.

Given a database D and a positive integer k, we consider two optimization problems on $P_{[k,2k-1]}$. The first one is the $[k, 2k-1]$-minimum clustering problem, in which we look for $\gamma \in P_{[k,2k-1]}$ that minimizes $d(\gamma)$. The second one is the k-anonymization problem, in which we look for $\gamma \in P_{[k,2k-1]}$ that minimizes $ANON(\gamma)$. The following theorem [8, 17] asserts that given an α-approximation algorithm for the $[k, 2k-1]-$minimum clustering problem, the k-anonymization problem can be approximated to within a factor of 2α.

Theorem 1 *Let $\alpha \geq 1$, and let γ be a $[k, 2k-1]$-clustering with cost at most α times that of an optimal solution to the $[k, 2k-1]$-minimum clustering problem. Then if we replace each record $R \in D$ with the closure of the cluster in γ to which it belongs, we obtain a $2\alpha-$approximation to the optimal k-anonymization problem.*

3.4 The basic $O(\ln k)$-approximation algorithm

We now proceed to describe the approximation algorithm of [8] to the problem of k-anonymization with minimal loss of information. The two main procedures in that algorithm are described in Algorithms 1 and 2. Algorithm 1, which will be referred to hereinafter as GEN-COVER, is the well-known greedy algorithm for approximating the weighted set cover problem. It receives a database D and a collection C of subsets of D and outputs a cover of D with subsets from C, that approximates an optimal cover to within $O(\ln \kappa(C))$, where $\kappa(C) := \max\{|S|, S \in C\}$. The approximation algorithm for the optimal k-anonymity problem, Algorithm 3 (k-ANON hereinafter), starts by invoking GEN-COVER with the special collection

$$C = F_{[k,2k-1]} := \{S \subset D : k \le |S| \le 2k-1\}\,. \tag{6}$$

Consequently, the resulting cover, γ, is an $O(\ln k)$-approximation to the problem of optimal $[k, 2k-1]$-cover. In the second phase, Algorithm 3 invokes Algorithm 2 which translates the cover γ to a $[k, 2k-1]$-clustering γ^0. Finally, that clustering is translated into the corresponding k-anonymization of D.

Algorithm 1 GEN-COVER
A naïve greedy approximation to an optimal cover

Input: Table D, a collection of subsets $C \subseteq \mathcal{P}(D)$.
Output: Cover γ of D with subsets from C.

```
 1: Set $\gamma = \emptyset$ and $E = \emptyset$.
 2: while $E \neq D$ do
 3:    for all $S \in C$ do
 4:       Compute the ratio $\rho(S) = \frac{d(S)}{|S \cap (D \setminus E)|}$ .
 5:    end for
 6:    Choose $S$ that minimizes $\rho(S)$.
 7:    $E = E \cup S$, $\gamma = \gamma \cup \{S\}$, $C = C \setminus \{S\}$.
 8: end while
```

Algorithm 2 Converting a cover to a clustering

Input: An integer k, and a $[k, 2k-1]$-cover $\gamma = \{S_1, \ldots, S_t\}$ of $D = \{R_1, \ldots, R_n\}$.
Output: A $[k, 2k-1]$-clustering, γ^0, of D.

```
 1: Set $\gamma^0 = \gamma$.
 2: while $\gamma^0$ has intersecting subsets do
 3:    Let $S_j, S_l \in \gamma^0$ be such that $S_j \cap S_l \neq \emptyset$ and let $R \in S_j \cap S_l$.
 4:    if $|S_j| > k$ then
 5:       Set $S_j = S_j \setminus \{R\}$.
 6:    else if $|S_l| > k$ then
 7:       $S_l = S_l \setminus \{R\}$
 8:    else $\{|S_j| = |S_l| = k\}$
 9:       Remove $S_l$ and $S_j$ from $\gamma^0$ and replace them with $S_j \cup S_l$.
10:    end if
11: end while
```

Theorem 2 *The k-anonymization g that is produced by Algorithm k-ANON satisfies*

$$\Pi(D, g(D)) \le 2(1 + \ln 2k) \cdot OPT(D), \tag{7}$$

Algorithm 3 k-ANON
k-anonymization via set-cover

```
Input: Table $D$, integer $k$.
Output: Table $g(D)$ that satisfies $k$-anonymity.
 1: Invoke Algorithm 1 with $C = F_{[k,2k-1]}$ (see Eq. (6)).
 2: Convert the resulting $[k, 2k-1]$-cover $\gamma$ into a $[k, 2k-1]$-clustering, $\gamma^0$, by invoking Algorithm 2
 3: Output the $k$-anonymization $g(D)$ of $D$ that corresponds to $\gamma^0$.
```

where $OPT(D)$ is the cost of an optimal k-anonymization.

Proof Let $\gamma \in \Gamma_{[k,2k-1]}$ be the cover that is produced by GEN-COVER (Algorithm 1), and let γ_{OPT} be the optimal cover (one that minimizes $d(\cdot)$ in $\Gamma_{[k,2k-1]}$). As the greedy algorithm approximates the optimal solution of the set-cover problem to within a factor of $(1 + \ln 2k)$, we have $d(\gamma) \leq (1 + \ln 2k) \cdot d(\gamma_{OPT})$.

Now, let γ^0 be the clustering that is achieved by Algorithm 2. In each iteration of the algorithm, it performs one of two possible operations — either the deletion of a record from a subset of the cover, or the unification of two intersecting subsets. As implied by our monotonicity assumption and by Lemma 1, neither of these operations increases the cost of the cover, whence $d(\gamma^0) \leq d(\gamma) \leq (1 + \ln 2k) \cdot d(\gamma_{OPT})$. Finally, by Theorem 1, the resulting k-anonymization satisfies inequality (7). □

Algorithm k-ANON (Algorithm 3) has an impractical runtime because of its first phase, GEN-COVER. The runtime of GEN-COVER is $O(|C||D|)$ where C is the input collection of subsets from which a cover is to be selected. Since Algorithm k-ANON invokes Algorithm GEN-COVER with an input collection of size $|C| = O(n^{2k-1})$, we end up with an impractical runtime of $O(n^{2k})$.

3.5 An improved version using closed frequent generalized records

To improve the runtime of k-ANON, Park and Shim [19] considered the case of generalization by suppression only, and introduced the usage of closed frequent itemsets in order to reduce the size of the collection C, while still guaranteeing the approximation ratio of $2(1 + \ln 2k)$.

They replaced the collection $C = F_{[k,2k-1]}$, that consists of all subsets of D of size between k and $2k - 1$ (see Eq. (6)) with a much smaller collection $C = F_{CF}$, which we will describe in Section 4. As the latter collection may have subsets of size greater than $2k - 1$, Park and Shim modified the basic algorithm GEN-COVER so that it outputs a cover consisting of subsets of size between k and $2k-1$ only. They prove that each of the possible outputs of their modified algorithm is also a possible output of the original GEN-COVER with the original input collection $C = F_{[k,2k-1]}$. Hence, each of the possible outputs of the modified algorithm is a cover that approximates the optimal cover to within $O(\ln k)$. This, in turn, implies that the k-anonymizations that are obtained by k-ANON, where the first phase invokes the modified algorithm instead of GEN-COVER, is also an $O(\ln k)$-approximation of an optimal k-anonymization.

The runtime of the modified first phase is $O(|F_{CF}||D|)$, while that of the original algorithm is $O(|F_{[k,2k-1]}||D|)$. Since, typically, $|F_{CF}|$ is much smaller than $|F_{[k,2k-1]}| = O(n^{2k-1})$, the modified algorithm is a practical version of k-ANON. However, as mentioned above, it is restricted to generalization by suppression only. We proceed to describe the extension of that algorithm for all proper generalizations and any monotone measure of loss of information.

4 A General $O(\log k)$-Approximation Algorithm for k-Anonymity

In this section we describe a general $O(\log k)$-approximation algorithm for k-anonymity, which is an adaptation of the algorithm that was proposed in [19] for the case of k-anonymization by suppression. The structure of our algorithm is similar to the structure of the algorithms of [8, 19] that we described in the previous section. Namely, it too (see Algorithm 5 below), like Algorithm 3, has two phases: In the first phase it produces a $[k, 2k - 1]$-cover of D that approximates the optimal $[k, 2k - 1]$-cover of D to within an approximation ratio of $O(\ln k)$; it does so by solving a weighted set cover problem using the greedy algorithm. In the second phase it translates the found $[k, 2k - 1]$-cover into a $[k, 2k - 1]$-clustering. As shown in Theorem 2, that clustering induces a k-anonymization that approximates the optimal k-anonymization to within $O(\ln k)$.

The second phase is identical in both Algorithms 3 and 5; the cover is translated to a clustering by invoking Algorithm 2. The difference between the two algorithms is in the first phase. While k-ANON (Algorithm 3) produces the cover by solving a weighted set cover problem with the collection of subsets $F_{[k,2k-1]}$, Algorithm 5 produces such a cover by solving a weighted set cover problem with a much smaller collection of subsets. This is a key issue in rendering the algorithm practical, as we proceed to explain.

The main disadvantage of k-ANON is the runtime of its first phase, GEN-COVER, which is $O(n^{2k})$. The modification of that algorithm that we present here (GEN-COVER-CF, Algorithm 4) also produces a cover of D that approximates the optimal $[k, 2k - 1]$-cover of D to within an approximation ratio of $O(\ln k)$; however, its runtime is significantly reduced. Both algorithms, GEN-COVER and GEN-COVER-CF, receive as an input a collection of subsets, $C \subseteq \mathcal{P}(D)$, from which they select the subsets for the cover. The runtime of both algorithms is bounded by $O(|C||D|)$. Hence, the key idea is to reduce the size of the input collection C. In the original algorithm GEN-COVER, the input collection is $C = F_{[k,2k-1]} := \{S \subset D : k \leq |S| \leq 2k - 1\}$, which is of size $O(n^{2k-1})$. In the modified algorithm GEN-COVER-CF, the input collection is $C = F_{CF}$, where F_{CF} contains only the supports of closed frequent generalized records.

Definition 7 Let $Q \in \overline{A}_1 \times \cdots \times \overline{A}_r$ be a generalized record. Its *support* is the subset of records $R \in D$ such that $R \sqsubseteq Q$. Q is called (k-)*frequent* if its support size is at least k. Q is called *closed* if for all generalized records $Q' \sqsubset Q$, the support of Q' is strictly smaller than that of Q.

Generally, the size of F_{CF} is much smaller than that of $F_{[k,2k-1]}$. Hence, the runtime of GEN-COVER-CF, which is $O(n \cdot |F_{CF}|)$, is much smaller than that of GEN-COVER, which is $O(n \cdot |F_{[k,2k-1]}|)$ (even though the theoretical time complexity of the two algorithms is the same — $O(n^{2k})$).

Example. Consider the following database that consists of ten transaction records,

$$D = \{R_1 = \text{shoes},\ R_2 = \text{shoes},\ R_3 = \text{shoes},\ R_4 = \text{boots},\ R_5 = \text{boots},\ R_6 = \text{sandals},\ R_7 = \text{ski-pants},\ R_8 = \text{sport-jackets},\ R_9 = \text{sport-jackets},\ R_{10} = \text{sport-jackets}\}.$$

Assume that the corresponding generalization taxonomy is as given in Figure 1. If $k = 3$, the frequent generalized records and their supports are given in Table 1. The frequent generalized record “clothes” is not closed, since it has the same support as that of the generalized record “outdoors” that narrows it. All other five frequent generalized records are closed. Hence, in this example we have

$$F_{CF} = \{\text{shoes, footwear, sport-jackets, outdoors, clothing}\} .$$

(More precisely, F_{CF} contains the supports of those five closed frequent generalized records; e.g., instead of “shoes” it contains the corresponding support $\{R_1, R_2, R_3\}$.) On the other hand, $F_{[k,2k-1]} = F_{[3,5]}$ includes in this case all subsets of D of cardinality between 3 and 5. The size of that collection is $\binom{10}{3} + \binom{10}{4} + \binom{10}{5} = 582$, which is much larger. As an example to the economization of replacing $F_{[k,2k-1]}$ with F_{CF}, consider the subset of records $S = \{R_1, R_2, R_3, R_4, R_5, R_6\}$. It has $\binom{6}{3} + \binom{6}{4} + \binom{6}{5} = 41$ subsets of sizes between 3 and 5. Of those 41 subsets, 40 subsets have the closure “footwear” (the only exception is the subset $\{R_1, R_2, R_3\}$ whose closure is “shoes”). Hence, the closed frequent generalized record “footwear” is reproduced in $F_{[3,5]}$ 40 times. In F_{CF}, on the other hand, it appears only once through its support S.

frequent generalized records	supports
shoes	$\{R_1, R_2, R_3\}$
footwear	$\{R_1, R_2, R_3, R_4, R_5, R_6\}$
sport-jackets	$\{R_8, R_9, R_{10}\}$
outdoors	$\{R_7, R_8, R_9, R_{10}\}$
clothes	$\{R_7, R_8, R_9, R_{10}\}$
clothing	D

Table 1 List of frequent generalized records and the corresponding supports

This change in the input collection of subsets requires us also to modify the algorithm itself. In GEN-COVER, it was essential that all subsets in the cover are of size between k and $2k-1$. In GEN-COVER-CF, the input collection of subsets is F_{CF}, and it may include subsets of any size greater than or equal to k. Hence, whenever the greedily selected subset S has a size larger than $2k-1$, we randomly select a subset $S^R \subset S$ of size $2k-1$ at the most, in which the number of uncovered records is maximized, and then insert that subset into the cover γ.

The modified algorithm GEN-COVER-CF is given in Algorithm 4. That algorithm is then used as a procedure in k-ANON-CF, Algorithm 5, which is the approximation algorithm for k-anonymity.

Comment. In the case of suppressions only, as studied in [19], all generalized records are taken from $\overline{A}_1 \times \cdots \times \overline{A}_r$, where $\overline{A}_j = A_j \cup \{*\}$. In other words, a generalized record in that context is an exact record where some of the components are suppressed. Such a generalized record may be identified with the *itemset* that consists of all record values that were not suppressed. For example, if $r = 3$ and the attributes are (Age, Gender, Zipcode), then the generalized record $(34, *, 93412)$ is identified with the itemset $\{A-34, Z-93412\}$ (where the prefixes A and Z are used to identify the attribute from which the following value was taken). Hence, the terminology and techniques used in [19] are those of itemsets and not generalized records. Herein, we need to adopt the wider terminology of generalized records, and apply also more general techniques for mining all closed frequent generalized records.

We proceed to show that GEN-COVER-CF is an appropriate substitute to GEN-COVER, in the sense that it produces a cover of D that approximates the optimal $[k, 2k-1]$-cover

Algorithm 4 GEN-COVER-CF
A greedy approximation to optimal cover by closed frequent generalized records

Input: Table D; the collection of the supports of all closed frequent generalized records, F_{CF}.
Output: Cover γ of D, where each set has size between k and $2k-1$.

```
1: $\gamma = \emptyset$ {the current cover}
2: $E = \emptyset$ {currently covered records in $D$}
3: while ($E \neq D$) do
4:   for all $S \in F_{CF}$ do
5:     Compute the ratio $\rho(S) = \frac{d(S)}{\min(|S \cap (D \setminus E)|, 2k-1)}$ .
6:   end for
7:   Choose a set $S$ for which $\rho(S)$ is minimized.
8:   if ($|S| \leq 2k-1$) then
9:     $S^R \leftarrow S$ {the set is in the right size}
10:  else if ($|S \cap (D \setminus E)| \geq 2k-1$) then
11:    Choose $S^R \subseteq S \cap (D \setminus E)$ such that $|S^R| = 2k-1$. {select $2k-1$ uncovered records}
12:  else {$|S| \geq 2k$ and $|S \cap (D \setminus E)| < 2k-1$}
13:    Choose $S^R \subseteq S$ such that $S^R \supseteq S \cap (D \setminus E)$ and $|S^R| = \max(k, |S \cap (D \setminus E)|)$
14:  end if
15:  $E \leftarrow E \cup S^R$
16:  $\gamma \leftarrow \gamma \cup \{S^R\}$
17: end while
18: return $\gamma$
```

Algorithm 5 k-ANON-CF
k-anonymization via set-cover using closed frequent generalized records

Input: Table D, integer k.
Output: Table $g(D)$ that satisfies $k-$anonymity

```
1: Find all closed generalized records that have support of size at least $k$ in $D$ (Algorithm 6).
2: Set $F_{CF}$ to be the set of supports of all the found closed frequent generalized records.
3: Produce a cover $\gamma$ of $D$, by using Algorithm 4, with $F_{CF}$ as an input.
4: Convert the resulting $[k, 2k-1]$-cover $\gamma$ into a $[k, 2k-1]$-clustering, $\gamma^0$, by invoking Algorithm 2
5: Output the $k$-anonymization $g(D)$ of $D$ that corresponds to $\gamma^0$.
```

to within $O(\ln k)$. Once we show that, we may conclude that k-ANON-CF (Algorithm 5) produces a k-anonymization that approximates the optimal one to within $O(\ln k)$ by arguing along the same lines as in Theorem 2.

First of all, it is clear that the cover γ produced by GEN-COVER-CF includes only subsets of size between k and $2k-1$. It remains to show that γ achieves the same approximation ratio as GEN-COVER.

Lemma 2 *If S and S^R are the subsets that are selected in each iteration of* GEN-COVER-CF*, then $\rho(S^R) \leq \rho(S)$.*

Proof We will prove this claim for every set S and its corresponding subset S^R (namely, not only for sets S that minimize ρ). Since $S^R \subseteq S$ then, by monotonicity, $d(S^R) \leq d(S)$. Furthermore, it is easy to see that in each of the three cases in GEN-COVER-CF, we have

$$\min\{|S \cap (D \setminus E)|, 2k-1\} = \min\{|S^R \cap (D \setminus E)|, 2k-1\} .$$

This implies that $\rho(S^R) \leq \rho(S)$. $\square$

Given a table D and an integer k, we may apply GEN-COVER on the collection of subsets of records $F_{[k,2k-1]}$, or we may apply GEN-COVER-CF on the collection F_{CF}. In both cases we shall get as an output a cover of D by subsets of records of D whose size is between k and $2k-1$. The cover that GEN-COVER outputs might differ from the one that GEN-COVER-CF outputs due to the random selections made by the two algorithms. (Specifically, in each iteration there may be more than one set that minimizes $\rho(\cdot)$, and the algorithms randomly select one of them.) However, the following relation holds.

Theorem 3 *Every cover that may be produced by* GEN-COVER-CF*, given F_{CF}, is also a possible output cover for* GEN-COVER*, given $F_{[k,2k-1]}$.*

Proof Let Cov and Cov_{CF} be the sets of all possible output covers that may be produced by GEN-COVER with $F_{[k,2k-1]}$ and GEN-COVER-CF with F_{CF}, respectively. Consider any cover $\gamma_{CF} \in Cov_{CF}$. We will show that there exists a cover $\gamma \in Cov$ such that $\gamma = \gamma_{CF}$.

Let $\gamma_{CF} = \{S_1^{'R}, S_2^{'R}, \cdots, S_m^{'R}\}$ where the subscript denotes the selection order by GEN-COVER-CF. $S_i^{'R}$ is selected in lines 8-14 of GEN-COVER-CF as a subset of S_i' which was selected greedily among the subsets in F_{CF} in lines 4-7. We claim that there exists a cover $\gamma = \{S_1, S_2 \ldots S_m\} \in Cov$, where the subscript denotes the selection order of the subsets by GEN-COVER, in which $S_i = S_i^{'R}$ for $1 \le i \le m$. That will prove that $\gamma_{CF} = \gamma \in Cov$. We will prove this claim by induction on the length of prefixes of γ_{CF} that may be found in covers in Cov.

Assume that there exists a cover $\gamma \in Cov$ such that $S_i = S_i^{'R}$ for $1 \le i \le j-1$, where $1 \le j \le m$. We will prove that such an assumption implies that there exists a cover $\gamma' \in Cov$ for which $S_i = S_i^{'R}$ for $1 \le i \le j$. (The case $j = 1$, in which the assumption is empty, is the induction base.) Assume, towards contradiction, that there does not exist such a cover. We consider two cases:

Case 1. $\rho(S_j) < \rho(S_j^{'R})$: Let $S^{''}$ be the support of $\overline{S_j}$ in D. Then the closures of $S^{''}$ and S_j coincide, $\overline{S''} = \overline{S_j}$, whence $d(S^{''}) = d(S_j)$. Therefore, since $S^{''} \supseteq S_j$, we conclude that $\rho(S^{''}) \le \rho(S_j) < \rho(S_j^{'R})$. Since S_j' is the greedily selected subset at the jth iteration of GEN-COVER-CF, we have $\rho(S_j^{'R}) \le \rho(S_j')$ by Lemma 2. Therefore, $\rho(S^{''}) < \rho(S_j')$. Since $S^{''}$ is the support of the generalized record $\overline{S_j}$, it is also the support of some closed generalized record that narrows $\overline{S_j}$; hence, as $|S^{''}| \ge k$, we infer that $S^{''} \in F_{CF}$. But that contradicts our assumption that S_j' minimized the value of $\rho(\cdot)$ in F_{CF} at the jth iteration of GEN-COVER-CF.

Case 2. $\rho(S_j) > \rho(S_j^{'R})$: Since $k \le |S_j^{'R}| \le 2k-1$ and $F_{[k,2k-1]}$ is the collection of all subsets of D of size between k and $2k-1$, we infer that $S_j^{'R} \in F_{[k,2k-1]}$. Then we should have selected $S_j^{'R}$ instead of S_j at the jth iteration of GEN-COVER, which, ones again, leads to a contradiction.

We conclude that $\rho(S_j) = \rho(S_j^{'R})$. But $S_j^{'R} \in F_{[k,2k-1]}$, since $k \le |S_j^{'R}| \le 2k-1$. Then, if GEN-COVER selected $\{S_1, \ldots, S_{j-1}\}$ until the $(j-1)$th iteration, it can select $S_j^{'R}$ instead of S_j at the jth iteration. Hence, there exists a possible output cover of GEN-COVER in which the first j selected subsets are $S_1^{'R}, \ldots, S_{j-1}^{'R}$ and $S_j^{'R}$. Setting $j = m$, where m is the number of subsets in γ_{CF}, yields the desired result that $\gamma_{CF} \in Cov$. $\square$

Finally, combining Theorem 3 and Theorem 2 we arrive at the following conclusion.

Theorem 4 *The k-anonymization g that is produced by Algorithm k-*ANON-CF *satisfies*

$$\Pi(D, g(D)) \le 2(1 + \ln 2k) \cdot OPT(D),$$

where $OPT(D)$ is the cost of an optimal k-anonymization.

5 An Algorithm for Mining Closed Frequent Generalized Records

As stated previously, the input to GEN-COVER-CF is the collection $C = F_{CF}$, where F_{CF} contains only sets of records, S, whose closure, $\overline{S}$, is a closed frequent generalized record. In this section we describe an algorithm that mines all closed frequent generalized records.

The problem of mining frequent itemsets was first introduced by Agrawal et al. [2]. They proposed an algorithm called Apriori for mining frequent itemsets. Apriori employs a bottom-up, breadth-first search that enumerates every single frequent itemset. Apriori uses the downward closure property of itemsets in order to prune the search space. Thus, only the frequent k-itemsets are used to construct candidate $(k+1)$-itemsets. The algorithm starts by finding all itemsets of size 1 that are frequent, and then generates itemsets of size 2 that are candidate to being frequent. It then scans the table in order to detect which of the candidates is indeed a frequent itemset. Frequent itemsets of size 2 are joined in order to create candidate itemsets of size 3, and then another table scan finds the frequent itemsets among those candidates. This procedure is repeated until no more frequent itemsets are found.

Subsequent algorithms were devised for mining closed frequent itemsets [11,10,20], but they too apply only for regular itemsets, and not for generalized itemsets (or records) as we need here. An algorithm for mining frequent generalized itemsets is also available [23], but as it is Apriori-based, it suffers from the drawbacks of Apriori (iterated candidate generation and repeated database scans) and performs poorly in our application.

Here, we deal with structured databases and not with transactional databases. In structured databases, a generalized itemset is a generalized record. Hence, each itemset is of size r, where r is the number of quasi-identifiers in the table, and it includes exactly one item from each of the taxonomies. This distinguishes structured databases from transactional ones, where itemsets may be of different sizes and may contain any number of items from each taxonomy. The algorithm that we present here takes advantage of those special characteristics of structured databases, in order to mine efficiently all closed frequent generalized records.

5.1 Overview of the algorithm

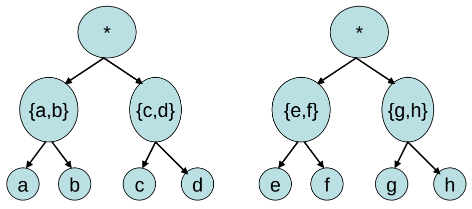
Let $D = \{R_1, \dots, R_n\}$ be a database table with records $R_i \in A_1 \times \cdots \times A_r$, $1 \le i \le n$. For each $1 \le j \le r$, let $\overline{A}_j \subseteq \mathcal{P}(A_j)$ be a proper collection of subsets of A_j (Definition 3). As implied by [8, Lemma 3.3], such collections form a (possibly unbalanced) hierarchical clustering tree (or a taxonomy) for A_j. Our goal here is to find all generalized records $Q \in \overline{A}_1 \times \cdots \times \overline{A}_r$ that are supported by at least k records in D (namely, there are at least k records $R_i \in D$ such that $R_i \sqsubseteq Q$) and, in addition to that, are closed (Definition 7).

Figure 2(a) illustrates an example of a database with ten records and two attributes ($n = 10$, $r = 2$) and the accompanying taxonomies for each of the two attributes. That toy example will be used in order to exemplify the operation of the algorithm.

The algorithm starts with the most generalized record, $(A_1, \cdots, A_r)$, where the jth entry holds the entire set of possible values for that entry, namely, A_j. That generalized record may be thought of as the totally suppressed record, $(*, \dots, *)$. It is supported by all records of D (since every record in D belongs to $A_1 \times \cdots \times A_r$), whence, it is frequent. The algorithm then starts exploring the search space by narrowing the generalized record in a depth-first manner. It keeps narrowing the generalized record until it can no longer be narrowed and still remain frequent. If the current record turns out to be closed, in the sense that all of

A	B
a	e
d	h
b	h
c	g
d	e
b	g
b	e
c	f
a	e
b	g

(a)



(b)

Fig. 2 (a) Database (b) Taxonomies

the generalized records that narrow it have smaller supports, it is added to the list of closed frequent generalized records that the algorithm eventually outputs.

The collection of all generalized records, $\overline{A}_1 \times \cdots \times \overline{A}_r$, forms a lattice, in which there are many paths, or branches, of narrowing. The algorithm searches all possible branches of narrowing. However, as the size of the lattice may be large, the algorithm reduces the search space by implementing two basic ideas.

The first idea exploits the following property of anti-monotonicity: If Q and Q' are two generalized records and $Q \sqsubseteq Q'$, then $|support(Q)| \leq |support(Q')|$. Therefore, if during the search we reached a record that is infrequent, there is no point in examining the records that narrow it, since they must be infrequent as well.

The next lemma provides another idea that allows the algorithm to prune branches of the search space that cannot lead to yet undiscovered closed frequent generalized records. The basic idea is that once a generalized record X is found to be not closed, namely, one of the generalized records that narrows it, say X', has the same support, we need to process only that narrowed record X' and none of the other generalized records that narrow X.

Lemma 3 *Let X be a frequent generalized record, and assume that $X' \sqsubset X$. If $support(X) = support(X')$, then X is not closed. Moreover, assume that Y is a generalized record such that $Y \sqsubset X$ and there exists $1 \leq j \leq r$ for which $X'(j) \subsetneq Y(j)$. Then Y is not closed either.*

Proof Since $X' \sqsubset X$, but the two generalized records have the same support, X is not closed. Now, consider a generalized record $Y = (Y(1), \ldots, (Y(r))$ as described in the second part of the lemma. Since $Y \sqsubset X$ then

$$support(Y) \subseteq support(X) = support(X') \,. \tag{8}$$

As $\overline{A}_j$, $1 \leq j \leq r$, are proper, one of the following holds for each $1 \leq j \leq r$:

$$X'(j) \subseteq Y(j) \,, \quad \text{or } X'(j) \supseteq Y(j) \,, \quad \text{or } X'(j) \cap Y(j) = \emptyset \,.$$

We separate our discussion into two cases.

Case 1. Assume that there exists $1 \leq j \leq r$ for which $X'(j) \cap Y(j) = \emptyset$. In this case, $support(X'(j)) \cap support(Y(j)) = \emptyset$, since no record can contain in its jth entry items from both $X'(j)$ and $Y(j)$. Consequently, $support(X') \cap support(Y) = \emptyset$, because for any generalized record Z, $support(Z(j)) \supseteq support(Z)$ for every $1 \leq j \leq r$. As $support(X') =$

$support(X)$, we infer that $support(X) \cap support(Y) = \emptyset$. Finally, in view of (8) we arrive at the conclusion that $support(Y) = \emptyset$.

Consider now the generalized record

$$Y' = (Y(1), \ldots, Y(j-1), X'(j), Y(j+1), \ldots, Y(r))$$

where j is the index for which $X'(j) \subsetneqq Y(j)$. As $Y' \sqsubset Y$ and $support(Y) = \emptyset$, we conclude that $support(Y') = \emptyset$. Hence, Y has a proper narrowing, Y', that has the same support. Therefore, Y is not closed. This settles the proof in Case 1.

Case 2. Assume that for all $1 \leq j \leq r$, $X'(j) \subseteq Y(j)$ or $X'(j) \supseteq Y(j)$. Consider the generalized record $Y' = (Y'(1), \ldots, Y'(r))$, where:

$$Y'(j) = \begin{cases} X'(j) & X'(j) \subsetneqq Y(j) \\ Y(j) & X'(j) \supseteq Y(j) \end{cases} \quad , \quad 1 \leq j \leq r\,.$$

In other words, $Y'(j) = Y(j) \cap X'(j)$ for all $1 \leq j \leq r$. Therefore,

$$support(Y') = support(Y) \cap support(X') = support(Y)\,.$$

Hence, we found a generalized record $Y' \sqsubset Y$ for which $support(Y') = support(Y)$. Consequently, Y is not closed. This settles the proof in the second case as well. $\Box$

Example. Consider the database in Figure 2 and take

$$X = (\{a, b\}, \{g, h\})\,, \quad X' = (\{b\}, \{g, h\})\,.$$

Clearly, $X' \sqsubset X$, but they both have the same support, namely $\{3, 6, 10\}$. As implied by Lemma 3, there is no need to process generalized records such as $Y = (\{a, b\}, \{g\})$, since $Y \sqsubset X$ and $X'(1) \subsetneqq Y(1)$. Indeed, the support of Y equals that of $Y' = (\{b\}, \{g\})$ (the set of records $\{6, 10\}$), whence Y is not closed.

5.2 Efficient computation of supports of generalized records

Each proper collection of subsets, $\overline{A_j}$, $1 \leq j \leq r$, is a hierarchical clustering tree, or a taxonomy, for the attribute A_j. We will augment the taxonomy structures in a manner that will allow us to compute the support of any given generalized record efficiently. To that end, we associate with each leaf in the taxonomy a list that holds the indices of all records in D that contain the value in that leaf. This way, the union of the lists of all leaves under a given node in the taxonomy will be the support for that node (in other words, the list of all records in D that contain a value that appears in that node). Figure 3 illustrates the augmented taxonomies for the example in Figure 2(b).

Assume next that we wish to compute the support of the generalized record $(a_1, \ldots, a_r)$, where a_j is a node in the taxonomy $\overline{A_j}$. Then all we need to do is to intersect the supports of each of the sets of records that are associated with the r nodes a_j. For example, if we wish to compute the support of the generalized record $(\{a, b\}, \{g\})$, we first compute the list of records in which the first entry supports the set $\{a, b\}$, then we compute the list of records in which the second entry supports the set $\{g\}$, and then intersect the two lists. The first list is $\{1, 3, 6, 7, 9, 10\}$ (it is the union of the two lists under the node $\{a, b\}$ in the first taxonomy in Figure 3), while the second list is $\{4, 6, 10\}$. The support of $(\{a, b\}, \{g\})$ is the corresponding intersection, $\{6, 10\}$.

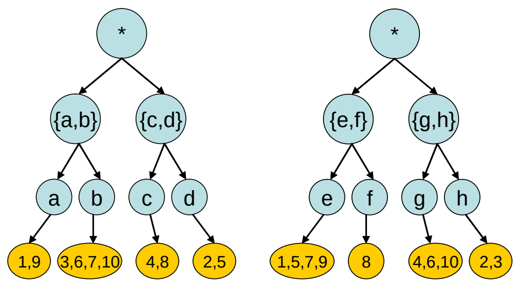


Fig. 3 Taxonomy structures

Computing the generalized record support in such a manner is much more efficient than traversing the complete database and inspecting each record. After making one pass over the entire database, the algorithm needs only to mine these compact taxonomy structures instead of going again over the actual database.

5.3 The algorithm

Algorithm 6 is the algorithm for mining closed frequent generalized records. It starts by augmenting the taxonomies with the lists of supports for each leaf in each taxonomy (Step 1). It then prunes from $\overline{A}_j$ nodes whose support is of size less than k since no generalized record that includes the corresponding generalized value in its jth entry can be frequent (Steps 2-4). Then it starts with the most generalized record, $(A_1, \ldots, A_r) = (*, \ldots, *)$ (which is obviously frequent, but not necessarily closed), and starts traversing the search space in a depth-first manner. A generalized record will be added to the output list, F_{CF}, if it is frequent and closed, i.e., none of its direct narrowings has the same support as it does. Both checks can be made quickly and easily using the special data structures that were constructed in Step 1. In order to avoid processing a generalized record more than once, we keep a hash table (called *processed*) of all generalized records that were already examined.

Algorithm 6 Mining closed frequent generalized records

Input:

- Database D of n records and r quasi-identifiers
- Minimum support threshold, k
- Taxonomy trees, $\overline{A}_1, \cdots, \overline{A}_r$

Output: F_{CF} — the list of all closed generalized records of support size at least k.

```
1: Make a single pass over $D$ and augment the trees $\overline{A}_1, \cdots, \overline{A}_r$ with the supporting record ids.
2: for all $1 \leq j \leq r$ do
3:     Remove from $\overline{A}_j$ all nodes that are supported by less than $k$ records.
4: end for
5: $root \leftarrow (A_1, \ldots, A_r)$
6: $processed \leftarrow \emptyset$
7: $F_{CF} \leftarrow \emptyset$
8: Call Algorithm 7 with inputs $root$, $k$, $processed$, and $F_{CF}$.
9: return $F_{CF}$
```

The main function in this algorithm is called in Step 8: That function, which is implemented in Algorithm 7, receives as an input a root generalized record, $(a_1, \ldots, a_r)$, and adds to the list of closed frequent generalized records all the closed frequent generalized records that are narrowings of the given root. That function implements the depth-first search, starting from the given root downward.

Algorithm 7 Updating the list of closed frequent generalized records

Input:

- A generalized record, $root = (a_1, \ldots, a_r)$
- Minimum support threshold, k
- A set of processed generalized records, $processed$
- A set of closed frequent generalized records, $closed$

Output: Adding to the given list $closed$ all the closed frequent generalized records that are narrowings of $root$.

```
1:  if $((root \in processed)\ OR\ (|support(root)| < k))$ then
2:      return
3:  end if
4:  $processed \leftarrow processed \cup \{root\}$
5:  $isRootClosed \leftarrow true$
6:  for $j = 1$ to $j = r$ do
7:      for all children $b_j$ of $a_j$ in $\overline{A}_j$ do
8:          $new\_support \leftarrow support(b_j) \cap support(root)$
9:          if $(|new\_support| \geq k)$ then
10:             if $(|new\_support| = |support(root)|)$ then
11:                 $isRootClosed \leftarrow false$
12:             end if
13:             $new\_root \leftarrow (a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \cdots, a_r)$
14:             Call Algorithm 7 (recursively) with inputs $new\_root$, $k$, $processed$, $closed$;
15:             if $(isRootClosed = false)$ then
16:                 break double loop;
17:             end if
18:         end if
19:     end for
20: end for
21: if $(isRootClosed = true)$ then
22:     $closed \leftarrow closed \cup \{root\}$
23: end if
```

It starts (Steps 1-3) by checking whether the given root was already processed, or if its support is of size less than k; in either case it returns without going on further. It then continues and adds the root to the list of processed generalized records (Step 4) and marks it as closed (Step 5). The mark of the root as being closed will remain so until we are convinced otherwise.

Next, we start the loop over all quasi-identifiers ($j = 1, \ldots, r$) (Step 6) and for each identifier over all the children of a_j (Step 7). This double loop scans all generalized records that are immediate narrowings of the input generalized record. (To this end, a generalized record X' is an immediate narrowing of the generalized record X if X and X' coincide in all entries except for one entry, say the jth entry, in which $X'(j)$ is an immediate child of $X(j)$.) We first compute the support of that narrowed record (Step 8). If the support of that narrowed record is of size less than k (Step 9), there is no need to further process it, or any of its narrowings, as that generalized record is not frequent. Furthermore, such an infrequent narrowing does not provide any evidence against our current assumption that the input generalized record is closed, since the input generalized record has a support of size at least k, while the current narrowing has a support of size less than k.

Next (Steps 10-12), if the support of that narrowed record equals that of the input generalized record then we mark the input generalized record as not closed. Then we activate the depth-first strategy by calling the function, recursively, with that narrowed record as an input. Finally, if the input generalized record has been established as not closed, we can break out of the double-loop and not proceed to check the rest of the narrowings. To see why, let us assume that the root is $X = (a_1, \ldots, a_r)$ and assume that a_j has in $\overline{A}_j$ the children $b_1, \ldots, b_t$. Assume that while examining the narrowed record $X' = (a_1, \ldots, a_{j-1}, b_1, a_{j+1}, \ldots, a_r)$, we discovered that X and X' have the same support (whence, we set *isRootClosed* to be false). Then that means that the support of $(a_1, \ldots, a_{j-1}, b_i, a_{j+1}, \ldots, a_r)$ for all $i > 1$ is empty and so we can break out of the inner loop on the children. Moreover, there is no need to continue the loop over j since the resulting narrowed records there cannot be closed and cannot lead to other closed generalized records as implied by Lemma 3.

Finally, at the end of the loop over all immediate narrowings, if the root survived all checks and proved to be closed, we add it to the list of closed frequent generalized records (Steps 21-23).

6 Enhancing k-Anonymity by ℓ-Diversity

6.1 About k-anonymity and ℓ-diversity

Several studies have pointed out weaknesses of the k-anonymity model and suggested more secure measures such as ℓ-diversity [16], t-closeness [15], or p-sensitivity [26]. The main weakness of k-anonymity is that it does not guarantee sufficient diversity in the private attribute in each equivalence class of indistinguishable records. Namely, even though it guarantees that every record in the anonymized table is indistinguishable from at least $k-1$ other records, when projected on the subset of quasi-identifiers, it is possible that all of those records have the same private value. Therefore, an adversary who is capable of locating his target individual in that block of records, will be able to infer the private value of that individual. Machanavajjhala et al. [16] proposed the security measure of ℓ-diversity. They suggested that the private attribute in each block will have at least ℓ "well represented" values. They offered two interpretations of that measure. In one interpretation, the entropy of the private values in every block should be at least $\log \ell$, for some predetermined value of the parameter ℓ. The other interpretation is that of recursive (c, ℓ)-diversity (see [16] for its definition). In practice, a simpler interpretation of ℓ-diversity is usually applied [27,28]: A block is ℓ-diverse if the relative frequency of each of the private values within each block does not exceed $1/\ell$.

The notion of t-closeness is stronger than ℓ-diversity since it demands that the distribution of the private values within every block of indistinguishable records would be sufficiently close to its general distribution in the table. The notion of p-sensitivity, on the other hand, relaxes the notion of ℓ-diversity as it only requires each block to have p distinct private values, but does not impose any condition on their distribution.

It is important to understand that those notions do not and can not replace k-anonymity. They offer essential *enhancements* to k-anonymity in the sense that one must require them *in addition* to k-anonymity. In accord with this, Truta et al. [26] propose algorithms that generate tables that are both k-anonymous and p-sensitive, and Wong et al. [27] consider the conjunction of k-anonymity with the last interpretation of ℓ-diversity (they call this conjunction of conditions $(1/\ell, k)$-anonymity).

In order to clarify that point, let us focus on the most important notion of the three that we mentioned above — ℓ-diversity. The diversity of a table is bounded from above by the number of possible private values (equality holds if and only if the distribution of the private values is uniform). The diversity of any anonymization of the table is bounded from above by the diversity of the entire table (equality holds if and only if the distribution in each block equals the global distribution). Therefore, if the table has a private attribute with a small number of possible values, all of its anonymizations will respect ℓ-diversity with ℓ that does not exceed this number. For example, in the case of a binary private attribute, one can aim at achieving ℓ-diverse anonymizations with $\ell \leq 2$ only. In such a case, if one imposes only ℓ-diversity, the blocks of indistinguishable records could be of size 2. Such small blocks do not provide enough privacy for the individuals in them, because if an adversary may be able to learn the private value of one of those individuals, he may infer that of the other one as well. If, on the other hand, we demand that such ℓ-diverse anonymizations are also k-anonymous, for a suitable selection of k, then the adversary would have to find out the private values of at least $k/2$ individuals before he would be able to infer the private value of his target individual.

6.2 Modifying k-anonymizations to meet the ℓ-diversity constraint

As noted in [16], any algorithm for k-anonymization may be enhanced so that it issues k-anonymized tables that are also ℓ-diverse. The idea is simple: The diversity of the union of two clusters of records is a convex combination of the diversities of the two clusters (this is true for all acceptable definitions of diversity). Hence, if there are clusters of records that violate ℓ-diversity, one can start unifying them until ℓ-diversity is met. Such a procedure, as explained in [16], will always stop successfully if the target diversity parameter ℓ is a legitimate one (namely, if the global diversity in D is at least ℓ). Therefore, in order to convert *any* algorithm that is designed to achieve only k-anonymity into one that achieves k-anonymity *and* ℓ-diversity, it is needed to post-process the output clustering by unifying clusters that violate ℓ-diversity until all clusters are ℓ-diverse. Herein, we adopt the definition of diversity from [27,28] (see Definition 8); however, the algorithm that we present later on can be applied also with other definitions of diversity, such as the entropy-based definition that was proposed in [16].

Definition 8 Let $C = \{R_{i_1}, \ldots, R_{i_{|C|}}\}$ be a cluster of records in D and let $C' := \{S_{i_1}, \ldots, S_{i_{|C|}}\}$ be the private values of those records. Let f be the number of occurrences of the most frequent value in C'. Then the diversity of C is div$(C) := |C|/f$. Let $\gamma = \{C_1, \ldots, C_t\}$ be a clustering of the records of the table D. Then its diversity is div$(\gamma) := \min_{1 \leq i \leq t} \text{div}(C_i)$.

Algorithm 8, that is described below, is a post-processing procedure that may be applied on top of any algorithm of k-anonymization. Its input is any clustering of the records of the table D, and a target diversity parameter $\ell \geq 1$. Its output is a coarser clustering in which all clusters are ℓ-diverse. (By coarser clustering we mean that it is derived from the input clustering only by means of unifying clusters.) We shall apply Algorithm 8 on clusterings issued by k-anonymization algorithms, namely, clusterings in which all clusters are of size at least k. Since the output clustering is coarser than the input clustering, all of the clusters in it will be of size at least k, and, in addition, they will all be ℓ-diverse.

First, the algorithm computes the diversities of all clusters in the input clustering γ. It selects the cluster C_m with minimal diversity. If that cluster is already ℓ-diverse, the clustering is ripe to be output. Otherwise, we look for the best cluster with which C_m can be unified. Once such a cluster is found, we unify it with C_m and then repeat the procedure until all clusters are ℓ-diverse.

The algorithm uses a cost function in order to decide about the most profitable unification. On one hand, unifying the least diverse cluster with another cluster brings us closer to meeting the ℓ-diversity requirement. On the other hand, unifying clusters increases the information loss. It is our goal to achieve a maximal gain towards meeting the ℓ-diversity requirement, but at the same time we wish to favor unifications that will incur smaller additions to the information loss. Hence, we define the cost function as a weighted average between an information cost and a diversity cost.

Let $\gamma = \{C_1, \dots, C_t\}$ be a clustering of the records in the table D. For any two clusters in γ, say C_i, C_j, we let $\gamma_{C_i,C_j} = \big(\gamma \setminus \{C_i, C_j\}\big) \cup \{C_i \cup C_j\}$ denote the clustering that would be obtained from γ if C_i and C_j were unified. Let

$$\text{cost}_I(C_i, C_j) = ANON(\gamma_{C_i,C_j}) - ANON(\gamma) \tag{9}$$

be the added information loss if C_i and C_j were unified; in view of Eq. (5) it equals

$$|C_i \cup C_j| \cdot d(C_i \cup C_j) - |C_i| \cdot d(C_i) - |C_j| \cdot d(C_j)\,.$$

The diversity cost, $\text{cost}_D(C_i, C_j)$, is defined as the remaining gap between the diversity of the unified cluster $C_i \cup C_j$ and the target level ℓ:

$$\text{cost}_D(C_i, C_j) = \max\left\{\ell - \text{div}(C_i \cup C_j)\ ,\ 0\right\}\ . \tag{10}$$

Finally, we define

$$\text{cost}(C_i, C_j) = w \cdot \text{cost}_I(C_i, C_j) + (1 - w) \cdot \text{cost}_D(C_i, C_j)\,, \tag{11}$$

where w is a weight between 0 and 1 that can be tuned experimentally.

Algorithm 8 A post-processing algorithm to achieve ℓ-diverse anonymizations

```
Input: A clustering $\gamma = \{C_1, \dots, C_t\}$ of the records in a table $D$; a target diversity parameter $\ell \geq 1$.
Output: A coarser clustering that respects $\ell$-diversity.
1: Compute div$(C_i)$ for all $C_i \in \gamma$.
2: Let $C_m$ be the cluster with minimal diversity in $\gamma$.
3: if div$(C_m) \geq \ell$ then
4:     Output $\gamma$ and stop.
5: end if
6: Compute cost$(C_i, C_m)$ for all $C_i \in \gamma \setminus \{C_m\}$.
7: Find the cluster $C_i \in \gamma \setminus \{C_m\}$ for which cost$(C_i, C_m)$ is minimal.
8: Remove $C_i$ and $C_m$ from $\gamma$ and add to $\gamma$ the cluster $C_i \cup C_m$.
9: Go to Step 2.
```

It should be noted that by converting our algorithm to an algorithm which issues k-anonymizations that respect also ℓ-diversity, the logarithmic approximation ratio is no longer guaranteed. However, the anonymizations issued by our algorithm are characterized not only by a proven approximation guarantee, but also by lower information losses than those of other leading k-anonymization algorithms, as demonstrated by our experimental evaluation (Section 7). Our experiments show that this advantage is maintained also after applying the ℓ-diversity post-processing.

7 Experiments

We tested our algorithm versus three algorithms: the Forest algorithm [1], the Agglomerative algorithm [7], and the Hilbert-curve algorithm[1] [6]. We also tested it against the recent sequential clustering algorithm [9], but as it issues results that are very close to those issued by the agglomerative algorithm we do not include a separate curve for that algorithm in the figures below.

The tests were conducted on the following three datasets from the UCI Machine Learning Repository:

- Adult: The Adult dataset was extracted from the US census Bureau Data Extraction System. It contains demographic information of a sample of US population with 14 public attributes such as age, education-level, marital-status, occupation, and native-country. The private information is an indication whether that individual earns more or less that 50 thousand dollars annually. The Adult data contains 45,222 records after the records with missing values are removed.
- Nursery: The Nursery dataset was derived from a hierarchical decision model that was originally developed to rank applications for nursery schools. The Nursery dataset contains 12,960 records after deleting those with missing values. It has 8 quasi-identifier attributes.
- Coil2000: This dataset used in the CoIL 2000 Challenge contains information on customers of an insurance company. The data consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. The Coil2000 dataset contains 5,822 records after deleting those with missing values. We used the first 9 quasi-identifiers out of the 86 available ones.

We ran each of the three algorithms with seven values of the anonymity parameter, $k = 10, 30, 50, 75, 100, 150, 200$. The information loss measure that we used in order to compare the algorithms was the LM measure (see Eq. (3)). We also verified the results on the entropy measure [8].

All of the algorithms that we tested were implemented in Java and run on a core 2 (R) quad (Q6600) CPU 2.4 GHz, 8GB of RAM.

7.1 Adult dataset experiments

We created taxonomies for each the categorical attributes in that database, as follows:

1. Taxonomies of height 1 (suppression only): In the three attributes Relationship, Race, and Sex we used suppression only since those attributes were not rich enough to support meaningful intermediate levels of generalization.
2. Taxonomies of height 2 (one intermediate level of generalization). Such taxonomies were applied in the following attributes (for each of those attributes we list the groups of values in the intermediate level and the number of exact values in each of those groups):
   - Workclass: Government (3), Private employment (3), Not employed (2)
   - Education: Academic (3), Some school (9), Other education (4).

[1] We used a modification of the original algorithm in which we rescale all table attributes, prior to applying the Hilbert mapping, so that they vary along intervals of the same length. Such a modification improves the results of the algorithm, especially in cases where the original attributes vary along intervals of sizes that differ significantly. For a detailed discussion and justification of that modification, the reader is referred to [9].

- Marital-status: Married (3), Unmarried (4).
- Occupation: Business (3), Technical (3), Protective (2), Other (6)

3. Taxonomies of height 3:
   - Native-country: The first level consists of North America, Central America and Carribean, South America, Europe, and Asia. The second level consists of West (Americas and the Carribean) and East (Europe and Asia).

In the remaining six attributes that are numeric we used generalization by interval ranges. The comparison between the various algorithms can be seen in the following figures. Figure 4 compares between the algorithms when run using the LM and entropy (EM) measures. Figure 5 displays the runtime differences between the algorithms in the case of the LM measure; the graph for the EM measure is similar.

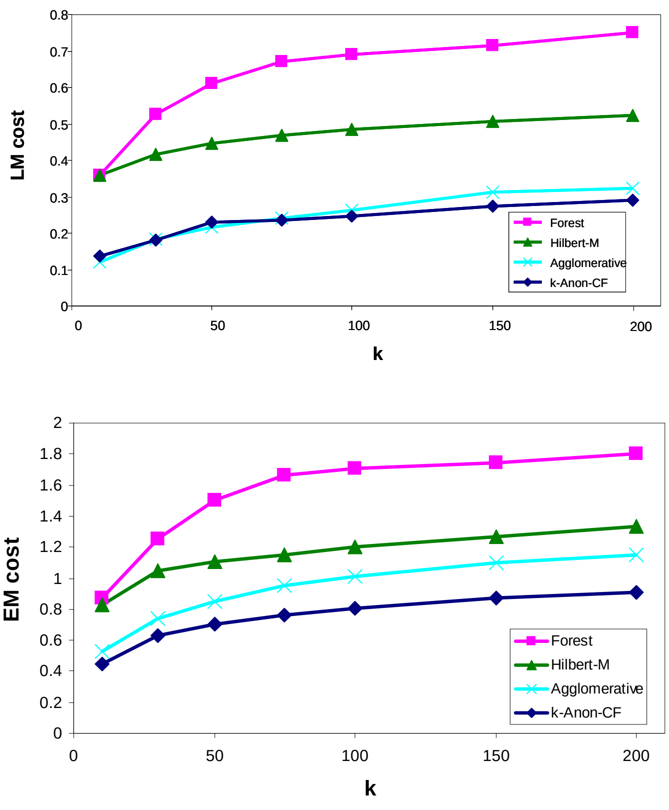


Fig. 4 Algorithm score comparison — Adult dataset

As we can see, our anonymization algorithm, k-ANON-CF, provides much better anonymizations than the Forest algorithm, in consistency with the improvement in the approximation factor from $O(k)$ (for the Forest algorithm) to $O(\ln k)$ (for k-ANON-CF). The informa-

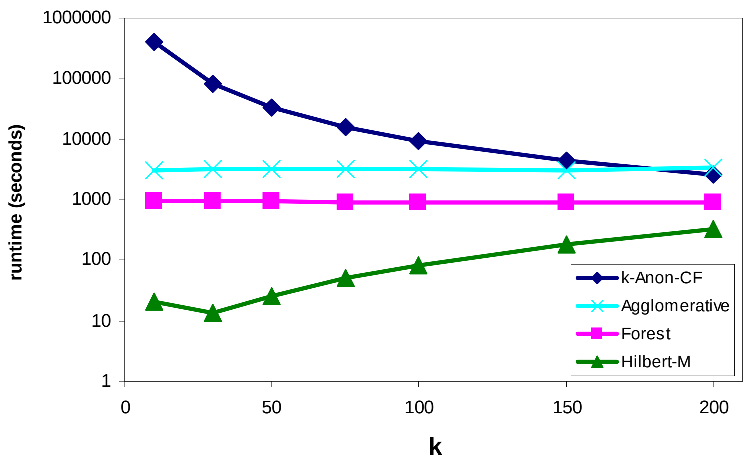


Fig. 5 Algorithm runtime comparison — Adult dataset

tion loss in the anonymizations produced by k-ANON-CF are also better than that in the anonymizations issued by the Agglomerative and modified Hilbert algorithms, which are heuristical algorithms with no approximation guarantee. This better performance in terms of information loss is accompanied by slower (but still practical) runtimes.

7.1.1 Scalability

We examined the scalability of k-ANON-CF with respect to two parameters — the size of the table, n, and the number of attributes r. In the first test we ran the algorithm over four artificial tables with 30, 60, 90, and 120 thousand records (and 14 quasi-identifiers) that were generated from the Adult database. According to these experiments we see that the number of generalized closed frequent itemsets grows roughly linearly with the table size (Figure 6 (a)). We also observe that the runtime of k-ANON-CF grows quadratically with the table size (Figure 6 (b)). The quadratic growth is due to the fact that the algorithm's runtime depends on the number of generalized closed frequent records and the time to compute the support of each such generalized closed frequent record also depends linearly on the table size.

In the second test we investigated the dependence of the runtime on the number of quasi-identifiers and the structure of the generalization hierarchies. To that end, we ran k-ANON-CF on four versions of the Adult dataset. The first was the original dataset, which has all 14 public attributes. The second was a reduced version that had 11 public attributes; we removed attributes fnlwgt, capital-gain and capital-loss. The third dataset had 8 public attributes; we further removed attributes education-num, hours-per-week and native-country. The fourth dataset had five public attributes; we further removed marital-status, relationship and race attributes. We also ran the algorithm on the full Adult dataset using "flat" hierarchies, namely, generalization by suppression only. Figure 7 displays the total algorithm runtime as a function of k in each of the above described tests. The vast differences in the running times as a function of the number of public attributes is due to the fact that the num-

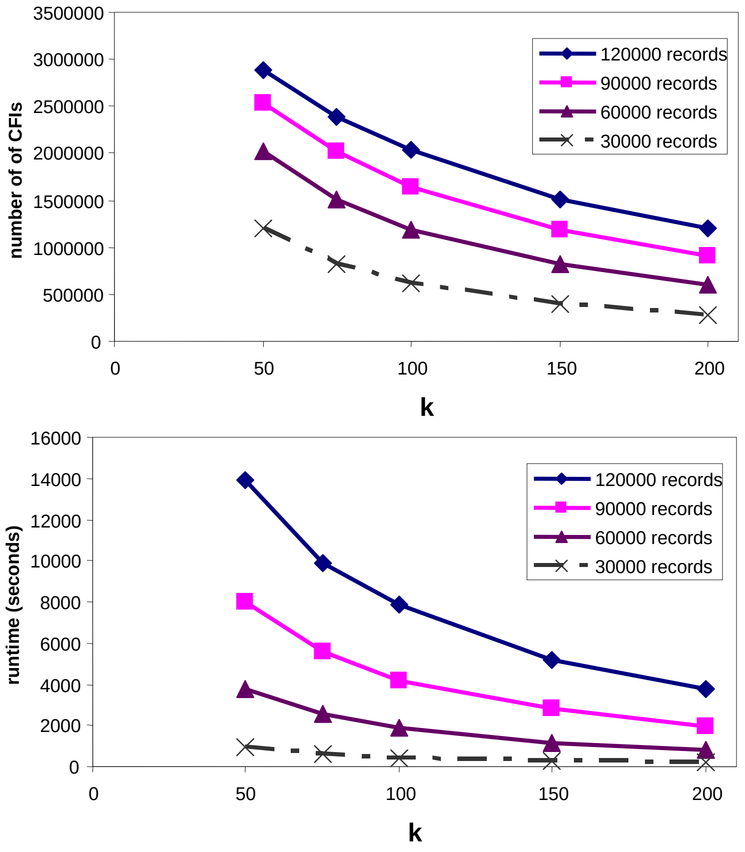


Fig. 6 Table size scaling: (a) Number of CFIs; (b) Runtime (seconds)

ber of closed frequent generalized records depends exponentially on the number of public attributes.

7.2 Nursery dataset experiments

We created taxonomies for each of the 8 attributes as follows:

1. Taxonomies of height 1 (suppression only): The hierarchies for attributes form, health, parents, finance, housing and social were defined as suppression only. The reason for this is that the values of these attributes are derived from a small domain.
2. Taxonomies of height 2 (one intermediate level of generalization):
   – children: 1-2 (2), 3-more (2).
   – has_nurs: not proper (2), critical (2).

Figure 8 displays the LM-scores achieved by the four algorithms on the Nursery dataset. Figure 9 displays the runtimes of those algorithms when run on this dataset.

Also here k-ANON-CF provides the best results in terms of information loss. The runtimes of k-ANON-CF here are much better since the Nursery dataset includes only 8 public

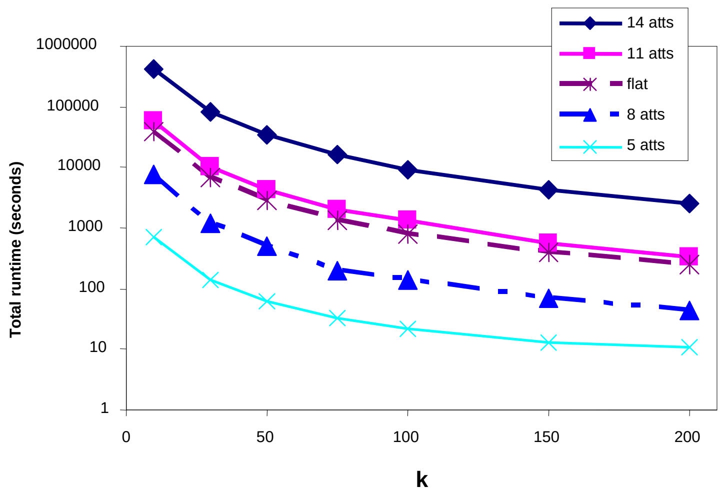


Fig. 7 k vs. total runtime — Adult dataset

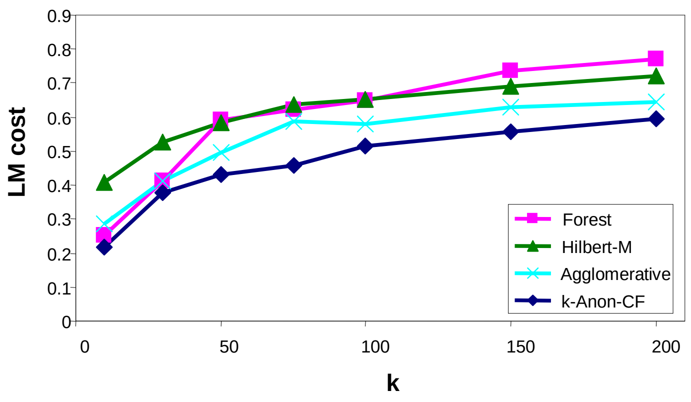


Fig. 8 Algorithm score comparison (LM) — Nursery dataset

attributes and, as exemplified earlier on the reduced versions of the Adult dataset, the number of public attributes affects dramatically the overall runtime of the algorithm.

7.3 Coil2000 dataset experiments

We created taxonomies of height 2 (suppressions only) for attributes 3 and 4, taxonomies of height 3 for attributes 2, 5, 6, 7, 8 and 9, and a taxonomy of height 4 for attribute 1. Figure 10 displays the LM-scores achieved by the four algorithms on this dataset, and Figure 11 displays the corresponding runtimes.

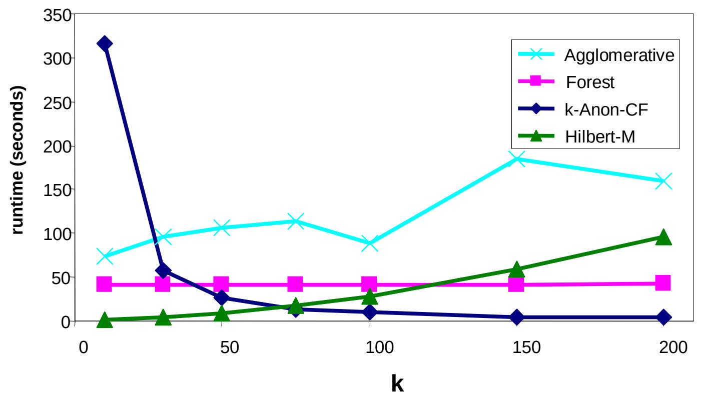


Fig. 9 Algorithm runtime comparison — Nursery dataset

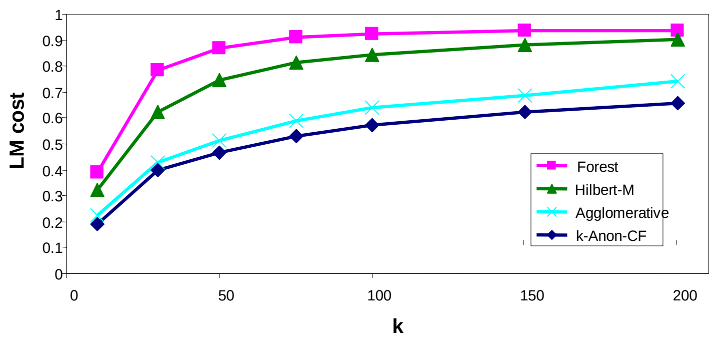


Fig. 10 Algorithm Comparison — Coil2000 dataset

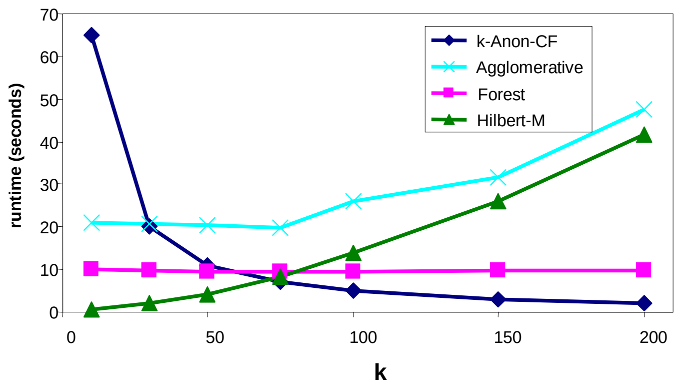


Fig. 11 Algorithm runtime Comparison — Coil2000 dataset

7.4 ℓ-diversity experiments

In Section 6.2 we described Algorithm 8 that is designed to post-process any anonymization by means of unifying clusters so that it issues anonymizations that respect ℓ-diversity. When applied on inputs that are already k-anonymized, it issues outputs that respect k-anonymity as well as ℓ-diversity. Each of the four algorithms with which we experimented in the previous sections is designed to produce k-anonymizations only. (The single exception is the Hilbert curve algorithm [6] that has also a version that achieves ℓ-diverse anonymizations; however, [6] does not present an algorithm for achieving anonymizations that respect the conjunction of the two privacy measures.) Coupling them with Algorithm 8 converts them into algorithms that respect k-anonymity as well as ℓ-diversity.

We tested all those algorithms with the three datasets. In the Adult dataset the private attribute is binary and the overall diversity in the entire table is 1.33. Hence, as discussed in Section 6, all anonymizations of that dataset are ℓ-diverse with $\ell \leq 1.33$. In the Nursery dataset the private attribute has 5 values and the overall diversity is 3. In the Coil2000 dataset, that has no distinguished private attribute, we used the tenth attribute as the private one. It has 9 values, but the overall diversity is only 3.46.

We ran Algorithm 8 with several values of w in the range $0 \leq w \leq 0.75$ (recall that setting w to values closer to 1 diminishes the effect of the diversity factor when deciding which clusters to unify, see Eq. (11)). We found out that in the range $0.15 \leq w \leq 0.75$ the sensitivity of the results to w was quite modest. In three of the experiments that we present below we used $w = 0.15$: Figure 12 shows the LM information loss on the Adult dataset, with $k = 50$, as a function of the constraint diversity parameter ℓ, as achieved by postprocessing the output of each of the four anonymization algorithms by means of Algorithm 8. Figure 13 shows a similar graph for the Coil2000 dataset with $k = 100$, and Figure 14 shows the results for the Nursery dataset with $k = 30$. Figure 15 shows the results in the same setting as the last one (i.e., the Nursery dataset with $k = 30$), this time using $w = 0$. Finally, Figure 16 shows the information losses in the k-ANON-CF algorithm for the Nursery dataset, for various values of k and ℓ.

As we can see from those figures, also when one imposes ℓ-diversity as an additional constraint, the k-ANON-CF algorithm usually keeps issuing the smallest information losses.

8 Conclusion

In this study we described a practical and general anonymization algorithm that approximates optimal k-anonymity to within a guaranteed factor of $O(\ln k)$. One of the main ingredients in that approximation algorithm is an algorithm for mining closed frequent generalized records. Experiments show that the proposed algorithm provides smaller information losses than the best known approximation algorithm as well as the best known heuristic algorithms.

This study raises three theoretical research problems:

(1) To devise approximation algorithms with an approximation guarantee that is smaller than $O(\log k)$, or to prove that the logarithmic approximation factor is optimal.

(2) The logarithmic approximation factor applies only to our basic k-anonymity algorithm; it does not hold for the modified version that satisfies also ℓ-diversity. To the best of our knowledge, no approximation guarantees were established thus far for algorithms that are designed to issue ℓ-diverse anonymizations. Can approximation factors be established for such algorithms?

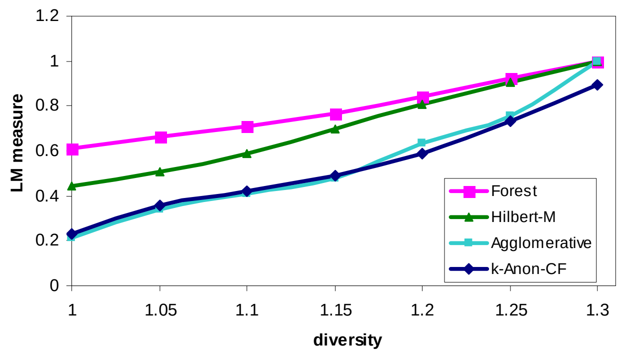


Fig. 12 Information loss in the Adult dataset as a function of the diversity

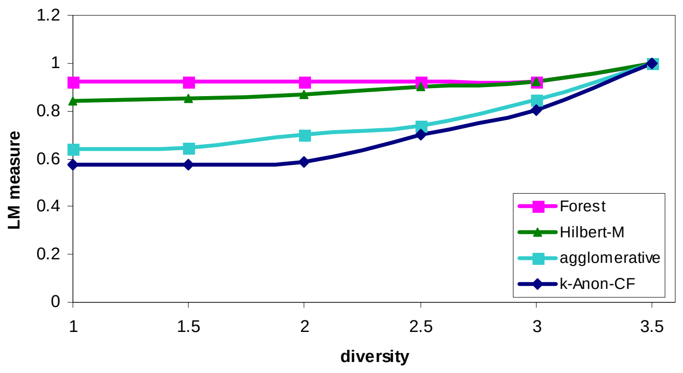


Fig. 13 Information loss in the Coil2000 dataset as a function of the diversity

(3) Are the assumptions of monotone measures of information loss or proper generalization operators essential for the proof of the approximation guarantee? While those two assumptions are quite natural in this context, they do not always apply. (For example, the mutual information measure of [9] is a non-monotone measure of information loss.)

Acknowledgements We would like to thank the first two authors of [6], Gabriel Ghinita and Panagiotis Karras, for providing us the source code of the Hilbert curve mapping.

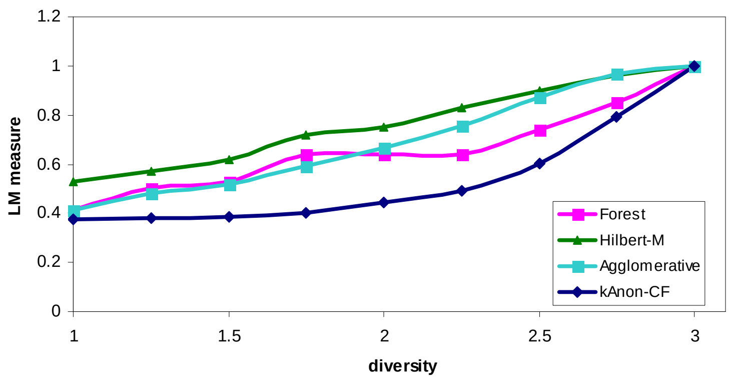


Fig. 14 Information loss in the Nursery dataset as a function of the diversity ($w = 0.15$)

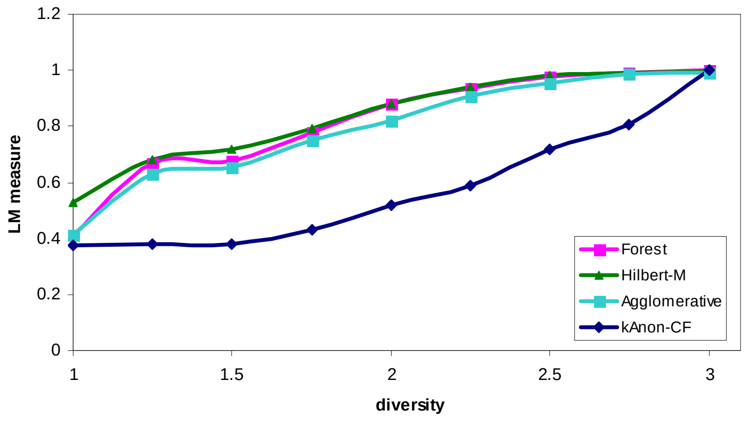


Fig. 15 Information loss in the Nursery dataset as a function of the diversity ($w = 0$)

References

1. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, pages 246–258, 2005.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.
3. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM-SIGMOD Conference on Management of Data*, pages 439–450, May 2000.

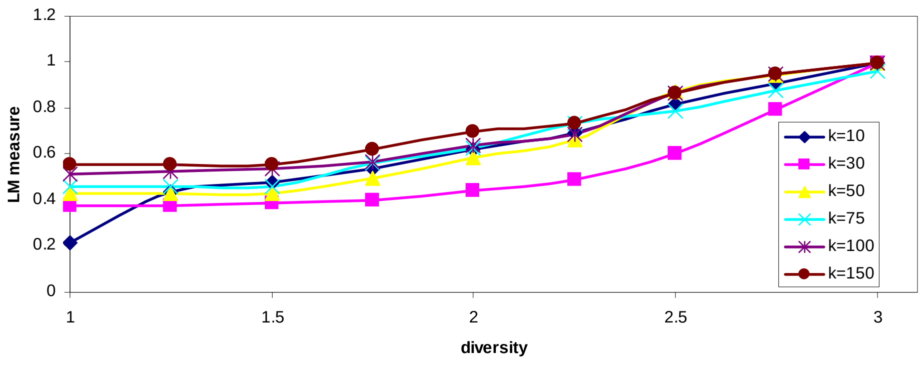


Fig. 16 Information loss in the k-ANON-CF algorithm on the Nursery dataset as a function of k and ℓ

4. R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *International Conference on Data Engineering (ICDE)*, pages 217–228, 2005.
5. J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymization using clustering techniques. In *DASFAA*, pages 188–200, 2007.
6. G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM Trans. Database Syst.*, 34(2), 2009.
7. A. Gionis, A. Mazza, and T. Tassa. k-anonymization revisited. In *International Conference on Data Engineering (ICDE)*, pages 744–753, 2008.
8. A. Gionis and T. Tassa. k-anonymization with minimal loss of information. *IEEE Trans. on Knowl. and Data Eng.*, 21(2):206–219, 2009.
9. J. Goldberger and T. Tassa. Efficient anonymizations with enhanced utility. *TDP*, 3:149–175, 2010.
10. G. Grahne and J. Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Trans. on Knowl. and Data Eng.*, 17(10):1347–1362, 2005.
11. C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):462–478, 2005.
12. V. Iyengar. Transforming data to satisfy privacy constraints. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288, 2002.
13. K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *ACM-SIGMOD Conference on Management of Data*, pages 49–60, 2005.
14. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.
15. N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proceedings of IEEE International Conference on Data Engineering (ICDE) 2007*, pages 106–115, 2007.
16. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.
17. A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228, 2004.
18. M. Nergiz and C. Clifton. Thoughts on k-anonymization. *Data Knowl. Eng.*, 63(3):622–645, 2007.
19. H. Park and K. Shim. Approximate algorithms for k-anonymity. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 67–78, 2007.
20. J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD*, pages 21–30, 2000.

21. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.
22. P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, page 188, 1998.
23. R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.
24. L. Sweeney. Uniqueness of simple demographics in the U.S. population. *Laboratory for international Data Privacy (LIDAP-WP4)*, 2000.
25. L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
26. T. Truta, A. Campan, and P. Meyer. Generating microdata with p-sensitive k-anonymity property. In *SDM*, pages 124–141, 2007.
27. R. Wong, J. Li, A. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *In ACM SIGKDD*, pages 754–759, 2006.
28. X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, pages 139–150, 2006.