Published in final edited form as:

Expert Syst Appl. 2012 August ; 39(10): 9764–9777. doi:10.1016/j.eswa.2012.02.179.

Utility-preserving transaction data anonymization with low information loss

Grigorios Loukides^{a,1} and Aris Gkoulalas-Divanis^b

Grigorios Loukides: g.loukides@cs.cf.ac.uk; Aris Gkoulalas-Divanis: agd@zurich.ibm.com

^aSchool of Computer Science & Informatics, Cardiff University, UK ^bIBM Research – Zurich, Zurich, Switzerland

Abstract

Transaction data record various information about individuals, including their purchases and diagnoses, and are increasingly published to support large-scale and low-cost studies in domains such as marketing and medicine. However, the dissemination of transaction data may lead to privacy breaches, as it allows an attacker to link an individual's record to their identity. Approaches that anonymize data by eliminating certain values in an individual's record or by replacing them with more general values have been proposed recently, but they often produce data of limited usefulness. This is because these approaches adopt value transformation strategies that do not guarantee data utility in intended applications and objective measures that may lead to excessive data distortion. In this paper, we propose a novel approach for anonymizing data in a way that satisfies data publishers' utility requirements and incurs low information loss. To achieve this, we introduce an accurate information loss measure and an effective anonymization algorithm that explores a large part of the problem space. An extensive experimental study, using click-stream and medical data, demonstrates that our approach permits many times more accurate query answering than the state-of-the-art methods, while it is comparable to them in terms of efficiency.

Keywords

anonymization; transaction data; data utility; information loss

1. Introduction

Publishing data about individuals is increasingly practised nowadays. For example, the National Institutes of Health (NIH) in the US and the Medical Research Council (MRC) in the UK emphasize the need for data collected from their funded projects to be shared for research purposes [1, 2]. Published data are, in fact, crucial for performing large-scale and low-cost analytic tasks, ranging from query answering [3, 4] to data mining [5], in domains as diverse as marketing and medicine. Alarming, however, privacy breaches are being constantly reported and have serious legal, financial, and emotional consequences to organizations and individuals. For instance, individuals' privacy has been compromised as a

¹Part of this work was done when the author was in Vanderbilt University.

© 2012 Elsevier Ltd. All rights reserved.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

result of sharing search query terms, recommendation rates, and health data about them [6–8], while a privacy breach costs an organization \$6.75M and \$3.44M on average in the US and the UK, respectively [9].

In response, several methods for publishing data while preventing the disclosure of individuals' identities (*identity disclosure*) and/or sensitive information (*sensitive information disclosure*) have been developed [5, 10]. These methods aim at producing data that can be shared according to data sharing policies and regulations [1, 11, 12], and they can be broadly categorized into perturbative and non-perturbative [13, 14]. The former methods include noise addition, data swapping, and rounding, and attempt to produce data that preserve aggregate statistics [13]. However, these data cannot be analyzed at a record level, since, in these data, individuals may no longer be associated with true information about them. On the other hand, non-perturbative methods focus on publishing data that can still be analyzed individually and include *generalization*, which replaces a value with a more general one (e.g., HIV to sexually transmitted disease), and *suppression*, which removes a value from the published data [8, 15].

1.1. Motivation

Using non-perturbative methods to thwart identity disclosure has been studied in the context of relational [8, 15–20], graph [21], trajectory [22], and transaction [23–27] data. Transaction data, in particular, are increasingly published by organizations and businesses to support a wide spectrum of applications, including e-commerce [28] and biomedicine [4]. Transaction datasets are comprised of records, called *transactions*, which consist of sets of *items* (also referred to as *itemsets*), such as the products purchased by customers from a supermarket, or the diagnoses contained in patients' electronic medical records. Publishing transaction data can lead to identity disclosure, as shown in the following example.

Example 1.1. Consider that a hospital publishes the dataset shown in Fig. 1(a), after removing patient names. Each transaction corresponds to a different patient and contains the diagnoses assigned to them. Observe that, knowing that Anne is diagnosed with *b*, *e*, and *f*, an attacker can associate Anne with her transaction and infer all of her diagnoses, since no other transaction contains these three items together.

Recent research has proposed several principles to prevent identity disclosure in transaction data publishing, such as complete *k*-anonymity [25], *k^m*-anonymity [23], and privacy-constrained anonymity [4]. All these principles limit the probability of identity disclosure to

$\frac{1}{k}$, where *k* is a parameter set by data publishers, and are enforced using algorithms that employ generalization and/or suppression. The primary difference between these principles is that they compute the latter probability based on different assumptions about adversarial knowledge. For instance, the attack considered in Example 1.1 can be prevented by applying the Apriori algorithm [4] to the data of Fig. 1(a) using *k* = 3 and the hierarchy shown in Fig. 1(g). As can be seen in Fig. 1(b), each item *a* to *g* is now replaced by a *generalized item* (*a*, *b*, *c*, *d*, *e*, *f*, *g*), which lies in the root of the hierarchy and is interpreted as any combination of these items. Thus, the probability of associating Anne with any combination of *b*, *e*, and *f*, using the data of Fig. 1(b), is no more than $\frac{1}{3}$.

Unfortunately, existing algorithms [4, 23–27] may produce data of subpar utility, because they have at least one of the following shortcomings:

- They consider a small number of possible generalizations and suppressions [4, 23–27]. For example, Apriori [23] cannot replace *f* and *g* by (*f*, *g*), as only generalized

items that correspond to a node in the hierarchy, shown in Fig. 1(g), can be constructed.

- They do not take into account data publishers' utility requirements regarding the items that can be generalized together [23–25, 27]. For instance, the result of Apriori, which is illustrated in Fig. 1(b), cannot support a medical study in which the number of patients diagnosed with f or g needs to be accurately determined. This is because every record in Fig. 1(b) may be associated with any combination of the items a to g .
- They use objective measures that fail to accurately capture data utility [4, 23–27]. The measure used in Apriori, for example, does not quantify the level of information loss incurred to produce the anonymized datasets shown in Figs. 1(c) and 1(d), as we will show later.

1.2. Our contributions

In this work, we propose a new anonymization method that can produce practically useful transaction data with minimal information loss. Our work makes the following contributions.

First, we introduce *Utility Criterion (UC)*, a measure that can quantify data utility under different generalization models and be employed by effective anonymization algorithms [26, 27]. *UC* captures information loss more accurately than existing metrics, as it considers the length, support, and anticipated utility of the items that are being generalized.

Second, we develop a novel anonymization algorithm, called Update-Anonymize-Reorder (UAR). UAR applies generalization and suppression to carefully selected items, in accordance with data publishers' utility requirements, and is guided by the *UC* measure. Thus, our algorithm is able to generate data that remain useful in intended applications, while incurring low information loss. For example, UAR produced the anonymized data shown in Fig. 1(d) from the original data in Fig. 1(a), when it was configured using $k = 3$, and the privacy and utility requirements of Figs. 1(e) and 1(f), respectively. Note, the dataset produced by UAR can support the aforementioned medical study, since every record that contains (f, g) in this dataset harbors either f or g in the data of Fig. 1(a). Also, UAR incurred less information loss than Apriori. For instance, the generalized items (a, b, c, d, e) and (f, g) in Fig. 1(d) are easier to interpret than (a, b, c, d, e, f, g) in Fig. 1(b).

Third, we experimentally evaluate our approach using two datasets, containing click-stream data, and a dataset containing electronic medical records derived from the Vanderbilt University Medical Center, a large healthcare provider in the US. Our results show that UAR is very effective at preserving data utility, as it permits many times more accurate query answering than the state-of-the-art methods [23, 26, 27], while maintaining good scalability.

1.3. Paper organization

The rest of this paper is organized as follows. Sections 2 and 3 discuss related work and provide the necessary background, respectively. In Section 4, we introduce the UAR algorithm, and, in Section 5, we evaluate it against the state-of-the-art methods. Finally, Section 6 concludes the paper.

2. Related work

In this section, we review privacy-preserving approaches for transaction data publishing, with an emphasis on those that guard against identity disclosure using non-perturbative

techniques. Producing data that prevent different attacks, such as the mining of sensitive knowledge patterns (e.g., frequent itemsets [29–31] or sequences [32, 33]), or the inference of individuals' sensitive information [24, 34, 35] is also possible but outside the focus of the paper. We also consider the non-interactive setting, in which the entire dataset is published once. This differs from the interactive setting, in which users receive perturbed query responses [36–38]. In the remainder of the section, we discuss privacy principles and anonymization algorithms that are most relevant to our work.

2.1. Privacy principles

A well-established and widely used anonymization principle to prevent identity disclosure is *k*-anonymity [8, 15]. Simply put, *k*-anonymity requires each record of the published dataset to be indistinguishable from at least $k - 1$ other records in the dataset with respect to a set of *quasi-identifiers*. Quasi-identifiers are attributes based on which the published dataset can be linked to external sources, and k is a parameter specified by data publishers, according to their expectations about attackers' background knowledge. *k*-anonymity was originally proposed for relational data [8, 15], but has been recently adapted to transaction data.

He et al. [25] proposed *complete k-anonymity*, a *k*-anonymity-based principle for transaction data. The latter principle assumes that any combination of items in a transaction can lead to identity disclosure and requires each transaction in the published dataset to be indistinguishable from at least $k - 1$ other transactions in the dataset, based on any of these combinations. Thus, satisfying complete *k*-anonymity guarantees that an attacker cannot link an individual to less than k transactions of the published dataset or, equivalently, that the

probability of associating an individual with their transaction is no more than $\frac{1}{k}$.

Terrovitis et al. [23] argued that it may be difficult for an attacker to acquire knowledge about all items of a transaction, because transaction data are typically high-dimensional and sparse. Based on this observation, the authors of [23] proposed the *k^m-anonymity* principle, which thwarts attackers who know up to m items in an individual's transaction. Specifically, *k^m-anonymity* ensures that any combination of these m items cannot be used to associate the individual with less than k transactions of the released dataset.

Loukides et al. [26] observed that, in several applications (e.g. in biomedical and mobility data analysis [4, 39, 40]), data publishers are able to specify *privacy constraints*, i.e., sets of items that lead to identity disclosure. For instance, not all diagnoses given to a patient can be used to link up with an external data source to disclose their identity [4]. In these cases, applying complete *k*-anonymity or *k^m-anonymity* would overprotect data, incurring information loss unnecessarily. To avoid this, the authors of [26] proposed the principle of *privacy-constrained anonymity*, which is enforced when privacy constraints are *satisfied*. That is, when any combination of items in a privacy constraint appears in at least k or none of the transactions of the published dataset (details will be provided later on). Privacy-constrained anonymity is more general than complete *k*-anonymity and *k^m-anonymity*, which are the special cases of it [26], and therefore we adopt it in this work.

We also note that there are privacy principles [24, 34] for preventing sensitive information disclosure. Xu et al. [24] introduced (h, k, p) -coherence, which treats items that can lead to identity disclosure (*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 *sensitive items* using a parameter h . Cao et al. [34] introduced ρ -uncertainty to prevent sensitive information disclosure. Their work guards against attackers who can use any combination of items to infer an individual's sensitive information, but does not prevent identity disclosure. In another line of research, Loukides et al. [34] proposed limiting the probability of identity

disclosure, based on specified sets of public items, as well as the probability of inferring specified sets of sensitive items. Privacy requirements are expressed based on implications, called PS-rules, each between a set of public items and a set of sensitive items. Extensions to our approach to guard against sensitive information disclosure are possible (e.g., by following the methodology discussed in [41]). However, in this work, we focus on eliminating identity disclosure, which is essential to publishing data in compliance with related policies and regulations [1, 11], and leave such extensions as future work.

2.2. Anonymization algorithms

Several anonymization algorithms for transaction data have been recently proposed [23, 25–27, 41]. These algorithms can be classified based on the privacy principle they adopt, as illustrated in Table 1. Since complete k -anonymity is the special case of k^m -anonymity, which, in turn, is the special case of privacy-constrained anonymity [26], Apriori can be configured to satisfy complete k -anonymity, while COAT and PCTA can enforce both the latter principle and k^m -anonymity. In the following, we present the algorithms reported in Table 1 in more detail, reviewing the search and data transformation strategies they employ.

He et al. [25] proposed a top-down algorithm, called *Partition*, that uses a local generalization model (i.e., different occurrences of the same item can be replaced by different generalized items). Partition starts by generalizing all items to the most generalized item lying in the root of a generalization hierarchy, which is specified by data publishers and describes all the ways items (leaf-level nodes in the hierarchy) can be replaced by generalized items (non-leaf nodes). Then, it replaces the most generalized item with its immediate descendants in the hierarchy, if complete k -anonymity is satisfied. For example, given the hierarchy shown in Fig. 1(g), Partition starts with the most generalized item (a, b, c, d, e, f, g), which is interpreted as any non-empty subset of $\{a, b, c, d, e, f, g\}$. The latter generalized item is then replaced by (a, b, c), (d, e, f), and g , if the anonymized dataset that contains only these items satisfies complete k -anonymity. In subsequent iterations, Partition replaces generalized items 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.

Terrovitis et al. proposed an algorithm, called Apriori, to enforce k^m -anonymity [23]. This algorithm works in a bottom-up fashion and uses the full-subtree, global generalization model [17], i.e., it replaces entire subtrees of items in the generalization hierarchy with the generalized item that corresponds to one of their ascendants in the hierarchy. Apriori operates by protecting increasingly larger combinations of items iteratively; from single items to combinations of m items. In each step, it examines all possible generalizations that are consistent with the full-subtree model, and it finds one that incurs the least information loss and satisfies k^m -anonymity. For example, given the hierarchy of Fig. 1(g), the release of original items is considered first, then the generalization of $\{a, b, c\}$ to (a, b, c) , and finally the generalization of all items to (a, b, c, d, e, f, g) . In [41], Terrovitis et al. introduced two more algorithms to produce k^m -anonymous data. One of these algorithms, called Vertical Partitioning Anonymization (VPA), first partitions the domain of items into sets and then generalizes items in each set, using global generalization. Next, VPA merges the generalized items to ensure that the entire dataset satisfies k^m -anonymity. The other algorithm, called Local Recoding Anonymization (LRA), partitions a dataset horizontally into sets that can be anonymized with low information loss, and then generalizes items in each set separately.

However, the aforementioned algorithms may produce data with high information loss for two reasons. First, they cannot be readily extended to support privacy-constrained anonymity, hence they overprotect data when certain sets of items in a transaction lead to identity disclosure. Second, they explore a small number of possible generalizations due to

the hierarchy-based generalization model they employ [26]. Furthermore, Partition and LRA use local generalization, which makes the datasets they produce difficult to be used in practice. This is because mining algorithms and analysis tools cannot work effectively on them [10].

An algorithm that overcomes these limitations is Privacy-constrained Clustering-based Transaction Anonymization (PCTA) [27]. PCTA aims to satisfy the privacy-constrained anonymity principle using a set-based, global generalization model. The latter model allows any group of items in the original dataset to form a generalized item, it does not assume the existence of generalization hierarchies, and it has been shown to help data utility [27]. PCTA adopts a bottom-up approach that iteratively merges clusters formed by the items of the original dataset. Each original item initially forms a singleton cluster and, subsequently, singleton clusters are merged, in a way that is reminiscent of hierarchical agglomerative clustering algorithms, to satisfy privacy constraints with low information loss. In contrast to our UAR algorithm, PCTA does not guarantee that the produced anonymized data will be useful in intended analysis and incurs more information loss, as shown in our experiments.

The closest algorithm to UAR is COAT [26]. COAT operates in a bottom-up, greedy fashion and performs global generalization and suppression. The selection of items that are generalized by COAT is governed by *utility constraints*, which limit the possible generalizations to those that are acceptable for intended applications (details will be provided later). Given a set of privacy constraints, COAT first sorts these constraints in descending order according to the number of transactions in the original dataset they appear in. This *fixed* order is adopted for efficiency reasons, as privacy constraints that appear in “many” transactions are expected to require few generalizations, and thus a small amount of time, to be satisfied [26]. However, the amount of information loss required to satisfy privacy constraints depends on generalization and suppression decisions that are taken *during* anonymization. This implies that the order in which COAT processes privacy constraints may not help data utility. After sorting the privacy constraints, COAT considers a single constraint and generalizes the item of the constraint that appears in the fewest transactions in the anonymized dataset, until either the selected privacy constraint is satisfied, or until further generalization would violate the utility constraints. This heuristic has been shown to help COAT reduce information loss [26]. For example, COAT starts by selecting the privacy constraint $\{g\}$ in Fig. 1(e) and generalizes g to (f, g) , because this does not violate the utility constraint $\{f, g\}$ in Fig. 1(f). If it would, COAT would suppress items in the selected privacy constraint to satisfy it. UAR is able to offer the same level of privacy protection as COAT, while preserving data utility significantly better, as our experiments verify. This is because our algorithm employs the more accurate *UC* measure and effective heuristics to select items that can be generalized with low information loss.

3. Background and problem statement

In this section, we discuss generalization models and utility measures for anonymizing transaction data and then formulate the problem that our approach attempts to solve.

3.1. Preliminaries

Let $\mathcal{Q} = \{i_1, \dots, i_M\}$ be a finite set of literals, called *items*. Any subset $I \subseteq \mathcal{Q}$ is called an *itemset* over $\mathcal{Q}$, 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 $\mathcal{D} = \{T_1, \dots, T_N\}$ is a set of N transactions. Each *transaction* T_n , $n = 1, \dots, N$, 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 $T_n = \langle tid, I \rangle$ *supports* an itemset I , if $I \subseteq J$. Given

an itemset I in $\mathcal{D}$ we use $\text{sup}(I, \mathcal{D})$ to represent the number of transactions $T_n \in \mathcal{D}$ that support I .

3.2. Generalization models

Generalization transforms an original dataset $\mathcal{D}$ to an anonymized dataset $\tilde{\mathcal{D}}$ by mapping original items in $\mathcal{D}$ to generalized items [23, 26]. In the set-based anonymization model [26], a generalized item $\tilde{I}$ is a non-empty unique subset of $\mathcal{Q}$, whereas a suppressed item is the empty subset. When the members of $\tilde{I}$ are known, we may also express $\tilde{I}$ by listing its item(s) in brackets. In Fig. 2, for example, item a is mapped to the generalized item (a, b, c, d, e) , which is a subset of $\mathcal{Q} = \{a, b, c, d, e, f, g\}$.

The set-based anonymization model requires each item of $\mathcal{Q}$ to be mapped to either a generalized item that contains i , or to the suppressed item. Also, all instances of an item in $\mathcal{Q}$ need to be mapped to the same generalized item, hence a global generalization model is adopted. Applying set-based anonymization, creates a set of generalized and suppressed items, which collectively form the domain $\tilde{\mathcal{Q}}$. Note in Fig. 2, for example, that each item is mapped to exactly one generalized item and that $\tilde{\mathcal{Q}}$ is comprised of (a, b, c, d, e) and (f, g) . Furthermore, contrary to the full-subtree generalization model, set-based anonymization does not require the specification of a generalization hierarchy nor restricts the generalized items of $\tilde{\mathcal{Q}}$ to be non-leaf level nodes in such a hierarchy. In contrast, each of the $2^{|\mathcal{Q}|} - 1$ possible non-empty sets of items in $\mathcal{Q}$ may be mapped to a different generalized item. Thus, the set-based anonymization has been shown to help data utility [4, 26], and therefore we use it in this work.

3.3. Data Utility Measures

A transaction dataset can be anonymized in many different ways, but the one that harms data utility the least, is typically preferred. Existing measures to capture data utility in transaction data anonymization can be broadly classified into *task-based* and *information-loss based* measures, according to the way they work.

The first category of measures assume that anonymized data are intended for a specific task and measure how accurately they can support this task compared to original data. Terrovitis et al. [41] considered the task of mining frequent itemsets at multiple levels of a generalization hierarchy, while LeFevre et al. [16] the answering of a workload of aggregate queries. A popular measure for the latter task is Average Relative Error (*ARE*), which reflects the average number of transactions that are retrieved incorrectly when the query workload is applied to an anonymized dataset [26]. Consider, for example, the COUNT query illustrated in Fig. 3(a). Assuming that $I = f$ and $\mathcal{D}$ is the dataset of Fig. 1(a), we can derive an answer of 3 for this query. However, this query cannot be answered accurately using the anonymized dataset $\tilde{\mathcal{D}}$ of Fig. 1(c), and an estimated answer needs to be derived. Based on the method of [4], for example, the estimated answer for this query is 2.67, and the

Relative Error is $\frac{|3 - 2.67|}{3} = 0.11$. Given a number of such queries, *ARE* is computed as the mean of their Relative Error scores.

On the other hand, information-loss based measures quantify the amount of information loss due to anonymization without considering any particular task [23, 26, 27]. Thus, these measures are more generic and typically used as objective criteria in anonymization algorithms [23, 26, 27, 34]. Two popular information-loss based measures are Normalized Certainty Penalty (*NCP*), which is used by Apriori [23], and Utility Loss (*UL*), which guides COAT [26] and PCTA [27]. However, these measures may lead to the production of anonymized data of subpar utility, as we show in Section 4.1.

Definition 3.1 (Normalized Certainty Penalty NCP)—Given a generalization hierarchy $\mathcal{H}$, the Normalized Certainty Penalty (NCP) for an item i in $\mathcal{Q}$ is defined as

$$NCP(i) = \begin{cases} 0, & \text{subtr}(\tilde{i})=1 \\ \frac{\text{subtr}(\tilde{i})}{|\mathcal{J}|}, & \text{otherwise} \end{cases}$$

where $\tilde{i}$ denotes the generalized item, which is represented as a non-leaf level node in $\mathcal{H}$ and to which i is mapped, and $\text{subtr}: \tilde{\mathcal{Q}} \rightarrow [1, |\mathcal{Q}|]$ is a function that counts the number of leaf-level descendants of $\tilde{i}$ in $\mathcal{H}$. Based on this definition, the NCP for a dataset $\mathcal{D}$ is defined as

$$NCP(\mathcal{D}) = \frac{\sum_{i \in \mathcal{I}} (\text{sup}(i, \mathcal{D}) \times NCP(i))}{\sum_{i \in \mathcal{I}} (\text{sup}(i, \mathcal{D}))}$$

NCP penalizes items based on the way they are generalized. When computed for an anonymized dataset, NCP assigns large penalties to items with high support in the original dataset, which are mapped to generalized items that replace many other items. Consider, for example, that each of the items a to g in the dataset of Fig. 1(a) is replaced by the generalized item (a, b, c, d, e, f, g) , which is represented by the root in the hierarchy of Fig.

1(g). In this case, each item has an NCP of $\frac{7}{7}=1$, because (a, b, c, d, e, f, g) has 7 leaf-level descendants, and the dataset that results from this generalization and is shown in Fig. 1(b) an

$$NCP \text{ of } \frac{(4+3+2+1+3+3+2) \times \frac{7}{7}}{4+3+2+1+3+3+2} = 1.$$

The following definition explains the Utility Loss (UL) measure [26].

Definition 3.2 (Utility loss)—The Utility Loss (UL) for a generalized item $\tilde{i}$ is defined as

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

where $|\tilde{i}|$ denotes the number of items in $\mathcal{Q}$ that are mapped to $\tilde{i}$, and $w: \tilde{\mathcal{Q}} \rightarrow [0, 1]$ is a function assigning a weight according to the perceived importance of $\tilde{i}$ in analysis. Based on this definition, the Utility Loss (UL) for an anonymized dataset $\tilde{\mathcal{D}}$ is defined as

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

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

UL quantifies information loss incurred by both generalization and suppression. Generalized items are penalized based on their size, weight and support in the anonymized dataset, while both the number and the perceived importance of items that have been suppressed are taken into account. The size of generalized items is taken into account because, in the set-based anonymization model, $\tilde{i}$ can represent any of the $(2^{|\mathcal{I}|} - 1)$ non-empty subsets of the items mapped to it. Moreover, the support of $\tilde{i}$ is taken into consideration, as highly supported

items will affect more transactions, resulting in higher distortion. Note that the computation of UL does not require a generalization hierarchy. Instead, it assumes the existence of a weight w , which is specified by data publishers and used to penalize generalizations exercised on more “important” items. For example, to compute the UL score for $\tilde{r} = (f, g)$ in

Fig. 1(d) assuming $w(\tilde{r}) = 1$, we have $UL(\tilde{r}) = \frac{2^2 - 1}{2^7 - 1} \times 1 \times \frac{4}{6} \approx 0.016$.

3.4. Problem statement

Before presenting the problem we consider in this paper, we provide the Definitions 3.3 and 3.4, which illustrate the notions of privacy and utility constraint satisfiability that were introduced in [26]. Recall that privacy constraints specify the sets of items in $\mathcal{Q}$ that require protection from identity disclosure, while utility constraints are used to limit the generalized items that an item can be mapped to in order to produce practically useful anonymizations. Privacy constraints and utility constraints are specified by data publishers.

Definition 3.3 (Privacy constraint set and its satisfiability)—A privacy constraint set $\mathcal{P}$ is a non-empty set of privacy constraints, which is satisfied in $\tilde{\mathcal{D}}$ when (1) $\text{sup}(p, \tilde{\mathcal{D}}) \geq k$, or (2) $\text{sup}(p, \tilde{\mathcal{D}}) = 0$ and each proper subset of p is either supported by at least k transactions in $\tilde{\mathcal{D}}$ or not supported in $\tilde{\mathcal{D}}$, for each $p \in \mathcal{P}$ and a given parameter k .

Definition 3.4 (Utility constraint set and its satisfiability)—A utility constraint set $\mathcal{U}$ is a partition of $\mathcal{Q}$, which is comprised of utility constraints, and specifies the set of allowable mappings of the items from $\mathcal{Q}$ to those of $\tilde{\mathcal{Q}}$. $\mathcal{U}$ is satisfied if and only if (1) for each non-empty $i_m \in \tilde{\mathcal{I}}$, $\exists u \in \mathcal{U}$ such that all items from $\mathcal{Q}$ in $\mathcal{D}$ that are mapped to i_m are also contained in u , and (2) the fraction of items in $\mathcal{Q}$ contained in the set of suppressed items $\mathcal{S}$ is at most $s\%$.

As an example, observe that the dataset in Fig. 1(d) satisfies the privacy and utility constraint sets shown in Figs. 1(e) and 1(f), respectively, for $k = 3$ and $s = 5\%$. This is because, in this dataset, each of the privacy constraints is supported by at least 3 transactions, each of the items a to g is generalized together with items from the utility constraint it is contained in, and no item is suppressed. Based on these definitions, our problem can be formulated, as follows.

Problem Given a transactional dataset $\mathcal{D}$ a privacy constraint set $\mathcal{P}$ a utility constraint set $\mathcal{U}$ and parameters k, s , construct an anonymized version $\tilde{\mathcal{D}}$ of $\mathcal{D}$ using the set-based anonymization model such that: (1) $\mathcal{P}$ and $\mathcal{U}$ are both satisfied and (2) the amount of utility loss $UL(\tilde{\mathcal{D}})$ is minimal.

A solution to this problem ensures that anonymized data satisfies both of the specified privacy and utility constraint sets, while incurring the smallest possible amount of information loss. However, tackling this problem is challenging, because it is NP-hard and its feasibility depends on the specification of privacy and utility constraints, as proven in [26].

4. Update-Anonymize-Reorder: A novel anonymization algorithm

To deal with the aforementioned problem, we introduce the Update-Anonymize-Reorder (UAR) algorithm. Before discussing UAR, we present the criterion it employs to quantify data utility.

4.1. Utility Criterion

Anonymization typically incurs information loss that we seek to minimize. Thus, designing a measure that can deal with generalized items constructed by the flexible set-based anonymization model and be employed by effective algorithms [26, 27] is particularly important. However, the *NCP* and *UL* measures do not satisfy at least one of these properties.

In fact, *NCP* may misestimate the amount of information-loss incurred by set-based anonymization. This is because it does not take into account the size of generalized items, which determines the amount of information loss [26], as shown in the following example.

Example 4.1. Consider the datasets in Figs. 4(a) and 4(b), which illustrate an original dataset and an anonymized version of it (constructed using the set-based anonymization model), respectively. Given the generalization hierarchy of Fig. 4(c), we have that $NCP(a) + NCP(e) = NCP(d) + NCP(e) + NCP(f) = 2$. Thus, (a, e) and (d, e, f) are considered to incur the same amount of information loss. This is counterintuitive, as a data recipient can interpret (a, e) more easily than (d, e, f) , because there are 3 possible interpretations for (a, e) , but 7 for (d, e, f) . On the other hand, *UL* identifies (a, e) as incurring lower information loss than (d, e, f) , because it considers the size of a generalized item when penalizing it.

Furthermore, neither *NCP* nor *UL* are suitable for use in algorithms that employ effective item grouping strategies based on greedy search [26] or clustering [27]. This is because these measures do not consider the impact of different generalization decisions taken during anonymization. As a result, an algorithm guided by *NCP* or *UL* may overgeneralize certain items, harming data utility significantly, as illustrated below.

Example 4.2. Consider the anonymized dataset in Fig. 4(b) and that the PCTA algorithm attempts to merge either (a, e) with (b, c, d) to create (a, b, c, d, e) , or f with (g, h, i, j) to generate (f, g, h, i, j) . Since (a, b, c, d, e) and (f, g, h, i, j) are deemed equally useful by *NCP* and *UL* (we henceforth assume that the weights w are computed based on the dissimilarity measure of [42] and the generalization hierarchy of Fig. 4(c)), (f, g, h, i, j) may be created. Intuitively, this incurs more information loss than the generation of (a, b, c, d, e) does, because the original item f is generalized, which means that all queries involving this item cannot be answered accurately anymore, while (g, h, i, j) is overgeneralized.

We now introduce our Utility Criterion (*UC*) measure and explain why it can guide anonymization algorithms towards finding high-quality solutions. The following definition explains *UC*.

Definition 4.1 (Utility Criterion)—Given two generalized items $\tilde{t}_l$ and $\tilde{t}_m$, and the generalized item $\tilde{t}$, constructed from the set of items that is mapped to $\tilde{t}_l$ or $\tilde{t}_m$, the Utility Criterion (*UC*) is defined as

$$UC(\tilde{t}_l, \tilde{t}_m, \tilde{t}) = \left[|\tilde{t}| \times \left(\frac{|\tilde{t}_l|}{|\tilde{t}|} + \frac{|\tilde{t}_m|}{|\tilde{t}|} \right) \right] \times \left[w(\tilde{t}) \times \left(\frac{1}{w(\tilde{t}_l)} + \frac{1}{w(\tilde{t}_m)} \right) \right] \times \left[\sup(\tilde{t}, \tilde{\mathcal{D}}) \times \left(\frac{\sup(\tilde{t}_l, \tilde{\mathcal{D}}')}{\sup(\tilde{t}_m, \tilde{\mathcal{D}}')} + \frac{\sup(\tilde{t}_m, \tilde{\mathcal{D}}')}{\sup(\tilde{t}_l, \tilde{\mathcal{D}}')} \right) \right]$$

where $|\tilde{t}_l|$, $|\tilde{t}_m|$, and $|\tilde{t}|$ denote the number of items in $\mathcal{Q}$ that are mapped to $\tilde{t}_l$, $\tilde{t}_m$, and $\tilde{t}$, respectively, $\tilde{\mathcal{D}}$ is a dataset that contains $\tilde{t}_l$ and $\tilde{t}_m$, and $w: \tilde{\mathcal{Q}} \rightarrow [0, 1]$ is a function assigning a weight according to the perceived importance of a generalized item in analysis.

UC captures the amount of information loss incurred by generalization, based on the size, weight, and support, of the generalized items $\tilde{t}$, $\tilde{t}_l$, and $\tilde{t}_m$. As we will explain in Section 4.2,

UAR produces a new generalized item $\tilde{f}$ from $\tilde{f}_l$ and $\tilde{f}_r$, by mapping any item, which was mapped to $\tilde{f}_l$ or $\tilde{f}_r$, to $\tilde{f}$. Thus, the amount of information loss incurred by the construction of $\tilde{f}$ depends on the properties of $\tilde{f}_l$ and $\tilde{f}_r$, which are neglected by *NCP* and *UL*. Each term in square brackets in Definition 4.1 corresponds to a different property of the generalized items, as explained below:

- The first term rewards the construction small generalized items, based on generalized items that have similar lengths. This is because $\frac{|\tilde{f}_r|}{|\tilde{f}_l|} + \frac{|\tilde{f}_l|}{|\tilde{f}_r|}$ decreases with the difference of the lengths of $\tilde{f}_l$ and $\tilde{f}_r$, and it is minimized when these generalized items have equal lengths.
- The second term prevents the generation of generalized items that hinder analysis. This is because it favors the construction of a generalized item $\tilde{f}$ with small weight, based on $\tilde{f}_l$ and $\tilde{f}_r$ that have small weights themselves.
- The third term leads to the creation of generalized items with small support, based on generalized items that have small support themselves. In other words, this term attempts to reduce the number of transactions that are affected by generalization, similarly to what the first term does for length.

To see why using *UC* helps the production of data with low information loss, let us revisit Example 4.2. Observe that, contrary to *NCP* and *UL*, the *UC* measure correctly indicates that creating (a, b, c, d, e) (from (a, e) and (b, c, d)) incurs less information loss than constructing (f, g, h, i, j) (from f and (g, h, i, j)), because the length and supports of (a, e) and (b, c, d) in the dataset of Fig. 4(b) are equal. Specifically,

$$UC((a, e), (b, c, d), (a, b, c, d, e)) = \left[\left(\frac{2}{3} + \frac{3}{2} \right) \times 5 \right] \times \left[1 \times \left(\frac{1}{1} + \frac{1}{1} \right) \right] \times \left[4 \times \left(\frac{2}{2} + \frac{2}{2} \right) \right] = 173.3, \text{ whereas} \\ UC(f, (g, h, i, j), (f, g, h, i, j)) = 3541.7.$$

4.2. Anonymization algorithm

The *Update-Anonymize-Reorder* (UAR) algorithm aims at anonymizing a dataset so that it satisfies the specified privacy and utility constraints and incurs minimal information loss. To achieve this, UAR stores the privacy constraints that require protection in a priority queue, whose sorting criterion will be explained later. Then, UAR selects a privacy constraint from the priority queue and processes it in three phases. In the first, *Update phase*, the items of the selected privacy constraint are updated to reflect the generalizations and/or suppressions that have occurred in previous iterations. This way, the algorithm avoids the computational overhead that updating all privacy and utility constraints after each item transformation would bring. Then, in the *Anonymize phase*, UAR attempts to generalize a selected item (or generalized item) in the privacy constraint that is being processed, using the set-based anonymization model. If this would violate the utility constraints, UAR applies suppression to satisfy the selected privacy constraint. Otherwise, in the *Reorder phase*, auxiliary information about some of the privacy constraints in the priority queue is updated, and the priority queue is reorganized accordingly. This ensures that the selected privacy constraint in the next iteration of UAR incurs minimal information loss when processed. The process continues until all privacy constraints are satisfied.

4.2.1. Main operation of UAR—We now explain how UAR works in detail. The pseudocode of this algorithm is presented in Algorithm 1.

In steps 1–7, UAR initializes $\mathcal{Q}$ to $\mathcal{D}$ and a priority queue *PQ*, which stores triples containing each privacy constraint in $\mathcal{P}$ along with its support in $\mathcal{Q}$ and a variable *age*, which counts the number of generalizations that have been applied to the items of a privacy constraint. The

priority queue implements the usual operations *top()* and *pop()*. The former operation retrieves the privacy constraint that corresponds to an itemset with the maximum support in $\mathcal{D}$ without deleting it, whereas the latter one deletes the privacy constraint with the maximum support. Furthermore, PQ orders its elements with respect to the support in $\mathcal{D}$ in descending order, breaking ties with smallest *age*. This order is effective at minimizing information loss and guides UAR towards producing anonymizations of high data utility. This is because privacy constraints with large support in $\mathcal{D}$ are likely to require a small number of generalizations to be protected, while those with small *age* values do not contain heavily generalized items. Note also that this ordering criterion differs from the one employed by COAT (see Section 2.2) in that the support of privacy constraints in $\mathcal{D}$ changes as the items in these constraints are generalized or suppressed, and in that it differentiates between privacy constraints that have the same support in $\mathcal{D}$.

In the *Update* phase (steps 11 – 27), UAR assigns the top element of PQ (i.e., the one corresponding to the constraint with the largest support in $\mathcal{D}$ and smallest *age*) to the triple e (step 12). Then, the support of the selected privacy constraint $e.p$ is assigned to σ , and $e.p$ is updated by replacing its items to reflect the generalizations and suppressions that have occurred in previous iterations of UAR (steps 13–22). This strategy is performed using a hashtable H , which has each original item in $\mathcal{D}$ as key, and the generalized or suppressed item that this item is mapped to in $\mathcal{D}$ as value. Furthermore, it is significantly more efficient than updating all privacy constraints after an item transformation (the strategy employed by COAT), as shown in our experiments. If the privacy constraint $e.p$ is not satisfied, UAR selects it for further processing (steps 23 – 25). Otherwise, the top element is removed from PQ (step 26), and the next element of PQ is considered.

Next, in the *Anonymization* phase, UAR applies a function *Anonymize* to the privacy constraint $e.p$ that corresponds to the top element of PQ (step 30). The latter function generalizes or suppresses items in $e.p$ and returns the privacy constraint after transforming its items, which is assigned to $e.p$, as well as the generalized or suppressed item constructed by *Anonymize*, which is assigned to $\tilde{I}$ (details will be given later). Thus, the top element of PQ and $e.p$ contain the set of items in the privacy constraint before and after transformation, respectively. Following that, UAR computes the support of $e.p$ in $\mathcal{D}$, which is assigned to $e.sup$ (step 31), and checks whether $e.p$ is satisfied. In case it is, the top element of PQ is removed from PQ (steps 32–33).

Otherwise, UAR proceeds into the *Reordering* phase. In this phase, the top element of PQ , which corresponds to the selected privacy constraint before *Anonymize*, is removed from PQ (step 36), and e , which corresponds to the selected privacy constraint after applying *Anonymize*, is inserted into PQ with an increased *age* value (step 38). In addition, we update PQ by removing each element that has a support equal to σ and contains at least an item that has been mapped to the generalized item $\tilde{I}$, and reinserting this element into PQ after increasing its *age* value. After the Reordering phase, UAR selects the next element of PQ , if there is one. Otherwise, it returns the anonymized dataset (step 41).

The reordering strategy prevents privacy constraints that share items and have the same support with the selected privacy constraint from being processed in the next iterations of UAR. This would result in overgeneralizing certain items in many such privacy constraints, which exist as transaction data are typically sparse and k is much lower than the dataset size. Worse, these constraints are difficult to be identified based on a fixed ordering criterion, such as the one adopted by COAT or PCTA. This is because the support of a privacy constraint and the generalized items contained in the constraint are not known prior to anonymization. Thus, reordering allows UAR to produce anonymized data that permit more accurate analysis than those generated by COAT or PCTA, as we verify experimentally.

We now explain the way *Anonymize* works. As illustrated in Algorithm 2, this function starts by examining whether an item in the privacy constraint p can be generalized without violating the specified utility constraints. If this is possible, then we know from Definition 3.4 that there exists a utility constraint $u \in \mathcal{U}$ with two or more items, at least one of which is in p (step 1). In this case, *Anonymize* generalizes an item $i_m \in p$ together with another item from u . To find a generalization that helps data utility, all possible generalizations for such a pair of items are examined, and the one whose generalization would incur the minimum information loss, as measured by the UC measure, is selected (steps 3–9). When a pair of items v is found, *Anonymize* constructs a new generalized item $\tilde{a}$ by mapping to it each item in v (step 11), and then updates the transactions in $\mathcal{D}$ that support any of the items in v . Following that, the items in p and those in u , and the hashtable H are all updated to reflect the generalization of items in v to $\tilde{a}$, and p together with $\tilde{a}$ are returned (steps 12–16). If generalizing any item in p would result in violating the specified utility constraints, then suppression is applied (steps 18–26). Specifically, the item i_m in p that is supported by the fewest transactions in $\mathcal{D}$ is identified and removed from p , from its corresponding utility constraint, and from $\mathcal{D}$ (steps 19–23). In addition, the hashtable H is updated to reflect the suppression of i_m (step 24), and the data publisher is notified if the number of suppressed items exceeds $s\%$ (steps 25–26). In this case, the utility constraint set $\mathcal{U}$ has been violated and the algorithm terminates. Last, *Anonymize* returns the privacy constraint p together with $\tilde{a}$, which indicates that suppression has occurred (steps 27–28).

4.2.2. UAR through an example—Consider that UAR is applied to the original dataset in Fig. 1(a), with $k = 3$, $s = 0\%$, and the privacy and utility constraints shown in Figs. 1(e) and 1(f), respectively. UAR initializes the anonymized dataset $\mathcal{D}$ with the original dataset, and constructs the PQ that is illustrated in Fig. 5(b). Then, after creating an empty triple e and initializing σ to zero, UAR proceeds into the *Update phase*. It retrieves $\langle 2, 0, \{g\} \rangle$ from PQ and assigns it to e , while the support of the privacy constraint $\{g\}$ is assigned to σ .

Since no items have been transformed, $\{g\}$ is not updated. Furthermore, $\{g\}$ is not satisfied, so UAR enters the *Anonymization phase*. In this phase, *Anonymize* is executed and constructs (f, g) (steps 1–9 of Algorithm 2), because the corresponding utility constraint of g contains only f . *Anonymize* also updates the original dataset by replacing f and g with (f, g) , which results in the dataset of Fig. 5(a). Next, UAR computes the support of $\{(f, g)\}$, the privacy constraint after generalization, and assigns it to $e.sup$. As $e.sup < k$, the top element $\langle 2, 0, \{g\} \rangle$ of PQ is removed from PQ , and UAR proceeds into the next iteration. Now, e is assigned to $\langle 2, 0, \{b, e, f\} \rangle$, the current top element of PQ , and $\{b, e, f\}$ is updated to $\{b, e, (f, g)\}$. Since the support of $\{b, e, (f, g)\}$ in the dataset of Fig. 5(a) is less than k , UAR calls *Anonymize* to generalize b , or e , or (f, g) . The latter function examines all possible generalizations, which are shown in Fig. 6 along with their corresponding UC scores (in red), and constructs (b, e) , which has the best UC score. Thus, *Anonymize* returns $\{(b, e)(f, g), (b, e)\}$.

However, $\{(b, e), (f, g)\}$ is not yet satisfied, so UAR removes $\langle 2, 0, b, e, (f, g) \rangle$ from PQ , increases the age of $\{(b, e), (f, g)\}$ to 1, and inserts it into PQ , whose elements are shown in Fig. 5(c). Note that the increase of the age of $\{(b, e), (f, g)\}$ would force UAR to select another privacy constraint that has the same support as $\{(b, e), (f, g)\}$ and shares none of its items in the next iteration, to prevent further generalizing (b, e) and (f, g) . Since no such privacy constraint exists, UAR considers $\{(b, e), (f, g)\}$ again and eventually generalizes (b, e) to (a, b, c, d, e) , producing the anonymized dataset shown in Fig. 1(d). Note that the latter dataset satisfies both the specified utility and privacy constraints and incurs less information loss than that produced by COAT, shown in Fig. 1(c).

4.2.3. 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, the worst-case time complexity for UAR is

$$O(|P| \times |p| \times (\log(|\mathcal{U}|) + |u|^2 + N \times (|\mathcal{U}| + |u| + |p| + N)))$$

This corresponds to when each privacy constraint in $\mathcal{P}$ will be protected by first applying all possible generalizations that do not violate the utility constraints in $\mathcal{U}$ and then by suppressing all of its items one by one. In this case, we need $O(|\mathcal{P}| \times \log(|\mathcal{P}|))$ time to initialize *PQ* (steps 2–7 of Algorithm 1) and $O(|\mathcal{P}| \times |p| \times (\log(|\mathcal{U}|) + |u|^2 + N \times (|\mathcal{U}| + |u| + |p| + N)))$ time for anonymization (steps 8 – 40 of Algorithm 1).

Specifically, the while loop in step 8 of Algorithm 1 is executed $|\mathcal{P}|$ times, steps 11 to 22 take $O(|p| \times \log(|\mathcal{U}|))$ time, and *Anonymize* takes $O(|p| \times |u|^2)$ and $O(|p| \times N \times (|\mathcal{U}| + |u| + |p| + N))$ time to apply generalization and suppression to a privacy constraint p , respectively.

UAR scales linearly with $|\mathcal{P}|$, which is expected to be smaller than N when data publishers are able to specify detailed privacy requirements [4, 26, 27], but is quadratic to N . Thus, it is expected to be less efficient than COAT, which is linear to both $|\mathcal{P}|$ and N . However, UAR runs in less than a minute in most cases, and, since anonymization is an one-off task, data publishers may prefer to use UAR instead of a more efficient, but less effective in terms of preserving data utility algorithm, such as COAT.

5. Experimental evaluation

In this section, we experimentally demonstrate that UAR can produce anonymized data that satisfy both privacy and utility requirements in an efficient way. After discussing the experimental setup and the datasets we used, we compare our algorithm against three popular algorithms for preventing identity disclosure: (1) Apriori [23], which is specifically designed for k_m -anonymity, (2) COAT [26], which is the only algorithm able to take into account utility requirements, and (3) PCTA [27], which is the state-of-the-art method for minimizing information loss. The results of the comparison in terms of data utility and efficiency are presented in Sections 5.2 and 5.3, respectively, and show that UAR allows aggregate queries to be answered many times more accurately, while being scalable with respect to dataset size, k , and number of specified privacy constraints.

5.1. Experimental setup and datasets

We configured the tested algorithms as in [27] and transformed the anonymized datasets produced by Apriori by replacing each generalized item with the set of items it contains. All algorithms were implemented in C++ and executed on an Intel 2.8GHz machine with 4GB of RAM.

In our experiments, we used *AvgRE* to capture data utility and constructed workloads comprised of 1000 COUNT() queries similar to that of Fig. 3. The queries in each workload require retrieving the set of supporting transactions of 5-itemsets, which are comprised of randomly selected items, as in [26, 27]. We used the *BMS-Web View-1* and *BMS-Web View-2* datasets (henceforth referred to as *BMS1* and *BMS2*), which contain click-stream data from two e-commerce sites and have been used extensively in evaluating prior work [23, 26, 27, 43]. Furthermore, we used the Vanderbilt Native Electrical Conduction *VNEC* dataset [4], which contains de-identified electronic medical records from Vanderbilt University Medical Center [44]. Each transaction in *VNEC* corresponds to a different patient and contains diagnoses in the form of *ICD-9* codes³. An *ICD-9* code denotes a

disease of a certain type (e.g., 493.01 corresponds to *Extrinsic asthma with exacerbation*), and, in most cases, multiple *ICD-9* codes correspond to the same disease (e.g., 492.00, 493.01, and 493.02 all correspond to *Asthma*). Table 2 summarizes the characteristics of the datasets we used.

The default values for k and s were 5 and 0%, respectively, and, unless otherwise stated, COAT and PCTA were configured using one utility constraint that contains all items in $\mathcal{Q}$.

5.2. Data utility evaluation

In this section, we evaluate the effectiveness of the algorithms at preserving data utility in different data publishing scenarios.

5.2.1. Data utility for k^m -anonymity—We first assumed that data publishers have no specific utility requirements and that all 2-itemsets can lead to identity disclosure. To achieve protection, we configured Apriori by using $m = 2$ and used all 2-itemsets as privacy constraints for the other algorithms. Fig. 7(a) illustrates the *AvgRE* scores for *BMS1* and k varying in $[2, 50]$. As can be seen, UAR incurred significantly less information loss than the other algorithms for all tested k values. The *AvgRE* scores for UAR were at least 13 and up to 45 times lower than those of Apriori, which confirms that the set-based anonymization model can help UAR retain data utility. Furthermore, UAR permits on average 66% more accurate query answering than PCTA, the second best algorithm in this experiment, and outperforms all other algorithms particularly for larger k values. For instance, the *AvgRE* scores for PCTA were larger than those of UAR by 83% for $k = 50$. This is because, due to the reordering strategy and the *UC* measure it employs, UAR prevents overgeneralization. These conclusions can also be drawn from the results of the same experiment in *BMS2*, which is illustrated in Fig. 7(b).

Next, we considered a scenario that involves protecting all combinations of 1 to 3 items. The result for *BMS1* can be seen in Fig. 8(a). While all methods incur more information loss to prevent attacks in which more items are expected to be known, due to the privacy/utility trade-off [45], the increase in information loss caused by using a larger m is much smaller for UAR. This is because, UAR is able to generalize items that help preserve data utility by reordering the large number of privacy constraints that share items and have the same support, as explained in Section 4.2. As a result, UAR *permitted at least 19, 6, 4, and 4.5 times more accurate query answering than Apriori, COAT, and PCTA, respectively*. Similar results were obtained for *BMS2*, as can be seen in Fig. 9(a).

In a different scenario, we assumed that the published data need to satisfy 5^2 anonymity, as well as specific utility requirements, expressed using the utility constraint sets $U1, \dots, U7$. Each of these sets contains utility constraints that are comprised of a certain number of semantically close items⁴ (i.e., sibling items in the hierarchy), as shown in Table 3. The parameter s in this experiment was set to a small value of 0.5%, because the number of items that are allowed to be generalized together is much smaller than the scenario considered above. We present the results for UAR and COAT, the only algorithms that take into account utility constraints⁵ in Fig. 9(a), for *BMS1*, and in Fig. 9(b), for *BMS2*. Observe that UAR performed much better than COAT, incurring at least 2 and up to 13.4 times less information loss.

²This is a proprietary dataset that was made available to the authors while working at Vanderbilt University.

³ICD-9 is the official system of assigning health insurance billing codes to diagnoses in the United States.

⁴The last privacy constraint in each set may have fewer items than others.

⁵Apriori and PCTA were unable to produce data that satisfy the utility constraints in our experiments. This confirms the finding reported in [26].

5.2.2. Data utility for privacy-constrained anonymity—For this set of experiments, we considered two types of privacy constraints sets, following the setup of [26]. The first type contains a privacy constraint for every 2-itemset that can be constructed from items in a subset of $\mathcal{Q}$. Specifically, we created 5 sets of privacy constraints, namely $P_1, \dots, P_5$. The items in the privacy constraints of each set are selected from a random subset of $\mathcal{Q}$ that contains 10, 25, 50, 125, and 250 items for P_1, P_2, P_3, P_4 , and P_5 , respectively.

The *AvgRE* scores for all tested methods, when applied on *BMS1* and *BMS2*, are shown in Figs. 10(a) and 10(b), respectively. Again, UAR outperformed both COAT and PCTA, achieving *up to 16 and 2.6 times lower AvgRE scores*. Apriori performed much worse than any other methods in this test, so we do not present the results for it in the rest of this section. Interestingly, the difference in *AvgRE* scores between UAR and these algorithms gets larger as privacy constraints become more stringent. This again shows the ability of UAR to explore the space of generalizations effectively.

We also considered another type of privacy constraints sets, which are comprised of 1000 privacy constraints of different sizes, as shown in Table 4. For instance, P_8 contains an equal number of privacy constraints that contain 2, 3, and 4 items each. Figs. 11(a) and 11(b) illustrate the *AvgRE* scores for *BMS1* and for *BMS2*, respectively. Note that, due to the effective way it deals with privacy constraints, UAR produced data that allows up to 6.7 and 4.4 times more accurate query answering than that generated by COAT and PCTA.

5.2.3. Data utility in electronic medical record publishing—We compared the effectiveness of UAR to that of COAT in a real-world scenario, which involves publishing *VNEC* in a way that allows genetic studies, related to 18 diseases [46], as well as accurate biomedical query answering, to be performed. To achieve this, data recipients need to be able to accurately determine the number of patients suffering a disease, and information loss must be kept at a minimum. At the same time, the disclosure of patients' identity, which is possible as ICD-9 codes are contained in publicly available hospital discharge summaries [47], must be prevented. In this set of experiments, we used two query workloads, namely W_1 and W_2 . The first of these workloads is similar to those described in Section 5.1, whereas the second contains 1000 COUNT queries, each corresponding to a frequent itemset mined with minimum support threshold of 5%. Producing anonymized data that support W_2 is particularly important, as several biomedical data analysis tasks are based on frequent itemsets [48]. In this set of experiments, we configured UAR and COAT as in [4] and set s to 2%.

The *AvgRE* scores for W_1 and W_2 and k varying in [2, 25] are shown in Figs. 12(a) and 12(b), respectively. Observe that UAR consistently outperformed COAT, achieving scores that are at least 2.3 times better. Due to its superiority in retaining data utility and its ability to guarantee the published data will be useful in analysis, UAR can be extremely useful in privacy-preserving medical data publishing.

5.3. Efficiency evaluation

In this section, we examine the impact of dataset size, parameter k , and number of specified privacy constraints on efficiency. We start by reporting the time needed to anonymize datasets constructed by selecting increasingly larger subsets of *BMS1*. For consistency, the transactions in each of these subsets were selected uniformly at random but contained in all larger sets. As illustrated in Fig. 13(a), UAR was faster than Apriori for all tested k values, by 55% on average. Also, although slower than COAT and PCTA, UAR scales sublinearly with the dataset size. Then, we examined the effect of varying k and $|\mathcal{P}|$ on runtime. From Figs. 13(b) and 13(c), it can be seen that the efficiency of UAR is comparable to that of Apriori and that our algorithm scales well with respect to both of these parameters. This

indicates that the privacy constraint reordering strategy that helps UAR preserve data utility, as shown above, does not incur a significant computational overhead.

6. Conclusions

Publishing individuals' data that permit accurate data analysis is essential to support a growing number of real-world applications, but may lead to privacy breaches. To facilitate this task in a privacy-preserving manner, we introduced a novel approach to producing practically useful anonymized data. At the heart of our approach lies the UAR anonymization algorithm, which enforces the privacy-constraint anonymity principle and is guided by an accurate objective measure. UAR incurs significantly lower information loss than the state-of-the-art methods [23, 26, 27], as it processes the specified privacy constraints based on a flexible ordering scheme and transforms data in a way that satisfies the specified utility constraints with minimal information loss.

This work opens up several promising avenues for future research. These include examining how UAR can be extended to guard against both identity and sensitive information disclosure and how to produce anonymized data with guaranteed utility in certain data mining tasks, such as classification and association rule mining.

- ▶ Published transaction data may be linked back to individuals' identities.
- ▶ Existing anonymization approaches may incur excessive information loss.
- ▶ Our approach produces data with guaranteed utility and low information loss.
- ▶ A novel utility measure and an effective anonymization algorithm are developed.
- ▶ Experiments with click-stream and medical data show the effectiveness of our method.

Acknowledgments

We would like to thank Manolis Terrovitis, Nikos Mamoulis and Panos Kalnis for providing the implementation of the Apriori anonymization algorithm [23], as well as Bradley Malin for helpful discussions. Part of the research was funded by National Human Genome Research Institute Grant U01HG004603, National Library of Medicine Grant 1R01LM009989, and a Royal Academy of Engineering Research Fellowship.

References

1. National Institutes of Health. [NOT-OD-07-088] Policy for sharing of data obtained in NIH supported or conducted genome-wide association studies. 2007.
2. Medical Research Council. MRC data sharing and preservation initiative policy. 2006 <http://www.mrc.ac.uk/ourresearch/ethicsresearchguidance/datasharinginitiative>.
3. Kohane I. Using electronic health records to drive discovery in disease genomics. *Nature Review Genetics*. 2011; 12:417–428.
4. Loukides G, Gkoulalas-Divanis A, Malin B. Anonymization of electronic medical records for validating genome-wide association studies. *Proceedings of the National Academy of Sciences*. 2010; 17:7898–7903.
5. Aggarwal, CC.; Yu, PS. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer; 2008.
6. Gotz M, Machanavajjhala A, Wang G, Xiao X, Gehrke J. Publishing search logs - a comparative study of privacy guarantees. *IEEE Transactions on Knowledge and Data Engineering* 99 (PrePrints). doi:<http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.26>.

7. Narayanan A, Shmatikov V. Robust de-anonymization of large sparse datasets. *IEEE Security and Privacy*. 2008;111–125.
8. Samarati P. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*. 2001; 13(9):1010–1027.
9. Ponemon Institute. [last accessed Dec. 27, 2011] Annual Study: Global cost of a data breach. 2009. http://www.securityprivacyandthelaw.com/uploads/file/Ponemon_COB_2009_GL.pdf
10. Fung BCM, Wang K, Chen R, Yu PS. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*. 42
11. U.S. Department of Health and Human Services Office for Civil Rights. HIPAA administrative simplification regulation text. 2006
12. European Parliament. Council. EU Directive on privacy and electronic communications. 2002 <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:NOT>.
13. Domingo-Ferrer, J. *Encyclopedia of Database Systems*. US: Springer; 2009. Non-perturbative masking; p. 1912
14. di Vimercati SDC, Foresti S, Livraga G, Samarati P. Anonymization of statistical data (anonymisierung von statistischen daten). *IT - Information Technology*. 2011; 53(1):18–25.
15. Sweeney L. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*. 2002; 10:557–570.
16. LeFevre K, DeWitt D, Ramakrishnan R. Mondrian multidimensional k-anonymity. *ICDE*. 2006;25.
17. LeFevre K, DeWitt D, Ramakrishnan R. Incognito: efficient full-domain k-anonymity. *SIGMOD*. 2005;49–60.
18. Loukides G, Shao J. Preventing range disclosure in k-anonymised data. *Expert Systems with Applications*. 2011; 38(4):4559–4574.
19. Loukides G, Tziatzios A, Shao J. Towards preference-constrained - anonymisation. *DASFAA International Workshop on Privacy- Preserving Data Analysis (PPDA)*. 2009;231–245.
20. Loukides G, Shao J. An efficient clustering algorithm for - anonymisation. *Journal of Computer Science and Technology*. 2008; 23(2):188–202.
21. Liu K, Terzi E. Towards identity anonymization on graphs. 2008 *SIGMOD*. 2008;93–106.
22. Mohammed N, Fung BCM, Debbabi M. Walking in the crowd: anonymizing trajectory data for pattern analysis. *CIKM*. 2009;1441–1444.
23. Terrovitis M, Mamoulis N, Kalnis P. Privacy-preserving anonymization of set-valued data. *PVLDB*. 2008; 1(1):115–125.
24. Xu Y, Wang K, Fu AW-C, Yu PS. Anonymizing transaction databases for publication. *KDD*. 2008;767–775.
25. He Y, Naughton JF. Anonymization of set-valued data via top-down, local generalization. *PVLDB*. 2009; 2(1):934–945.
26. Loukides G, Gkoulalas-Divanis A, Malin B. COAT: Constraint-based anonymization of transactions. *Knowledge and Information Systems*. 2011; 28(2):251–282.
27. Gkoulalas-Divanis A, Loukides G. PCTA: Privacy-constrained Clustering-based Transaction Data Anonymization. *EDBT PAIS*. 2011;5.
28. Zheng Z, Kohavi R, Mason L. Real world performance of association rule algorithms. *KDD*. 2001;401–406.
29. Oliveira SRM, Zaiane OR. Protecting sensitive knowledge by data sanitization. *ICDM*. 2003;613–616.
30. Sun X, Yu PS. A border-based approach for hiding sensitive frequent itemsets. *ICDM*. 2005;8.
31. Gkoulalas-Divanis A, Verykios VS. Hiding sensitive knowledge without side effects. *Knowledge and Information Systems*. 2009; 20(3):263–299.
32. Gkoulalas-Divanis A, Loukides G. Revisiting sequential pattern hiding to enhance utility. *KDD*. 2011;1316–1324.
33. Abul O, Atzori M, Bonchi F, Giannotti F. Hiding sequences. *ICDE Workshop*. 2007;147–156.
34. Loukides G, Gkoulalas-Divanis A, Shao J. Anonymizing transaction data to eliminate sensitive inferences. *DEXA*. 2010;400–415.

35. Cao J, Karras P, Raïssi C, Tan K. *rho*-uncertainty: Inference-proof transaction anonymization. PVLDB. 2010; 3(1):1033–1044.
36. Dwork C. Differential privacy. ICALP. 2006:1–12.
37. Blum A, Dwork C, McSherry F, Nissim K. Practical privacy: the sulq framework. PODS. 2005:128–138.
38. Friedman A, Schuster A. Data mining with differential privacy. KDD. 2010:493–502.
39. Texas Department of State Health Services. [last accessed Dec. 27, 2011] User manual of texas hospital inpatient discharge public use data file. 2008. <http://www.dshs.state.tx.us/THCIC/>
40. Terrovitis M, Mamoulis N. Privacy preservation in the publication of trajectories. MDM. 2008:65–72.
41. Terrovitis M, Mamoulis N, Kalnis P. Local and global recoding methods for anonymizing set-valued data. VLDB J. 2011; 20(1):83–106.
42. Xu J, Wang W, Pei J, Wang X, Shi B, Fu AW-C. Utility-based anonymization using local recoding. KDD. 2006:785–790.
43. Ghinita G, Tao Y, Kalnis P. On the anonymization of sparse high-dimensional data. ICDE. 2008:715–724.
44. Roden D, Pulley J, Basford M, Bernard G, Clayton E, Balser J, Masys D. Development of a large scale de-identified dna biobank to enable personalized medicine. Clinical Pharmacology and Therapeutics. 2008; 84(3):362–369. [PubMed: 18500243]
45. Loukides G, Gkoulalas-Divanis A, Shao J. On balancing disclosure risk and data utility in transaction data sharing using R-U confidentiality map. Joint UNECE/Eurostat work session on statistical data confidentiality (to appear). 2011
46. Manolio T, Brooks L, Collins F. A hapmap harvest of insights into the genetics of common disease. Journal of Clinical Investigation. 2008; 118:1590–1605. [PubMed: 18451988]
47. Loukides G, Denny J, Malin B. The disclosure of diagnosis codes can breach research participants' privacy. Journal of the American Medical Informatics Association. 2010; 17:322–327. [PubMed: 20442151]
48. Ordóñez C. Association rule discovery with the train and test approach for heart disease prediction. IEEE Transactions on Information Technology in Biomedicine. 2006; 10(2):334–343. [PubMed: 16617622]

Name	Diagnoses
Anne	a b c d e f
Greg	a b e g
Jack	a e
Tom	b f g
Mary	a b
Jim	c f

(a)

Diagnoses
(a, b, c, d, e, f, g)
(a, b, c, d, e, f, g)
(a, b, c, d, e, f, g)
(a, b, c, d, e, f, g)
(a, b, c, d, e, f, g)
(a, b, c, d, e, f, g)

(b)

Diagnoses
(a, b, c, d, e) (f, g)
(a, b, c, d, e) (f, g)
(a, b, c, d, e)
(a, b, c, d, e) (f, g)
(a, b, c, d, e)
(a, b, c, d, e) (f, g)

(c)

Diagnoses
(a, b, d, e) c (f, g)
(a, b, d, e) (f, g)
(a, b, d, e)
(a, b, d, e) (f, g)
(a, b, d, e)
c (f, g)

(d)

Privacy constraints
{b, e, f}
{d, e}
{g}

(e)

Utility constraints
{a, b, c, d, e}
{f, g}

(f)

Figure 1.

An example of: (a) original dataset, (b) output of Apriori [23] (c) output of COAT [26], (d) output of UAR, (e) Privacy constraint set, (f) Utility constraint set, and (g) generalization hierarchy

Figure 2.
An example of generalizing items based on the set-based anonymization model

SELECT COUNT(T_n) FROM $\mathcal{D}$
WHERE T_n supports I in $\mathcal{D}$
(a)

SELECT COUNT($\tilde{T}_n$) FROM $\tilde{\mathcal{D}}$
WHERE $\tilde{T}_n$ supports $\tilde{I}$ in $\tilde{\mathcal{D}}$
(b)

Figure 3.

COUNT query applied to (a) original dataset, and (b) anonymized dataset

Figure 4.
Example of (a) original dataset, (b) a set-based anonymization of the dataset, and (c) generalization hierarchy

<i>Diagnoses</i>
a b c d e (f, g)
a b e (f, g)
a e
b (f, g)
a b
c (f, g)

(a)

$\langle 2, 0, \{g\} \rangle$
$\langle 1, 0, \{b, e, f\} \rangle$
$\langle 1, 0, \{d, f\} \rangle$

(b)

$\langle 2, 1, \{b, e, (f, g)\} \rangle$
$\langle 1, 0, \{d, f\} \rangle$

(c)

Figure 5.
(a) A dataset constructed during the first iteration of UAR, and the contents of *PQ* during (b) the first iteration of UAR, and (c) after the construction of (f, g)

generalized items and their UC scores

items in $\{b, e, (f, g)\}$ that can be generalized

Figure 6.

Possible generalizations for items in the privacy constraint $\{b, e, (f, g)\}$ and their corresponding *UC* scores (shown in red)

Figure 7.
AvgRE for 5^2 -anonymity vs k for (a) BMS1 and (b) BMS2

Figure 8.
AvgRE for 5^2 -anonymity vs m for (a) *BMS1* and (b) *BMS2*

Figure 9. *AvgRE* for 5^2 -anonymity and various utility constraint sets for (a) *BMS1* and (b) *BMS2*

Figure 10.
AvgRE vs. $P1, \dots, P5$ for (a) *BMS1*, and (b) *BMS2*

Figure 11.
AvgRE vs. $P_6, \dots, P_9$ for (a) *BMS1*, and (b) *BMS2*

Figure 12.
AvgRE vs. k for $VNEC$ and for (a) W_1 , and (b) W_2

Figure 13.
Runtime vs. (a) $|D|$, (b) k , and (c) $|\mathcal{P}|$ (BMSI)

Table 1

Summary of algorithms for preventing identity disclosure in transaction data publishing

Algorithm	Principle	Search strategy	Transformation
Partition [25]	complete k -anonymity	top-down partitioning	local generalization
Apriori [23]	k^m -anonymity	bottom-up traversal	global generalization
VPA [41]	k^m -anonymity	vertical partitioning	global generalization
LRA [41]	k^m -anonymity	horizontal partitioning	local generalization
PCTA [27]	privacy-constrained anonymity	item clustering	global generalization
COAT [26]	privacy-constrained anonymity	greedy search	global generalization and suppression

Table 2

Description of used datasets

Dataset	N	$ Q $	Max. size of T	Avg. size of T
<i>BMS1</i>	59602	497	267	2.5
<i>BMS2</i>	77512	3340	161	5.0
<i>VNEC</i>	2762	5830	25	3.1

Table 3

Summary of utility constraint sets used

Utility Constraint set	size of group of semantically close items
<i>U1</i>	5
<i>U2</i>	10
<i>U3</i>	25
<i>U4</i>	50
<i>U5</i>	125
<i>U6</i>	250
<i>U7</i>	500

Table 4Summary of sets of privacy constraints $P6$, $P7$, $P8$ and $P9$

Privacy constraint sets	% of items	% of 2-itemsets	% of 3-itemsets	% of 4-itemsets
$P6$	33%	33%	33%	1%
$P7$	30%	30%	30%	10%
$P8$	25%	25%	25%	25%
$P9$	16.7%	16.7%	16.7%	50%

Algorithm 1

UAR anonymization algorithm

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

output: Anonymized dataset $\tilde{\mathcal{D}}$

1. $\tilde{\mathcal{D}} \leftarrow \mathcal{D}; PQ \leftarrow \emptyset$
- /* Initialize privacy constraint queue PQ */*
2. **foreach** ($p_i \in \mathcal{P}$)
3. create triple $e \leftarrow \langle e.sup, e.age, e.p \rangle$
4. $e.sup \leftarrow sup(p_i, \tilde{\mathcal{D}})$
5. $e.age \leftarrow 0$
6. $e.p \leftarrow p_i$
7. $PQ \leftarrow PQ \cup e$
8. **while** ($PQ \neq \emptyset$)
9. create empty triple e
10. $\sigma \leftarrow 0$
- /* Update phase - updates privacy constraints based on previous item transformations */*
11. **do**
12. $e \leftarrow PQ.top()$ *// initialization of triple e*
13. $\sigma \leftarrow e.sup$
14. **foreach** ($i_m \in e.p$)
15. **if** ($H(i_m) = *$) *// i_m has been suppressed in a previous iteration*
16. $e.p \leftarrow e.p \setminus i_m$
17. **else if** ($H(i_m) = i_m$) *// i_m has been generalized in a previous iteration*
18. $\tilde{i}_m \leftarrow H(i_m)$
19. **if** ($\tilde{i}_m \in e.p$)
20. $e.p \leftarrow e.p \setminus i_m$
21. **else**
22. $e.p \leftarrow (e.p \setminus i_m) \cup \tilde{i}_m$
23. **if** ($sup(e.p, \tilde{\mathcal{D}}) \in (0, k)$)
24. $e.sup \leftarrow sup(e.p, \tilde{\mathcal{D}})$
25. **break**
26. $PQ.pop()$
27. **while** ($PQ \neq \emptyset$)
28. **if** ($PQ \neq \emptyset$)
29. **break**
- /* Anonymization phase - transformation of items in a selected privacy constraint */*
30. $\{e.p, \tilde{i}\} \leftarrow Anonymize(e.p, \mathcal{U}, \tilde{\mathcal{D}}, H, s)$
31. $e.sup \leftarrow sup(e.p, \tilde{\mathcal{D}})$
32. **if** ($e.sup \leq k$ or $e.sup = 0$)
33. $PQ.pop()$

```

34.  else
      /* Reordering phase - reorders privacy constraints in  $PQ$  to avoid overgeneralization */
35.  do
36.     $PQ.pop()$ 
37.     $e.age \leftarrow e.age + 1$ 
38.     $PQ.insert(e)$ 
39.     $e \leftarrow PQ.top()$ 
40.    while ( $e.sup = \sigma$  and  $e$  contains item  $i$  s.t.  $H(i) = \bar{i}$ )
41.  return  $\mathcal{G}$ 

```

Algorithm 2

Anonymize($p, \mathcal{U}, \tilde{\mathcal{D}}, H, s$)

input: Privacy constraint p , set of utility constraints $\mathcal{U}$, hashtable H , parameter s
output: Set of a privacy constraint p and a generalized item $\tilde{r}$

1. **if** (exists $u \in \mathcal{U}$ s.t. it contains at least 2 items and at least one of them is in p)
/ apply generalization to protect p */*
 2. $\mu \leftarrow 1$ *// maximum UC score*
 3. **foreach** ($i_m \in p$)
 4. $u \leftarrow$ the utility constraint from $\mathcal{U}$ that contains i_m
 5. **if** (u contains at least 2 items, one of which is in p)
 6. $i_s \leftarrow \arg \min_{i_r \in H, i_r \neq i_m} UC(i_m, i_r, (i_m, i_r))$
 7. **if** ($UC(i_m, i_s, (i_m, i_s)) < \mu$)
 8. $\mu \leftarrow UC(i_m, i_s, (i_m, i_s))$
 9. $v \leftarrow \{i_m, i_s\}$
 10. update transactions of $\tilde{\mathcal{D}}$ based on v
 11. $\tilde{r} \leftarrow (i_m, i_s)$ *// generalize the pair of items v*
 12. $p \leftarrow (p \cup \{\tilde{r}\}) \setminus v$
 13. $u \leftarrow u \cup \tilde{r} \setminus v$
 14. **foreach** ($i_r \in v$)
 15. $H(i_r) \leftarrow \tilde{r}$
 16. **return** $\{p, \tilde{r}\}$
17. **else** */* apply suppression to protect p */*
 18. **while** ($\text{sup}(p, \tilde{\mathcal{D}}) \in (0, k)$)
 19. $i_m \leftarrow \arg \min_{i_r \in p} \text{sup}(i_r, \tilde{\mathcal{D}})$
 20. $u \leftarrow$ the utility constraint from $\mathcal{U}$ that contains i_m
 21. $u \leftarrow u \setminus \{i_m\}$
 22. $p \leftarrow p \setminus \{i_m\}$
 23. Remove i_m from all transactions of $\tilde{\mathcal{D}}$
 24. $H(i_m) \leftarrow *$ *// update H to reflect the suppression of i_m*
 25. **if** more than $s\%$ of items are suppressed
 26. Error: $\mathcal{U}$ is violated
 27. $\tilde{r} \leftarrow ()$
 28. **return** $\{p, \tilde{r}\}$