TRANSACTIONS ON DATA PRIVACY 5 (2012) 223–251

,

Utility-guided Clustering-based Transaction Data Anonymization†

Aris Gkoulalas-Divanis*, **Grigorios Loukides****,‡

* Information Analytics Lab, IBM Research – Zurich, Switzerland.

** School of Computer Science & Informatics, Cardiff University, UK.

E-mail: agd@zurich.ibm.com, g.loukides@cs.cf.ac.uk

Abstract. Transaction data about individuals are increasingly collected to support a plethora of applications, spanning from marketing to biomedical studies. Publishing these data is required by many organizations, but may result in privacy breaches, if an attacker exploits potentially identifying information to link individuals to their records in the published data. Algorithms that prevent this threat by transforming transaction data prior to their release have been proposed recently, but they may incur significant utility loss due to their inability to: (i) accommodate a range of different privacy requirements that data owners often have, and (ii) guarantee that the produced data will satisfy data owners' utility requirements. To address this issue, we propose a novel clustering-based framework to anonymizing transaction data, which provides the basis for designing algorithms that better preserve data utility. Based on this framework, we develop two anonymization algorithms which explore a larger solution space than existing methods and can satisfy a wide range of privacy requirements. Additionally, the second algorithm allows the specification and enforcement of utility requirements, thereby ensuring that the anonymized data remain useful in intended tasks. Experiments with both benchmark and real medical datasets verify that our algorithms significantly outperform the current state-of-the-art algorithms in terms of data utility, while being comparable in terms of efficiency.

Keywords. Data Privacy, Anonymization, Transaction Data, Clustering-based Algorithms

1 Introduction

Transaction datasets, containing information about individuals' behaviors or activities, are commonly used in a wide spectrum of applications, including recommendation systems [7], e-commerce [51], and biomedical studies [28]. These datasets are comprised of records, called *transactions*, which consist of a set of items, such as the products purchased by a customer of a supermarket, or the diagnosis codes contained in the electronic medical record of a patient.

Unfortunately, publishing transaction data may lead to privacy breaches, even when explicit identifiers, such as individuals' names or social security numbers, have been removed

†A preliminary version of this work appeared at the 4th EDBT International Workshop on Privacy and Anonymity in the Information Society (PAIS), March 25, 2011, Uppsala, Sweden.

‡Part of the work was done when the author was at the Health Information Privacy Lab, Vanderbilt University.

prior to data release. This became evident when New York Times journalists managed to re-identify an individual from a de-identified dataset of search keywords that was released by AOL Research [3]. The reason such privacy breaches may occur is because potentially identifying information (e.g., the diagnosis codes given to the individual during a hospital visit [6]) can be used to link an individual to her transaction in the published dataset, as explained in the following example.

Example 1. Consider releasing the dataset in Fig. 1(a), which records the items purchased by customers of a supermarket, after removing customers' names. This allows an attacker, who knows that *Anne* has purchased the items a, b, and c during her visit to this supermarket, to associate *Anne* with her transaction, since no other transaction in this dataset contains these 3 items together. □

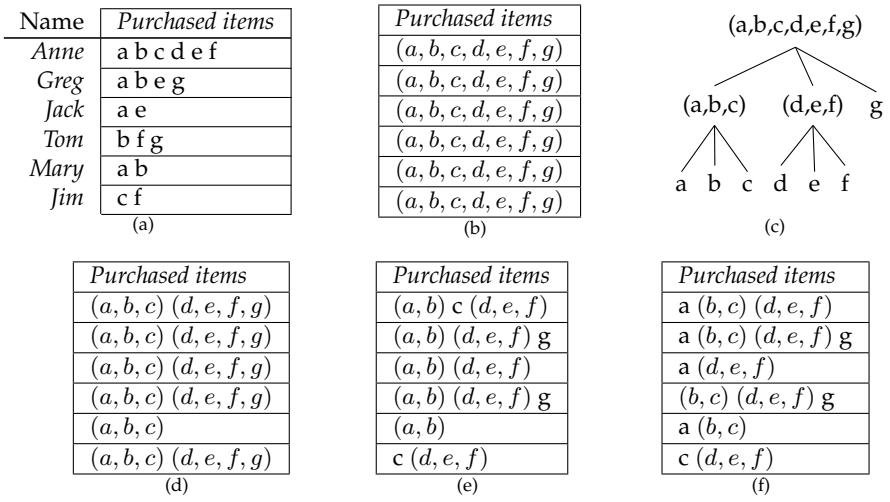

Figure 1: An example of: (a) original dataset, (b) output of Apriori Anonymization (AA), (c) item generalization hierarchy, (d) output of COAT, (e) output of PCTA, and (f) output of UPCTA.

Re-identification is a major privacy threat to individuals' privacy, which needs to be addressed for three main reasons. First, this is a requirement to comply with legislation and regulations that govern data sharing, such as regulations related to medical data sharing [1,2] or the EU Directive on privacy and electronic communications[1]. Second, avoiding to forestall re-identification may endanger future data collection [22], for example, in applications related to the sharing of electronic medical records [28], statistical [22] and video rental data [33], even when the identified transactions contain no sensitive information. Last, thwarting identity disclosure allows for better protection of potentially sensitive information contained in individuals' transactions. This is because it helps prevent attackers, whose knowledge is expected to be limited [29, 47, 50], from obtaining additional knowledge needed to infer sensitive information. For instance, having associated *Anne* with her transaction, the attacker considered in Example 1 can infer any other item purchased by *Anne*.

[1] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:NOT

Several methods for anonymizing transaction data have been proposed recently [6,12,29, 30,46,47,50], but they all produce solutions that may incur a large amount of information loss. This is because these methods consider only a small number of possible transformations to anonymize data and are unable to accommodate specific privacy requirements that data owners often have. For instance, the method introduced in [19] assumes that an item in the original data, represented as a leaf-level node in a generalization hierarchy such as the one shown in Fig. 1(c), can only be replaced by a node lying in the path between itself and the root of the hierarchy, and that an attacker has knowledge of all items associated with an individual transaction. This may lead to producing data with unnecessarily low data utility, particularly when attackers have knowledge of only some of the items that are associated with an individual, as is the case for transaction data that are high-dimensional and sparse [29,47,50]. Note that these characteristics of transaction data make it difficult to use a gamut of methods that have been developed for anonymizing relational data, such as those proposed in [4,5,9,11,24,31,34,41].

In this work, we propose a clustering-based approach to anonymizing transaction data with "low" information loss. Our work makes the following contributions:

- We propose a novel clustering-based framework inspired from agglomerative clustering [49]. This framework is independent of the way items are transformed and allows flexible algorithms that can anonymize transactions with low information loss, under various privacy requirements and utility requirements, to be developed.

- We design two effective and efficient anonymization algorithms based on our clustering-based framework. Both algorithms explore a large number of possible data transformations, which helps produce data which incur a small amount of information loss, and exploit a lazy updating strategy, which is crucial to achieving efficiency. Our first algorithm (called *PCTA*) focuses on dealing with privacy requirements, while the second one (called *UPCTA*) additionally ensures that utility requirements data owners often have can be expressed and satisfied by the anonymized result.

- We perform extensive experiments using two benchmark datasets containing clickstream data, as well as real patient data derived from the Vanderbilt University Medical Center to evaluate our algorithms. The results of our experiments confirm that both our algorithms significantly outperform the state-of-the-art algorithms [29, 47] in terms of retaining data utility, while maintaining good scalability.

The rest of this paper is organized as follows. Section 2 provides the necessary background and discusses related work. Section 3 introduces our clustering-based framework. In Section 4, we present our anonymization algorithms and, in Section 5, we evaluate them against the state-of-the-art methods. Finally, Section 6 concludes the paper.

2 Background and related work

In this section, we review transformation strategies for anonymizing transaction data, discuss measures that capture data utility, and provide an overview of anonymization principles and algorithms, focusing on those designed for transaction data.

2.1 Notation

Let $\mathcal{I} = \{i_1, ..., i_M\}$ be a finite set of literals, called *items*. Any subset $I \subseteq \mathcal{I}$ is called an *itemset* over $\mathcal{I}$, and is represented as the concatenation of the items it contains. An itemset that has m items, or equivalently a *size* of m, is called an m-itemset and its size is denoted with $|I|$. A dataset $\mathcal{D} = \{T_1, ..., T_N\}$ is a set of N transactions. Each *transaction* T_n, $n = 1, ..., N$, corresponds to a unique individual and is a pair $T_n = \langle tid, I \rangle$, where *tid* is a unique identifier and I is the itemset. A transaction $T_n = \langle tid, J \rangle$ *supports* an itemset I, if $I \subseteq J$. Given an itemset I in $\mathcal{D}$, we use $sup(I, \mathcal{D})$ to represent the number of transactions $T_n \in \mathcal{D}$ that support I. This set is called the set of *supporting transactions* of I in $\mathcal{D}$.

2.2 Data transformation strategies

Constructing an anonymous transaction dataset is possible through techniques that transform items. One such technique is perturbation [8, 10], which operates by adding or removing items from individuals' transactions with certain probability [39]. While the data produced by perturbation can be used to build accurate data mining models, perturbation produces falsified data that cannot be analyzed meaningfully at a transaction level. As a result, perturbation is not suitable for anonymizing data intended to support several applications, such as in biomedical analysis [10, 28], where data should by no means contain false observations. On the other hand, the techniques of suppression and generalization produce data that are not falsified. Both of these techniques can be applied either *globally*, in which case all items of the dataset undergo the same type of transformation, or *locally*, where items of certain transactions of the dataset are transformed. Suppression is an operation which removes items from the dataset before it is anonymized [50]. Global suppression is generally preferred, because it produces data in which all items have the same support as in the original dataset. This is important in building accurate data mining models using the anonymized data [50].

Generalization transforms an original dataset $\mathcal{D}$ to an anonymized dataset $\tilde{\mathcal{D}}$ by mapping original items in $\mathcal{D}$ to generalized items [29,47]. This technique often retains more information than suppression, as suppression is a special case of generalization where an original item is mapped to a generalized item that is not released [29]. Thus, global generalization can be considered as a mapping function from $\mathcal{I}$ to the space of generalized items $\tilde{\mathcal{I}}$, which is constructed by assigning each item $i \in \mathcal{I}$ to a unique generalized item $\tilde{i} \in \tilde{\mathcal{I}}$ that contains i. It is easy to observe that, given an item i that is mapped to a generalized item $\tilde{i}$, it holds that $sup(i, \mathcal{D}) \leq sup(\tilde{i}, \tilde{\mathcal{D}})$, because each transaction of $\mathcal{D}$ that supports i corresponds to at least one transaction of $\tilde{\mathcal{D}}$ that supports $\tilde{i}$. The following example illustrates the concept of global generalization.

Example 2. Consider Fig. 2 which illustrates the mapping of original items, contained in the dataset of Fig. 1(a), to the anonymized items of the dataset shown in Fig. 1(e). Based on this mapping, item a is mapped to the generalized item (a, b), and item c to the generalized item (c). We note that we may skip notation $()$ from a generalized item when a single item is mapped to it. The generalized item (a, b) is interpreted as a, or b, or a and b, and appears in the same transactions as those that have these items in the data of Fig. 1(a). Observe also that the generalized item (a, b) is supported by 5 transactions of the dataset of Fig. 1(e) (those that contained a and/or b before the anonymization), while each of the a and b is supported by 4 transactions of the dataset shown in Fig. 1(a). □

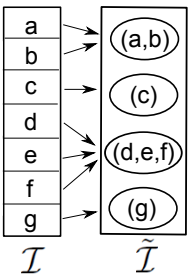

Figure 2: Mapping original to generalized items using global generalization.

2.3 Information loss measures

There are numerous ways to anonymize a transaction dataset, but the one that harms data utility the least, is typically preferred. To capture data utility, many criteria measure the information loss that is incurred by generalization based on item generalization hierarchies [46,48]. One of them is the Normalized Certainty Penalty (*NCP*) measure, which was introduced in [48] and employed in [46, 47]. *NCP* is expressed as the weighted average of the information loss of all generalized items, which are penalized based on the number of ascendants they have in the hierarchy. Other measures are the *multiple level mining loss* (*ML*^{2}), and *differential multiple level mining loss* (*dML*^{2}), which express utility based on how well anonymized data supports frequent itemset mining [46]. However, all the above measures require the items to be generalized according to hierarchies. A measure that can be used in the absence of hierarchies and captures the information loss incurred by both generalization and suppression is *Utility Loss* (*UL*) [29], which is defined below.

Definition 1 (Utility loss for a generalized item)**.** The Utility Loss (*UL*) for a generalized item $\tilde{i}$ is defined as

$$UL(\tilde{i}) = \frac{2^{|\tilde{i}|} - 1}{2^{|\mathcal{I}|} - 1} \times w(\tilde{i}) \times \frac{sup(\tilde{i}, \tilde{\mathcal{D}})}{N}$$

where $|\tilde{i}|$ denotes the number of items in $\mathcal{I}$ that are mapped to $\tilde{i}$, and $w : \tilde{\mathcal{I}} \rightarrow [0, 1]$ is a function assigning a weight according to the perceived usefulness of $\tilde{i}$ in analysis.

Definition 2 (Utility loss for an anonymized dataset)**.** The Utility Loss (*UL*) for an anonymized dataset $\tilde{\mathcal{D}}$ is defined as

$$UL(\tilde{\mathcal{D}}) = \sum_{\forall \tilde{i} \in \tilde{\mathcal{I}}} UL(\tilde{i}) + \sum_{\forall \textit{suppressed item } i_m \in \mathcal{I}} \mathcal{Y}(i_m)$$

where $\mathcal{Y} : \mathcal{I} \rightarrow \Re$ is a function that assigns a penalty, which is specified by data owners, to each suppressed item.

UL quantifies information loss based on the size, weight and support of generalized items, imposing a "large" penalty on generalized items that are comprised of a large number of "important" items that appear in many transactions. The size is taken into account because $\tilde{i}$ can represent any of the $(2^{|\tilde{i}|} - 1)$ non-empty subsets of the items mapped to it. That is, the larger $\tilde{i}$ is, the less certain we are about the set of original items represented by $\tilde{i}$. The support of $\tilde{i}$ also contributes to the loss of utility, as highly supported items will affect more transactions, resulting in more distortion. The denominators $(2^{|\mathcal{I}|} - 1)$ and N in Definition

1 are used for normalization purposes, so that the scores for *UL* are in $[0, 1]$. Moreover, a weight w is used to penalize generalizations exercised on more "important" items. This weight is specified by the data owner based on the perceived importance of the items to the subsequent analysis tasks. We note, however, that w can also be computed based on the semantic similarity of the items that are mapped to a generalized item [21, 29]. The following example illustrates how *UL* is computed.

Example 3. Consider that we want to compute the *UL* score for the generalized item $\tilde{i} = (a, b)$ in the dataset of Fig. 1(e) that was produced by anonymizing the dataset of Fig. 1(a), assuming that $w(\tilde{i}) = 1$. Since 2 out of the 7 items of the dataset of Fig. 1(a) are mapped to the generalized item (a, b), which is supported by 5 out of 6 transactions of the anonymized dataset of Fig. 1(e), we have that $UL(\tilde{i}) = \frac{2^2-1}{2^7-1} \times 1 \times \frac{5}{6} \approx 0.02$. □

2.4 Principles and algorithms for transaction data anonymization

Anonymization principles have been proposed for various types of data, such as relational [20, 24, 31, 35], sequential [38], trajectory [15, 23], and graph data [26]. In this section, we review the privacy principles that have been proposed for anonymizing transaction data and explain why and how they offer privacy protection from the main threats in data publishing, namely identity [47] and sensitive itemset disclosure [30, 50]. We also survey anonymization algorithms that can be used to enforce these principles.

Identity disclosure. A well-established and widely used anonymization principle is k-anonymity, which was originally proposed for relational data [41, 45], but has been recently adapted to other data types, including sequential [38], mobility [15, 18, 23], and graph data [26]. He et al. [19] applied a k-anonymity-based principle, called *complete k-anonymity*, to transaction datasets, requiring each transaction to be indistinguishable from at least $k - 1$ other transactions, as explained below.

Definition 3 (Complete k-anonymity)**.** Given a parameter k, a dataset $\mathcal{D}$ satisfies complete k-anonymity when $sup(I_j, \mathcal{D}) \geq k$, for each itemset I_j of a transaction $T_j = \langle tid_j, I_j \rangle$ in $\mathcal{D}$, with $j \in [1, N]$.

Satisfying complete k-anonymity guarantees protection against identity disclosure because it ensures that an attacker cannot link an individual to less than k transactions of the released dataset, even when this attacker knows all the items of a transaction. To enforce this principle, He et al. [19] proposed a top-down algorithm, called *Partition*, that uses a local generalization model. *Partition* starts by generalizing all items to the most generalized item lying in the root of the hierarchy and then replaces this item with its immediate descendants in the hierarchy if complete k-anonymity is satisfied. In subsequent iterations, generalized items are replaced with less general items (one at a time, starting with the one that incurs the least amount of data distortion), as long as complete k-anonymity is satisfied, or the generalized items are replaced by leaf-level items in the hierarchy. As mentioned in the Introduction, *Partition* has two shortcomings that lead to producing data with excessive information loss: (i) it cannot be readily extended to accommodate various privacy requirements that data owners may have, since its effectiveness and efficiency depend on the use of complete k-anonymity, and (ii) it explores a small number of possible generalizations due to the hierarchy-based model it uses to generalize data.

Terrovitis et al. [47] argued that it may be difficult for an attacker to acquire knowledge about all items of a transaction, in which case protecting all items would unnecessarily

incur excessive information loss. In response, the authors proposed the k^m-anonymity principle, defined as follows.

Definition 4 (k^m-anonymity)**.** Given parameters k and m, a dataset $\mathcal{D}$ satisfies k^m-anonymity when $sup(I, \mathcal{D}) \geq k$, for each m-itemset I in $\mathcal{D}$.

A k^m-anonymous dataset offers protection from attackers who know up to m items of an individual, because it ensures that these items cannot be used to associate this individual with less than k transactions of the released dataset. Terrovitis et al. [47] designed the Apriori algorithm to efficiently construct k^m-anonymous datasets. Apriori operates in a bottom-up fashion, beginning with 1-itemsets (items) and subsequently considering incrementally larger itemsets. In each iteration, the proposed algorithm enforces k^m-anonymity using the full-subtree, global generalization model [20]. The same authors have recently proposed two other algorithms to enforce k^m-anonymity [46], namely Vertical Partitioning Anonymization (VPA) and Local Recoding Anonymization (LRA). These algorithms operate in the following way. VPA first partitions the domain of items into sets and then generalizes items in each set to achieve k^m-anonymity. Next, the algorithm merges the generalized items to ensure that the entire dataset satisfies k^m-anonymity. LRA, on the other hand, partitions a dataset horizontally into sets in a way that would result in low information loss when the data is anonymized, and then generalizes items in each set separately, using local generalization. These algorithms are more flexible than *Partition* in the sense that they can be configured to offer protection against attackers who do not know all items of a transaction, but, contrary to our approach, perform hierarchy-based generalization.

Loukides et al. [29] proposed a privacy principle that imposes a lower bound of k to the support of combinations of items that need to be protected from identity disclosure. Different from previous works, the approach of [29] limits the amount of allowable generalization for each item to ensure that the generalized dataset remains useful for specific data analysis requirements. To satisfy this principle, the authors of [29] proposed COAT, an algorithm that operates in a greedy fashion and employs both generalization and suppression. The choice of the items generalized by COAT is governed by utility constraints that model data analysis requirements and correspond to the most generalized item that can replace a set of items. Thus, COAT allows constructing any generalized item that is not more general than an owner-specified utility constraint. When such an item is not found, COAT selectively suppresses a minimum number of items from the corresponding utility constraint to ensure privacy. Our method is similar to COAT in that it addresses the aforementioned limitations of the approaches of [19, 46, 47], but it preserves data utility better than COAT due to the use of clustering-based heuristics, as our experiments verify.

Sensitive itemset disclosure. Beyond identity disclosure is the threat of sensitive itemset disclosure, in which an individual is associated with an itemset that reveals some sensitive information, e.g., purchased items an individual would not be willing to be associated with. The afore-mentioned principles do not guarantee preventing sensitive itemset disclosure, since a large number of transactions that have the same generalized item can all contain the same sensitive itemset. To guard against this type of inferences, several approaches have been recently proposed. Ghinita et al. [12] developed an approach that releases transactions in groups, each of which contains public items in their original form and a summary of the frequencies of the sensitive items, while Cao et al. [6] introduced ρ-uncertainty, a privacy principle that limits the probability of inferring any sensitive itemset and a greedy algorithm to enforce it. The proposed algorithm for ρ-uncertainty iteratively suppresses sensitive items and then generalizes non-sensitive ones using the generalization

model of [47].

Identity and sensitive itemset disclosure. Different from [12] and [6], which provide no protection guarantees against identity disclosure, the works of [50] and [30] are able to prevent both identity and sensitive itemset disclosure. In particular, Xu et al. [50] proposed (h, k, p)-coherence, a privacy principle which treats public items similarly to k^m-anonymity (the function of parameter p is the same as m in k^m-anonymity) and additionally limits the probability of inferring any sensitive item using a parameter h. More recently, Loukides et al. [30] examined how to anonymize data to ensure that owner-specified itemsets are sufficiently protected. The authors proposed the notion of PS-rules to effectively capture privacy protection requirements and designed a generalization-based anonymization algorithm. This algorithm operates in a top-down fashion, starting with the most generalized transaction dataset, and then gradually replaces generalized items with less general ones, as long as the data remain protected. Our approach focuses on guarding against identity disclosure but can be extended to additionally prevent sensitive itemset disclosure. However, we leave this extension for future work.

Finally, we note that our work is orthogonal to approaches that investigate how to mine data in a way that the resulting data mining model will preserve privacy [43], or others that focus on preventing the mining of sensitive knowledge patterns, such as frequent itemsets [14, 16, 17, 36, 42, 44] or sequential patterns [13], from the released data.

3 Achieving anonymity through clustering

This section presents our clustering-based formulation of the transaction data anonymization problem. After introducing the main framework that satisfies detailed privacy requirements, we discuss how this framework can be extended to allow the specification and enforcement of utility requirements.

3.1 Dealing with privacy requirements

We model the task of anonymizing transaction data as a clustering problem. The latter problem requires assigning a label to each record of a dataset so that the records that are similar, according to an objective function, are assigned the same label. A series of papers, such as [5,25,31], have shown that anonymized relational datasets can be constructed based on clustering. In these approaches, records that incur low information loss when anonymized end up in the same cluster, and each cluster needs to contain at least k records to satisfy k-anonymity. To anonymize a transaction dataset $\mathcal{D}$, in this work, we aim at solving the following problem.

Problem 1. Construct a set of clusters $\mathcal{C}$ of generalized items such that: (i) each cluster $c \in \mathcal{C}$ corresponds to a unique generalized item, (ii) $\mathcal{C}$ satisfies the owner-specified privacy constraints, and (iii) the anonymized version $\tilde{\mathcal{D}}$ of $\mathcal{D}$, constructed based on $\mathcal{C}$, incurs minimal Utility Loss.

We note that Problem 1 is fundamentally different from the one considered in [5, 25, 31]. First, clusters are built around generalized items, and not transactions. As a result, a cluster that represents a generalized item $\tilde{i}$ may be associated with more than one transactions, since it is associated with the supporting transactions of $\tilde{i}$ in $\tilde{\mathcal{D}}$. Second, instead of requiring all clusters to have at least k records for achieving k-anonymity, we require the entire

anonymized dataset $\tilde{\mathcal{D}}$ to adhere to a set of specified privacy constraints that can span clusters. A privacy constraint [29] is modeled as a set of potentially linkable items from $\mathcal{I}$ and needs to be satisfied to thwart identity disclosure, as explained below.

Definition 5 (Privacy constraint)**.** A *privacy constraint* $p = \{i_1, ..., i_r\}$ is a set of potentially linkable items in $\mathcal{I}$. Given a parameter k of anonymity, p is *satisfied* in $\tilde{\mathcal{D}}$ when either $sup(p, \tilde{\mathcal{D}}) \geq k$ or $sup(p, \tilde{\mathcal{D}}) = 0$.

It is easy to observe that a set of privacy constraints can be specified to offer protection based on detailed privacy requirements and, alternatively, on complete k-anonymity [19] or k^m-anonymity [47], as explained in [29]. Privacy constraints can be satisfied by generalization and suppression. This is because the support of a generalized item in the anonymized dataset $\tilde{\mathcal{D}}$ is greater than or equal to the support of any item mapped to it in the original dataset $\mathcal{D}$, as discussed in Section 2.2, while suppressed items do not appear in $\tilde{\mathcal{D}}$. It is also worth noting that an attacker does not gain any advantage by using subsets of a privacy constraint p in linkage attacks, since, for every $p' \subseteq p$ and anonymized dataset $\tilde{\mathcal{D}}$, it holds that either $sup(p', \tilde{\mathcal{D}}) \geq sup(p, \tilde{\mathcal{D}})$ or $sup(p', \tilde{\mathcal{D}}) = 0$. The concept of privacy constraint and its satisfaction are illustrated in the following example.

Example 4. Consider the privacy constraint $\{a, d\}$ (which translates to "at least k transactions of the anonymized dataset should be associated with a, or d, or a and d") and that $k = 4$. This privacy constraint is satisfied in the anonymized data of Fig. 1(d) because the generalized item $(a, b, c) \cup (d, e, f, g)$ to which a and d are mapped, is supported by 5 transactions. □

The clustering-based model we propose aims to satisfy privacy constraints by progressively merging clusters as hierarchical agglomerative clustering algorithms do [49]. Since the support of a privacy constraint in $\tilde{\mathcal{D}}$ can become at least k or 0 as a result of applying generalization and suppression, respectively, a clustering that satisfies the specified privacy constraints will eventually be found by following a bottom-up approach that iteratively merges clusters formed by the items in $\mathcal{D}$, for any $k \in [2, N]$. This approach initially considers each original item as a singleton cluster and then iteratively merges singleton clusters (leading to the corresponding item generalizations) until the privacy constraints are satisfied. Although there are alternative approaches, such as divisive methods that split large clusters in a top-down fashion, these approaches have been shown to incur more information loss than bottom-up methods [48]. Since disparate item generalization decisions may incur a substantially different amount of information loss, the entire clustering process is driven by the *UL* measure, so that the two clusters that lead to minimizing information loss are merged at each step. This process is illustrated in Example 5.

Example 5. Assume that the original dataset of Fig. 4(a) needs to be anonymized to satisfy the privacy constraints $p_1 = \{i_1\}$ and $p_2 = \{i_5, i_6\}$ for $k = 3$. First, a set of singleton clusters are constructed, each built around one of the (generalized) items (i_1) to (i_7), as shown in Fig. 3. That is, the data of Fig. 4(a) are transformed as shown in Fig. 4(b). Since the specified privacy constraints are not satisfied in the dataset of Fig. 4(b), the current (singleton) clusters are subsequently merged. Among the different merging options, assume that merging the clusters for (i_1) and (i_2) incurs the minimum amount of utility loss, as measured by *UL*. This merging operation leads to a new cluster for the generalized item (i_1, i_2), which is associated with transactions T_1 and T_2, as shown in Fig. 3. Note that the latter cluster will always have a higher *UL* score than each of the clusters from which it

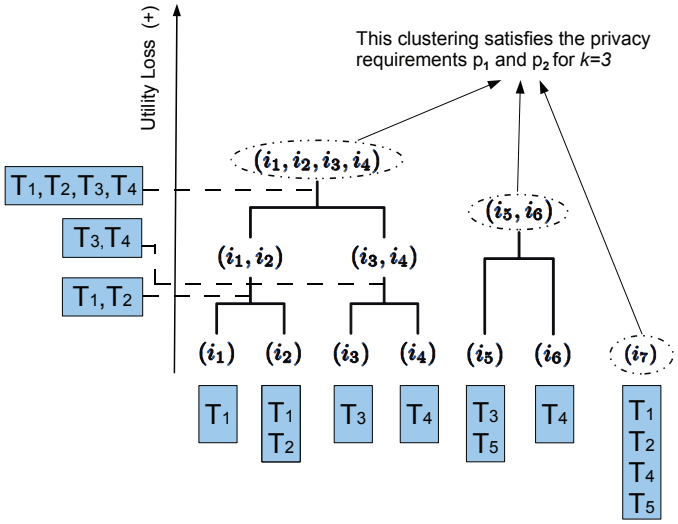

Figure 3: Data anonymization as a clustering problem.

tid	*Items*
T_1	$i_1\ i_2\ i_7$
T_2	$i_2\ i_7$
T_3	$i_3\ i_5$
T_4	$i_4\ i_6\ i_7$
T_5	$i_5\ i_7$

(a)

tid	*Items*
T_1	$(i_1)\ (i_2)\ (i_7)$
T_2	$(i_2)\ (i_7)$
T_3	$(i_3)\ (i_5)$
T_4	$(i_4)\ (i_6)\ (i_7)$
T_5	$(i_5)\ (i_7)$

(b)

tid	*Items*
T_1	$(i_1, i_2)\ (i_7)$
T_2	$(i_1, i_2)\ (i_7)$
T_3	$(i_3)\ (i_5)$
T_4	$(i_4)\ (i_6)\ (i_7)$
T_5	$(i_5)\ (i_7)$

(c)

tid	*Items*
T_1	$(i_1, i_2)\ (i_7)$
T_2	$(i_1, i_2)\ (i_7)$
T_3	$(i_3, i_4)\ (i_5)$
T_4	$(i_3, i_4)\ (i_6)\ (i_7)$
T_5	$(i_5)\ (i_7)$

(d)

tid	*Items*
T_1	$(i_1, i_2, i_3, i_4)\ (i_7)$
T_2	$(i_1, i_2, i_3, i_4)\ (i_7)$
T_3	$(i_1, i_2, i_3, i_4)\ (i_5, i_6)$
T_4	$(i_1, i_2, i_3, i_4)\ (i_5, i_6)\ (i_7)$
T_5	$(i_5, i_6)\ (i_7)$

(e)

Figure 4: Example of original data and its different anonymizations for Fig. 3.

was constructed. Still, the dataset produced by this clustering, shown in Fig. 4(c), does not satisfy the privacy constraints, because (i_1, i_2) is associated with less than k transactions. As a next step, the clusters for (i_3) and (i_4) are merged to create the cluster (i_3, i_4) that has the lowest *UL* score. This produces the dataset of Fig. 4(d). After additional cluster merging operations, shown in Fig. 3, the clusters (i_1, i_2, i_3, i_4), (i_5, i_6), and (i_7), are obtained and not extended any further, as they correspond to the dataset of Fig. 4(e) which satisfies the specified privacy constraints. This dataset can be safely released. □

An important benefit of adopting our clustering-based framework when designing anonymization algorithms is that it is general enough to accommodate different generalization models and anonymization requirements. This allows algorithms that exploit several generalization and privacy models to be developed. In terms of generalization models,

the soft (overlapping) clustering solution that is produced in the transaction-space by our clustering-based framework, leads to the generation of a *cover* instead of a *partition* of the original transactions, thereby allowing each produced cluster to be anonymized differently. This is important because it can lead to anonymizations with significantly less information loss [46]. Furthermore, the proposed model can be easily employed to anonymize data that satisfy stringent privacy and utility constraints, as we will discuss shortly. In any case, we note that finding the clustering that incurs minimum information loss is an NP-hard problem (the proof follows from [29]), and thus one needs to resort to heuristics to tackle it.

3.2 Incorporating utility requirements

Producing an anonymized dataset based on Problem 1, or by using the approaches of [12, 19, 47], guarantees that the dataset is protected from identity disclosure, but not that it will be useful in intended applications. This is because, even when the level of incurred information loss is minimal, generalized items that are difficult to be interpreted in these applications may be produced during anonymization. For instance, the dataset of Fig. 4(e) is not useful to a data recipient who is interested in counting the number of individuals associated with i_3, because the generalized item (i_1, i_2, i_3, i_4) may be interpreted as any non-empty subset of the itemset $i_1 i_2 i_3 i_4$. In fact, this requirement can only be satisfied when no other item is mapped to the same generalized item as i_3.

To satisfy such requirements, we employ the notion of *utility constraint* [29]. Utility constraints specify the generalized items that each item in the original dataset is allowed to be mapped to, effectively limiting the amount of generalization these items can receive. The specification of utility constraints is performed by data owners based on application requirements. The concepts of utility constraint and its satisfiability are explained in Definitions 6 and 7, and illustrated in Example 6.

Definition 6 (Utility constraint set)**.** A *utility constraint set* $\mathcal{U}$ is a partition of $\mathcal{I}$ that specifies the set of allowable mappings of the items from $\mathcal{I}$ to those of $\tilde{\mathcal{I}}$. Each element of $\mathcal{U}$ is called a *utility constraint*.

Definition 7 (Utility constraint set satisfiability)**.** Given a parameter s and an anonymized dataset $\tilde{\mathcal{D}}$, a utility constraint set $\mathcal{U}$ is *satisfied* if and only if (1) for each non-empty $\tilde{i_m} \in \tilde{\mathcal{I}}$, $\exists u \in \mathcal{U}$ such that all items from $\mathcal{I}$ in $\mathcal{D}$ that are mapped to $\tilde{i_m}$ are also contained in u, and (2) the fraction of items in $\mathcal{I}$ contained in the set of suppressed items $\mathcal{S}$ is at most $s\%$.

Utility constraint
$u_1 = \{i_1, i_2\}$
$u_2 = \{i_3\}$
$u_3 = \{i_4\}$
$u_4 = \{i_5, i_6, i_7\}$

Table 1: An example of a utility constraint set.

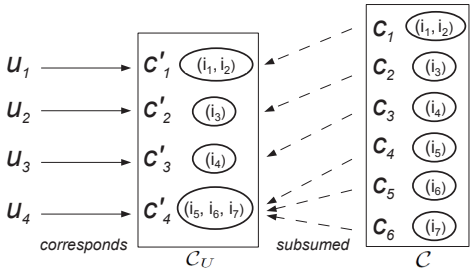

Figure 5: Utility-guided anonymization as a clustering problem.

Example 6. Consider the utility constraint set $\mathcal{U} = \{u_1, u_2, u_3, u_4\}$, which is shown in Table 1, and let $s = 0\%$. Satisfying $\mathcal{U}$ requires mapping i_1 and i_2 to the same generalized item, or releasing them intact. Furthermore, i_3 and i_4 need to be released intact, and each of the items i_5, i_6, and i_7 must be mapped to a unique generalized item to which no item other than these three items can be mapped. The anonymized dataset of Fig. 4(c) satisfies U, because all items i_1 to i_7 are generalized as specified by the utility constraints u_1 to u_4, and none of these items is suppressed. □

Notice that we limit the level of suppression incurred during anonymization by bounding the number of items that can be suppressed using a threshold s. Although suppression is not necessary to satisfying privacy requirements when there are no utility constraints, as in Problem 1, it is essential and beneficial when there are strict utility requirements. This is because it offers our approach the ability to deal with items that are difficult to be generalized together with others in a way that satisfies the privacy and utility constraints, without violating the latter constraints.

An anonymized dataset that satisfies a utility constraint set ensures that the number of transactions in the original dataset $\mathcal{D}$ that support at least one item mapped to a generalized item $\tilde{i}$ is equal to the number of transactions that support $\tilde{i}$ in the anonymized dataset $\tilde{\mathcal{D}}$. Producing such datasets is crucial in many applications, e.g., in biomedicine, where counting the number of patients suffering from any type of a certain disease is required [32].

In what follows, we explain how our clustering-based framework can be enhanced to produce anonymized data that respect the specified utility constraints. Since the set of utility constraints $\mathcal{U}$ is a partition of $\mathcal{I}$ (i.e., every item in $\mathcal{I}$ belongs in exactly one utility constraint), we can view the set of generalized items constructed by mapping all items in each $u \in \mathcal{U}$ to the same generalized item $\tilde{i}_{\tilde{u}}$ as a clustering $\mathcal{C}_{\mathcal{U}}$. Also, we state that a cluster $c \in \mathcal{C}$ is *subsumed* by a cluster $c' \in \mathcal{C}_U$ when, for each set of items that is mapped to a generalized item $\tilde{i}$ (represented as $c \in \mathcal{C}$), this set of items is mapped to exactly one generalized item $\tilde{i}'$ (represented as $c' \in \mathcal{C}_{\mathcal{U}}$). Thus, a utility constraint set is satisfied when each cluster $c \in \mathcal{C}$ is subsumed by exactly one cluster $c' \in \mathcal{C}_{\mathcal{U}}$, except those clusters in $\mathcal{C}$ that correspond to suppressed items. The following example illustrates these concepts.

Example 7. Consider the utility constraint set $\mathcal{U}$ of Example 6 and that $s = 0\%$. Each of the utility constraints u_1, u_2, u_3, and u_4 in $\mathcal{U}$ corresponds to a different cluster c'_1,c'_2, c'_3, and c'_4, which are the elements of the clustering $\mathcal{C}_{\mathcal{U}}$, as shown in Fig. 5. The utility constraint u_4, for example, corresponds to c'_4, which represents the generalized item (i_5, i_6, i_7). The cluster c_4, which belongs to the clustering $\mathcal{C}$, is subsumed by c'_4, because the generalized item (i_5) is mapped to the generalized item (i_5, i_6, i_7). Since each other cluster in $\mathcal{C}$ is subsumed by one cluster in $\mathcal{C}_{\mathcal{U}}$, the utility constraint set $\mathcal{U}$ is satisfied. □

The problem of anonymizing a transaction dataset $\mathcal{D}$ based on the specified utility constraints can be expressed as follows.

Problem 2. Construct a set of clusters $\mathcal{C}$ such that: (i) each cluster $c \in \mathcal{C}$ corresponds to a unique generalized item or a suppressed item, (ii) for each cluster $c \in \mathcal{C}$ that is mapped to a generalized item, there exists exactly one cluster in $\mathcal{C}_{\mathcal{U}}$ that *subsumes* c, (iii) there are at most $s\%$ items in $\mathcal{I}$ that correspond to suppressed items, (iv) $\mathcal{C}$ satisfies the owner-specified privacy constraints, (v) the anonymized version $\tilde{\mathcal{D}}$ of $\mathcal{D}$, constructed based on $\mathcal{C}$, incurs minimal Utility Loss.

Problem 2 is based on Problem 1 and remains NP-hard (the proof follows from [29]). However, its feasibility depends on the specification of utility and privacy constraints and that

of threshold s. To see this, assume that we need to anonymize the dataset of Fig. 4(a), using a single privacy constraint $p = \{i_1\}$, a utility constraint set comprised of one utility constraint $u_1 = \{i_1\}$, and parameters $k = 2$ and $s = 0\%$. Observe that the support of i_1 in this dataset is 1, so p is not satisfied, but neither generalizing nor suppressing i_1 would satisfy both p and $\mathcal{U}$. Nevertheless, as we will show in our experiments, using benchmark and real patient data, anonymizations that satisfy various privacy and utility requirements without incurring "high" information loss can be found.

4 Clustering-based anonymization algorithms

In this section, we present two algorithms that employ our clustering-based framework for anonymizing transaction data with "low" information loss. Section 4.1 presents our Privacy-Constrained Clustering-based Transaction Anonymization (PCTA) algorithm, which attempts to solve Problem 1, while Section 4.2 introduces a heuristic algorithm that tackles Problem 2 to produce anonymized data that satisfy utility requirements.

4.1 PCTA algorithm

Dealing with Problem 1 is possible by mapping original items to generalized items in order to construct a clustering, and then examining whether this clustering satisfies the specified privacy constraints. This is conceptually similar to how Apriori [47] and Partition [19] algorithms work. However, this strategy is likely to incur excessive information loss, because generalization is not "focused" on the items that are potentially linkable and need to be protected. For this reason, we opt for a different strategy that exploits the knowledge of which items need to be protected by targeting items contained in privacy constraints. Our strategy considers the imposed privacy constraints one at a time, selecting the privacy constraint p that is most likely to require a small amount of generalization in order to be satisfied. Then, it examines all possible cluster merging decisions that correspond to items in p and applies the one that leads to the minimum utility loss. The same process continues until the privacy constraint is satisfied, at which point the next non-satisfied privacy constraint is selected. Both this strategy and a novel lazy cluster-updating heuristic are used in the PCTA algorithm, whose pseudocode is provided in Algorithm 1.

The PCTA algorithm works as follows. In steps 1 and 2, we initialize $\tilde{\mathcal{D}}$ to $\mathcal{D}$ and a priority queue PQ to the set containing all the specified privacy constraints $\mathcal{P}$. PQ orders the constraints with respect to their support in decreasing order and implements the usual operations $top()$, which retrieves the privacy constraint that corresponds to an itemset with the maximum support in $\tilde{\mathcal{D}}$ without deleting it from PQ, and $pop()$, which deletes the privacy constraint with the maximum support from PQ. In steps $3-27$, PCTA iteratively merges clusters to increase the support of each privacy constraint in PQ to at least k, so that the constraint is satisfied in $\tilde{\mathcal{D}}$. More specifically, we assign the privacy constraint that lies in the top of PQ to p (step 4) and update its items to reflect the generalizations that have occurred in previous iterations of PCTA (steps $5-11$). This lazy updating strategy significantly improves the runtime cost of PCTA, as experimentally verified in Section 5.2.4, since the generalized items that are needed to update p are retrieved without scanning the anonymized dataset. This leads to considerably better efficiency, particularly when many clusters need to be merged, as is the case for large k values. For this purpose, we use a hashtable H which has each item of $\tilde{\mathcal{D}}$ as key and the generalized item that corresponds to this item as value. Then, we remove the privacy constraint p from PQ if its support is

Algorithm 1 PCTA($\mathcal{D}, \mathcal{P}, k$)

input: Dataset $\mathcal{D}$, set of privacy constraints $\mathcal{P}$, parameter k
output: Anonymous dataset $\tilde{\mathcal{D}}$

1. $\tilde{\mathcal{D}} \leftarrow \mathcal{D}$
2. $PQ \leftarrow$ privacy constraints of $\mathcal{P}$
3. **while** ($PQ \neq \emptyset$)
4. $p \leftarrow PQ.top()$
5. **foreach** ($i_m \in p$) *//lazy updating strategy*
6. **if** ($H(i_m) \neq i_m$)
7. $\tilde{i_m} \leftarrow H(i_m)$
8. **if** ($\tilde{i_m} \in p$)
9. $p \leftarrow p \backslash i_m$
10. **else**
11. $p \leftarrow (p \backslash i_m) \cup \tilde{i_m}$
12. **if** ($sup(p, \tilde{\mathcal{D}}) \geq k$) *//p is protected*
13. $PQ.pop()$
14. **else** *// apply generalization to protect p*
15. **while** ($sup(p, \tilde{\mathcal{D}}) < k$)
16. $\mu \leftarrow 1$ *// maximum UL score*
17. **foreach** ($i_m \in p$)
18. $i_s \leftarrow \arg \min_{\forall i_r \in H, i_r \neq i_m} UL(\ (i_m, i_r)\ )$
19. **if** ($UL(\ (i_m, i_s)\ ) < \mu$)
20. $\mu \leftarrow UL(\ (i_m, i_s)\ )$
21. $\sigma \leftarrow \{i_m, i_s\}$
22. $\tilde{i} \leftarrow (i_m, i_s)$ *// generalize σ (cluster merging)*
23. update transactions of $\tilde{\mathcal{D}}$ based on σ
24. $p \leftarrow (p \cup \{\tilde{i}\}) \backslash \sigma$ *// update p to reflect the generalization of i_m to σ*
25. **foreach** ($i_r \in \sigma$)
26. $H(i_r) \leftarrow \tilde{i}$
27. $PQ.pop()$
28. **return** $\tilde{\mathcal{D}}$

at least k (step 13), in which case p is satisfied by the current clustering solution, or we merge clusters to protect it (steps $11 - 27$), if p is still unprotected in $\tilde{\mathcal{D}}$. In steps $16 - 21$, we select the best cluster merging decision among the clusters that affect the support of privacy constraint p. This is achieved by identifying the item i_m that can be generalized with another item i_s such that the resultant item σ incurs the least amount of information loss as measured by UL. When the best pair of clusters is found, PCTA performs the merging of the clusters by generalizing the items' pair σ to construct a new generalized item $\tilde{i}$ (step 22). Following that, the affected transactions in $\tilde{\mathcal{D}}$, the items in privacy constraint p, and the hashtable H, are all updated to reflect the new generalization (steps $23 - 26$). Steps $15 - 26$ are repeated until the support of p becomes at least k, in which case the current clustering satisfies the privacy constraint p. Then, p is removed from PQ in step 27. Finally, the dataset $\tilde{\mathcal{D}}$ is returned in step 28.

To illustrate the operation of the PCTA algorithm, we provide the following example.

Example 8. Consider applying PCTA to the dataset of Fig. 1(a), assuming a single privacy constraint $p = \{a, b, e, f\}$ and $k = 3$. In steps 1 and 2, we initialize the anonymized dataset $\tilde{\mathcal{D}}$ to the original dataset $\mathcal{D}$ and add p to the priority queue PQ. Then, in step 4, we retrieve

p from PQ and subsequently (steps 5-11) iterate over its items a, b, e and f, replacing each of them with its value in the hashtable H. Since these items have not been generalized before, their values in H contain the items themselves and thus p is left intact. Next, in step 12, we compute the support of p in $\tilde{\mathcal{D}}$, and, since it is less than k, we execute the loop beginning in step 15. In steps 16 to 21, PCTA considers all possible cluster merging operations that affect the privacy constraint p. Put in terms of item generalization decisions, the algorithm considers generalizing each of the items $\{a, b, e, f\}$ together with any other item in the domain $\mathcal{I}$ and constructs the generalized item (d, f), which incurs the minimum utility loss among all the examined generalized items. Next, the algorithm assigns (d, f) to $\tilde{i}$, in step 22, and updates $\tilde{\mathcal{D}}$, p and H (steps 23-26). Specifically, the generalized item (d, f) replaces d, and the values of d and f in H are updated. Since the support of p remains less than k after generalizing d to (d, f), the loop of step 15 is executed again. Now, PCTA considers a, b, e, and (d, f) for generalization and constructs (a, b) that has the minimum utility loss. While p is updated to $(a, b)e(d, f)$, it still has a support of less than k, and thus PCTA performs another iteration of the loop of step 15. In the latter, p is updated to $\{(a, b)(d, e, f)\}$, which has a support of 3. Thus, in step 27, p is removed from PQ and, in step 28, the anonymized dataset of Fig. 1(e) is returned. Figure 6 illustrates the anonymization process. □

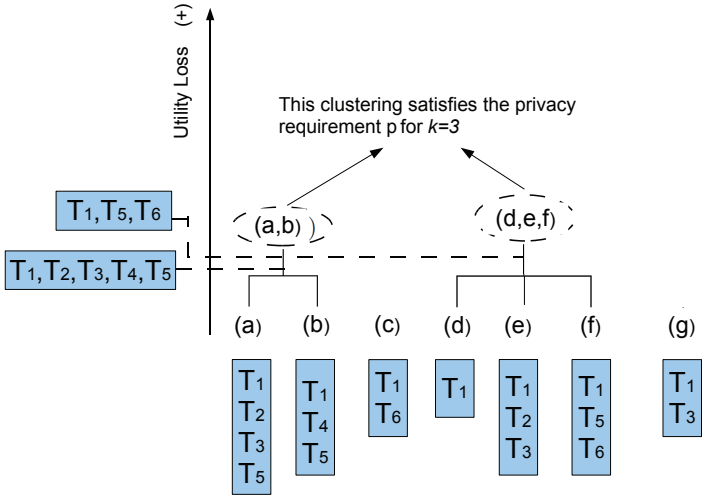

Figure 6: Anonymizing the data of Fig. 1(a) using PCTA

Cost analysis. Assuming that we have $|\mathcal{P}|$ privacy constraints and each of them has $|p|$ items, PCTA takes $O(|\mathcal{P}| \times |p| \times (N + |\mathcal{I}|^2))$ time. This is because we need $O(|\mathcal{P}| \times log(|\mathcal{P}|))$ time to build PQ and $O(|\mathcal{P}| \times |p| \times (log(|\mathcal{I}|) + N + |\mathcal{I}|^2))$ time for steps $3-27$. More specifically, the lazy updating strategy for the items of p in steps $5 - 11$ takes $O(|p| \times log(|\mathcal{I}|))$ time, the support computation of p in step 12 takes $O(|p| \times N)$ time, and the while loop in step 15 takes $O(|p| \times (|\mathcal{I}| - 1) + (|p| - 1) \times (|\mathcal{I}| - 2) + \ldots + 1 \times 1) \approx O(|p| \times |\mathcal{I}|^2)$ time.

4.2 UPCTA algorithm

In this section, we present UPCTA (Utility-guided Privacy-constrained Clustering-based Transaction Anonymization), an algorithm that takes into account both privacy and utility

requirements when anonymizing transaction data. Specifically, given an original dataset $\mathcal{D}$, sets of utility and privacy constraints $\mathcal{P}$ and $\mathcal{U}$, respectively, and parameters k and s, UPCTA selects the privacy constraint $p \in \mathcal{P}$ that is supported by the most transactions in $\mathcal{D}$, among the privacy constraints whose support is in $(0, k)$ (splitting ties arbitrarily), and generalizes and/or suppresses items in p to satisfy it. Different from PCTA, generalization in UPCTA is performed in accordance with the utility constraint set $\mathcal{U}$, while suppression is also employed when generalization alone is not sufficient to protect p. The process is repeated for all privacy constraints until $\mathcal{P}$ is satisfied.

UPCTA begins by initializing the original dataset $\tilde{\mathcal{D}}$ to $\mathcal{D}$, and a priority queue PQ, which orders its elements in decreasing order of support, to the set containing all the privacy constraints in $\mathcal{P}$ (steps 1-2). Then, it selects the privacy constraint p provided by $PQ.top()$ (step 4) and updates its items to reflect the cluster merging operations and/or suppressions that may have occurred in previous iterations (steps $5-13$). If p is supported by at least k or none of the transactions of $\tilde{\mathcal{D}}$, it is removed from PQ, as it is satisfied (steps 14-15). Otherwise, UPCTA keeps merging clusters until either p is satisfied or further merging would violate the set of utility constraints $\mathcal{U}$ (steps $17-31$).

Specifically, the algorithm first identifies the utility constraint u_p in which i_m belongs and then selects the best cluster merging decision, among the clusters that affect the support of p (steps 20-25). That is, a merging that has the lowest *UL* sore and does not violate $\mathcal{U}$. When the best pair of clusters that can be merged is found, UPCTA performs their merging by generalizing this pair of items to construct a new generalized item $\tilde{i}$ (step 26). Following that, the affected transactions in $\tilde{\mathcal{D}}$, the items in p and those in u, and the hashtable H are all updated to reflect the merging operation (steps 27-31). Steps $17-31$ are repeated until p is satisfied or until every utility constraint in $\mathcal{U}$ contains a single item (or generalized item).

If p is not satisfied after UPCTA exits from the loop of step 17, no further generalization that satisfies $\mathcal{U}$ is possible, and we apply suppression (steps 33-40). In steps 33-34, the algorithm identifies the item (or generalized item) i_m that is contained in p and has the minimum support in $\tilde{\mathcal{D}}$ (breaking ties arbitrarily), and the utility constraint u that contains i_m. Then, it removes i_m from u, $\tilde{\mathcal{D}}$, and p, and updates H to reflect the suppression of i_m (steps 35-38). Next, UPCTA checks if the number of suppressed items exceed $s\%$ (step 39). In this case, it notifies data owners that the utility constraint set $\mathcal{U}$ has been violated and terminates (step 40), otherwise it proceeds to checking whether p is now satisfied. After protecting p, UPCTA removes it from PQ (step 41) and continues to examining the next element of PQ, if there is one. Finally, the dataset $\tilde{\mathcal{D}}$ is returned (step 42).

The example below illustrates how UPCTA works.

Example 9. Consider applying UPCTA on the dataset of Fig. 1(a), using a set of privacy constraints $\mathcal{P} = \{p_1, p_2\}$, where $p_1 = \{b, e, f\}$ and $p_2 = \{g\}$, a set of utility constraints $\mathcal{U} = \{u_1, u_2, u_3, u_4\}$, where $u_1 = \{a\}$, $u_2 = \{b, c\}$, $u_3 = \{d, e, f\}$, and $u_4 = \{g\}$, and the parameters k and s to be 3 and 15%, respectively. Both p_1 and p_2 are inserted into PQ, and UPCTA retrieves p_2, because it is supported by more transactions than p_1 in the dataset of Fig. 1(a). Since p_2 is not satisfied, the algorithm attempts to generalize the item g contained in it. As the utility constraint $u_4 = g$, and 2 transactions of the dataset of Fig. 1(a) support p_2, the algorithm has to suppress g. After that, p_1, the next privacy constraint retrieved by PQ, is considered. This privacy constraint is not satisfied, $u_3 = \{d, e, f\}$, and both e and f are contained in p_1. Consequently, UPCTA considers all possible cluster merging operations that do not violate $\mathcal{U}$ for the items in p_1 (i.e., generalizing b with c, e with d or f, and f with e or d), and constructs the generalized item (d, f), which has the lowest *UL* score. Then, UPCTA assigns (d, f) to $\tilde{i}$ and updates $\tilde{\mathcal{D}}$, p, u_3, and H. Next, the clustering

Algorithm 2 UPCTA($\mathcal{D}, \mathcal{U}, \mathcal{P}, k, s$)

input: Dataset $\mathcal{D}$, set of utility constraints $\mathcal{U}$, set of privacy constraints $\mathcal{P}$, parameters k and s
output: Anonymous dataset $\tilde{\mathcal{D}}$

1. $\tilde{\mathcal{D}} \leftarrow \mathcal{D}$
2. $PQ \leftarrow$ privacy constraints of $\mathcal{P}$
3. **while** ($PQ \neq \emptyset$)
4. $p \leftarrow PQ.top()$
5. **foreach** ($i_m \in p$) *//lazy updating strategy*
6. **if** ($H(i_m) = *$) *//i_m has been suppressed in a previous iteration*
7. $p \leftarrow p \backslash i_m$
8. **else if** ($H(i_m) \neq i_m$) *//i_m has been generalized in a previous iteration*
9. $\tilde{i_m} \leftarrow H(i_m)$
10. **if** ($\tilde{i_m} \in p$)
11. $p \leftarrow p \backslash i_m$
12. **else**
13. $p \leftarrow (p \backslash i_m) \cup \tilde{i_m}$
14. **if** ($sup(p, \tilde{\mathcal{D}}) \geq k$ or $sup(p, \tilde{\mathcal{D}}) = 0$) *//p is protected*
15. $PQ.pop()$
16. **else** *// apply utility-guided generalization and/or suppression to protect p*
17. **while** ($sup(p, \tilde{\mathcal{D}}) \in (0, k)$ and $\exists u \in \mathcal{U}$ s.t. it contains at least 2 items, one of which is in p)
18. $\mu \leftarrow 1$ *// maximum UL score*
19. **foreach** ($i_m \in p$)
20. $u_p \leftarrow$ the utility constraint from $\mathcal{U}$ that contains i_m
21. **if** (u contains at least 2 items, one of which is in p)
22. $i_s \leftarrow \arg \min_{\forall i_r \in H, i_r \neq i_m} UL(\ (i_m, i_r)\ )$
23. **if** ($UL(\ (i_m, i_s)\ ) < \mu$)
24. $\mu \leftarrow UL(\ (i_m, i_s)\ )$
25. $\sigma \leftarrow \{i_m, i_s\}$
26. $\tilde{i} \leftarrow (i_m, i_s)$ *// generalize σ (cluster merging)*
27. update transactions of $\tilde{\mathcal{D}}$ based on σ
28. $p \leftarrow (p \cup \{\tilde{i}\}) \backslash \sigma$
29. $u \leftarrow u \cup \tilde{i} \backslash \sigma$
30. **foreach** ($i_r \in \sigma$)
31. $H(i_r) \leftarrow \tilde{i}$
32. **while** ($sup(p, \tilde{\mathcal{D}}) \in (0, k)$) *// apply suppression to protect p*
33. $i_m \leftarrow \arg \min_{\forall i_r \in p} \sup(i_r, \tilde{\mathcal{D}})$
34. $u \leftarrow$ the utility constraint from $\mathcal{U}$ that contains i_m
35. $u \leftarrow u \backslash \{i_m\}$
36. $p \leftarrow p \backslash \{i_m\}$
37. Remove i_m from all transactions of $\tilde{\mathcal{D}}$
38. $H(i_m) \leftarrow *$ *//update H to reflect the suppression of i_m*
39. **if** more than $s\%$ of items are suppressed
40. Error: $\mathcal{U}$ is violated
41. $PQ.pop()$
42. **return** $\tilde{\mathcal{D}}$

merging operation that has the minimum utility loss and does not violate u_2 is performed, since p_1 is still not satisfied and $u_2 = \{(d, f), e\}$. This results in creating (d, e, f) and in updating p_1, u_2, and H. After that, UPCTA generalizes b to (b, c), since p_1 is not satisfied

and $u_1 = \{(b, c), e\}$. At this point, the privacy constraint $p_2 = \{(b, c,), (d, e, f)\}$ is satisfied and the dataset of Fig. 1(f) is returned. □

Cost analysis. Assuming that we have $|\mathcal{P}|$ privacy constraints and each of them has $|p|$ items, as well as $|\mathcal{U}|$ utility constraints, each of which has $|u|$ items, UPCTA takes $O(|P| \times |p| \times (log(|\mathcal{I}|) + |u|^2 + N \times (|\mathcal{U}| + |u| + |p| + N)))$ time. Specifically, we need $O(|\mathcal{P}| \times log(|\mathcal{P}|))$ time to build PQ and $O(|\mathcal{P}| \times |p| \times (log(|\mathcal{I}|) + |u|^2 + N \times (|\mathcal{U}| + |u| + |p| + N)))$ time for steps $17 - 40$. This is because steps 5-13 take $O(|p| \times log(|\mathcal{I}|))$ time, step 14 takes $O(|p| \times N)$ time, and the while loops in steps 17 and 32 take $O(|p| \times |u|^2)$ and $O(|p| \times N \times (|\mathcal{U}| + |u| + |p| + N))$ time, respectively.

5 Experimental Evaluation

In this section, we present extensive experiments to evaluate the ability of our algorithms to produce anonymized data with low information loss efficiently. Section 5.1 discusses the experimental setup and the datasets we used. In Section 5.2, we evaluate PCTA against Apriori [47] and COAT [29], in terms of data utility, under several different privacy requirements, as well as in terms of efficiency. The results of this set of experiments confirm that PCTA is able to retain much more data utility when compared to the other methods under all tested scenarios, as: (1) it allows aggregate queries to be answered many times more accurately (e.g., the average error for PCTA was up to 26 and 6 times lower than that of Apriori and COAT respectively), and (2) it incurs an amount of information loss that is smaller by several orders of magnitude, while being scalable with respect to both dataset size and k. In Section 5.3, we compare UPCTA against COAT, the algorithm that, to the best of our knowledge, is the only one that can take into account utility requirements. Our results show that UPCTA produces data that permit up to 3.8 times more accurate aggregate query answering than those generated by COAT, incurs significantly lower information loss, and scales similarly to COAT.

5.1 Experimental setup and data

To allow a direct comparison between the tested algorithms, we configured all of them as in [29] and transformed the anonymized datasets produced by them by replacing each generalized item with the set of items it contains. We used a C++ implementation of Apriori provided by the authors of [47] and implemented COAT and our algorithms in C++. All methods ran on an Intel 2.8GHz machine with 4GB of RAM and were tested using a common framework to measure data utility that was built in Java.

We compared the amount of data utility preserved by all methods by considering three utility measures: Average Relative Error (*AvgRE*) [12, 24], Utility Loss (*UL*) [29], and Normalized Certainty Penalty (*NCP*) [48]. *AvgRE* captures the accuracy of query answering on anonymized data. It is computed as the mean error of answering a workload of queries and reflects the average number of transactions that are retrieved incorrectly as part of query answers. To measure *AvgRE*, we constructed workloads comprised of 1000 COUNT() queries that retrieve the set of supporting transactions of 5-itemsets, following the methodology of [12, 29]. The items participating in these queries were selected randomly from the generalized items.

In our experiments, we used the datasets *BMS-WebView-1* and *BMS-WebView-2* (referred to as *BMS1* and *BMS2* respectively), which contain click-stream data from two e-commerce

sites and have been used extensively in evaluating prior work [12, 29, 47]. In addition, we used 2 real datasets that contain de-identified patient records derived from the Electronic Medical Record (EMR) system of Vanderbilt University Medical Center [40]. These datasets are referred to *VNEC* and $VNEC_{KC}$ and were introduced in [28]. The datasets we used have different characteristics, shown in Table 2.

Dataset	N	$\|\mathcal{I}\|$	Max. size of T	Avg. size of T
BMS1	59602	497	267	2.5
BMS2	77512	3340	161	5.0
VNEC	2762	5830	25	3.1
$VNEC_{KC}$	1335	305	3.1	5.0

Table 2: Description of used datasets

5.2 Evaluation of PCTA

In this section, we compare the PCTA algorithm to Apriori and COAT. For a fair comparison, COAT was configured to neglect the specified utility constraints i.e, to allow an item to be mapped to any possible generalized item.

5.2.1 Data utility for k^m-anonymity

We first evaluated data utility assuming that combinations of up to 2 items need to be protected. Thus, we set $m = 2$ and configured COAT and PCTA by using all 2-itemsets as privacy constraints. We also used various k values in $[2, 50]$. Fig. 7 illustrates the result with respect to *AvgRE* for *BMS1* and Fig. 8 for *BMS2*. It can be seen that PCTA allows at least 7 and 2 times (and up to 26 and 6 times) more accurate query answering than Apriori and COAT respectively. Furthermore, PCTA incurred significantly less information loss to anonymize data, as shown in Fig. 9, which illustrates the *UL* scores for *BMS1*. These results verify that the clustering-based search strategy that is employed by PCTA is much more powerful than the space partitioning strategies of Apriori and COAT.

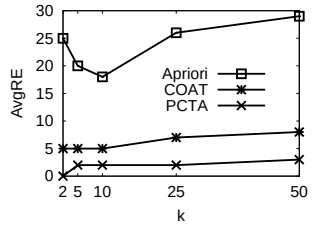

Figure 7: *AvgRE* vs. k (*BMS1*)

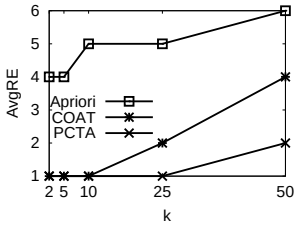

Figure 8: *AvgRE* vs. k (*BMS2*)

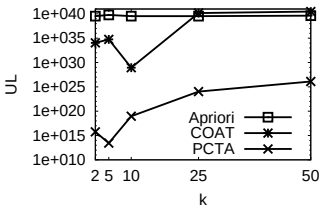

Figure 9: UL vs. k (*BMS1*)

Another observation is that the performance of both Apriori and COAT in terms of preserving data utility deteriorates much faster when k increases. This is mainly because these algorithms create much larger groups of items than PCTA. Specifically, Apriori considers fixed groups of items, whose size depends on the fan-out of the hierarchy, and generalizes together all items in each group, while COAT partitions items based on the utility loss incurred by generalizing a single item in a group.

Next, we assumed that combinations of 1 to 3 items need to be protected using $k = 5$. Figure 10 illustrates the result with respect to *AvgRE* for *BMS1*. Observe that the amount of

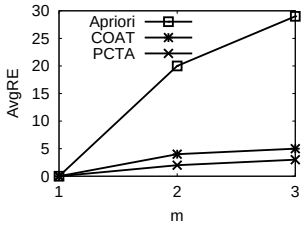

Figure 10: *AvgRE* vs. m (*BMS1*)

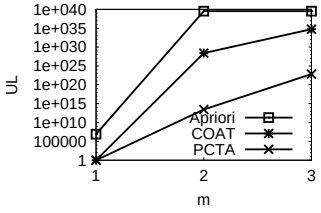

Figure 11: UL vs. m (*BMS1*)

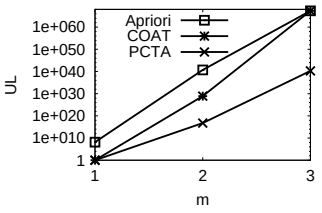

Figure 12: UL vs. m (*BMS2*)

information loss incurred by all methods decreases as a function of the number of items attackers are expected to know, as more generalization is required to preserve privacy. However, PCTA outperformed Apriori and COAT in all cases, permitting queries to be answered with an error that is *at least* 8 *times lower than that of Apriori and* 1.6 *times lower than that of COAT*. Similar results were obtained when *UL* was used to capture data utility, as can be seen in Figs. 11 and 12 for *BMS1* and *BMS2*, respectively. This demonstrates that protecting incrementally larger itemsets as Apriori does, leads to significantly more generalization compared to applying generalization to protect that items in each privacy constraint as in the PCTA algorithm.

5.2.2 Data utility for privacy constraints with various characteristics

For this set of experiments, we constructed 5 sets of privacy constraints: *PR1*,...,*PR5*, comprised of 2-itemsets, and we assumed that they need protection with $k = 5$. Each set contains a certain percentage of randomly selected items, which is 2% for *PR1*, 5% for *PR2*, 10% for *PR3*, 25% for *PR4*, and 50% for *PR5*. The *AvgRE* scores for all tested methods, when applied on *BMS1*, are shown in Fig. 13.

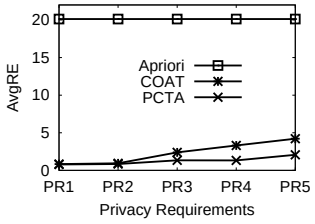

Figure 13: *AvgRE* vs. PR (*BMS1*)

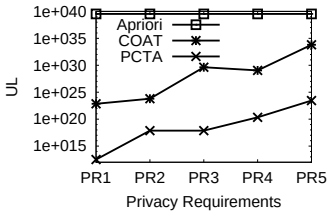

Figure 14: *UL* vs. PR (*BMS1*)

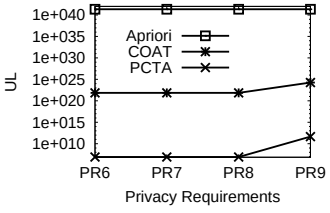

Figure 15: *UL* vs. PR (*BMS1*)

Since Apriori does not take into account the specified privacy constraints, its performance remains constant in this experiment and is the worst among the tested algorithms. PCTA outperformed both Apriori and COAT, achieving *up to* 26 *times lower AvgRE scores than those of Apriori and* 2.5 *times lower than those of COAT*. Furthermore, the difference in *AvgRE* scores between PCTA and COAT increases as policies become more stringent, which confirms the benefit of our clustering-based strategy. The ability of PCTA to preserve data utility better than Apriori and COAT was also confirmed when *UL* was used, as shown in Fig. 14.

Next, we constructed 4 sets of privacy constraints *PR6*,...,*PR9* that are comprised of 1000 itemsets and need to be protected with $k = 5$. A summary of these constraints appears in

Table 3. Figure 15 illustrates the *NCP* scores for BMS1 and Fig. 16 the *UL* scores for BMS2. As can be seen, PCTA consistently outperformed Apriori and COAT, being able to incur less information loss. This again demonstrates the ability of the clustering-based strategy employed in PCTA to preserve data utility.

Privacy Constraints	% of items	% of 2-itemsets	% of 3-itemsets	% of 4-itemsets
PR6	33%	33%	33%	1%
PR7	30%	30%	30%	10%
PR8	25%	25%	25%	25%
PR9	16.7%	16.7%	16.7%	50%

Table 3: Summary of sets of privacy constraints *PR6*, *PR7*, *PR8* and *PR9*

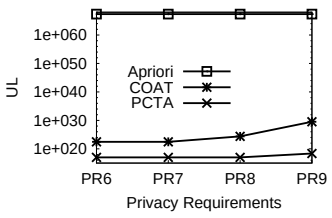

Figure 16: *UL* vs. PR (*BMS2*)

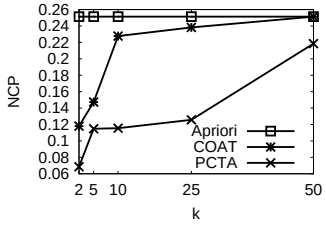

Figure 17: *NCP* vs. k (*BMS1*)

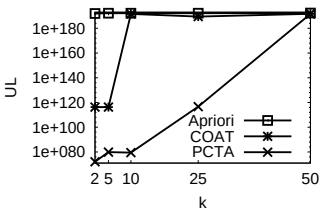

Figure 18: *UL* vs. k (*BMS2*)

5.2.3 Data utility for complete k-anonymity

We evaluated the effectiveness of the methods when privacy is enforced through complete k-anonymity, which requires protecting all items of a transaction. To achieve this, we configured Apriori by setting m to the size of the largest transaction of each dataset, and COAT and PCTA by generating itemsets using the *Pgen* algorithm introduced in [29]. As shown in Figs. 17 and 18, which present results for *NCP* and *UL* respectively, PCTA performs better than Apriori and COAT, while Apriori and COAT incur much information loss particularly when k is 10 or larger. In fact, in these cases, these algorithms created generalized items whose size was much larger than those constructed by PCTA. This again shows that PCTA is more effective in retaining information loss.

5.2.4 Efficiency

We used BMS1 to evaluate the runtime efficiency of PCTA, assuming that all 2-itemsets require protection. We first tested scalability in terms of dataset size, using increasingly larger subsets of *BMS1*. Since the size and items of a transaction can affect the runtime of the algorithms, we require the transactions of a subset to be contained in all larger subsets. From Fig. 19 we can see that PCTA is more efficient than Apriori, because it discards protected itemsets, whereas Apriori considers all m-itemsets, as well as their possible generalizations. This incurs more overhead, particularly for large datasets. However, PCTA is less efficient than COAT, as it explores a larger number of possible generalizations. Thus, PCTA performs a larger number of dataset scans to compute the support of privacy constraints and measure *UL* during generalization.

Finally, we examined how PCTA scales with respect to k and report the result in Fig. 20. Observe that Apriori becomes slightly more efficient as k increases, while the runtime of both PCTA and COAT follows an opposite trend. This is because Apriori generalizes entire subtrees of items, while both PCTA and COAT generalize one item (or generalized item) at a time. Nevertheless, PCTA was found to be *up to* 44% *more efficient than COAT*. This is attributed to the lazy updating strategy that it adopts. Since data needs to be scanned after each item generalization, the savings from using this strategy increase as k gets larger, as discussed in Section 4.1.

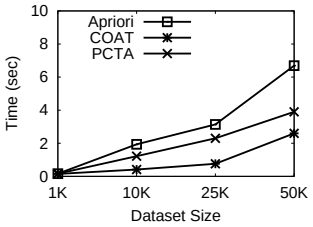

Figure 19: Runtime vs. $|D|$

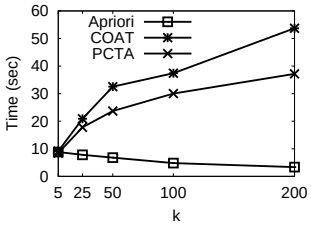

Figure 20: Runtime vs. k

5.3 Evaluation of UPCTA

We now test UPCTA against COAT with respect to both data utility and efficiency. Both algorithms were configured using the same utility constraints, $k = 5$, and $s = 15\%$. We do not report the results for Apriori and PCTA, because these algorithms do not guarantee data utility. We assume 5 sets of utility constraints: *UR1*, ..., *UR5*. Each of these sets contains groups of a certain number of semantically close items (i.e., sibling leaf-level nodes in the hierarchy). The mappings between each set of utility constraints and the size of these groups are shown in Table 4. Note that *UR1* is quite restrictive, since it requires a small number of items to be generalized together. Thus, both algorithms had to suppress a small percentage of items, which was the same for both algorithms and less than s, to satisfy *UR1*.

Utility Requirement	size of group of semantically close items
UR1	25
UR2	50
UR3	125
UR4	250
UR5	500

Table 4: Summary of sets of utility constraints *UR1*, *UR2*, ..., *UR5*

5.3.1 Data utility for k^m-anonymity

We first evaluated data utility when COAT and UPCTA were configured to achieve 5^2-anonymity. Figs. 21 and 22 show the result with respect to *AvgRE* for *BMS1* and *BMS2*, respectively. Note that the *AvgRE* scores of UPCTA were lower (better) than those of COAT by 61% for *BMS1* and by 33% for *BMS2* (on average). Furthermore, UPCTA outperformed

COAT with respect to the *UL* measure, as can be seen in Figs. 23 and 24. These results suggest that the clustering-based heuristic we propose enables UPCTA to produce anonymized data that satisfy the specified utility requirements, permit more accurate query answering, and retain more information than those generated by COAT.

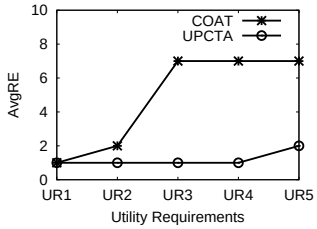

Figure 21: *AvgRE* vs. UR (*BMS1*)

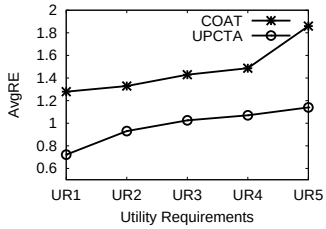

Figure 22: *AvgRE* vs. UR (*BMS2*)

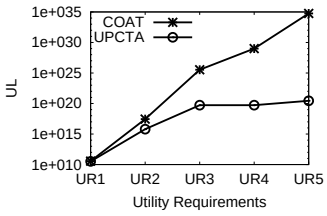

Figure 23: UL vs. UR (*BMS1*)

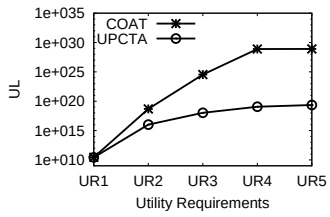

Figure 24: *UL* vs. UR (*BMS2*)

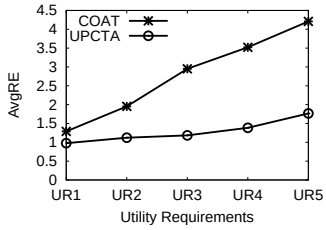

Figure 25: *AvgRE* vs. UR (*BMS1*)

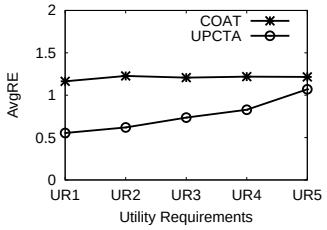

Figure 26: AvgRE vs. UR (*BMS2*)

5.3.2 Data utility for privacy constraints with various characteristics

Subsequently, we configured COAT and UPCTA using the set of privacy constraints *PR5*, which is the most stringent among those considered in Section 5.2.2. The *AvgRE* scores for *BMS1* and *BMS2* and sets of utility constraints *UR1* to *UR5* are reported in Figs. 25 and 26, respectively. These results together with those of Figs. 27 and 28, which show the *UL* scores for *BMS1* and *BMS2*, confirm the superiority of UPCTA in retaining data utility. It is also interesting to observe that the scores in both *AvgRE* and *UL* for UPCTA increase much less than those for *COAT*. This is because UPCTA examines a larger number of generalizations than COAT, which leads UPCTA to finding solutions with better data utility.

We also applied UPCTA and COAT using the set of privacy constraints *PR9*, which requires protecting 50% of items selected uniformly at random (see Section 5.2.2). The results with respect to *AvgRE* and *UL* are illustrated in Figs. 29 to 32, and they are qualitatively similar to those of Figs. 25 to 28.

5.3.3 Data utility for complete k-anonymity

We compared UPCTA and COAT in terms of data utility when they enforce complete 5-anonymity. As illustrated in Fig. 33, which shows the result for *AvgRE* and for BMS2, UPCTA outperformed COAT across all tested utility requirements, allowing up to 2.5 times more accurate aggregate query answering. In addition, we observed that the higher level

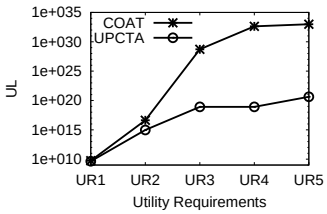

Figure 27: UL vs. UR (*BMS1*)

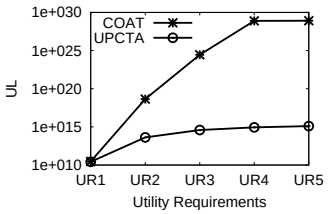

Figure 28: *UL* vs. UR (*BMS2*)

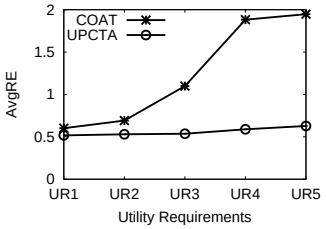

Figure 29: *AvgRE* vs. UR (*BMS1*)

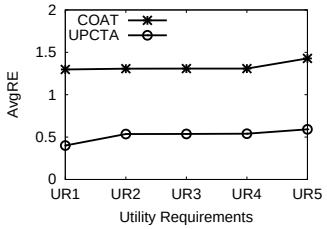

Figure 30: AvgRE vs. UR (*BMS2*)

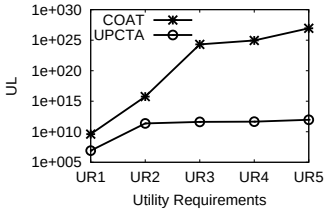

Figure 31: UL vs. UR (*BMS1*)

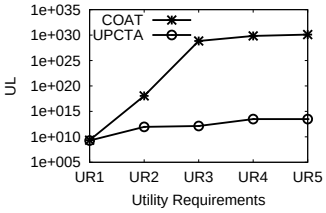

Figure 32: *UL* vs. UR (*BMS2*)

of privacy provided by complete k-anonymity forced both algorithms to incur more information loss compared to the case of *PR9*, which requires protecting only certain items. For example, the mean of the *AvgRE* scores for COAT was 4.4 times larger when this algorithm enforced complete 5-anonymity instead of *PR9*, while the same statistic for UPCTA was 4.8 times larger. The result with respect to *UL* for BMS2 are shown in Fig. 34 and confirm the above observations.

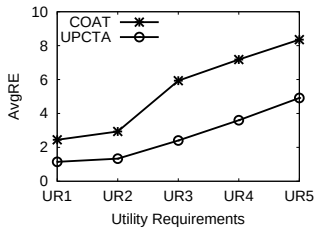

Figure 33: AvgRE vs. UR (*BMS2*)

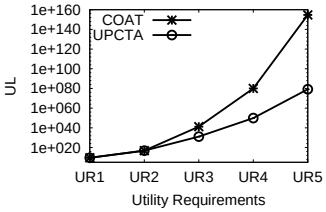

Figure 34: *UL* vs. UR (*BMS2*)

5.3.4 Data utility in electronic medical record publishing

Having examined the effectiveness of UPCTA using benchmark data, we proceed to investigating whether it can produce anonymized data that permit accurate analysis in a real-world scenario, which involves publishing the *VNEC* and $VNEC_{kc}$ datasets. Each transaction in these datasets corresponds to a different patient and contains one or more *ICD-9* codes [2]. An *ICD-9* code denotes a disease of a certain type (e.g., 493.01 corresponds to *Ex-*

[2]ICD-9 is the official system of assigning health insurance billing codes to diagnoses in the United States.

trinsic asthma with exacerbation), and, in most cases, more than one *ICD-9* codes correspond to the same disease (e.g, 492.00, 493.01, and 493.02 all correspond to *Asthma*).

The goal of publishing *VNEC* and *VNEC*$_{KC}$ is to enable studies related to the 18 diseases considered in [32] (i.e., data recipients need to be able to accurately determine the number of patients suffering a disease), while keeping information loss at a minimum to allow general biomedical analysis. At the same time, the association between transactions and patients' identities, which is possible as ICD-9 codes are contained in publicly available hospital discharge summaries [27], must be prevented. To examine whether COAT and UPCTA can achieve both of these goals, we configured them by specifying a set of 18 utility constraints, one for each disease. We also considered all ICD-9 codes a patient was diagnosed with during a single hospital visit as potentially identifying following [28], and used $s = 2\%$ and various k values in $\{2, 5, 10, 25\}$.

We first examined whether the anonymizations generated by COAT and PCTA satisfy the specified utility constraint set. In fact, we found that anonymizations constructed by both algorithms managed to accomplish this goal for all tested k values. Thus, we then compared the amount of information loss incurred by the tested methods by measuring *AvgRE* for a workload of 1000 COUNT queries asking for sets of *ICD-9* codes supported by at least 5% of the transactions in each dataset[3]. This type of queries is important in various biomedical data analysis applications [37].

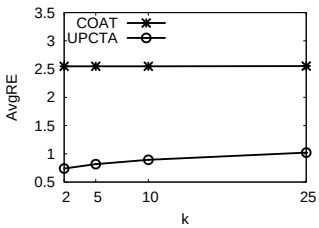

Figure 35: AvgRE vs. k (*VNEC*)

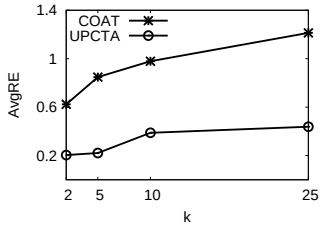

Figure 36: AvgRE vs. k (*VNEC*$_{kc}$)

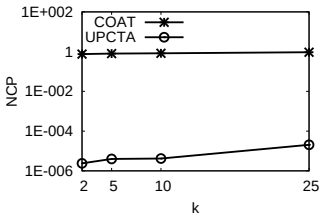

Figure 37: *NCP* vs. k (*VNEC*)

The results with respect to *AvgRE* for *VNEC* and *VNEC*$_{KC}$ for various k values are shown in Figs. 35 and 36, respectively. Observe that PCTA outperformed COAT for all tested k values, being able to generate anonymizations that permit at least 2.5 and up to 3.8 times more accurate querying answering. In addition, we investigated the amount of data utility retained by the tested algorithms using the *NCP* and *UL* measures. Figs. 37 and 38 show the results with respect to *NCP* for *VNEC* and *VNEC*$_{kc}$, respectively, while Fig. 39 illustrates the *UL* scores for *VNEC*$_{kc}$. These results together with those of Figs. 35 and 36 demonstrate that PCTA is significantly more effective than COAT at minimizing information loss.

5.3.5 Efficiency

In Fig. 40, we report the result for datasets constructed by selecting increasingly larger subsets of BMS1 when both algorithms ran using *UR2*. The transactions in each subset were selected uniformly at random but contained in all larger sets. COAT was more efficient than UPCTA for all tested utility requirements, requiring 65% less time on average. This is because it explores a larger number of possible generalizations than COAT, which increases runtime significantly, as discussed in Section 5.2.4.

[3]The sets that are retrieved by the queries in the workload were selected randomly from the frequent itemsets that were mined with minimum support threshold of 5%

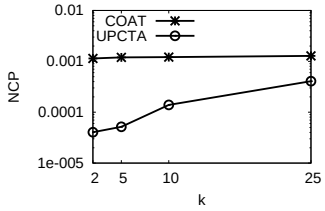

Figure 38: NCP vs. k ($VNEC_{kc}$)

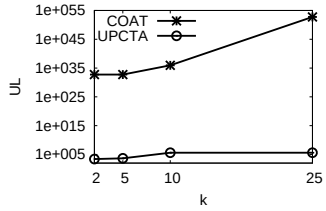

Figure 39: *UL* vs. k ($VNEC_{kc}$)

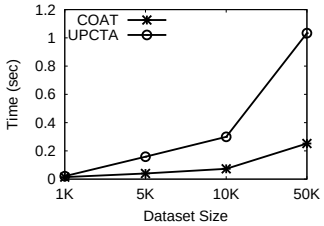

Figure 40: Runtime vs. $|\mathcal{D}|$ (*BMS1*)

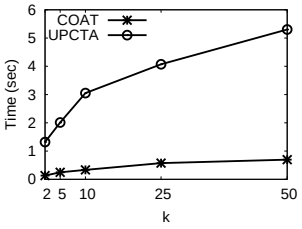

Figure 41: Runtime vs. k (*BMS1*)

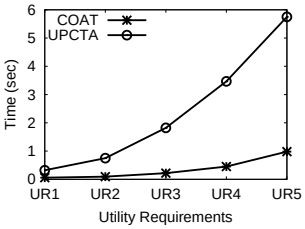

Figure 42: Runtime vs. UR (*BMS1*)

Then, we examined how UPCTA scales with respect to k. From Fig. 41, it can be seen that both PCTA and COAT scale sublinearly with k, with UPCTA being more scalable due to the lazy updating strategy it adopts. For instance, the runtime of COAT was 5.3 times higher for $k = 50$ than what it was for $k = 2$, while the runtime of PCTA 4 times higher. Also, we observed that COAT was more efficient than UPCTA for all k values. The reason is that the latter algorithm examines more generalizations than COAT.

Last, the impact of considering different utility requirements on runtime was investigated. As can be seen in Fig. 42, both algorithms needed more time as the specified utility requirements get coarser, because the number of generalized items an item can be mapped to increases. For example, the runtime of COAT and PCTA was 16.7 and 17.9 times higher when these algorithms were configured with *UR5* instead of *UR1*. PCTA was less efficient than COAT across all tested utility requirements, for the reasons explained above.

6 Conclusions

Existing algorithms are unable to anonymize transaction data under a range of different privacy requirements without incurring excessive information loss, because they are built upon the intrinsic properties of a single privacy model. To address this issue, we introduced a novel formulation of the problem of transaction data anonymization based on clustering. The generality of this formulation allows the design of algorithms that are independent of generalization strategies and privacy models, and are able to achieve high data utility and privacy. We also proposed two algorithms that are based on clustering and can produce a significantly better result than the state-of-the-art methods in terms of data utility. Specifically, PCTA employs generalization, while UPCTA uses a combination of generalization and suppression-based heuristics and is able to satisfy utility requirements that are common in real-world applications.

We believe that this work opens up several important research directions, which we aim

to pursue in the future. A first issue is how we can extend the proposed clustering-based framework to produce an anonymized dataset that prevents both identity and sensitive itemset disclosure. One way to achieve this is to assume a classification of items into public and sensitive and then guarantee that an attacker who knows public, and potentially some sensitive items as well, cannot associate an individual to their transaction and infer all sensitive items. Another interesting issue is how we can incorporate other types of utility requirements into the anonymization process, such as, for example, to mine certain association rules from the anonymized data.

Acknowledgements

We would like to thank Manolis Terrovitis, Nikos Mamoulis and Panos Kalnis for providing the implementation of the Apriori anonymization algorithm [47].

References

[1] National Institutes of Health. Policy for sharing of data obtained in NIH supported or conducted genome-wide association studies. NOT-OD-07-088. 2007.

[2] Health insurance portability and accountability act of 1996 united states public law.

[3] M. Barbaro and T. Zeller. A face is exposed for aol searcher no. 4417749. New York Times, Aug 2006.

[4] R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *21st ICDE*, pages 217–228, 2005.

[5] J. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymity using clustering technique. In *DASFAA*, pages 188–200, 2007.

[6] J. Cao, P. Karras, C. Raïssi, and K. Tan. rho-uncertainty: Inference-proof transaction anonymization. *PVLDB*, 3(1):1033–1044, 2010.

[7] C.C. Chang, B.Thompson, H. Wang, and D. Yao. Towards publishing recommendation data with predictive anonymization. In *5th ACM Symposium on Information, Computer and Communications Security*, pages 24–35, 2010.

[8] B. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala. Privacy-preserving data publishing. *Found. Trends databases*, 2(1–2):1–167, 2009.

[9] J. Domingo-Ferrer and V. Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *DMKD*, 11(2):195–212, 2005.

[10] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Comput. Surv.*, 42, 2010.

[11] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, pages 205–216, 2005.

[12] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, pages 715–724, 2008.

[13] A. Gkoulalas-Divanis and G. Loukides. Revisiting sequential pattern hiding to enhance utility. To appear in KDD, 2011.

[14] A. Gkoulalas-Divanis and V. S. Verykios. Exact knowledge hiding through database extension. *TKDE*, 21(5):699–713, 2009.

[15] A. Gkoulalas-Divanis and V.S. Verykios. A free terrain model for trajectory k-anonymity. In *DEXA*, pages 49–56, 2008.

[16] A. Gkoulalas-Divanis and V.S. Verykios. Hiding sensitive knowledge without side effects. *Knowledge and Information Systems*, 20(3):263–299, 2009.

[17] A. Gkoulalas-Divanis and V.S. Verykios. *Association rule hiding for data mining*. Advances in Database Systems series, vol. 41. Springer, 2010.

[18] A. Gkoulalas-Divanis, V.S. Verykios, and Mohamed F. Mokbel. Identifying unsafe routes for network-based trajectory privacy. In *SDM*, pages 942–953, 2009.

[19] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.

[20] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *KDD*, pages 279–288, 2002.

[21] S. Jha, L. Kruger, and P. McDaniel. Privacy preserving clustering. In *ESORICS*, pages 397–417, 2005.

[22] A.F. Karr, C.N. Kohnen, A. Oganian, J.P. Reiter, and A.P. Sanil. A framework for evaluating the utility of data altered to protect confidentiality. *The American Statistician*, 60:224 – 232, 2006.

[23] S. Kisilevich, L. Rokach, Y. Elovici, and B. Shapira. Efficient multidimensional suppression for k-anonymity. *TKDE*, 22:334–347, 2010.

[24] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.

[25] J. Li, R. Wong, A. Fu, and J. Pei. Achieving -anonymity by clustering in attribute hierarchical structures. In *DaWaK*, pages 405–416, 2006.

[26] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *2008 SIGMOD*, pages 93–106, 2008.

[27] G. Loukides, J.C. Denny, and B. Malin. The disclosure of diagnosis codes can breach research participants' privacy. *Journal of the American Medical Informatics Association*, 17:322–327, 2010.

[28] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. Anonymization of electronic medical records for validating genome-wide association studies. *Proceedings of the National Academy of Sciences*, 17:7898–7903, 2010.

[29] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowledge and Information Systems*, 2011. To Appear, DOI: 10.1007/s10115-010-0354-4.

[30] G. Loukides, A. Gkoulalas-Divanis, and J. Shao. Anonymizing transaction data to eliminate sensitive inferences. In *DEXA*, pages 400–415, 2010.

[31] G. Loukides and J. Shao. Capturing data usefulness and privacy protection in k-anonymisation. In *SAC*, pages 370–374, 2007.

[32] T.A. Manolio, L.D. Brooks, and F.S. Collins. A hapmap harvest of insights into the genetics of common disease. *Journal of Clinical Investigation*, 118:1590–1605, 2008.

[33] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE S&P*, pages 111–125, 2008.

[34] M. Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. *DKE*, 63(3):622–645, 2007.

[35] M.E. Nergiz, C. Clifton, and A.E. Nergiz. Multirelational k-anonymity. *TKDE*, 21(8):1104–1117, 2009.

[36] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *ICDM*, pages 613–616, 2003.

[37] C. Ordonez. Association rule discovery with the train and test approach for heart disease prediction. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):334–343, 2006.

[38] R. G. Pensa, A. Monreale, F. Pinelli, and D. Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *1st International Workshop on Privacy in Location-Based Applications*, 2008.

[39] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, pages 682–693, 2002.

[40] D. Roden, J. Pulley, M. Basford, G.R. Bernard, E.W. Clayton, J.R. Balser, and D.R. Masys. Development of a large scale de-identified dna biobank to enable personalized medicine. *Clinical Pharmacology and Therapeutics*, 84(3):362–369, 2008.

[41] P. Samarati. Protecting respondents identities in microdata release. *TKDE*, 13(9):1010–1027, 2001.

[42] Y. Saygin, V.S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[43] P. Sharkey, Hongwei H. Tian, W. Zhang, and S. Xu. Privacy-preserving data mining through knowledge model sharing. In *Privacy, security, and trust in KDD*, pages 97–115, 2008.

[44] X. Sun and P. S. Yu. A border-based approach for hiding sensitive frequent itemsets. *5th IEEE International Conference on Data Mining*, page 8 pp., 2005.

[45] L. Sweeney. k-anonymity: a model for protecting privacy. *IJUFKS*, 10:557–570, 2002.

[46] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB J.* To appear.

[47] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *PVLDB*, 1(1):115–125, 2008.

[48] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W-C. Fu. Utility-based anonymization using local recoding. In *KDD*, pages 785–790, 2006.

[49] R. Xu and D.C. Wunsch. *Clustering*. Wiley-IEEE Press, 2008.

[50] Y. Xu, K. Wang, A. W-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, pages 767–775, 2008.

[51] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *KDD*, pages 401–406, 2001.