Limiting Disclosure of Sensitive Data in Sequential Releases of Databases

Erez Shmueli[a,c], Tamir Tassa[b,c], Raz Wasserstein[a], Bracha Shapira[a], Lior Rokach[a]

[a] *Deutsche Telekom Laboratories and the Department of Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva, Israel*
[b] *Division of Computer Science, The Open University, Ra'anana, Israel*
[c] *E. Shmueli and T. Tassa contributed equally to this work.*

Abstract

Privacy Preserving Data Publishing (PPDP) is a research field that deals with the development of methods to enable publishing of data while minimizing distortion, for maintaining usability on one hand, and respecting privacy on the other hand. Sequential release is a scenario of data publishing where multiple releases of the same underlying table are published over a period of time. A violation of privacy, in this case, may emerge from any one of the releases, or as a result of joining information from different releases. Similarly to [37], our privacy definitions limit the ability of an adversary who combines information from all releases, to link values of the quasi-identifiers to sensitive values. We extend the framework that was considered in [37] in three ways: We allow a greater number of releases, we consider the more flexible local recoding model of "cell generalization" (as opposed to the global recoding model of "cut generalization" in [37]), and we include the case where records may be added to the underlying table from time to time. Our extension of the framework requires also to modify the manner in which privacy is evaluated. We show that while [37] based their privacy evaluation on the notion of the Match Join between the releases, it is no longer suitable for the extended framework considered here. We define more restrictive types of join between the published releases (the Full Match Join and the Kernel Match Join) that are more suitable for privacy evaluation in this context. We then present a top-down algorithm for anonymizing sequential releases in the cell generalization model, that is based on our modified privacy evaluations. Our theoretical study is followed by experimentation that demonstrates a staggering improvement in terms of utility due to the adoption of the cell generalization model, and exemplifies the correction in the privacy evaluation as offered by using the Full or Kernel Match Joins instead of the Match Join.

Keywords: Privacy Preserving Data Publishing, Sequential Release, Continuous Data Publishing, Multipartite Graphs, Matching

1. Introduction

Vast amount of information of all types is collected daily about people by governments, corporations and individuals. The information is collected, for example, when users register to or use online applications, receive health related services, use their mobile phones, utilize search engines, or perform common daily activities. As a result, there is an enormous quantity of privately-owned records that describe individuals' finances, interests, activities, and demographics. These records often include sensitive data and may violate the privacy of the users if published. In today's global network of organizational connections, the growing demand to disseminate and share this information is motivated by various academic, commercial and other benefits. This information is becoming a very important resource for many systems and corporations that may analyze the data in order to enhance and improve their services and performance. One common practice for publishing such data without violating privacy is applying regulations, policies and guiding principles for the use of the data. Such regulations usually entail data distortion for the sake of anonymizing the data. This approach is problematic since, on one hand, data leakage (intentional or unintentional) can still occur, and, on the other hand, the data may become unusable after distortion.

The recent developing research field of Privacy Preserving Data Publishing (PPDP) is targeting this challenge. It aims at developing techniques that enable publishing data while minimizing distortion for maintaining usability on one hand, and ensuring that privacy is preserved on the other hand. (The reader is referred to the recent survey [12] for further details.)

A closely related research area is Privacy Preserving Data Mining (PPDM) that was initiated in 2000 by [2]. For detailed surveys about PPDM see [1, 36]. PPDM algorithms aim at anonymizing data towards its release for specific data mining goals, so that distortion is minimized, on one hand, and privacy is preserved on the other hand. The developed algorithms are tailored to the relevant data mining goals. For example, if the data needs to be used for learning a classifier, the corresponding PPDM algorithm will aim at achieving anonymization while incurring a minimal loss of accuracy in the resulting classifier. In PPDP, on the other hand, the purposes of the data release are unknown and it is needed to anonymize the data using utility measures that are not targeted to specific data mining tasks.

PPDP basically assumes [7] that the candidate table to be published includes four types of attributes: explicit identifiers — attributes that uniquely identify an individual (e.g. `S.S.N.`); quasi-identifiers — attributes that do not offer unique identification but their combination might yield unique identification by means of linking attacks (e.g., `zipcode`, `age`, `gender`); sensitive data — personal attributes of private nature, such as health condition or financial data; and other attributes that are non-sensitive, on one hand, and cannot be used for identification on the other hand. (We refer hereinafter to the latter type of attributes as non-identifiers.) The usual practice in PPDP is to remove the identifiers and to generalize the quasi-identifiers in order to protect the sensitive data of individuals from being revealed. Generalization replaces a value with a subset of values of the relevant at-

tribute, typically a parent value in a taxonomy (a hierarchical generalization tree) of that attribute. In such practices, the sensitive data is usually retained intact in order to allow meaningful data mining on the anonymized data.

In the past years, several models were suggested for maintaining privacy when disseminating data. Most approaches evolved from the basic model of k-anonymity [34]. Algorithms for k-anonymization include those of [22, 32, 33] that are designed for obtaining minimum distortion, and [5, 14, 18] that are intended for classification. Variations and alternatives of k-anonymity were also studied. For example, [24] proposed the model of ℓ-diversity to address attacks that are based on lack of diversity in the sensitive data; t-closeness [9, 23] was suggested in order to maintain the distribution of the sensitive data; [31] proposed a privacy model similar to t-closeness called *average privacy risk*; [39, 38] proposed to limit the confidence of inferring a sensitive property for a group of individuals; and [45] proposed the notion of personalized anonymity.

Early models, such as k-anonymity or ℓ-diversity, referred to a scenario in which a single release is considered to be published. In practical applications, however, data publishing is more complicated and may involve several releases. This may occur, for example, when new information becomes available or when various parts of the data are published to different data recipients. A violation of privacy, in this case, may emerge from any one of the releases, or as a result of joining information from different releases. Several scenarios of publishing multiple releases are identified in [12]:

(1) Multiple Release Publishing [47, 20, 4, 11]: Several releases (views) of the same underlying table are published at one time.

(2) Sequential Release Publishing [37]: Different projections of a given table on different subsets of attributes are released in a sequential manner. For example, an organization makes a new release when new attributes become available, releases a separate view for each data sharing purpose (such as classifying a different target variable) [14, 18, 40], or makes separate releases for personally-identifiable data (e.g., names) and sensitive data (e.g., DNA sequences) [25]. The goal in such scenarios is to anonymize the new release so that its combination with all previous releases would be anonymous.

(3) Continuous Data Publishing [6, 8, 13, 30, 43, 46]: Several releases of the same underlying table are published over a period of time, where in between releases, records are inserted to or deleted from the table.

In the two last scenarios, several releases of partial views of the same basic table are published in a sequential manner. Releases that were already published cannot be modified. The goal is to anonymize the next release so that the combination of information from all releases does not lead to a privacy breach. In the sequential release scenario, the set of records (rows) is fixed, while the set of attributes (columns) changes from one release to another; in continuous data publishing, the set of attributes is fixed while the set of records is dynamic. In this study we consider a combination of the two scenarios. Most of the

study is devoted to the sequential release scenario, that was studied so far only in [37]. We then extend the discussion to include cases in which records may be added to the table.

As in [37], we consider two privacy goals: k-linkability and k-diversity. k-Linkability mandates that even if an adversary combines information from all releases of the underlying table, he would not be able to link any selection of values of the quasi-identifiers to less than k distinct values of the sensitive attribute. k-Diversity demands that such an adversary would not be able to link any selection of values of the quasi-identifiers with any sensitive value with probability greater than $1/k$.[1] Given a desired linkability or diversity level, k, the problem of anonymizing sequential release is as follows: Assume that the data holder has previously published several releases of the table, where each release is a generalized view of the table. The data holder wishes to issue a new release. The goal is to find the most "useful" generalized release, bearing in mind that previous releases that were already published can no longer be modified, so that the required level of privacy is still respected. We illustrate these ideas in the following example.

Example 1.1. Consider the table and its corresponding two releases that are given in Table 1. The table has two quasi-identifiers, `age` and `gender`, and one sensitive attribute, `disease`. Clearly, each of the two releases, on its own, satisfies 2-linkability, as well as 2-diversity; indeed, each release enables to link any specific value of the quasi-identifiers to exactly two sensitive values with equal probabilities. For instance, if an adversary wishes to find sensitive information about Alice, a female of age 30, then the first release would allow him to infer that she has either flu or angina, while the second release would allow him to infer that she has either hepatitis or angina. However, the sequential release that consists of those two releases satisfies only 1-linkability/diversity since if the adversary is exposed to both releases, he may combine the information that is disclosed by each of them and find that Alice has angina. Therefore, in order to retain 2-linkability/diversity, the second release would have to be further generalized.

age	gender	disease
20	male	measles
20	female	hepatitis
30	male	flu
30	female	angina

age	disease
20	measles
20	hepatitis
30	flu
30	angina

gender	disease
male	measles
female	hepatitis
male	flu
female	angina

Table 1: A table (left) and two corresponding releases (middle and right)

□

Two naïve solutions exist for this scenario [33, 34]: (1) Anonymizing the underlying table once and then publishing releases based on the anonymized table; and (2) Generalizing each

[1] Our terminology differs from that of [37]: We used "linkability" instead of "anonymity, since k-anonymity is a notion of privacy that is oblivious of the sensitive values; we used "diversity" instead of "linkability" since that is the more accepted term [24].

new release based on previous releases by guaranteeing that each value in the new release is not more specialized than the same value as published in previous releases. However, both of these solutions imply excessive generalizations that increase the information loss and hence reduce the utility of the releases unnecessarily (see [37]).

[37] provided a formal definition of the sequential release privacy problem and developed a Top-Down Specialization Algorithm for Anonymizing Sequential Releases (TDS4ASR). Their work concentrated on the case in which only one previous release was published, and the data holder wishes to publish a second release of the same table while maintaining the desired level of linkability. The algorithm generalizes the second release so that the required privacy goal is met. [37] considered a generalization model which is called "cut generalization". In that model, every attribute has a corresponding taxonomy and the generalization of any given attribute in any release is a global recoding generalization that corresponds to a cut in the graph representation of the taxonomy tree. Herein, we extend the framework that was considered in [37] in three ways: We allow a greater number of releases, we consider the more flexible local recoding model of "cell generalization", in which each cell may be generalized independently, and we allow adding records to the underlying table in between releases. We show that, owing to its flexible local nature, cell generalization implies smaller information losses and hence it offers substantially better utility results than cut generalization.[2]

The above described extension of the framework requires also to modify the manner in which the level of privacy (linkability or diversity) is evaluated. In [37], the authors based their evaluation on the notion of the Match Join (MJ). We show that when using cell generalization, or when the number of releases is greater than two, the MJ is no longer suitable for such evaluation. We define a more restrictive type of join between the published releases, which we call the Full Match Join (FMJ), and show that the privacy level must be evaluated based on that join. Specifically, when using the MJ (rather than the FMJ) in order to assess the level of privacy that is delivered by the sequential release, one may be led to excessive (namely, too optimistic) estimates. Switching from the MJ to the FMJ provides the required remedy.

Unfortunately, computing the FMJ for more than two releases is an NP-Hard problem, as we prove later on. We therefore define an intermediate type of join, called the Kernel Match Join (KMJ), and prove that basing the evaluation on it, rather than on the MJ, provides better approximation of the actual level of privacy. The KMJ, as opposed to the FMJ, can be computed in polynomial time.

Our theoretical study is followed by extensive experimentation. We tested our algorithm, which is designed to evaluate privacy based on the FMJ (in the case of two releases)

[2]Standard data mining tools cannot be applied directly on tables that were generalized using local recoding, since such tables, as opposed to tables that were generalized using global recoding, include intersecting subsets of values. The recent study [21] discusses that issue and proposes a way for preprocessing such tables for data mining purposes. It is also shown there that the lower information losses that characterize local recoding generalizations yield more accurate data mining results.

or KMJ (for more than two releases) and achieving the required level of privacy using the cell generalization model, on several large datasets. Our experiments demonstrate the corrected level of linkability or diversity as measured by our modified evaluations, as well as a staggering improvement in terms of utility due to the adoption of the cell generalization model.

The paper is organized as follows: Section 2 provides basic definitions. In Section 3 we define the three types of joins, discuss their properties and present algorithms for their computation. In Section 4 we define the notions of k-linkability and k-diversity and the corresponding privacy problem in sequential releases. Then, we describe in Section 5 an algorithm for computing sequential releases that comply with either k-linkability or k-diversity while minimizing the information loss. In Section 6 we extend our discussion to include dynamically changing tables to which new records may be added from time to time. Section 7 portrays the experimental evaluation and its results. Finally, we conclude in Section 8 and describe some future research directions.

2. Basic Definitions

2.1. Releases of a table and their graph representation

In this section we define the basic notions of a release and a sequential release of a given database table. For convenience, we include in Section 9.1 in the Appendix a table that summarizes the main notations that we introduce here and in the next section.

Let $A_1, \ldots, A_M$ be M attributes and $T = \{S_1, \ldots, S_N\}$ be a table of N records in $A_1 \times \cdots \times A_M$. Here, A_m, $1 \leq m \leq M$, denotes the m-th attribute as well as the domain in which it takes values. Hence, the n-th record in T is $S_n = (S_n(1), \ldots, S_n(M))$ and $S_n(m) \in A_m$, $1 \leq m \leq M$. We shall assume that the quasi-identifiers are $A_1, \ldots, A_t$, the non-identifiers are $A_{t+1}, \ldots, A_{M-1}$, and the sensitive attribute is A_M. (We make the common assumption of a single sensitive attribute; the generalization to scenarios with more sensitive attributes dictates itself.)

For each $1 \leq m \leq M$, we let $\overline{A}_m$ denote a collection of subsets of A_m that covers the entire set A_m. A generalization of the table T is a table of the form $\overline{T} = \{\overline{S}_1, \ldots, \overline{S}_N\}$ where the n-th generalized record in $\overline{T}$ is $\overline{S}_n = (\overline{S}_n(1), \ldots, \overline{S}(M)) \in \overline{A}_1 \times \cdots \times \overline{A}_M$ and, in addition, the following inclusion holds: For all $1 \leq n \leq N$ and $1 \leq m \leq M$, the subset $\overline{S}_n(m) \subseteq A_m$, which is one of the subsets in the corresponding collection of subsets $\overline{A}_m$, includes the original element $S_n(m)$.

A sequential release of T is a sequence $T_1, \ldots, T_R$ of releases of T, where each release T_r, $1 \leq r \leq R$, is a generalized view of T. (The generalization schemes, namely, the collections $\overline{A}_1, \ldots, \overline{A}_M$ of permissible subsets, may be different for different releases.) Typically in sequential releases, every release contains a different subset of the attributes, as dictated by the purposes of the release. If $I_r \subset [M] := \{1, \ldots, M\}$ denotes the subset of attribute indices that appear in the r-th release, and $\{S_1^r, \ldots, S_N^r\}$ are the generalized records in T_r, $1 \leq r \leq R$, then for every attribute index $m \in [M] \setminus I_r$, the m-th column in T_r is suppressed; namely, $S_n^r(m) = *$ for all $1 \leq n \leq N$ and $m \in [M] \setminus I_r$.

Example 2.1. Assume that T is the table given in Table 2 which consists of $N = 4$ records and $M = 4$ attributes — `age`, `zipcode`, `occupation`, and `disease`. Here, there are $t = 3$ quasi-identifiers (being the first three attributes) and one sensitive value. A_1, for example, denotes here the `age` attribute as well as the domain in which it takes values (say, $A_1 = \{0, 1, 2, \ldots, 120\}$). Assume the following generalization rules:

- In the `age` attribute we allow any generalization in the form of an interval of ages. Hence, $\overline{A}_1$ in this case is the set of all intervals of the form $[a, b]$ where $a, b \in A_1$ and $b \geq a$.
- In `zipcode` we allow any generalized zipcode in the form of a prefix (e.g., the exact zipcode 53120 may be generalized to any of the five prefix zipcodes 5312*, 531**, 53***, 5****, or *****).
- In `occupation` we allow generalization by suppression only (i.e., either the original value is retained, or it is totally suppressed). Hence, here $\overline{A}_3 = A_3 \cup \{A_3\}$; namely, the generalized subsets can be any of the singleton values in A_3 or the entire set (which corresponds to total suppression).
- In `disease` we allow no generalization. Namely, $\overline{A}_4 = A_4$.

Then Table 3 exemplifies two possible releases of T. The first one, T_1, includes attributes A_1 and A_2, while the second, T_2, includes attributes A_2, A_3 and A_4.

T	`age`	`zipcode`	`occupation`	`disease`
S_1:	30	53120	teacher	measles
S_2:	30	53425	engineer	hepatitis
S_3:	35	53890	singer	flu
S_4:	40	53764	actor	angina

Table 2: Basic table T

T_1	`age`	`zipcode`	`occupation`	`disease`
S_1^1:	30	53120	*	*
S_2^1:	30	53425	*	*
S_3^1:	35	53890	*	*
S_4^1:	40	53764	*	*

T_2	`age`	`zipcode`	`occupation`	`disease`
S_1^2:	*	53120	*	measles
S_2^2:	*	53***	*	hepatitis
S_3^2:	*	53890	singer	flu
S_4^2:	*	53***	actor	angina

Table 3: Releases T_1 (left) and T_2 (right) of T

□

Next, we define the important notion of *consistency* between records of different releases.

Definition 2.1. *Let T_r and $T_{r'}$ be two different releases of the same table T. The records $S_n^r \in T_r$ and $S_{n'}^{r'} \in T_{r'}$ are called consistent if $S_n^r(m) \cap S_{n'}^{r'}(m) \neq \emptyset$ for all $m \in [M]$.*

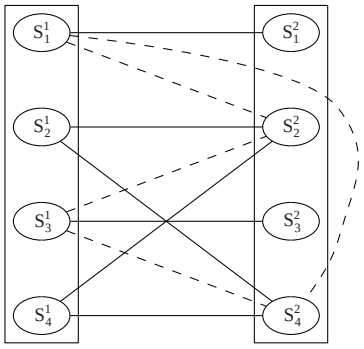


Figure 1: The graph to Example 2.2

Considering the two releases T_1 and T_2 of the basic table T in Example 2.1, the only attribute that appears in both releases is A_2 (`zipcode`). Hence, in order to check consistency between records in these two releases, it suffices to concentrate on the values of that attribute. The record S_1^1, for example, is consistent with S_1^2, S_2^2 and S_4^2, but it is not consistent with S_3^2.

If two records are not consistent, we may infer with certainty that they could not have originated from the same record in the basic table T. Indeed, if $S_n^r \in T_r$ and $S_{n'}^{r'} \in T_{r'}$ did originate from the same record in T, say S_1, then for all $m \in [M]$ we should have $S_1(m) \in S_n^r(m)$ and $S_1(m) \in S_{n'}^{r'}(m)$.

The notion of consistency gives rise to the following definition of the multipartite graph that corresponds to a given sequential release. Let $T_1, \ldots, T_R$ be R releases of the same table T, where $T_r = \{S_1^r, \ldots, S_N^r\}$, $1 \leq r \leq R$. Such a sequential release defines a multipartite consistency graph $G_{T[R]}$ on the set of nodes $T[R] = T_1 \cup \cdots \cup T_R$: An edge connects the records S_n^r and $S_{n'}^{r'}$ if and only if $r \neq r'$ and the records are consistent.

Among the edges of the graph, some connect records that originate from the same original record (such records are obviously consistent) and some connect records that happen to be consistent even though they did not originate from the same original record. We refer to the edges of the first kind as *horizontal*, since if we assume that the order of records in all releases is retained, those edges are graphically horizontal.

Example 2.2. Consider the basic table T and the two releases T_1 and T_2 as given in Tables 2 and 3. Then the corresponding graph in that case is given in Figure 1. That graph has four horizontal edges and six non-horizontal edges. (Some of the non-horizontal edges are denoted in the figure by solid lines and some are denoted by broken lines; the distinction between those two types of edges will be clarified later on.) □

2.2. Models of generalization

We define here two models of generalization that are based on hierarchical generalization trees (or taxonomies). Let $\mathcal{T}_m$ be a taxonomy for attribute A_m, $1 \leq m \leq M$; namely,

$\mathcal{T}_m$ is a tree in which each node represents a subset of A_m, the leaves are the singleton subsets, the root is the entire set and the edges describe the inclusions between the subsets. In the first model, called *cell generalization*, each entry in the m-th column of T may be generalized independently to any of the subsets in $\mathcal{T}_m$ that includes it (namely, the collection $\overline{A}_m$ of permissible subsets for generalization is all of $\mathcal{T}_m$). The second model, called *cut generalization*, is more restrictive. Its definition is based on the notion of a cut in a taxonomy.

Definition 2.2. *Let A be an attribute and $\mathcal{T}$ be a corresponding taxonomy. A cut in $\mathcal{T}$ is any selection of disjoint subsets from $\mathcal{T}$ whose union equals A.*

Example 2.3. Consider the hierarchical generalization tree in Figure 2 for an attribute of clothing items. Then [Footwear, Clothes] and [Footwear, Shirts, Pants, Outdoors] are two cuts in the tree. □

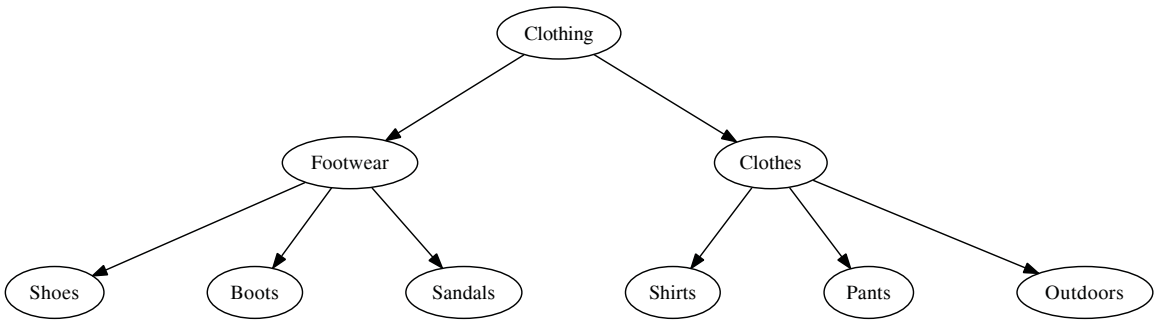


Figure 2: A hierarchical clustering tree (taxonomy)

In the case of cut generalization, the collection of subsets $\overline{A}_m$ for attribute A_m in each release T_r is some cut in the underlying taxonomy $\mathcal{T}_m$. Namely, the collection of subsets $\overline{A}_m$ may change from one release to another, but they are all cuts in the same taxonomy $\mathcal{T}_m$.

Note that in the case of cut generalization, all entries of the table T that have the same value will always be generalized in the same manner in any given release. For example, if one of the attributes that appears in a given release is `age`, then all entries with the value `age`=27 will always be generalized in the same manner (e.g., will be replaced by [20-29]). This generalization model is usually called *global recoding* (e.g. [5, 18, 22, 42]). In the case of cell generalization, we are flexible to generalize independently each value in the table's m-th column with any of the subsets in $\mathcal{T}_m$ that contains it. Such a model is usually called *local recoding* (e.g. [16, 27, 29, 42]). Local recoding is more flexible than global recoding, in the sense that the set of all generalizations of a given table in the local recoding model is a proper superset of the set of all generalizations of the same table in the global recoding model. Hence, local recoding is a preferable model since it allows achieving a certain privacy goal with lower costs in terms of information loss.

The sensitive attribute is typically not subjected to generalization. It is either included in a release, and then it appears fully specialized, or it does not, and then it is totally suppressed. The corresponding taxonomy $\mathcal{T}_M$ of A_M is therefore a trivial taxonomy that consists of the root, that corresponds to the whole set A_M, and the leaves, that correspond to all singleton values in A_M. In releases where A_M is included, we apply the trivial generalization that keeps all values in the leaf level of $\mathcal{T}_M$ (no generalization), while in other releases we generalize all values to the root value (total suppression).

3. Match Joins

3.1. The match join and the full match join

In a given sequential release of a table, the records in each release do not appear in any particular order. An adversary who wishes to link the values of the sensitive attribute to values of the quasi-identifiers needs first to find which records in the different releases correspond to the same original record in T.

Definition 3.1. *Two records in two different releases, $S_n^r \in T_r$ and $S_{n'}^{r'} \in T_{r'}$, are called siblings if they originated from the same record in T (namely, they are connected by a horizontal edge).*

For each $1 \leq n \leq N$, we let C_n^H denote the set of R records in $T_1, \ldots, T_R$ that originated from $S_n \in T$. Clearly, as all records in C_n^H are siblings, they are all consistent with each other. Therefore, C_n^H is a clique of size R (R-clique hereinafter) in the multipartite graph $G = G_{T[R]}$. (We refer to those cliques as *horizontal* cliques since all edges in them are horizontal.) Moreover, the collection $\{C_1^H, \ldots, C_N^H\}$ of all such R-cliques is a perfect matching in G:

Definition 3.2. *A collection of N R-cliques in $G = G_{T[R]}$ is called a perfect matching if those cliques are disjoint and their union covers all of the nodes in G.*

In view of the above, an adversary who examines the sequential release $T[R]$ may construct the multipartite consistency graph $G = G_{T[R]}$ and then, if an edge $\{S_n^r, S_{n'}^{r'}\}$ in that graph is not part of a perfect matching, he may deduce that the records S_n^r and $S_{n'}^{r'}$ cannot be siblings. On the other hand, every perfect matching in the graph represents a possible siblinghood linkage between the records in the sequential release. The adversary aims at learning the correct siblinghood linkage in the graph (namely, the perfect matching $\{C_1^H, \ldots, C_N^H\}$) since then he can make the correct linkage between values of the quasi-identifiers and values of the sensitive attribute. The goal of the data owner is to hide as much as possible that linkage. Namely, it is desired to hide the true perfect matching among a sufficiently large number of other perfect matchings so that the linkage between values of the quasi-identifiers and the sensitive attribute is obfuscated. Later on we state that goal more formally.

Consider, for example, the table T and releases T_1 and T_2 of Example 2.1. Each edge in the graph that is depicted in Figure 1 denotes a relation of consistency between records of the two releases. Of those edges, only the four horizontal ones connect sibling records. In that graph there are exactly two perfect matchings:

$$\{S_1^1, S_1^2\}\ ,\quad \{S_2^1, S_2^2\}\ ,\quad \{S_3^1, S_3^2\}\ ,\quad \{S_4^1, S_4^2\}\,, \tag{1}$$

and

$$\{S_1^1, S_1^2\}\ ,\quad \{S_2^1, S_4^2\}\ ,\quad \{S_3^1, S_3^2\}\ ,\quad \{S_4^1, S_2^2\}\,. \tag{2}$$

Therefore, an adversary who sees the two releases may deduce that all edges that are not part in neither of those perfect matchings do not connect sibling records (those edges are denoted by broken lines in Figure 1). However, he may not determine whether the sibling of S_2^1 is S_2^2 or S_4^2.

The above discussion motivates the following definition.

Definition 3.3. *Let $T_1, \ldots, T_R$ be R releases of the same table T and let $G = G_{T[R]}$ be the corresponding multipartite graph on the set of nodes $T[R] = T_1 \cup \cdots \cup T_R$.*

(1) An R-clique in G is called admissible if it is included in a perfect matching in G.

(2) The collection of all R-cliques in G is called the Match Join (MJ hereinafter) of $T_1, \ldots, T_R$ and it is denoted $\mathcal{MJ}_{T[R]}$.

(3) The collection of all admissible R-cliques in G is called the Full Match Join (FMJ hereinafter) of $T_1, \ldots, T_R$ and it is denoted $\mathcal{FMJ}_{T[R]}$.

We note that the notion of MJ as we defined in Definition 3.3 is equivalent to the join notion that was considered in [37].

Going back to Example 2.2 and the corresponding graph in Figure 1, $\mathcal{MJ}_{T[2]}$ in this case is the set of all ten edges in the graph, while $\mathcal{FMJ}_{T[2]}$ consists of just the six admissible edges that are denoted by solid lines.

Comment. Given a multipartite graph, we distinguish between two problems that relate to perfect matchings in it: (1) counting (or listing) all perfect matchings in that graph; and (2) computing its FMJ, namely, the subset of all edges in that graph that are part of any perfect matching. In this paper we are interested in the second problem only. That problem can be solved in polynomial time in bipartite graphs, but it is NP-hard in multipartite graphs with three or more parts (see Theorem 3.4). The first problem, on the other hand, is hard even in bipartite graphs. More specifically, the task of counting the number of perfect matchings in a bipartite graph is equivalent to the problem of computing the permanent of a binary matrix. The latter problem is #P-complete in the worst case as well as in the average case. The reader is referred to [35] for an elaborate discussion on this problem.

The rest of this section is organized as follows: In Section 3.1.1, we discuss the notions of MJ and FMJ in the case of two releases, $R = 2$. In Section 3.1.2, we prove that MJ coincides with FMJ in the special case which was considered in [37], namely, the case of cut generalization and $R = 2$ (Corollary 3.3). Then, in Section 3.1.3, we give two examples where MJ differs from FMJ that illustrate the necessity of the two conditions in Corollary 3.3. The first example is for the case of cut generalization and $R = 3$; the second example is for the case of cell generalization and $R = 2$.

3.1.1. MJ and FMJ for $R = 2$

When $R = 2$, an R-clique is just an edge. Hence, MJ is the set of all edges while FMJ is the subset of edges that are included in a perfect matching in the bipartite graph. Such edges were called *matches* in [15]. We shall adopt this term here in the following manner:

Definition 3.4. *Let $G_{r,r'}$ denote the bipartite graph that is obtained by restricting G to the subset of nodes $T_r \cup T_{r'}$, where $1 \leq r < r' \leq R$. An edge $e = \{S_n^r, S_{n'}^{r'}\}$ is called a match if it is an admissible 2-clique in $G_{r,r'}$; namely, it is included in a perfect matching in $G_{r,r'}$.*

In [35] a polynomial-time algorithm for detecting all matches in a bipartite graph was presented. It uses Tarjan algorithm for finding the strongly connected components in a related directed graph. It runs in linear time with respect to the number of nodes and number of edges in the bipartite graph. Therefore, the problem of computing the FMJ has an efficient solution when $R = 2$. Computing the FMJ when $R > 2$ is discussed in Section 3.2.

3.1.2. The case of cut generalization

Definition 3.5. *Let α and β be two values in the hierarchical generalization tree of A_m. Then α and β are on the same generalization path, denoted $\alpha \uparrow \beta$, if $\alpha = \beta$ or one of them is an ancestor of the other.*

Definition 3.6. *The complete bipartite graph $K_{d,d}$ is the bipartite graph where the set of nodes is $U = U_1 \cup U_2$, $|U_1| = |U_2| = d$, and every node in U_1 is connected to every node in U_2. A bipartite graph is called decomposable if each of its connected components is a complete bipartite graph.*

The graph in Figure 3 is decomposable since it has two connected components, which are $K_{3,3}$ and $K_{2,2}$.

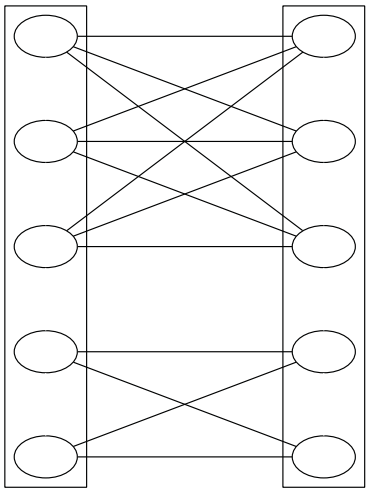


Figure 3: A decomposable bipartite graph

Our main result in that regard is the following theorem, whose proof is postponed to the Appendix (see Section 9.2 there).

Theorem 3.1. *In the case of cut generalization, $G_{r,r'}$ is decomposable for all $1 \leq r < r' \leq R$.*

Clearly, every edge in a decomposable bipartite graph is part of some perfect matching. Hence, an immediate consequence of Theorem 3.1 is as follows.

Corollary 3.2. *In the case of cut generalization, all edges are matches.*

From Corollary 3.2 we arrive at the following conclusion.

Corollary 3.3. *If $R = 2$, then in the case of cut generalization MJ=FMJ.*

3.1.3. MJ $\neq$ FMJ

Here we give two examples where MJ differs from FMJ. Example 3.1 is for the case of cut generalization and $R = 3$; hence, that example illustrates the necessity of the condition $R = 2$ in Corollary 3.3. Example 3.2 is for the case of cell generalization and $R = 2$; hence, it illustrates the necessity of the assumption about cut generalization in Corollary 3.3.

Example 3.1. Let $A_1 = \{a, b\}$, $A_2 = \{x, y\}$ and $A_3 = \{1, 2\}$ be three attributes and consider the table T and the corresponding releases T_1, T_2, T_3 as given in Table 4. Note that those releases comply with the cut generalization model (in which the taxonomies are of height 2). The corresponding tripartite graph is depicted in Figure 4.

T	T_1	T_2	T_3
$(a, x, 1)$	$(a, x, *)$	$(*, x, 1)$	$(a, *, 1)$
$(b, x, 2)$	$(b, x, *)$	$(*, x, 2)$	$(b, *, 2)$
$(a, y, 2)$	$(a, y, *)$	$(*, y, 2)$	$(a, *, 2)$

Table 4: A table T and corresponding three releases obtained by cut generalizations

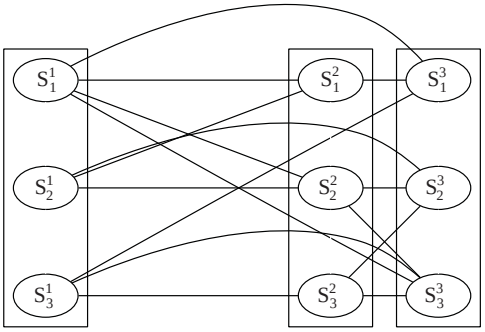


Figure 4: The graph to Example 3.1

That graph has only one perfect matching that consists of the three horizontal 3-cliques. Hence,

$$\mathcal{FMJ}_{T[3]} = \{\{S_1^1, S_1^2, S_1^3\}, \{S_2^1, S_2^2, S_2^3\}, \{S_3^1, S_3^2, S_3^3\}\}. \tag{3}$$

However, as the graph contains one more (non-admissible) clique — $\{S_1^1, S_2^2, S_3^3\}$, we have

$$\mathcal{MJ}_{T[3]} = \{\{S_1^1, S_1^2, S_1^3\}, \{S_2^1, S_2^2, S_2^3\}, \{S_3^1, S_3^2, S_3^3\}, \{S_1^1, S_2^2, S_3^3\}\}. \tag{4}$$

□

Example 3.2. Let $A_1 = \{a, b, c\}$, $A_2 = \{x, y, z\}$ and $A_3 = \{1, 2, 3\}$ be three attributes and consider the table T and the corresponding releases T_1, T_2 as given in Table 5. Because of the way in which the A_2 attribute is generalized in T_2, this sequential release complies with the cell (rather than cut) generalization model (with taxonomies of height 2). The corresponding bipartite graph is depicted in Figure 5.

T	T_1	T_2
$(a, x, 1)$	$(a, x, *)$	$(*, *, 1)$
$(b, y, 2)$	$(b, y, *)$	$(*, y, 2)$
$(c, z, 3)$	$(c, z, *)$	$(*, z, 3)$

Table 5: A table T and corresponding two releases obtained by cell generalizations

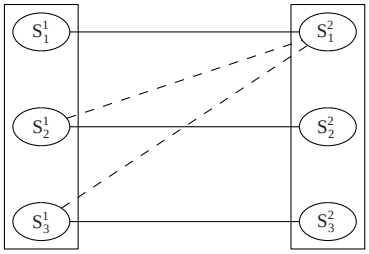


Figure 5: The graph to Example 3.2

Here,

$$\mathcal{FMJ}_{T[2]} = \{\{S_1^1, S_1^2\}, \{S_2^1, S_2^2\}, \{S_3^1, S_3^2\}\}. \tag{5}$$

However, as the graph contains two more (non-admissible) 2-cliques, we have

$$\mathcal{MJ}_{T[2]} = \{\{S_1^1, S_1^2\}, \{S_2^1, S_2^2\}, \{S_3^1, S_3^2\}, \{S_2^1, S_1^2\}, \{S_3^1, S_1^2\}\}. \tag{6}$$

□

3.2. The kernel match join

Here we define another type of join — the kernel match join. The motivation for this definition is computational. Computing the MJ is easy, as we show in Section 3.3. However, the situation with the FMJ is different. While it may be computed efficiently when $R = 2$, using the algorithm given in [35], it is hard to compute it when $R > 2$:

Theorem 3.4. *Computing the Full Match Join $\mathcal{FMJ}_{T[R]}$, for any constant $R > 2$, is NP-hard.*

The proof of Theorem 3.4 is given in the Appendix in Section 9.3. The kernel match join that we define here is an intermediate join that we use instead of the FMJ; as shown in Section 3.3, it can be computed in polynomial time.

Definition 3.7. *A subgraph $G'_{T[R]}$ of $G = G_{T[R]}$ is called rich if every edge in it is a match in $G'_{T[R]}$, and it also belongs to some R-clique in $G'_{T[R]}$. The kernel of G, denoted $K(G)$, is the maximal rich subgraph of G.*

Theorem 3.5. *There exists at least one rich subgraph of G. The kernel of G is the union of all rich subgraphs of G.*

Proof. The subgraph that consists of all $N\binom{R}{2}$ horizontal edges is clearly rich. If G' and G'' are both rich then so is their union. Hence, the union of all rich subgraphs is the maximal rich subgraph, namely, the kernel. □

Definition 3.8. *Let $T_1, \ldots, T_R$ be R releases of the same table T and let $G = G_{T[R]}$ be the corresponding multipartite graph on the set of nodes $T[R] = T_1 \cup \cdots \cup T_R$. The collection of all R-cliques in the kernel $K(G)$ of G is called the Kernel Match Join (KMJ hereinafter) of $T_1, \ldots, T_R$ and it is denoted $\mathcal{KMJ}_{T[R]}$.*

Example 3.3. Consider the sequential release in Example 3.1 and the corresponding tripartite graph in Figure 4. The three edges $e_1 := \{S_2^1, S_1^2\}$, $e_2 := \{S_3^1, S_1^3\}$, and $e_3 := \{S_3^2, S_2^3\}$ cannot be in the kernel since they are not part of any 3-clique. After removing them, we are left with a tripartite graph that includes, apart from the horizontal edges, three more edges, i.e., the three edges in the 3-clique $\{S_1^1, S_2^2, S_3^3\}$. Each of those three edges cannot be in the kernel either since they are no longer matches in the graph that is obtained after removing e_1, e_2, e_3. (For example, the edge $\{S_1^1, S_2^2\}$ is not a match since the corresponding bipartite graph, $G_{1,2}$, has only one perfect matching, consisting of the three horizontal edges.) After removing these edges too, we are left with a subgraph that consists of just the three horizontal 3-cliques. Since that subgraph is rich, it is the kernel.

The three joins that we defined form a chain in the following sense:

Theorem 3.6. *$\mathcal{FMJ}_{T[R]}$ is a subset of $\mathcal{KMJ}_{T[R]}$, while $\mathcal{KMJ}_{T[R]}$ is a proper subset of $\mathcal{MJ}_{T[R]}$.*

Proof. The first inclusion is obvious since $\mathcal{FMJ}_{T[R]}$ is clearly rich; indeed, as $\mathcal{FMJ}_{T[R]}$ is the union of all perfect matchings in the R-partite graph, every edge in $\mathcal{FMJ}_{T[R]}$ is part of some R-clique and it is also a match in $\mathcal{FMJ}_{T[R]}$. Also the second inclusion is clear since every edge in $\mathcal{KMJ}_{T[R]}$ is part of some R-clique and $\mathcal{MJ}_{T[R]}$ includes all R-cliques. It remains only to demonstrate the proper inclusion of the kernel within the match join. This is demonstrated by Examples 3.1 and 3.3. The KMJ in this case equals the FMJ, which is given in Eq. (3), and it differs from the MJ, which is given in Eq. (4). □

The definition of the KMJ implies that when $R = 2$ it coincides with the FMJ. On the other hand, when $R > 2$, there are cases in which the KMJ is a proper superset of the FMJ, assuming that P $\neq$ NP, as implied by Theorem 3.4 and the polynomial procedure to compute the KMJ that we describe in the next section.

3.3. Computing the MJ and KMJ

3.3.1. The MJ

Assume that we already have $\mathcal{MJ}_{T[R-1]}$ — the MJ on the first $R-1$ releases. Then $\mathcal{MJ}_{T[R]}$ may be computed by the straightforward Algorithm 1 which examines the possibility of expanding each of the cliques in $\mathcal{MJ}_{T[R-1]}$ by each of the records in T_R.

The runtime of Algorithm 1 is $\Theta(|\mathcal{MJ}_{T[R-1]}| \cdot NR)$. Since the size of $\mathcal{MJ}_{T[R-1]}$ is at most N^{R-1}, we get a theoretical runtime bound of $O(RN^R)$. It is impossible to improve upon this theoretical bound since if all releases are totally suppressed then the resulting $(R-1)$-partite graph would be the complete graph and then $|\mathcal{MJ}_{T[R-1]}| = N^{R-1}$. However, in practice, the $(R-1)$-partite is far from being complete. For example, if $R = 3$ then the MJ on the first two releases is just the set of edges in that bipartite graph. The number of edges in such graphs is typically $O(kN)$, which is much smaller than the theoretical bound of $O(N^2)$. (That fact was validated experimentally by us, and it was also validated in a similar setting that was studied in [35].) Having said that, Algorithm 1 and the subsequent algorithms that we present below are applicable for relatively small values of R. In order

Algorithm 1 Updating the MJ

Input: $T_1, \ldots, T_R$, $\mathcal{MJ}_{T[R-1]}$.
Output: $\mathcal{MJ}_{T[R]}$.

1: Set $\mathcal{MJ}_{T[R]} = \emptyset$.
2: **for all** $1 \leq n \leq N$ **do**
3: **for all** $C = \{S^1_{n_1}, \ldots, S^{R-1}_{n_{R-1}}\} \in \mathcal{MJ}_{T[R-1]}$ **do**
4: If $\{S^r_{n_r}, S^R_n\}$ is an edge in $G_{T[R]}$ for all $1 \leq r \leq R-1$, add $\{S^1_{n_1}, \ldots, S^{R-1}_{n_{R-1}}, S^R_n\}$ to $\mathcal{MJ}_{T[R]}$.
5: **end for**
6: **end for**
7: Output $\mathcal{MJ}_{T[R]}$.

to apply them for large values of R, it might be necessary to consider relaxed versions of those algorithms; for example, algorithms that partition the database into smaller parts and then work on each part separately, or randomizations of those algorithms.

3.3.2. The KMJ

We begin by describing a process that generates for a given graph G its kernel. Then, we describe a process for finding the kernel of $G = G_{T[R]}$ given that we already have the kernel of $G_{T[R-1]}$.

Define the following non-increasing sequence of subgraphs of G:

- $K_0(G) = G$.
- Given $K_j(G)$, $j \geq 0$, we define $K_{j+1}(G)$ in two steps.
  - (*i*) First, we remove from $K_j(G)$ all edges that are not matches and get an interim graph denoted $K_{j+1/2}(G)$; this is done by invoking the algorithm for detecting all matches in a bipartite graph on each of the bipartite subgraphs of $K_j(G)$, as described in Section 3.
  - (*ii*) Then we remove all edges in $K_{j+1/2}(G)$ that are not part of any R-clique in $K_{j+1/2}(G)$ and obtain $K_{j+1}(G)$. In other words, we retain only the edges that are part of some R-clique, namely, the edges that appear in the MJ of the graph $K_{j+1/2}(G)$. That computation can be carried out by applying Algorithm 1 on $K_{j+1/2}(G)$.

The computation of this sequence of graphs is repeated until a steady state, $K_{j_0}(G)$, is found (namely, a graph $K_{j_0}(G)$ where all edges are matches).

Theorem 3.7. *Consider the chain of non-increasing subgraphs* $G = K_0(G), K_1(G), K_2(G), \ldots$ *as defined above. Let* j_0 *be the first index for which* $K_{j_0+1}(G) = K_{j_0}(G)$. *Then* $K_{j_0}(G)$ *is*

the kernel $K(G)$. For fixed values of R, the time complexity of this algorithm is polynomial in the size of G.

Proof. Assume that $K_{j+1}(G) \neq K_j(G)$. Then the number of edges in $K_{j+1}(G)$ is strictly smaller than that in $K_j(G)$. Hence, the process must stop after no more than $|E| - N\binom{R}{2}$ steps (where $|E|$ denotes the number of edges in G), since in every iteration at least one edge was removed and it is clear that none of the $N\binom{R}{2}$ horizontal edges will be ever removed. Since the computation of $K_{j+1}(G)$ from $K_j(G)$ consists of two stages, each of which is polynomial, then the overall time complexity is also polynomial. Next, we prove that $K_{j_0}(G) = K(G)$ by induction on j_0. Clearly, if $j_0 = 0$ then $K_0(G)$, which equals G, must be the kernel, since all edges in the graph are matches and also belong to some R-clique. Assume next that $j_0 > 0$. Clearly, all edges in $K_0(G) \setminus K_{1/2}(G)$ cannot be in the kernel of $K_0(G)$ since they are not matches. Hence, the kernel of $K_0(G)$ must be included in $K_{1/2}(G)$. This implies that $K(K_0(G)) = K(K_{1/2}(G))$. By the same token, all edges in $K_{1/2}(G) \setminus K_1(G)$ cannot be in the kernel of $K_{1/2}(G)$ since they are not part of any R-clique. Hence, the kernel of $K_{1/2}(G)$ must be included in $K_1(G)$. This implies that $K(K_0(G)) = K(K_1(G))$. Finally, by the induction hypothesis on $K_1(G)$, we conclude that $K_{j_0}(G) = K(K_1(G)) = K(K_0(G)) = K(G)$. □

Next, we discuss the sequential computation of the kernel and the KMJ. To that end, we begin with the following observation.

Definition 3.9. *Let $H = (V, E)$ be a graph where V is the set of nodes and E is the set of edges. Let $V' \subset V$ be a subset of the nodes. Then the restriction of H to V' is the graph $H' = (V', E')$ where E' consists of all edges in E that connect two nodes in V'.*

Lemma 3.8. *Let K_{R-1} be the kernel of $G_{T[R-1]}$ (the multipartite graph of the first $R-1$ releases), and K_R be the kernel of $G_{T[R]}$. Then the restriction of K_R to $T[R-1]$ is a subgraph of K_{R-1}.*

Stated differently, Lemma 3.8 says that if we take any R-clique from K_R, say $C = \{S^1_{n_1}, S^2_{n_2}, \ldots, S^R_{n_R}\}$ and remove from it the node in T_R, we shall get a clique $C' = \{S^1_{n_1}, S^2_{n_2}, \ldots, S^{R-1}_{n_{R-1}}\}$ that belongs to the kernel K_{R-1} of $G_{T[R-1]}$. As an illustrating example, consider the tripartite graph in Figure 4. Its kernel, K_3, consists of the (nine) horizontal edges. When we restrict it to $T[2] = T_1 \cup T_2$ we get only three edges — $\{S^1_1, S^2_1\}$, $\{S^1_2, S^2_2\}$ and $\{S^1_3, S^2_3\}$. Those edges are included in K_2, the kernel of the restriction of the original graph to $T[2]$, which includes, in addition, also the two edges $\{S^1_1, S^2_2\}$ and $\{S^1_2, S^2_1\}$.

Proof. It is easy to see that the restriction of K_R to $T[R-1]$ is a rich subgraph of $G_{T[R-1]}$. Therefore, it is included in the kernel K_{R-1} of $G_{T[R-1]}$. □

Using the observation in Lemma 3.8 we may compute the kernel in a sequential manner using Algorithm 2. The input to that algorithm is K_{R-1}, the kernel of $G_{T[R-1]}$, and it outputs K_R, the kernel of $G_{T[R]}$.

Algorithm 2 Updating the kernel

Input: $T_1, \ldots, T_R$, K_{R-1}.
Output: K_R.

1: Set $K_R = K_{R-1}$.
2: Add to K_R all edges between a node in $T[R-1] = T_1 \cup \cdots \cup T_{R-1}$ and a node in T_R.
3: **while** K_R has edges that are not matches **do**
4: Remove all edges that are not matches.
5: Remove all edges that are not part of any R-clique.
6: **end while**

4. The privacy problem in sequential releases

Recall that among the attributes $A_1, \ldots, A_M$ of the table T, there are quasi-identifiers, $A_1, \ldots, A_t$, and a sensitive attribute A_M. As in [37], the goal is to limit the ability of the adversary to infer links between quasi-identifier values and sensitive values. Specifically, the adversary may use the sequential release in order to deduce that in records $R_i \in T$ where $(R_i(1), \ldots, R_i(t)) = (a_1, \ldots, a_t)$, for some quasi-identifier tuple of values $(a_1, \ldots, a_t) \in A_1 \times \cdots \times A_t$, the sensitive value $R_i(M)$ must belong to some subset, $S(a_1, \ldots, a_t)$, of the sensitive domain A_M. If, for every selection of $(a_1, \ldots, a_t) \in A_1 \times \cdots \times A_t$ that appears in T, the corresponding subset $S(a_1, \ldots, a_t)$ of sensitive values to which it can be linked is of size at least k, we say that the sequential release satisfies k-linkability. A stronger type of inference would allow the adversary to attach probabilities to each of the sensitive values in $S(a_1, \ldots, a_t)$. If, for every selection of $(a_1, \ldots, a_t) \in A_1 \times \cdots \times A_t$ that appears in T, the probabilities of all sensitive values in $S(a_1, \ldots, a_t)$ are at most $1/k$, we say that the sequential release satisfies k-diversity. We proceed to define these notions formally.

First, we define for every quasi-identifier tuple the set of cliques that are consistent with it.

Definition 4.1. *Let $T_1, \ldots, T_R$ be R releases of the same table T, and let $\mathcal{J}_{T[R]}$ be a collection of R-cliques in the corresponding multipartite graph $G_{T[R]}$. Fix a selection of quasi-identifier values, say $(a_1, \ldots, a_t) \in A_1 \times \cdots \times A_t$. Then the projection of $\mathcal{J}_{T[R]}$ onto $(a_1, \ldots, a_t)$, denoted $\langle \mathcal{J}_{T[R]} | (a_1, \ldots, a_t) \rangle$, is the collection of all R-cliques $\{S^1_{n_1}, \ldots, S^R_{n_R}\} \in \mathcal{J}_{T[R]}$ for which $a_m \in S^r_{n_r}(m)$, for all $1 \leq m \leq t$ and $1 \leq r \leq R$.*

In other words, given a quasi-identifier tuple $(a_1, \ldots, a_t)$ (in the sense of specifying a specific value for each of the t quasi-identifiers), the projection $\langle \mathcal{J}_{T[R]} | (a_1, \ldots, a_t) \rangle$ is the set of all R-cliques in $\mathcal{J}_{T[R]}$ that are consistent with that tuple.

Example 4.1. Consider a table with two quasi-identifiers — A_1=`age` and A_2=`gender`, and the sensitive attribute A_3=`disease`. Let T_1 be a release that includes $A_1 + A_2$, T_2 be a release that includes $A_1 + A_3$, and T_3 be a release that includes $A_2 + A_3$. (Namely, $R = 3$ in this case.) Assume that $\mathcal{J}_{T[R]}$ includes the following two R-cliques:

$$C_1 = \{ \quad S_1^1 = ([30\text{-}40],\text{male},*) \ , \quad S_1^2 = ([30\text{-}35],*,\text{flu}) \ , \quad S_1^3 = (*,*,\text{flu}) \quad \}$$
$$C_2 = \{ \quad S_2^1 = ([40\text{-}50],\text{male},*) \ , \quad S_2^2 = (*,*,\text{angina}) \ , \quad S_2^3 = (*,\text{male},\text{angina}) \quad \}$$

Then C_1 is in $\langle \mathcal{J}_{T[R]} | (a_1 = 31, a_2 = \text{male}) \rangle$ (because each of the 3 nodes in C_1 could be a generalization of $(a_1 = 31, a_2 = \text{male})$) while C_2 is not. □

Next, we define for every quasi-identifier tuple the set of sensitive values that may be linked to it. For any R-clique C, we denote by $s(C)$ its sensitive value. Recall that the sensitive attribute is not subjected to generalization. Hence, in any given release, it either appears fully specialized, or it is totally suppressed. If at least one of the releases so far included the sensitive attribute then $s(C)$ will be a specific sensitive value, otherwise $s(C) = *$. Then, if $(a_1, \ldots, a_t)$ is a quasi-identifier tuple that appears in T, we denote by

$$S(a_1, \ldots, a_t) := \{ s(C) : C \in \langle \mathcal{J}_{T[R]} | (a_1, \ldots, a_t) \rangle \}$$

the multiset[3] of all sensitive values that can be linked to $(a_1, \ldots, a_t)$ through some clique in $\langle \mathcal{J}_{T[R]} | (a_1, \ldots, a_t) \rangle$.

Finally, for any multiset A, we denote by $\gamma(A)$ the number of distinct values in A, and by $\delta(A)$ the inverse of the maximal relative frequency in A. For example, if $A = \{a, a, a, b, b, c\}$ then $\gamma(A) = 3$ (because A has 3 distinct values) and $\delta(A) = 2$ (because the relative frequency of a in A is maximal and it equals $1/2$).

We are now ready to define formally the notions of k-linkability and k-diversity:

Definition 4.2. *The sequential release $T[R]$ satisfies $[\mathcal{J}_{T[R]}, k]$-linkability (resp. diversity) with respect to a quasi-identifier tuple $(a_1, \ldots, a_t)$ that appears in T if:*

(a) $M \notin I_r$ for all $1 \leq r \leq R$ (namely, the sensitive attribute was suppressed in all R releases in $T[R]$); or

(b) $\gamma(S(a_1, \ldots, a_t)) \geq k$ (resp. $\delta(S(a_1, \ldots, a_t)) \geq k$).

The sequential release $T[R] = T_1 \cup \cdots \cup T_R$ satisfies $[\mathcal{J}_{T[R]}, k]$-linkability (resp. diversity) if it satisfies $[\mathcal{J}_{T[R]}, k]$-linkability (resp. diversity) with respect to every tuple $(a_1, \ldots, a_t) \in A_1 \times \cdots \times A_t$ that appears in T.

Example 4.2. Consider the same setting as in Example 4.1. Let us fix the tuple $(a_1 = 31, a_2 = \text{male})$. In order to evaluate the linkability and diversity levels provided by the FMJ for this tuple, we have to find all 3-cliques in $\mathcal{FMJ}_{T[3]}$ that are consistent with the above tuple (a_1, a_2). Assume that there are four such 3-cliques in $\mathcal{FMJ}_{T[3]}$; namely,

$$\langle \mathcal{FMJ}_{T[3]} | (a_1 = 31, a_2 = \text{male}) \rangle = \{C_1, C_2, C_3, C_4\}$$

and

[3] Recall that a multiset is a set where values may appear with repetitions.

$$
\begin{array}{llll}
C_1 = \{ & S_1^1 = ([30\text{-}40],\text{male},*)\ , & S_1^2 = ([30\text{-}35],*,\text{flu})\ , & S_1^3 = (*,*,\text{flu}) \quad \} \\
C_2 = \{ & S_1^1 = ([30\text{-}40],\text{male},*)\ , & S_2^2 = (*,*,\text{angina})\ , & S_2^3 = (*,\text{male},\text{angina}) \quad \} \\
C_3 = \{ & S_3^1 = (31,*,*)\ , & S_1^2 = ([30\text{-}35],*,\text{flu})\ , & S_1^3 = (*,*,\text{flu}) \quad \} \\
C_4 = \{ & S_3^1 = (31,*,*)\ , & S_2^2 = (*,*,\text{measles})\ , & S_2^3 = (*,\text{male},\text{measles}) \quad \}
\end{array}
$$

We see that $s(C_1) = s(C_3) = \text{flu}$, $s(C_2) = \text{angina}$, and $s(C_4) = \text{measles}$. Hence, the multiset of sensitive values that may be linked to $(a_1 = 31, a_2 = \text{male})$ through $\mathcal{FMJ}_{T[3]}$ is $S(a_1, a_2) = \{\text{flu,angina,flu,measles}\}$. Since $\gamma(S(a_1, a_2)) = 3$, $T[3]$ satisfies $[\mathcal{FMJ}_{T[3]}, 3]$-linkability with respect to $(a_1 = 31, a_2 = \text{male})$. As $\delta(S(a_1, a_2)) = 2$, $T[3]$ satisfies $[\mathcal{FMJ}_{T[3]}, 2]$-diversity with respect to $(a_1 = 31, a_2 = \text{male})$. □

Those notions give rise to the following privacy problem in sequential releases: Let T be a given table, and assume that we have already published $R-1$ releases of T and that those releases satisfy $[\mathcal{J}_{T[R-1]}, k]$-linkability/diversity, for a given parameter k and a join $\mathcal{J}_{T[R-1]}$ (that could be the MJ, KMJ, or FMJ). It is now needed to publish an R-th release of T on some subset of the attributes. It is needed to publish such an R-th release, for which the join $\mathcal{J}_{T[R]}$ still satisfies the same level of linkability/diversity, while the information loss is minimized. In the next section we describe an algorithm for that purpose.

Comment. All syntactic anonymization models (like k-anonymity, ℓ-diversity, or t-closeness) rely upon the the random worlds model [3]. According to that model, all tables with specific quasi-identifier values that are consistent with the published anonymized table are equally likely. In [19], Kifer showed that it is possible to extract from ℓ-diverse tables linkage probabilities between quasi-identifier tuples and sensitive values that are greater than $1/\ell$. He suggested the deFinetti attack that uses the anonymized table in order to learn a classifier that, given the quasi-identifier tuple of an individual in the underlying population, is able to predict the corresponding sensitive value with probability greater than the intended $1/\ell$ bound.

A natural question that arises in this context is whether the inference of a general behavior of the population in order to draw belief probabilities on individuals in that population constitutes a breach of privacy. To answer this question positively, the success of the attack when launched against tuples that are part of the table should be significantly higher than its success against tuples that were not part of the table. Such a comparison is yet to be performed.

The deFinetti attack and its implications were studied recently by Cormode [10]. He found that the accuracy of inference of sensitive attributes, by means of the deFinetti attack, for differentially private data and ℓ-diverse data can be quite similar, even though differential privacy does not assume the random worlds model. We believe that this finding supports our claim that using general trends to infer about individuals cannot constitute a privacy breach. We agree with Cormode's conclusion [10, Section 3.3] that "rejecting all syntactic anonymizations because the deFinetti attack exists is erroneous".

5. Achieving privacy in sequential releases

5.1. Overview

As in [37], our approach is top-down. We start with a release T_R in which all non-sensitive attributes, $A_1, \ldots, A_{M-1}$, are suppressed. We then start specializing the values of those attributes which are included in the new release T_R, while attempting to maximize the utility gain, as long as such specializations do not violate the required level of linkability. The differences between our approach and that of [37] are as follows:

- While [37] can evaluate the level of privacy only with respect to the MJ, we can evaluate it also with respect to the FMJ (in case $R = 2$) or with respect to the KMJ. As we demonstrate in Section 7, evaluating the privacy level with respect to the MJ might lead to wrong conclusions regarding the level of privacy that is actually provided by the sequential release; those evaluations should be made with respect to the FMJ, or with respect to the KMJ (which, by Theorem 3.6, is closer to the FMJ than MJ).

- We work in the much more flexible setting of cell generalization (rather than cut generalization). The differences in the quality of the outputs, as we demonstrate in Section 7, are staggering.

5.2. Measures of information loss

When generalizing table entries for the sake of privacy preservation, one must introduce a measure of information loss. Such a measure enables us to quantify the amount of information that is lost by generalization and to choose a generalization that achieves the underlying privacy goal with as small as possible information loss.

Many measures of information loss were used in the literature (see [17, Section 3] for a survey of such measures). We shall use here two of the more frequently used ones — the Loss Metric (LM) measure [18, 28] and the Entropy Measure (EM) [16]. Assume that the original table cell $S_n(m)$ was generalized to $\overline{S}_n(m)$, where the latter is a subset of the attribute domain A_m, $1 \leq m \leq M$, $1 \leq n \leq N$. Then both measures associate an information loss with that generalization and then the overall information loss is the sum of those losses over all NM cells in the table.

The information loss that the LM measure associates with each table cell is a number between 0 (no generalization at all) and 1 (total suppression) that is proportional to the size of the generalized subset in that cell. More precisely, if the table's m-th attribute has $|A_m|$ possible values, then the LM information loss in the cell $\overline{S}_n(m)$ is $(|\overline{S}_n(m)| - 1)/(|A_m| - 1)$.

The EM measure assumes that the general distribution in T of each of the attributes A_m, $1 \leq m \leq M$, is known. Then, if a cell in the m-th attribute was generalized to a subset $\overline{S}_n(m) \subseteq A_m$, the uncertainty regarding the exact value of $\overline{S}_n(m)$ that was in that cell originally may be quantified by the corresponding conditional probability. Specifically,

if $\overline{S}_n(m) = \{b_1, \ldots, b_L\}$ and p_ℓ is the probability of b_ℓ in A_m, $1 \leq \ell \leq L$, then the corresponding conditional entropy is

$$H(\overline{S}_n(m)) := -\sum_{\ell=1}^{L} q_\ell \log_2 q_\ell \ , \qquad \text{where } \ q_\ell = \frac{p_\ell}{p_1 + \cdots + p_L} \ , \ \ 1 \leq \ell \leq L \, .$$

5.3. The top-down algorithm

Our algorithm is given in Algorithm 3. It consists of two stages.

Stage 1: Cut specialization. The first stage in Algorithm 3 acts similarly to the algorithm of [37]. For each of the attributes, A_m, $1 \leq m \leq M$, the algorithm maintains and updates a cut $\mathcal{C}_m = \{B_1^m, \ldots, B_{Q_m}^m\}$ in the corresponding taxonomy $\mathcal{T}_m$; namely, B_q^m, $1 \leq q \leq Q_m$, are subsets (nodes) in $\mathcal{T}_m$ that are disjoint and their union equals A_m. Given such cuts, the corresponding generalization is that in which every value of A_m that belongs to the subset B_q^m, for some $1 \leq q \leq Q_m$, is replaced by that subset.

The initial cuts in each of the non-sensitive attributes, $A_1, \ldots, A_{M-1}$, is $\mathcal{C}_m = \{A_m\}$ (Step 8). In other words, the initial cut in those attributes is the trivial cut that includes just the root of the taxonomy (which describes the entire set of values, A_m). Such cuts induce a total suppression of the corresponding attributes. As for the sensitive attribute (Steps 3-7), if it is included in the new release, we set $\mathcal{C}_m = A_m$; namely, $\mathcal{C}_m$ is the cut which consists of all leaves in the taxonomy, which induces total specialization of that attribute (no generalization). If, on the other hand, the sensitive attribute is not included in the new release, we suppress it by setting $\mathcal{C}_m = \{A_m\}$.

Then, in each step, the algorithm scans all cuts that correspond to attributes that appear in the new release, and within each of those cuts it scans all subsets B_q^m, $1 \leq q \leq Q_m$. It checks whether it is possible to split the corresponding subset B_q^m to its immediate descendants in $\mathcal{T}_m$ without violating the necessary privacy constraint. (The algorithm skips attributes that do not appear in the new release since they should remain totally suppressed. It also skips the sensitive attribute: If it is not included in the new release then it should remain totally suppressed; otherwise, it is already fully specialized.) Among all split operations that are possible at that time, the algorithm selects the split that maximizes the following information-gain-privacy-loss ratio:

$$\rho := \frac{IL_B - IL_A}{PL_B - PL_A} \, ; \tag{7}$$

here, IL_B is the information loss before the split and IL_A is the information loss after the split, and PL_B and PL_A are the privacy levels before and after the split. This loop continues until the algorithm reaches a stage in which all subsets in all cuts are either leaves (namely, they cannot be further specialized) or they cannot be split without violating the required privacy constraint (be it k-linkability or k-diversity).

The algorithm maintains a set Θ that holds the union of nodes from all cuts that participate in the above described specialization process. Initially, all cuts consist of the roots of the relevant taxonomies (Step 8). The main loop of this stage (Steps 9-22) scans

Algorithm 3 A Top-Down Algorithm for Anonymizing Sequential Releases

Input: Generalized releases $T_1, \ldots, T_{R-1}$, a list of attribute indices $I_R \subseteq [M]$ to be included in T_R, a privacy requirement, and an information loss metric.

Output: A generalized release T_R such that the sequential release $T_1, \ldots, T_R$ satisfies the privacy requirement.

1: **Stage 1 (cut specialization):**
2: Suppress all non-sensitive attributes in T_R.
3: **if** $M \in I_R$ **then**
4: Leave A_M in its original form in T_R.
5: **else**
6: Suppress A_M.
7: **end if**
8: Set $\Theta \leftarrow \{A_m : m \in I_R \cap [M-1]\}$.
9: **for all** $\theta \in \Theta$ **do**
10: **if** θ can be specialized without violating the privacy requirement **then**
11: Calculate the information-gain-privacy-loss ratio in case we choose to specialize θ.
12: **else**
13: Remove θ from Θ.
14: **end if**
15: **end for**
16: **if** $\Theta = \emptyset$ **then**
17: Go to Step 23.
18: **end if**
19: Choose θ from Θ for which the information-gain-privacy-loss ratio is maximal.
20: Specialize all entries in T_R that have the value θ.
21: Replace θ in Θ with its immediate descendants in its taxonomy.
22: Go to Step 9.
23: **Stage 2 (cell specialization):**
24: Set $S_{cells} \leftarrow \{(n, m) : \ m \in I_R \cap [M-1], \ 1 \leq n \leq N, \ S_n^R(m)$ is not a leaf in $\mathcal{T}_m\}$.
25: Split S_{cells} randomly into buckets of size *bucket_size*.
26: Let $b_1, \ldots, b_p$ be a random ordering of those buckets.
27: **for** $1 \leq i \leq p$ **do**
28: **if** all cells in b_i can be specialized without violating the privacy requirement **then**
29: Specialize all cells in T_R that belong to b_i.
30: **end if**
31: **end for**
32: **if** *bucket_size* > 1 **then**
33: Decrease *bucket_size* and go to Step 24.
34: **else**
35: Return T_R.
36: **end if**

all nodes θ in the set Θ. For each such node, it checks whether it can be specialized without violating the required privacy constraint. If it can, the algorithm computes the information-gain-privacy-loss ratio ρ, Eq. (7), in case that specialization would take place (Steps 10-11), otherwise it removes that node from Θ so that it will not be checked again (Steps 12-13). If at the end of that loop Θ is empty, then there are no nodes in any of the taxonomies that could be further specialized, whence we proceed to Stage 2 (Steps 16-18). Otherwise, we choose the node that can be specialized and yields the maximal information-gain-privacy-loss ratio ρ, we specialize it and update Θ accordingly (Steps 19-21). (In Step 21, if the immediate descendants of the selected node θ are leaves of the corresponding taxonomy, we do not place them in Θ in lieu of their father node θ since leaves can not be candidates for further specialization.) We then return to Step 9 for another iteration of specialization.

The output of this stage is a cut generalization. Our algorithm then proceeds to the next stage in which it performs cell specialization.

Stage 2: Cell specialization. We first describe an idealized version of this stage of the algorithm. Then, we proceed to describe a relaxed version of that stage, which has significantly reduced runtimes.

In each iteration in this stage, the idealized algorithm scans all cells that are still not totally specialized. For each such cell, it checks whether replacing it with the immediate descendant in the taxonomy which generalizes the corresponding exact value maintains the privacy constraint. Then, the algorithm selects the legitimate specialization operation that yields the maximal information-gain-privacy-loss ratio ρ. The loop continues until it reaches a stage in which all possible cell specializations result in violating the privacy constraint.

For the sake of illustration: Assume that one of the attributes describes clothing items and that the corresponding taxonomy is as in Figure 2. If one of T's cells includes the value "sandals" and it is generalized in T_R to "clothing", the algorithm will check whether specializing "clothing" to "footwear" is allowed by the privacy constraint. If it is, then that cell would be a candidate for specialization. The cell that will be selected for specialization is the one that yields the greatest information-gain-privacy-loss ratio.

The runtime of the above described greedy cell specialization procedure is not appealing. Hence, Algorithm 3 utilizes a bucketing strategy that greatly reduces the runtimes, while still yielding generalized tables with low information losses. In Step 24, we place in S_{cells} all cells in the relevant attributes that do not contain leaf nodes (as such can not be further specialized). We split those cells into buckets of size *bucket_size* (Step 25). Then, we perform a loop over all buckets in a random order (Steps 26-31). For each bucket, we check whether we can simultaneously specialize all cells in that bucket. If so, then we specialize all of them. After we finish testing all buckets, and specializing the buckets that could be specialized, we reduce the bucket size (Steps 32-33) and return to Step 24; there, all cells that are currently not leaf nodes are reallocated again to buckets. The process continues down to the point of testing the specialization of singleton buckets. (We omit the technical details of how we select the initial bucket size and how we reduce it.)

Checking for privacy violation. The algorithm checks the privacy conditions in Steps 10 and 28. We ran Algorithm 3 with the $[\mathcal{J}_{T[R]}, k]$-linkability and $[\mathcal{J}_{T[R]}, k]$-diversity privacy conditions, with all three join notions — FMJ (when $R = 2$), KMJ, and MJ. In order to check those privacy conditions, we ran the algorithms described in Section 3 for computing and updating the relevant join. Then, after calculating that join, we verified the required privacy condition according to Definition 4.2.

Before concluding this section we note that since Algorithm 3 uses randomness (see Steps 25 and 26), minimality attacks [41] are ineffective against it.

6. Dynamically changing tables

So far we concentrated on the case where the underlying table is static, in the sense that its set of records is fixed. Namely, while our discussion thus far included cases where new attributes (columns) are added to T, it did not include the case where new records (rows) are added. Herein we extend our discussion to include also dynamics of the latter type.

Assume that after publishing R releases, $T_1, \ldots, T_R$, of $T = \{S_1, \ldots, S_N\}$, a set of new $N' - N$ records, $S_{N+1}, \ldots, S_{N'}$ is added to T. There are two approaches in which our algorithm may be extended to handle such additions of records.

The separative approach suggests to apply our algorithm, starting from the $(R + 1)$-th release, separately on the original set of N records and on the new set of $N' - N$ records. Namely, we view the addition of the new $N' - N$ records as the starting point of a new sequential release over those records only. That means that we perform two separate computations: A computation of the anonymization of the old N records for the $(R+1)$-th release, independently of the new addition of $N' - N$ records; and a computation of the anonymization of the new $N' - N$ records as if it is their first release, regardless of the old records. In doing so, we make sure that each of those sequential releases respects the required privacy constraint.

We illustrate that approach in Figure 6. The first three releases T_1, T_2, T_3 include the anonymization of only five records. In release T_4, a new set of two records is added, and then in release T_6 additional three records are added. Such a scenario triggers three parallel and independent sequential releases: A sequential release of the first five records, that consists in this example of six releases, T_1 through T_6; a sequential release of the next two records, that consists of T_4, T_5, T_6, and a third sequential release over the last three records, that starts in T_6.

The combined sequential release respects the required privacy constraint, because even if we had told the adversary which $N' - N$ generalized records in T_r, $r \geq R+1$, correspond to the newly added records, he would have not been able to make inferences that violate the required privacy constraint. In the example illustrated in Figure 6, even if we told the adversary how to correctly separate the second and the third release, and the privacy constraint is, say, ℓ-diversity, he would still be unable to link quasi-identifier tuples to sensitive values with probability greater than $1/\ell$. However, by considering each set of

records separately, we might make excessive generalizations that would result in higher information losses. Namely, by mixing the old and new records, we could achieve similar privacy goals with less generalizations, since we could use the quasi-identifier-proximity between old and new records in order to avoid unnecessary generalizations. This is the idea that underlies our second suggested approach.

The unifying approach considers all records together. In that approach, once the table T is augmented by the new records $S_n = (S_n(1), \ldots, S_n(M))$, $N < n \leq N'$, we augment each of the existing releases, $T_1, \ldots, T_R$, with the corresponding suppression of those records. Namely, we add to T_r, $1 \leq r \leq R$, the following $N' - N$ generalized records,

$$S_n^r = (*, \ldots, *, S_n(M)) \,, \quad N < n \leq N' \,.$$

Then, we proceed to compute the next releases on the unified set of records just as we did previously.

The unifying approach is illustrated in Figure 7. The first three releases, T_1, T_2, T_3, consist of five records only. In the subsequent release, T_4, two new records were added. In order to compute the generalization of that release, we augment at that point the first three releases with two corresponding suppressed records; those are denoted in Figure 7 by $*_6$ (standing for a record in which all quasi-identifiers are suppressed, and the sensitive value equals the sensitive value of the newly added record S_6) and $*_7$. Then, when three new records are introduced in T_6 (S_8, S_9, S_{10}) we augment all preceding releases, T_1 through T_5, with the corresponding suppressed records $*_8, *_9, *_{10}$ and then proceed to compute the appropriate generalization of T_6, such that the entire sequential release would respect the required privacy constraint.

Like previous studies that dealt with dynamically changing tables, e.g. [8, 46], we too assume that the set of newly added records complies with the required privacy constraint, in the following sense. Let $A := \{S_n(M) : N < n \leq N'\}$ be the multiset of sensitive values in the new records. If the privacy constraint is k-linkability (every quasi-identifier tuple must be linkable to no less than k distinct sensitive values), we should have $\gamma(A) \geq k$ (recall that $\gamma(\cdot)$ was defined in Section 4); if, however, the privacy constraint is k-diversity (every quasi-identifier tuple must be linkable to any sensitive value with probability at most $1/k$) then we should have $\delta(A) \geq k$. Such an assumption is inevitable. Indeed, an adversary who targets an individual that was part of that set of records can compare the sensitive values in releases before and after that set of records was added to T and then extract the multiset A of sensitive values of the newly added records. If $\gamma(A) < k$ or $\delta(A) < k$, he may draw inferences about his target individual that violate the required privacy constraint.

We conclude this section by a brief overview of recent studies that dealt with anonymization of dynamically changing tables. Byun et al. [8] focused, like us, on the case of adding records only. They considered anonymizations that respect ℓ-diversity in the sense of ensuring that each quasi-identifier tuple can be linked to at least ℓ distinct sensitive values, regardless of the linkage probabilities. Also [13] and [30] dealt with the case of record additions; the privacy measure which was applied there was that of k-linkability rather

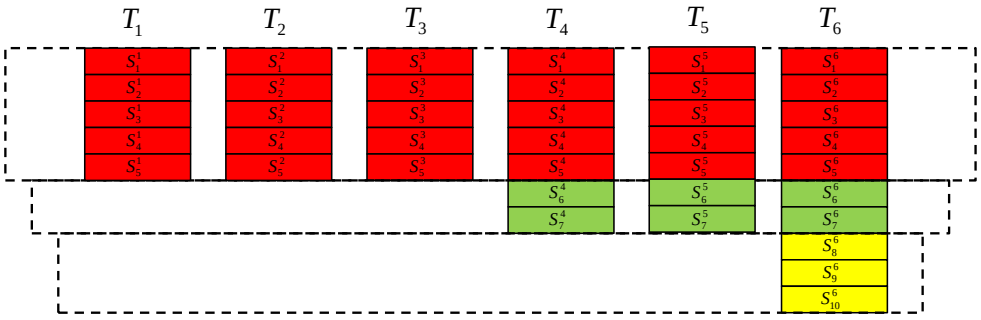


Figure 6: Illustrating the dynamic setting — the separative approach

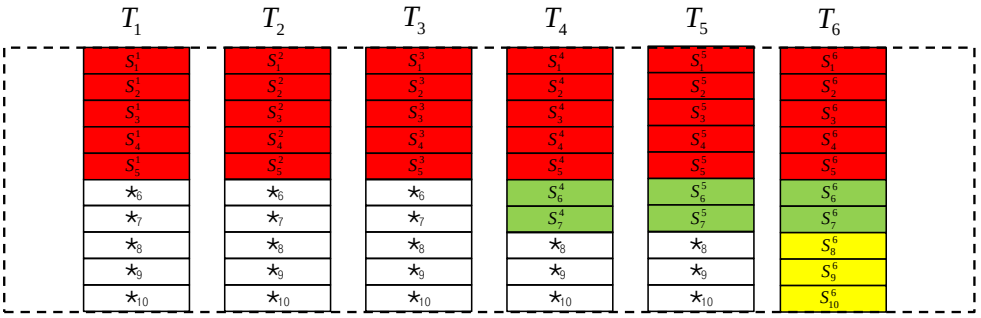


Figure 7: Illustrating the dynamic setting — the unifying approach

than ℓ-diversity. Xiao and Tao [46] were the first to include in their discussion the case of deletion of records. That case is harder, due to a phenomenon that was called in [46] *critical absence*. For example, if the table T had only one individual with some sensitive value, and that record was removed at some point, subsequent releases of T would not include that sensitive value. If the adversary knows when that individual was removed from the table, he may be able to infer that individual's sensitive value. That problem was addressed in [46] by introducing counterfeit records so that the multiset of sensitive values in the dynamic table will never narrow down. Xiao and Tao assumed that records remain unchanged during their lifespan in the sequential release. The subsequent studies [6, 43] considered the case where records can be updated between releases. The extension of our algorithms to the case of record deletions or updates is left as future research.

We note that all of the above mentioned studies, as opposed to ours, share two limiting features:

(1) They utilized methods of homogeneous anonymization. As shown theoretically and experimentally in [15, 35, 44], non-homogeneous anonymization offers significantly smaller information losses and enable more accurate data mining [21]. Our algorithms, in contrast to those in the above mentioned studies, adhere to the non-homogeneous framework since we do not limit ourselves to generalizations in which each release consists of blocks of "homogeneous" records. (Algorithm TDS4ASR of [37] also adheres to non-homogeneous anonymization.)

(2) Those studies assumed that all releases include all quasi-identifiers. Such an assumption represents an extreme (sometimes unnecessarily) case in which all releases contain the full linkage between all quasi-identifiers. However, typically in sequential releases, each release will be tailored for a different purpose and thus include a different subset of the quasi-identifiers. Hence, it is possible that none of the releases will include a full linkage between all quasi-identifiers and the adversary would face a puzzle-like challenge to correctly identify the links between the record "pieces" as they appear in the different releases. Our algorithms (and TDS4ASR of [37]) take advantage of that in order to reduce the information loss. Another support for the advantage in anonymizing separate projections of the table, instead of anonymizing the entire table, at can be found in [26]. It is shown there that partitioning the underlying table into several projections, such that the set of projections satisfies k-anonymity, results in better utility than anonymizing the entire table.

In addition to the above limitations, the m-invariance solution that was suggested in [46] has the following limitations (while our solution does not):

(1) It considers tables with only three types of attributes rather than four — it does not consider attributes that are non-sensitive on one hand, but cannot serve as identifiers on the other hand (i.e. attributes $A_{t+1}, \ldots, A_{M-1}$, see Section 2.1). Examples for such attributes in health information may include blood type, dietary information, or known allergies. It can be shown that m-invariance breaks, or may even totally collapse, in the presence of such attributes.

(2) It is limited to providing diversity only for integer values of the diversity parameter, what limits its applicability only to tables with a rich sensitive attribute. Specifically, it requires creating blocks of m identical generalized records having m distinct sensitive values. It is impossible to achieve that goal, for any $m > 1$, with databases having skew binary sensitive attributes (such as the well known Adult database that has a binary sensitive attribute, with a roughly 75%-25% split). It is also impossible to achieve m-invariance for other tables with a richer sensitive alphabet and a low general diversity.

7. Experimental evaluation

7.1. Experimental setup

Our experiments were conducted on two datasets:

- CENSUS[4]. That dataset has 500,000 records consisting of 7 quasi-identifiers (`age`, `gender`, `education level`, `marital status`, `race`, `work class`, `country`) and one sensitive attribute. We created two-level taxonomies for the attributes `gender` and

[4]http://www.ipums.org

`race` (namely, in those attributes we applied only suppressions), three-level taxonomies for the attributes `education level`, `marital status` and `country`, and four-level taxonomies for the attributes `age` and `work class`. The sensitive attribute in this dataset has 50 distinct values and its overall diversity (namely, the inverse of the maximal relative frequency) is 13.27.

- ADULT from the UCI Machine Learning Repository[5]. That dataset was extracted from the US Census Bureau Data Extraction System. It holds demographic information of a small sample of US population with 14 quasi-identifiers such as `age`, `education level`, `marital status`, and `native country` and contains 32,561 records. We adopted the taxonomies in [37] for this dataset. Since its sensitive attribute is too narrow (binary), we used one of the quasi-identifiers, `occupation`, as the sensitive attribute in our experiments. It has 15 distinct values and its overall diversity is 7.86.

All experiments were conducted on an Intel Core i7 CPU 2.67 GHz personal computer with 4 GB of RAM, running Windows 7 Enterprise edition.

The bulk of our experiments were conducted on static tables with the k-linkability privacy requirement. We then repeated some of the experiments with the k-diversity privacy requirement instead. Finally, we conducted experiments with dynamic tables.

We begin with a detailed description of our evaluation using static tables and the k-linkability privacy requirement. Our evaluation consisted of six sets of experiments. In the first set of experiments we ran our algorithm on different databases and releases: Two of the four experiments in this set were on sequential releases of the CENSUS dataset (with 100,000 records) and the other two experiments were on sequential releases of the ADULT dataset. In the second set of experiments we tested several values of k (the linkability level). In the third set we tested several values of N (the number of records in the database). In the fourth set we compared different information loss metrics. Finally, in the fifth and sixth sets we applied our algorithm on a sequential release of $R = 3$ and $R = 4$ releases, respectively. (In the preceding experiment sets there were $R = 2$ releases.)

Each experiment consisted of three stages:

(1) In the first stage, we found the best cut generalization that satisfied $[\mathcal{KMJ}_{T[R]}, k]$-linkability for the required linkability level k. (That stage represents, in fact, the application of the TDS4ASR algorithm [37].)

(2) In the second stage, we further specialized the cells in order to arrive at a cell generalization that satisfied the same privacy constraint. The goal of this stage was to demonstrate that the cell generalization model allows achieving a certain privacy goal with significantly smaller losses of information.

(3) In the third stage, we further specialized the cells in order to get $[\mathcal{MJ}_{T[R]}, k]$-linkability, instead of $[\mathcal{KMJ}_{T[R]}, k]$-linkability. Since MJ is a superset of the KMJ (Theorem 3.6),

[5] http://mlearn.ics.uci.edu/MLSummary.html

aiming at achieving a certain level of linkability with respect to the MJ instead of the KMJ allows making more specializations. The goal of this stage was to demonstrate the difference between the evaluation of the privacy condition with respect to the MJ and the KMJ (see Test (b) below).

In each of those stages we evaluated the information loss in the last release (T_R) and the execution runtime.

In addition to those three stages, we conducted two additional tests in each of the experiments:

- Test (a). In order to illustrate the disadvantage of the cut generalization from a different angle, we tested the cut generalization that we got in the first stage, when using some value of the linkability level k as the guiding privacy constraint, and checked what is the actual linkability level that the resulting generalization satisfied. We found out that the actual level of linkability was usually much higher than the intended level of linkability. That result illustrates the rigidity of the cut generalization model, as opposed to the flexibility of the cell generalization model.

- Test (b). In order to illustrate the breach of privacy in wake of using the MJ rather than the KMJ, in the case of cell generalization or in case $R > 2$, we took the generalization that we obtained in the third stage (which satisfied $[\mathcal{MJ}_{T[R]}, k]$-linkability) and evaluated the level k of $[\mathcal{KMJ}_{T[R]}, k]$-linkability that it satisfied (that is — the level of linkability with respect to the KMJ). As implied by Theorem 3.6, the level of linkability when tested with respect to the KMJ could be smaller than the level of linkability when tested with respect to the MJ. The purpose of this test was to demonstrate the difference between those two levels and, consequently, why using the MJ to verify linkability should be avoided in such cases.

Table 6 summarizes the details of the six experiment sets. For example, in the second experiment of the first set (configuration CENSUS-2), the dataset was CENSUS with 100,000 records. The first release T_1 included the attributes `gender`, `race`, `country` and `education level`, while T_2 included `gender`, `race`, `marital status` and the sensitive attribute. The privacy goal was to obtain 5-linkability (with respect to the KMJ in the first two stages and then with respect to the MJ in the third stage). The guiding utility measure was the LM measure of information loss.

We repeated all experiments with the k-diversity privacy requirement instead of the k-linkability one. The results were consistent with the ones obtained with the linkability requirement. We report herein the results of those repeated experiments only for experiments sets 2 and 3 in Table 6. In experiment set 3, the targeted diversity level was $k = 5$ (similarly to the linkability level in that experiment set). However, in experiment set 2, the sequence of diversity levels that we used was $k = 4, 6, 8, 10$ (instead of $k = 5, 10, 15, 20, 25$ as the linkability levels in that experiment set). Recall that for any multiset A, its diversity, which we denoted $\delta(A)$ (see Section 4), is bounded from above by the number of distinct values in it, $\gamma(A)$. Since the sensitive attribute in CENSUS has 50 distinct values, but its

Experiment Set	Configuration	Dataset	N	k	Inf.Loss	T_1	T_2	T_3	T_4
1. Different sequential releases	Census-1	Census	100,000	5	LM	Age,Edu	Age,Sen		
	Census-2	Census	100,000	5	LM	Gen,Rac,Cou,Edu	Gen,Rac,Marsta,Sen		
	Adult-1	Adult	32,561	5	LM	Age,Edu	Age,Occ		
	Adult-2	Adult	32,561	5	LM	Gen,Rac,Cou,Edu	Gen,Rac,Marsta,Occ		
2. Different values of required k	k=5	Census	100,000	5	LM	Age,Edu	Age,Sen		
	k=10	Census	100,000	10	LM	Age,Edu	Age,Sen		
	k=15	Census	100,000	15	LM	Age,Edu	Age,Sen		
	k=20	Census	100,000	20	LM	Age,Edu	Age,Sen		
	k=25	Census	100,000	25	LM	Age,Edu	Age,Sen		
3. Different Ns	N=100,000	Census	100,000	5	LM	Age,Edu	Age,Sen		
	N=200,000	Census	200,000	5	LM	Age,Edu	Age,Sen		
	N=300,000	Census	300,000	5	LM	Age,Edu	Age,Sen		
	N=400,000	Census	400,000	5	LM	Age,Edu	Age,Sen		
	N=500,000	Census	500,000	5	LM	Age,Edu	Age,Sen		
4. Different Inf. Loss Metrics	LM	Census	100,000	5	LM	Age,Edu	Age,Sen		
	EM	Census	100,000	5	EM	Age,Edu	Age,Sen		
5. Three Releases	R=3	Census	100,000	5	LM	Age,Gen	Gen,Sen	Age,Sen	
6. Four Releases	R=4	Census	100,000	5	LM	Age,Gen	Age,Sen	Age,Gen,Marsta	Gen,Sen

Table 6: Summary of experiments.

overall diversity is only 13.27, it is possible to achieve for that dataset k-linkability with k up to 50, but k-diversity with k only up to 13.27.

Finally, we conducted experiments with dynamic tables. We took the basic configuration Census-1 (see Table 6), and then examined scenarios where T_1 has $N = 100,000$ records and T_2 has additional $N' - N$ records. We examined five values of N': From $N' = 105,000$ (reflecting an increase of 5% in the number of records with respect to the first release) up to $N' = 200,000$ (reflecting an increase of 100%). We compared the performance of the two algorithms that were described in Section 6: The one that considers separately the two sets of records (old and new), and the other one that operates on the unified set of records. The goal was to show that the latter yields lower information losses. We repeated the test with a dynamic sequential release that consisted of $R = 3$ releases.

7.2. Results

The first experiment set compared four different configurations of sequential releases (two from the Census dataset and two from the Adult dataset). The information loss results of that set of experiments are given in Figure 8. For example, in the second configuration, the best cut generalization that satisfied 5-linkability was the one in which all entries of the attribute `race` had to be totally suppressed (whence, the LM information loss, which is the sum of information losses over all cells in the table, see Section 5.2, is 100000). When allowing cell generalization we were able to achieve 5-linkability with an overall LM information loss of 20 (i.e., an improvement factor of 5000). Then, when we relaxed our linkability testing and verified it with respect to the MJ rather than the FMJ, we were able to reduce the information loss from 20 to 16.

Table 7 includes the results of the two additional tests that we described in Section 7.1 (see Tests (a) and (b) there). In the first test (Test (a)) we checked the actual linkability level that is obtained by the cut generalization. For example, in the second configuration (Census-2), even though we aimed at achieving 5-linkability, the usage of cut generalization

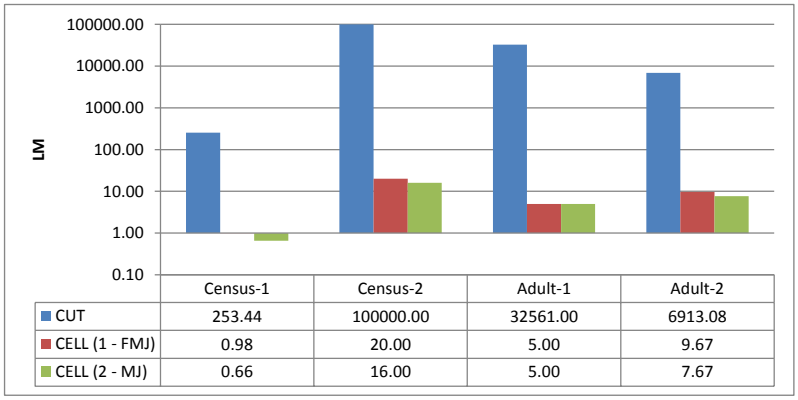


Figure 8: Experiment Set 1, information losses (linkability)

resulted in an extreme overshoot, since the generalization actually satisfies 43-linkability. In the second test (Test (b)) we took the generalization that we obtained in the third stage (which satisfied $[\mathcal{MJ}_{T[2]}, 5]$-linkability) and evaluated the level k of $[\mathcal{KMJ}_{T[2]}, k]$-linkability that it satisfied. In the second configuration, for example, even though we were aiming for $k = 5$, the achieved linkability level is actually $k = 1$.

Generalization	CENSUS-1	CENSUS-2	ADULT-1	ADULT-2
Cut (Test (a))	5	43	15	10
Cell (2-MJ) (Test (b))	3	1	5	2

Table 7: Experiment Set 1, actual linkability level in generalizations that targeted 5-linkability

The second experiment compared five different values of the required linkability level: $k = 5, 10, 15, 20, 25$. The results are given in Figure 9 and Table 8. We see that the improvement in the utility, due to adopting the cell generalization model, decreases with k but it remains significant (Figure 9). Here too, we see a significant overshoot in the linkability level when using the rigid cut generalization model. Also, evaluation of linkability that is based on the MJ completely misses the actual linkability level; for example, even though we aimed at a linkability level of $k = 25$, the actual level of linkability that we got was only $k = 3$.

Generalization	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
Cut (Test (a))	5	38	38	38	38
Cell (2-MJ)(Test (b))	3	3	3	3	3

Table 8: Experiment Set 2, actual linkability levels compared to the targeted ones

We repeated the experiment with k-diversity instead of k-linkability, with the following diversity levels: $k = 4, 6, 8, 10$. The results were consistent with the linkability results, see Figure 10 and Table 9.

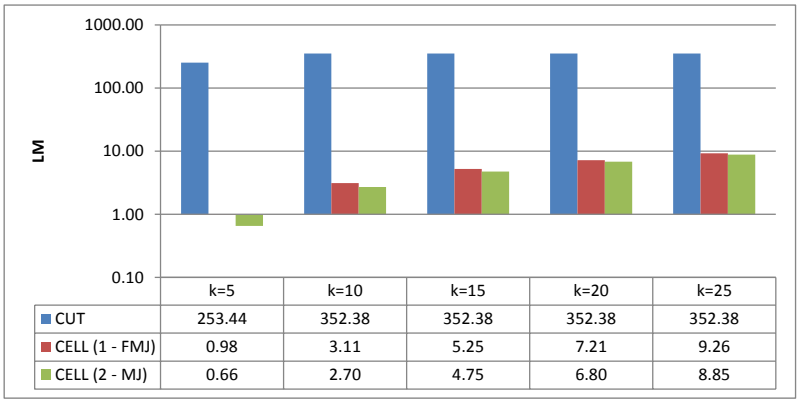


Figure 9: Experiment Set 2, different levels of linkability

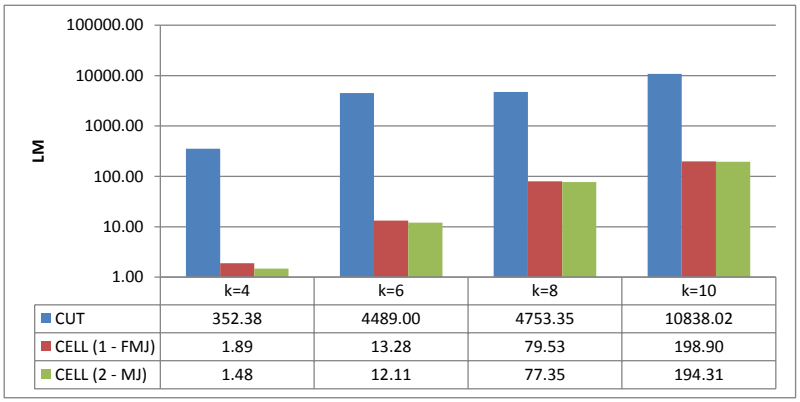


Figure 10: Experiment Set 2, different levels of diversity

Generalization	$k = 4$	$k = 6$	$k = 8$	$k = 10$
Cut (Test (a))	5.8	7.17	8.74	10.71
Cell (2-MJ)(Test (b))	2	2	5.2	2

Table 9: Experiment Set 2, actual diversity levels compared to the targeted ones

The third experiment compared databases of different sizes. Its results are given in Figure 11 and Table 10. We see that the improvement in the utility, due to adopting the cell generalization model, increases with N (Figure 11). As for the overshoot in the linkability level, due to the rigidity of the cut generalization model, it too increases with N (Table 10).

Generalization	$N = 100000$	$N = 200000$	$N = 300000$	$N = 400000$	$N = 500000$
Cut (Test (a))	5	9	11	12	15
Cell (2-MJ) (Test (b))	3	1	1	1	1

Table 10: Experiment Set 3, actual linkability level in generalizations that targeted 5-linkability

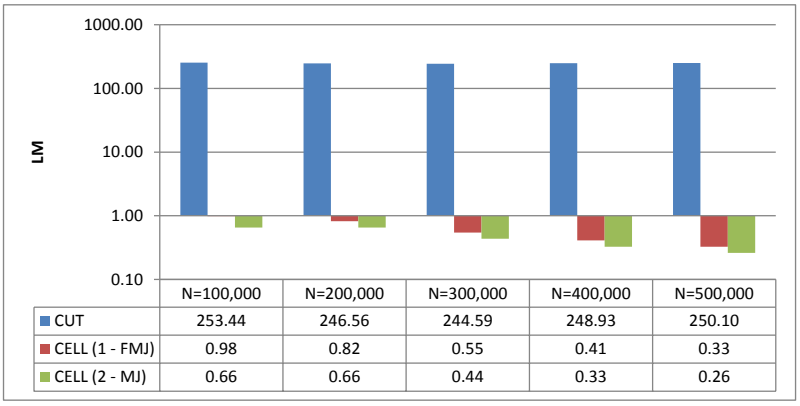


Figure 11: Experiment Set 3, different database sizes (linkability)

The results when the privacy goal was 5-diversity rather than 5-linkability are shown in Figure 12 and Table 11.

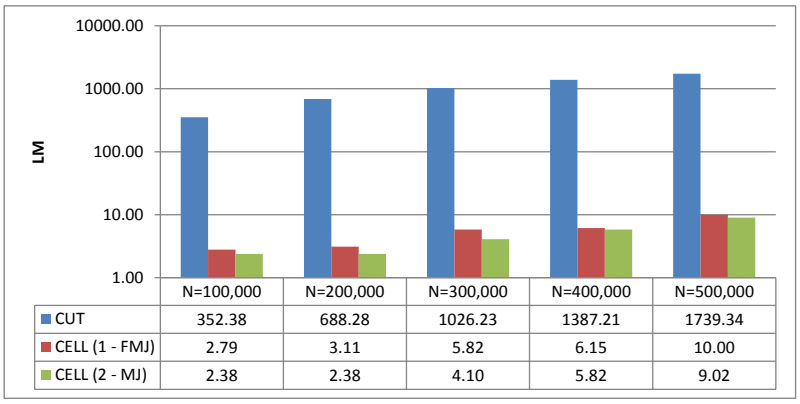


Figure 12: Experiment Set 3, different database sizes (diversity)

Generalization	$N = 100000$	$N = 200000$	$N = 300000$	$N = 400000$	$N = 500000$
Cut (Test (a))	5.8	5.94	5.75	5.71	5.64
Cell (2-MJ) (Test (b))	2	1	1	1	1

Table 11: Experiment Set 3, actual diversity level in generalizations that targeted 5-diversity

The fourth experiment compared two information loss metrics — the LM and the EM. The goal of that experiment was to verify that the pattern which emerges from our experiments is independent of the information loss metric. In all of our experiments with the EM we got the same pattern as in the experiments with the LM that were reported in the previous experiment sets. In Figure 13 and Table 12 we give the results for one of those configurations.

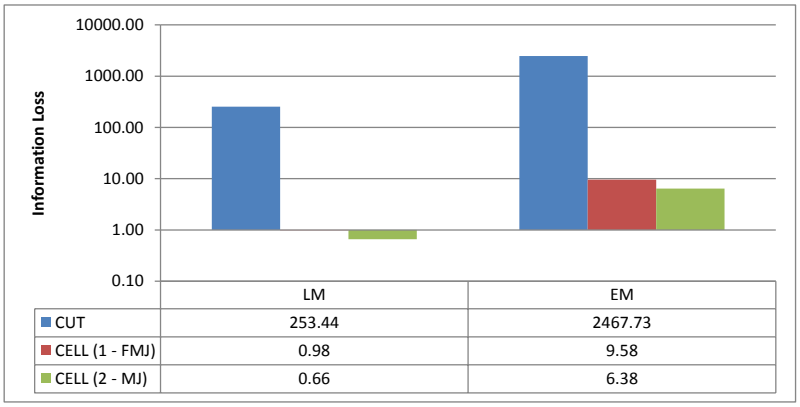


Figure 13: Experiment Set 4, different information loss metrics

Generalization	LM	EM
Cut (Test (a))	5	5
Cell (2-MJ) (Test (b))	3	3

Table 12: Experiment Set 4, actual linkability level in generalizations that were targeted at 5-linkability

In the fifth experiment we examined our algorithm in the case of $R = 3$ releases. When $R = 2$, the FMJ coincides with the KMJ and, therefore, it may be evaluated efficiently; however, when $R \geq 3$ we cannot compute the FMJ and, hence, we compute the KMJ instead. The resulting information losses are shown in Figure 14; here too, cell generalization allows significantly smaller information losses than cut generalization. Table 13 shows the results of Tests (a) and (b); as can be seen, using the MJ rather than the KMJ leads to a breach of privacy.

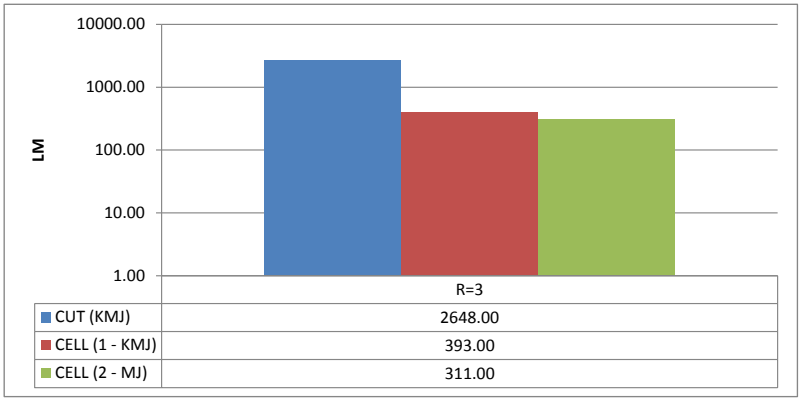


Figure 14: Experiment Set 5, three releases

In the sixth experiment we examined our algorithm in the case of $R = 4$ releases. The resulting information losses are shown in Figure 15 and the results of Tests (a) and (b) are shown in Table 14. As can be seen, the information losses in later releases ($R = 3$ and

Generalization	Actual k
Cut (Test (a))	5
Cell (2-MJ) (Test (b))	1

Table 13: Experiment Set 5, actual linkability in generalizations that were targeted at 5-linkability

$R = 4$) are higher than those in the second release, since the generalization of those releases is restricted by the previous releases.

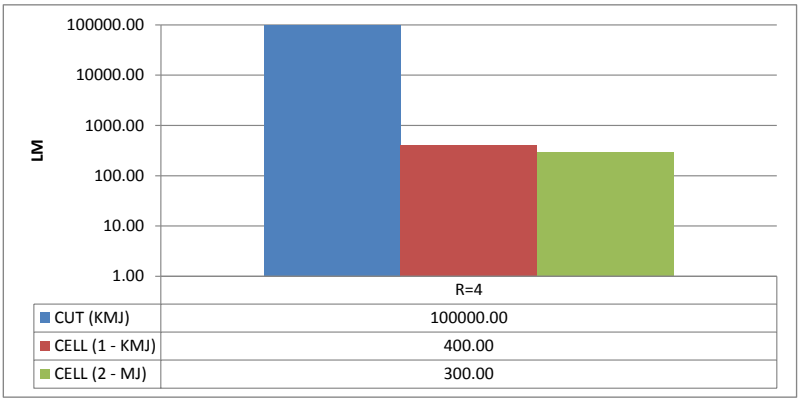


Figure 15: Experiment Set 6, four releases

Generalization	Actual k
Cut (Test (a))	6
Cell (2-MJ) (Test (b))	1

Table 14: Experiment Set 6, actual linkability in generalizations that were targeted at 5-linkability

In addition to the above described tests, we also checked the actual linkability level in the cell generalizations that we got in each experiment. In all of those experiments, the actual linkability level was equal to the targeted linkability level. Hence, while in cut generalization the actual linkability level was usually higher (and sometimes even much higher) than the targeted one, it is not the case when using cell generalization. We obtained similar findings also with regard to the diversity, in the experiments where the privacy goal was k-diversity.

Table 15 summarizes the execution runtimes for the different experiments; the runtime is indicated in seconds, apart from the last two entries that are indicated in hours. (We show runtimes for the experiments in which we used the linkability as the privacy goal; the runtimes in the experiments where the underlying privacy goal was the diversity are similar.) As expected, the runtime increases linearly with N. The dependence of the runtime on k appears to be sub-linear (in this experiment, the runtime grows roughly like $\Theta(k^{0.6})$). The runtime grows fast with the number of releases, R, and hence it is practical, in its present form, only for small values of R.

Experiment Set	Configuration	Runtime
1. Different sequential releases	CENSUS-1	52s
	CENSUS-2	115s
	ADULT-1	29s
	ADULT-2	41s
2. Different values of required k	k=5	52s
	k=10	80s
	k=15	115s
	k=20	121s
	k=25	144s
3. Different Ns	N=100,000	52s
	N=200,000	103s
	N=300,000	179s
	N=400,000	225s
	N=500,000	302s
4. Different Information Loss Metrics	LM	52s
	EM	43s
5. Three Releases	R=3	3.1h
6. Four Releases	R=4	24.8h

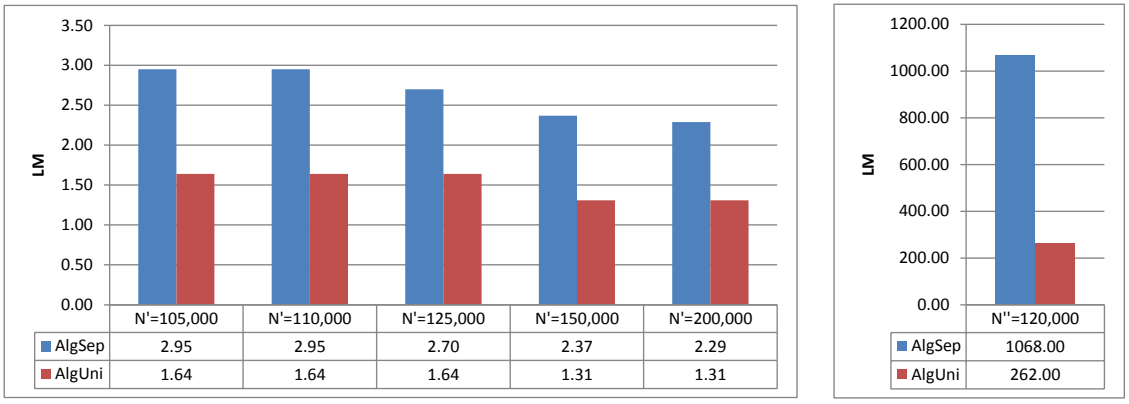
Table 15: Execution runtimes.

In the last set of experiments, we considered the scenario of a dynamic table. As explained earlier, we took the basic configuration CENSUS-1 (see Table 6), and then examined scenarios where T_1 has $N = 100,000$ records and T_2 has additional $N' - N$ records. Figure 16(a) shows the LM information losses in T_2 for five values of N' as achieved by the two algorithms described in Section 6 — the separative one that considers separately the two sets of records (ALGSEP in the figure) and the unifying one that operates on the unified set of records (ALGUNI). As expected, the latter issues significantly smaller information losses than the former, and the improvement is slightly larger when the number of new records is smaller, since then ALGSEP is more restricted in generalizing the new set of records.

It is interesting to compare the information losses of ALGUNI, as reported in Figure 16(a), to those in the third experiment set (Figure 11), which relates to similar static configurations. Comparing the total information loss of 1.64 in the dynamic case with $N' = 105,000$ to the total information loss of 0.98 in the static case with $N = 100,000$, and the information loss of 1.31 in the dynamic setting with $N' = 200,000$ to 0.82 in the corresponding static setting, we see that the information loss in the static setting are somewhat smaller (since in that setting the data owner has all records upfront and he may then perform better generalization decisions). However, the differences are very small, especially when we compare the information losses to those when using cut generalization in the static setting (see Figure 11) which were around 250.

In addition, we considered the case of dynamic table over three releases, T_1, T_2, T_3.

Those releases consisted of $N = 100,000$, $N' = 110,000$, and $N'' = 120,000$ records from CENSUS. T_1 included attributes `age` and `gender`, T_2 included `gender` and `sensitive`, and T_3 included `age` and `sensitive`. Figure 16(b) shows the LM information losses in T_3 as achieved by the separative and the unifying algorithms. Like in our previous experiments with $R > 2$ in the static setting, here too the information losses in T_3 are higher than those in T_2.



(a) Two releases

(b) Three releases

Figure 16: Dynamical tables

8. Conclusions

In this study we proposed a method for anonymizing sequential releases of databases in the sense of limiting the disclosure of sensitive data by the value of the quasi-identifiers. The proposed method allows any number of releases and the flexible local recoding model of "cell generalization". It also allows dynamics in the underlying database in the form of adding records to it. The experimental evaluation showed that cell generalization offers substantially better utility results than cut generalization, and targets much better the required level of privacy, as opposed to cut generalization. We also showed that privacy must be evaluated using the FMJ (in the case of 2 releases) or the KMJ (in the case of more than 2 releases) and must not be evaluated using the MJ. While switching from MJ to KMJ or FMJ has a significant effect on privacy, our experiments indicate that it has a small effect on the information loss. Possible future research directions are as follows:

(1) Our algorithm, like the one in [37], is a top-down algorithm. As shown in [17], bottom-up algorithms, or sequential clustering, achieve significantly better results than top-down algorithms. We intend to look for a practical bottom-up or sequential clustering algorithm that could be used and implemented efficiently in this context.

(2) In this study we concentrated on either static databases or dynamic databases to which records may only be added. In some cases, though, records may be also deleted or updated in between releases. Such a setting is more intricate due to a phenomenon that was called in [46] *critical absence*. For example, if the table T had only one individual with some sensitive value, and his record was removed at some point, subsequent releases of T would not include that sensitive value. If the adversary knows when that individual was removed from the table (since he knows, for example, when that individual was discharged from the hospital), he may be able to infer his sensitive value. That problem was addressed in [46] by introducing counterfeit tuples so that the multiset of sensitive values in the dynamic table will never narrow down. The extension of our algorithms to that case is left as future research.

(3) The underlying database T may be split horizontally or vertically between several data holders. We intend to devise a distributed protocol for implementing the algorithm that we presented here. To the best of our knowledge, all of previous studies of anonymizing distributed databases concentrated on the case of a single release.

References

[1] Aggarwal, C. and Yu, P. 2008. *Privacy-preserving data mining: models and algorithms*. Springer-Verlag New York Inc.

[2] Agrawal, R. and Srikant, R. 2000. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. 439–450.

[3] Bacchus, F., Grove, A. J., Koller, D., and Halpern, J. Y. 1992. From statistics to beliefs. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI '92)*. 602–608.

[4] Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., and Talwar, K. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '07)*. 273–282.

[5] Bayardo, R. and Agrawal, R. 2005. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*. 217–228.

[6] Bu, Y., Fu, A., Wong, R., Chen, L., and Li, J. 2008. Privacy preserving serial data publishing by role composition. *Proceedings of the 34th international conference on Very Large Data Bases (VLDB '08)*, 845–856.

[7] Burnett, L., Barlow-Stewart, K., Proos, A., and Aizenberg, H. 2003. The GeneTrustee: a universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine 10,* 4, 506–513.

[8] Byun, J.-W., Sohn, Y., Bertino, E., and Li, N. 2006. Secure anonymization for incremental datasets. In *Secure Data Management*. 48–63.

[9] Cao, J., Karras, P., Kalnis, P., and Tan, K. 2011. Sabre: a sensitive attribute bucketization and redistribution framework for t-closeness. *The VLDB Journal 20,* 1, 59–81.

[10] Cormode, G. 2011. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. 1253–1261.

[11] Dwork, C. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP '06)*. 1–12.

[12] Fung, B., Wang, K., Chen, R., and Yu, P. 2010. Privacy-preserving data publishing: a survey of recent developments. *ACM Computing Surveys (CSUR) 42,* 4, 1–53.

[13] Fung, B., Wang, K., Fu, A., and Pei, J. 2008. Anonymity for continuous data publishing. In *Proceedings of the 11th international conference on Extending Database Technology: advances in database technology (EDBT '08)*. ACM, 264–275.

[14] Fung, B., Wang, K., and Yu, P. 2005. Top-down specialization for information and privacy preservation. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*. IEEE, 205–216.

[15] Gionis, A., Mazza, A., and Tassa, T. 2008. k-Anonymization revisited. In *Proceedings of the 24th International Conference on Data Engineering (ICDE '08)*. 744–753.

[16] Gionis, A. and Tassa, T. 2009. k-Anonymization with minimal loss of information. *IEEE Transactions on Knowledge and Data Engineering 21*, 206–219.

[17] Goldberger, J. and Tassa, T. 2010. Efficient anonymizations with enhanced utility. *Transactions on Data Privacy 3*, 149–175.

[18] Iyengar, V. 2002. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. 279–288.

[19] Kifer, D. 2009. Attacks on privacy and definetti's theorem. In *Proceedings of the 2009 SIGMOD International Conference on Management of Data (SIGMOD '09)*. 127–138.

[20] Kifer, D. and Gehrke, J. Injecting utility into anonymized datasets. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*. 217–228.

[21] Last, M., Tassa, T., and Zhmudyak, A. Improving accuracy of classification models induced from anonymized datasets. *Submitted*.

[22] LeFevre, K., DeWitt, D., and Ramakrishnan, R. 2005. Incognito: efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of data (SIGMOD '05)*. 49–60.

[23] Li, N., Li, T., and Venkatasubramanian, S. 2010. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering 22,* 7, 943–956.

[24] Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. 2006. l-Diversity: privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*. 24.

[25] Malin, B. and Sweeney, L. 2004. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics 37,* 3, 179–192.

[26] Matatov, N., Rokach, L., and Maimon, O. 2010. Privacy-preserving data mining: a feature set partitioning approach. *Information Sciences 180,* 14, 2696–2720.

[27] Meyerson, A. and Williams, R. 2004. On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '04)*. 223–228.

[28] Nergiz, M. and Clifton, C. 2007. Thoughts on k-anonymization. *Data and Knowledge Engineering 63,* 3, 622–645.

[29] Park, H. and Shim, K. 2007. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. 67–78.

[30] Pei, J., Xu, J., Wang, Z., 0009, W. W., and Wang, K. 2007. Maintaining k-anonymity against incremental updates. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. 5.

[31] Rebollo-Monedero, D., Forne, J., and Domingo-Ferrer, J. 2010. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering 22,* 11, 1623–1636.

[32] Samarati, P. 2001. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering 13*, 1010–1027.

[33] Samarati, P. and Sweeney, L. 1998. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*. 188.

[34] Sweeney, L. 2002. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10,* 5, 557–570.

[35] Tassa, T., Mazza, A., and Gionis, A. 2012. k-Concealment: an alternative model of k-type anonymity. *Transactions on Data Privacy*.

[36] Wang, J., Luo, Y., Zhao, Y., and Le, J. 2009. A survey on privacy preserving data mining. In *Proceedings of the 1st International Workshop on Database Technology and Applications (DBTA '09)*. 111–114.

[37] Wang, K. and Fung, B. 2006. Anonymizing sequential release. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 414–423.

[38] Wang, K., Fung, B., and Yu, P. 2007. Handicapping attacker's confidence: an alternative to k-anonymization. *Knowledge and Information Systems 11,* 3, 345–368.

[39] Wang, K., Fung, B. C. M., and Yu, P. S. 2005. Template-based privacy preservation in classification problems. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM '05)*. 466–473.

[40] Wang, K., Yu, P. S., and Chakraborty, S. 2004. Bottom-up generalization: a data mining solution to privacy protection. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM '04)*. 249–256.

[41] Wong, R., Fu, A., Wang, K., and Pei, J. 2007. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*. 543–554.

[42] Wong, R., Li, J., Fu, A., and Wang, K. 2006. (α, k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 754–759.

[43] Wong, R. C.-W., Fu, A. W.-C., Liu, J., Wang, K., and Xu, Y. 2010. Global privacy guarantee in serial data publishing. In *Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE '10)*. 956–959.

[44] Wong, W. K., Mamoulis, N., and Cheung, D. W.-L. 2010. Non-homogeneous generalization in privacy preserving data publishing. In *Proceedings of the 2010 international conference on Management of Data (SIGMOD '10)*. 747–758.

[45] Xiao, X. and Tao, Y. Anatomy: simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*. 139–150.

[46] Xiao, X. and Tao, Y. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of Data (SIGMOD '07)*. 689–700.

[47] Yao, C., Wang, X., and Jajodia, S. 2005. Checking for k-anonymity violation by views. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*. 910–921.

9. Appendix

9.1. Summary of notations

Notation	Meaning
$A_1, \ldots, A_M$	The M attributes of the table, as well as the attribute domains
$A_1, \ldots, A_t$	The quasi-identifier attributes
$A_{t+1}, \ldots, A_{M-1}$	The non-identifier attributes
A_M	The sensitive attribute
$\overline{A}_m$	The collection of subsets of A_m that could be used for generalization, $1 \leq m \leq M$
T	The underlying table
S_n	The n-th record in T, $1 \leq n \leq N$
$S_n(m)$	The m-th entry in S_n, $1 \leq m \leq M$
T_r	The r-th release of T, $1 \leq r \leq R$
I_r	The set of indices of all attributes that are included in T_r
S_n^r	The n-th record in T_r, $1 \leq n \leq N$
$S_n^r(m)$	The m-th entry in S_n^r, $1 \leq m \leq M$
$T[R]$	The union of all records in all releases, $T[R] = T_1 \cup \cdots \cup T_R$
$G_{T[R]}$	The multipartite consistency graph on $T[R]$
C_n^H	The n-th R-clique in $G_{T[R]}$ that consists of the generalized images of S_n in $T_1, \ldots, T_R$
$\mathcal{MJ}_{T[R]}$	The Match Join (MJ) in $G_{T[R]}$
$\mathcal{FMJ}_{T[R]}$	The Full Match Join (FMJ) in $G_{T[R]}$
$\mathcal{KMJ}_{T[R]}$	The Kernel Match Join (KMJ) in $G_{T[R]}$

Table 16: Notation table

9.2. Proof of Theorem 3.1

Lemma 9.1. *In the case of cut generalization, if $\{S_n^r, S_{n'}^{r'}\}$ is an edge in $G = G_{T[R]}$, then $\{S_{n'}^r, S_n^{r'}\}$ is an edge too.*

Proof. Assume, for the sake of simplicity, that $r = n = 1$ and $r' = n' = 2$. Let $X \subseteq [M]$ be the subset of indices off all attributes that appear in both T_1 and T_2. Since $\{S_1^1, S_2^2\}$ is an edge in G, then $S_1^1(m) \uparrow S_2^2(m)$ for all $m \in X$. In order to prove that $\{S_2^1, S_1^2\}$ is an edge too, we need to show that $S_2^1(m) \uparrow S_1^2(m)$ for all $m \in X$. There are two cases to consider:

Case 1: In the m-th attribute $S_1^2(m) = S_2^2(m)$. As S_2^1 and S_2^2 are siblings, we know that $S_2^1(m) \uparrow S_2^2(m)$. Therefore $S_2^1(m) \uparrow S_1^2(m)$.

Case 2: In the m-th attribute $S_1^2(m) \cap S_2^2(m) = \emptyset$. In that case, as $S_1^1(m)$ is on the same generalization path with both $S_1^2(m)$ and $S_2^2(m)$, it must be an ancestor of both of them. Now, $S_2^1(m)$ either equals $S_1^1(m)$ or they are disjoint. They could not be disjoint since that would contradict the fact that $S_2^1(m) \uparrow S_2^2(m)$. Hence, $S_1^1(m) = S_2^1(m)$. Therefore, $S_2^1(m)$ is an ancestor of $S_1^2(m)$, namely $S_2^1(m) \uparrow S_1^2(m)$. □

Proof of Theorem 3.1. Assume, without loss of generality, that $r = 1$ and $r' = 2$. Assume that $S_1^1 \in T_1$ is connected by an edge to exactly t nodes in T_2, say $W := \{S_1^2, \ldots, S_t^2\}$. We shall prove three claims:

(1) Every node S_i^1, $1 \leq i \leq t$, is connected to each of the nodes in W.

(2) Every node S_i^1, $1 \leq i \leq t$, is not connected to any of the nodes in T_2 outside W.

(3) Every node S_i^1, $t+1 \leq i \leq N$, is not connected to any of the nodes in W.

Those three claims imply that the restriction of G to $T_1 \cup T_2$ has the complete bipartite graph on the first t nodes in T_1 and T_2 as one of its connected components. The proof may be thus completed by an inductive argument.

We begin by proving the first claim. If $t = 1$ the claim is trivial so we assume that $t \geq 2$. We shall show that S_2^1 must be connected to all nodes in W. Clearly, S_2^1 is connected to S_2^2 (by a horizontal edge) and also to S_1^2 (as implied by Lemma 9.1).It remains to prove that it must be connected also to S_i^2 for $3 \leq i \leq t$.

Fix $m \in X$, where $X \subseteq [M]$ is the subset of indices of all attributes that appear in both T_1 and T_2. Then

$$S_1^1(m) \uparrow S_2^2(m)\,, \tag{8}$$

$$S_1^1(m) \uparrow S_i^2(m)\,, \tag{9}$$

$$S_2^1(m) \uparrow S_2^2(m)\,, \tag{10}$$

and we wish to prove that also

$$S_2^1(m) \uparrow S_i^2(m)\,. \tag{11}$$

Eq. (8) implies that either $S_1^1(m) \subseteq S_2^2(m)$ or $S_1^1(m) \supset S_2^2(m)$. Let us consider first the case where $S_1^1(m) \subseteq S_2^2(m)$. The subsets $S_2^2(m)$ and $S_i^2(m)$ are either equal or disjoint. They could not be disjoint since that would contradict Eq. (9). Hence, $S_2^2(m) = S_i^2(m)$. Consequently, Eq. (11) follows from Eq. (10). Next, consider the case where $S_1^1(m) \supset S_2^2(m)$. In that case we conclude, in view of Eq. (10) and the fact that $S_1^1(m)$ and $S_2^1(m)$ are either equal or disjoint, that $S_1^1(m) = S_2^1(m)$. Hence, Eq. (11) follows from Eq. (9).

The second claim follows immediately. If, say, S_i^1 for some $1 \leq i \leq t$, would have been connected to a node outside W, then by arguing along the same lines as in the first claim, also S_1^1 would have to be connected to such a node, in contradiction to our assumption that W includes all neighbors of S_1^1 in T_2.

As for the third claim, assume, towards contradiction, that S_j^1, for $j > t$, is connected to $S_i^2 \in W$. Then, by Lemma 9.1, we should have that S_i^1 is connected to S_j^2, thus contradicting the second claim. □

9.3. Proof of Theorem 3.4

We shall prove Theorem 3.4 by showing a reduction from the problem of R-dimensional matching. In that problem, one is given an R-partite graph and it is needed to decide whether it has a perfect matching. That problem is known to be NP-complete for $R > 2$ (it appears in Karp's list of 21 NP-complete problems).

The proof consists of two stages. In Lemma 9.2 we show that the R-partite graphs $G_{T[R]}$ that correspond to sequential releases have no special structure, apart from the fact that

they have at least one perfect matching (the one that consists of all horizontal cliques). Then, in Lemma 9.3, we prove that the problem of computing the FMJ in such graphs is NP-hard.

Lemma 9.2. *Let G be an R-partite graph, where each of its parts consists of N nodes. If G has at least one perfect matching, there exists a table T and a corresponding sequential release $T[R] = T_1 \cup \cdots \cup T_R$ for which G is the consistency graph.*

Proof. Let us denote the nodes of G by $V = \{v_n^r : 1 \leq n \leq N,\ 1 \leq r \leq R\}$. Assume, without loss of generality, that the perfect matching in G consists of the cliques $C_n = \{v_n^1, v_n^2, \ldots, v_n^R\}$, $1 \leq n \leq N$. Given such a graph $G = (V, E)$, we proceed to define the corresponding table and sequential release.

The table T consists of N rows and $M := \binom{R}{2} N$ columns, and its entries are given by $S_n(m) = n$ for all $1 \leq n \leq N$, $1 \leq m \leq M$. In order to define the various releases, we need to introduce an alternative indexing for the columns of the table. Letting Ω denote the following domain of triplets,

$$\Omega = \{(r_1, r_2, n_1) : 1 \leq r_1 \leq R\,,\ r_1 + 1 \leq r_2 \leq R\,,\ 1 \leq n_1 \leq N\}\,,$$

it is clear that there is a one-to-one mapping between $[M] = \{1, \ldots, M\}$ and Ω. Given $m \in [M]$ we let $(r_1(m), r_2(m), n_1(m))$ denote its unique image in Ω under that mapping.

The n-th record in release T_r is denoted S_n^r; we describe its content, $S_n^r(m)$, $1 \leq m \leq M$, using the alternative triplet indexing:

$$S_n^r(m) = \begin{cases} n & r_1(m) = r \\ A(r_1(m), r, n) & r_2(m) = r, n_1(m) = n \\ * & r_2(m) = r, n_1(m) \neq n \\ * & \text{otherwise}\,, \end{cases}$$

where * denotes a totally suppressed entry and $A(r_1, r, n)$ is the set of indices ν of all neighbors of v_n^r from among $\{v_\nu^{r_1} : 1 \leq \nu \leq N\}$.

First, we claim that the above described releases comply with the model of cell generalization by taxonomies. The domain of each of the M attributes is $[N] := \{1, 2, \ldots, N\}$ and the generalized values in the m-th attribute are taken from the set $\{1, \ldots, N, A(r_1(m), r_2(m), n_1(m)), [N]\}$, which is a taxonomy for the set $[N]$.

We leave it for the reader to verify that the records $S_{n_1}^{r_1}$ and $S_{n_2}^{r_2}$ are consistent if and only if the nodes $v_{n_1}^{r_1}$ and $v_{n_2}^{r_2}$ are connected in G. That completes the proof. □

Example 9.1. Consider the bipartite graph given in Fig. 17. The table T and the two releases T_1 and T_2 that correspond to it, as described in the proof of Lemma 9.2, are given in Table 17.

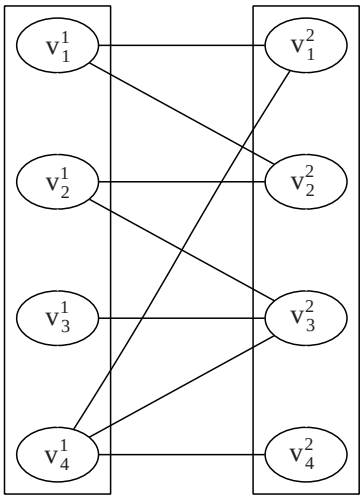


Figure 17: The graph to Example 9.1

T	A_1	A_2	A_3	A_4
S_1:	1	1	1	1
S_2:	2	2	2	2
S_3:	3	3	3	3
S_4:	4	4	4	4

T_1	A_1	A_2	A_3	A_4
S_1^1:	1	1	1	1
S_2^1:	2	2	2	2
S_3^1:	3	3	3	3
S_4^1:	4	4	4	4

T_2	A_1	A_2	A_3	A_4
S_1^2:	{1,4}	*	*	*
S_2^2:	*	{1,2}	*	*
S_3^2:	*	*	{2,3,4}	*
S_4^2:	*	*	*	{4}

Table 17: Table T (left), release T_1 (middle) and release T_2 (right)

□

Lemma 9.3. *The following problem is NP-complete for any constant $R > 2$: Given an R-partite graph G that has at least one perfect matching, and any R-clique in G, decide whether that clique is admissible (in the sense that there exists a perfect matching in G that contains it).*

Proof. Assume the existence of an oracle $\mathcal{O}$ that can decide in polynomial time the problem that is described in the lemma. We will use that oracle in order to solve in polynomial time the NP-complete problem of R-dimensional matching.

Let $G = (V, E)$ be an arbitrary R-partite graph with node set $V = \{v_n^r : 1 \leq n \leq N\,,\ 1 \leq r \leq R\}$. Define $G_0 = (V, E_0)$ where

$$E_0 = E \bigcup H\,, \quad H := \{\{v_n^{r_1}, v_n^{r_2}\} : 1 \leq r_1 < r_2 \leq R, 1 \leq n \leq N\}\,.$$

Namely, G_0 contains all edges of G and, in addition, it has all horizontal edges. Let us color all edges in E in black and all edges in $H \setminus E$ in red. Clearly, G_0 is a graph that has at least one perfect matching, which is given by the edges in H. The original graph G, on the other hand, has a perfect matching if and only if G_0 has an all-black perfect matching (that is, a perfect matching that does not involve any red edges).

Next, we will pick a red edge $e = \{u, v\}$ from G_0. Assume that we can decide in polynomial time whether $G_1 = (V, E_1 := E_0 \setminus \{e\})$ still has a perfect matching. Then, if the answer is NO, it means that the red edge e is essential for each of the perfect matchings in G_0, whence the original graph G does not have a perfect matching. If, on the other hand, the answer is YES, then G_1 is a graph that has at least one perfect matching, and, in addition, it has an all-black perfect matching if and only if G has a perfect matching. Therefore, we may repeat the above described procedure with G_1. Namely, we shall pick a red edge e' from G_1 and decide (by the assumed procedure that is yet to be described) whether $G_2 = (V, E_2 := E_1 \setminus \{e'\})$ still has a perfect matching.

This process may end in one of two ways. Either we shall get in one of the iterations a negative answer, indicating that G has no perfect matching; or we shall reach a graph G_k, for $k = |H \setminus E|$, that has no more red edges. That graph, which must have at least one perfect matching, is the original graph G. Hence, either way, we shall arrive at the sought-after decision regarding the original graph G. As the number of red edges is polynomial, the number of iterations of the above described procedure is polynomial in N.

It remains to show that if $G_i = (V, E_i)$ is any graph in the above described sequence and $e = \{u, v\} \in E_i$ is a red edge in it, we can decide in polynomial time whether $G_{i+1} = (V, E_{i+1} := E_i \setminus \{e\})$ still has a perfect matching. To that end, we compute two sets: The set C_e that consists of all admissible R-cliques in G_i that contain the edge e; and the set C_v that consists of all admissible R-cliques in G_i that contain the node v. We can compute those sets in polynomial time using the assumed oracle $\mathcal{O}$ since the number of R-cliques in the graph is polynomial. Note that $C_e \subseteq C_v$. There are two cases to consider:

- If $C_v \setminus C_e = \emptyset$ then all perfect matchings in G_i must contain the edge e (since every perfect matching must include the node v); hence, in that case we conclude that G_{i+1} does not have a perfect matching.

- If $C_v \setminus C_e \neq \emptyset$ then there exists an admissible R-clique that contains the node v but not the edge e; the perfect matching in G_i that contains that clique does not include the edge e. Such a perfect matching is therefore also a perfect matching in G_{i+1}.

The proof is thus complete. □