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

ρ-uncertainty Anonymization by Partial Suppression

Xiao Jia · Hongfei Hu · Chao Pan · Yanping Xie · Xinhui Xu · Kenny Q. Zhu · Eric Lo

the date of receipt and acceptance should be inserted later

Abstract ρ-uncertainty is a privacy model which guarantees that no sensitive association rules can be inferred with confidence larger than ρ from a set of transactions, regardless of the background knowledge from the attacker. Traditional approaches to achieve ρ-uncertainty either generalize the data according to a concept hierarchy or globally suppress some data types which hurts the overall usability of the data. We present a novel partial suppression framework which can be adapted to both space-time and quality-time trade-offs in a "pay-as-you-go" approach. While minimizing the number of item suppressions, the framework attempts to either preserve the original data distribution or retain minable useful association rules, which respectively targets statistical analysis and association mining, two major data mining applications on set-valued data.

Keywords data privacy; set-valued data; data anonymization; partial suppression; association rule mining; data distribution

1 Introduction

Set-valued data sources are valuable in many data mining and data analysis tasks. For example, retail companies may want to know what items are top sellers (e.g., milk), or whether there is an association between the purchase of two or more items (e.g., people who buy flour also buy milk). According to our observation there are two main categories of set-valued data analysis: one is *statistical analysis* such as computing max, min and average values; the other is *mining of association rules* between items. In many cases, analysis tasks are

Xiao Jia · Hongfei Hu · Chao Pan · Yanping Xie · Xinhui Xu · Kenny Q. Zhu
Shanghai Jiao Tong University, China
E-mail: kzhu@cs.sjtu.edu.cn

Eric Lo
Hong Kong Polytechnic University, China

outsourced to other external companies or individuals, or simply *published* to the general masses for scientific and public research purposes.

Publishing set-valued, and especially transactional data, can pose significant privacy risks. Set-valued transactions consist of one or more data items, which can be divided into two categories: *non-sensitive* and *sensitive.* Privacy is in general associated with the sensitive items. Table 1(a) shows an example of retail transactions in which each record (row) represents a set of items purchased in a single transaction by an individual. All the items are non-sensitive, except the *condom* which is sensitive. An individual's privacy is breached if he or she can be *re-identified*, or associated with a record in the data which contains one or more sensitive items. Past research has shown that such breach is possible through *linking attacks* [**?**,**?**], e.g. linking beer with condom in Table 1(a).

Table 1 A Retail Dataset and Anonymization Results

(a) Original Dataset

ID	**Transaction**
1	bread, beer, *condom*
2	coffee, fruits
3	beer, *condom*
4	coffee, fruits
5	flour, *condom*
6	bread, coffee
7	fruits, *condom*

(b) Global Suppression

ID	**Transaction**
1	bread, beer, ~~*condom*~~
2	coffee, fruits
3	beer, ~~*condom*~~
4	coffee, fruits
5	flour, ~~*condom*~~
6	bread, coffee
7	fruits, ~~*condom*~~

(c) Our Approach 1

ID	**Transaction**
1	bread, beer, ~~*condom*~~
2	coffee, fruits
3	beer, *condom*
4	coffee, fruits
5	~~flour~~, *condom*
6	bread, coffee
7	fruits, *condom*

(d) Our Approach 2

ID	**Transaction**
1	bread, beer, ~~*condom*~~
2	coffee, fruits
3	beer, *condom*
4	coffee, fruits
5	flour, ~~*condom*~~
6	bread, coffee
7	fruits, *condom*

The privacy model we want to achieve is called ρ-uncertainty, where no sensitive association rules can be inferred with a confidence higher than ρ [**?**]. The most popular approach to achieve ρ-uncertainty is called "global suppression" [**?**] in which once an occurrence of an item t is determined to be removed from one record, all occurrences of t are removed from the whole dataset. We instead opt to *partially* suppress the data set so only *some* occurrences of item t are deleted. Table 1 shows the example original dataset and three anonymized datasets produced by global suppression and our approaches. The orginal dataset is not safe because sensitive rules such as beer $\rightarrow$ *condom* and flour $\rightarrow$ *condom* can be inferred with confidence 100%, which is greater than our threshold $\rho = 50\%$. Table 1(b) is the anonymized dataset where all the occurrences of the sensitive item *condom* are deleted due to global suppression. Table 1(c) shows the anonymized dataset produced by our first approach, which is optimized to preserve data distribution, so different items (*condom* and flour) are deleted to make the dataset safe. Table 1(d) is the result of our second approach, which is optimized to preserve important data association in the original dataset, so only two occurrences of the item *condom* are deleted for safety. Even in such a small dataset, both our approaches outperform global suppression in the number of deletions (2 vs. 4) while retaining useful information at the same time.

To the best of our knowledge, the partial suppression technique has not been studied in the context of set-valued data anonymization before. We choose to solve the set-valued data anonymization problem by partial suppression because global suppression tends to delete more items than necessary, and the removal of all occurrences of the same item not only changes the data distribution significantly but also makes mining association rules about the deleted items impossible. The problem of anonymization by suppression (global or partial) is very challenging [**?**,**?**], exactly because, (i) the number of possible inferences from a given dataset is exponential, and (ii) the size of the search space, i.e. the number of ways to suppress the data is also exponential to the number of data items. We therefore propose two heuristics in this paper to anonymize input data in two different ways, giving rise to the two kinds of output in Table 1.

The main contributions of this paper are as follows.

1. To the best of our knowledge, we are the first to propose an effective *partial* suppression framework for anonymizing set-valued data (Section 2 and 3).
2. We adopt a “pay-as-you-go” approach based on divide-and-conquer, which can be adapted to achieve both space-time and quality-time trade-offs (Section 3 and 4). Our two heuristics can be adapted to either preserving data distribution or retaining useful association in the data (Section 3).
3. Experiments show that our algorithm outperforms the peers in preserving the original data distribution (more than 100 times better than peers) or retaining mineable useful association rules while reducing the item deletions by large margins (Section 4).

2 Problem Definition

This section introduces the privacy model and data utility before formally describing the problem of partial suppression.

2.1 Privacy Model

$X \rightarrow Y$ is a *sensitive association rule* iff Y contains at least one sensitive item. The privacy model of this paper stipulates that, a dataset T is *safe* iff no sensitive rules can be inferred from it with a confidence higher than ρ [**?**]. It is clear that, if all sensitive association rules with consequence of exactly one item, can't be inferred from T with a confidence higher than ρ, then all sensitive association rules can't be inferred with a confidence higher than ρ, and hence T is safe.

Formally, we define quasi-identifier (a.k.a. *qid*) to be any itemset (including sensitive items) drawn from any record in table T. A *qid* q is safe w.r.t. ρ iff $conf(q \to e) \leq \rho$, for any sensitive item e in T. We say T is safe w.r.t. ρ iff q is safe w.r.t. ρ for any *qid* q in T. A *suppressor* is a function $S : T \mapsto T'$ where T' is a suppressed table which is safe w.r.t. ρ. There are many different ways to suppress a table. The goal is to find a suppressor that maximizes the *utility* of the suppressed table.

2.2 Data Utility

In this paper, we identify two major uses of an anonymized table: *statistical analysis* and *association rule mining*. In the first case, we want the anonymized table to have a distribution as close to the original table as possible; in the second case, we would like the anonymized data to retain all non-sensitive association rules while introducing few or no spurious rules. In both scenarios, the common goal is to minimize the *information loss*, i.e., the total number of items suppressed.

With these two scenarios in mind, we define two variants of an objective function $f(T, T')$ as:

$$f(T,T') = \begin{cases} NS(T,T') \cdot KL(T' \mid\mid T) & \text{(data distribution)} \\ \\ \frac{NS(T,T')}{J(R(T),R(T'))} & \text{(rule mining)} \end{cases} \tag{1}$$

where

$$NS(T,T') = \frac{\sum_{e \in D(T)} (sup_T(e) - sup_{T'}(e))}{\sum_{e \in D(T)} sup_T(e)} \tag{2}$$

$$KL(P \mid\mid Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{3}$$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

$$R(T) = \text{ set of rules minable from dataset } T \tag{5}$$

Here $D(T)$ denotes the domain of items in T, $sup_T(e)$ denotes the support of item e in T, while the support of item e means the number of instances of item e in a certain set. $P(i)$ and $Q(i)$ denote the probability of item i in distribution P and Q, respectively. The functions NS, KL and J represent total number of suppressions (normalized to 1), K-L divergence[**?**] and Jaccard similarity[**?**], respectively. K-L divergence measures the distance between two probability distributions, while Jaccard similarity measures the similarity between two sets. $R(T)$ is the set of rules mined from dataset T with a given confidence. In this paper, we set the this confidence to be ρ, same as the parameter of our privacy model. In our model, ρ denotes that an attacker is only interested in mining sensitive rules with at least a confidence of ρ.

2.3 Optimal Partial Suppression Problem

The optimal partial supression problem is to find a *Partial Suppressor* S which anonymizes an input set-valued table T to minimize the objective function:

$$\min_{S} f(T, S(T))$$

such that $S(T)$ is *safe* w.r.t. to our privacy model.

3 Partial Suppression Algorithm

The Optimal Suppression Problem defined in Section 2 is an NP-hard problem. We therefore present the partial suppression algorithm as a heuristic solution to the Optimal Suppression Problem. To simplify the discussion of the algorithm, we make the following definitions.

Definition 1 (Number of Suppressions) To disable an unsafe rule $q \to e$, the minimum number of instances of item $t \in q \cup \{e\}$ that needs to be suppressed is

$$N_s(t, q \to e) = \begin{cases} sup_T(q \cup \{e\}) - sup(q)\rho & t = e \\ \frac{sup_T(q \cup \{e\}) - sup_T(q)\rho}{1-\rho} & t \in q \end{cases}$$

Proof If $t = e$,

$$\begin{aligned} conf(q \to e) &= \frac{\sup_{T'}(q \cup \{e\})}{\sup_{T'}(q)} \\ &= \frac{\sup_T(q \cup \{e\}) - N_s(t, q \to e)}{\sup_T(q)} \\ &= \frac{\sup_T(q \cup \{e\}) - (\sup_T(q \cup \{e\}) - \sup_T(q)\rho)}{\sup_T(q)} \\ &= \rho \end{aligned}$$

If $t \in q$,

$$\begin{aligned} conf(q \to e) &= \frac{\sup_{T'}(q \cup \{e\})}{\sup_{T'}(q)} \\ &= \frac{\sup_T(q \cup \{e\}) - N_s(t, q \to e)}{\sup_T(q) - N_s(t, q \to e)} \\ &= \frac{\sup_T(q \cup \{e\}) - \frac{\sup_T(q \cup \{e\}) - \sup_T(q)\rho}{1-\rho}}{\sup_T(q) - \frac{\sup_T(q \cup \{e\}) - \sup_T(q)\rho}{1-\rho}} \\ &= \rho \end{aligned}$$

In other words, for each sensitive rule r, we need to delete at least $N_s(t, r)$ instances of item t to make it safe. In this work, we select these instances randomly for deletion.

Definition 2 (Leftover Items) The leftover of item type t is defined as

$$leftover(t) = sup_{T'}(\{t\})/sup_T(\{t\})$$

T is the original data and T' is the intermediate suppressing result. The ratio shows the percentage of remaining instances of item t in the intermediate result T'.

The key intuition of our algorithm is that although the total number of "bad" sensitive association rules maybe, in the worst case, exponential in the original data, incremental "invalidation" of some of the rules through partial suppression of a small number of affected items can massively reduce the number of these bad rules, which leads to quick convergence to a solution, that is, a safe data set. Next we present the basic algorithm of this framework.

3.1 The Basic Algorithm

PartialSuppressor (Algorithm 1) presents the top-level algorithm. The partial suppressor iterates over the table T, and for each record $T[i]$, the algorithm first generates *qid*s from $T[i]$ and sanitizes the unsafe ones. The suppressor terminates when the whole table is scanned and there is no unsafe *qid*.

Algorithm 1: PartialSuppressor($T, b_{\max}$)

```
1: $T_0 \leftarrow T$ (original table)
2: loop
3:   Initialize the $sup$ of all $qid$s to 0
4:   while $|B| < b_{max}$ and $i \leq |T|$ do
5:     Fill $B$ with $qid$s generated by $T[i]$
6:     Update $sup$ of all $qid$s
7:     $i \leftarrow i + 1$
8:   end while
9:   if $B$ contains an unsafe $qids$ then
10:    SanitizeBuffer($T_0, T, B$)
11:    $safe \leftarrow$ false
12:  end if
13:  if $i \geq |T|$ and $safe$ then
14:    break
15:  else if $i \geq |T|$ then
16:    $i \leftarrow 1$
17:    $safe \leftarrow$ true
18:  end if
19: end loop
```

A *qid* is a combination of different items, and the number of distinct *qid*s to be enumerated is exponential. We therefore introduce a *qid* buffer of capacity

$b_{\max}$ to balance the space consumption with the generation time. The value of $b_{\max}$ is significant. Small $b_{\max}$ values cause repetitive generation of *qid*s, while large $b_{\max}$ values cause useless generation of *qid*s which do not exist by the time to process them in the queue.

3.2 Buffer Sanitization

Each time *qid* buffer B is ready, SanitizeBuffer (Algorithm 2) is invoked to start processing *qid*s in B and make all of them safe. $D_S(T)$ denotes the domain of all sensitive items in T. We first partition *qid*s in B into two groups, *safe* and *unsafe*. Then in each iteration (Lines 2-18), SanitizeBuffer picks the "best" (according to heuristic functions H) unsafe sensitive association rule to sanitize (Lines 6 and 8). SuppressionPolicy in SanitizeBuffer uses one of the the following two heuristic functions.

Algorithm 2: SanitizeBuffer(T_0, T, B)

```
1: $\mathcal{P} \leftarrow$ SuppressionPolicy()
2: repeat
3:   pick an unsafe qid $q$ from $B$
4:   $E \leftarrow \{e \mid conf(q \to e) > \rho \wedge e \in D_S(T)\}$
5:   if $\mathcal{P} = Distribution$ then
6:     $(d, q, e) \leftarrow \underset{d \in q \cup E, q, e \in E}{\arg\max} H_{dist}(d, q, e, T_0, T)$
7:   else if $\mathcal{P} = Mine$ then
8:     $(d, q, e) \leftarrow \underset{d \in q \cup E, q, e \in E}{\arg\min} H_{mine}(d, q, e)$
9:   end if
10:  $X \leftarrow q \cup \{e\}$
11:  $k \leftarrow N_s(d, q \to e)$
12:  while $k > 0$ do
13:    pick a record $R$ from $T$ where $R \subseteq \mathcal{C}(X)$
14:    $R \leftarrow R - \{d\}$
15:    Update sup of qids contained in $R$
16:    $k \leftarrow k - 1$
17:  end while
18: until there is no unsafe qid in $B$
```

3.2.1 Preservation of Data Distribution

Consider an unsafe sensitive association rule $q \to e$ where $conf(q \to e) > \rho$, and $q \in B$. To reduce $conf(q, e)$ below ρ, we suppress a number of instances of item $t \in q \cup \{e\}$ from $\mathcal{C}(q \cup \{e\})$.[1] We hope to minimize $KL(T \parallel T_0)$ (see Equation (3)). From Equation (3), we observe that by suppressing some

[1] We define $\mathcal{C}(X) = \{T[i] | X \subseteq T[i], 1 \leq i \leq |T|\}$.

instances of item t where $T(t) > T_0(t)$,[2] the KL divergence tends to decrease, thus we define the following heuristic function

$$H_{dist}(t, q, e, T_0, T) = \frac{T(t)log\frac{T(t)}{T_0(t)}}{N_s(t, q \to e)}. \tag{6}$$

Maximizing this function aims at suppressing item t which maximally recovers the original data distribution and minimizes the number of deletions.

3.2.2 Preservation of Useful Rules

A spurious rule $(q \to e)$ is introduced when the denominator of $conf(q \to e)$, $sup(q)$, is sufficiently small so that the confidence appears large enough. However, if $sup(q)$ is too small, the rule would not have enough support and can be ignored. Therefore, our objective is to continue suppressing those items that have been suppressed before, to minimize the support of the potential spurious rules. Therefore, we seek to minimize

$$H_{mine}(t, q, e) = leftover(t) \cdot N_s(t, q \to e) \tag{7}$$

3.2.3 Regression

After suppressing the item d, unsafe *qid*s may become safe, while safe ones may become unsafe again, an undesirable situation known as *regression*. Line 15 of Algorithm SanitizeBuffer determines the set of *qid*s that would be affected by regression. This step is like the *qid* generation step in Algorithm PartialSuppressor Line 5 and 6. There are also different ways to pick a record from $\mathcal{C}(q \cup \{e\})$ to suppress item d (Line 13), and currently we pick a random record from $\mathcal{C}(X)$ for simplicity.

3.3 Optimization with Divide-and-Conquer

When data is very large we can speed up by a divide-and-conquer (DnC) framework that partitions the input data dynamically, runs PartialSuppressor on them individually and combines the results in the end. This approach is correct in the sense that if each suppressed partition is safe, so is the combined data. This approach also gives rise to the parallel execution on multi-core or distributed environments which provides further speed-up (this will be shown in Section 4).

Algorithm 3 splits the input table whenever the estimated cost of suppressing that table is greater than $t_{\max}$. Cost is estimated as:

$$Cost(T) = \frac{|T| \cdot 2^{\frac{N}{|T|}}}{|D(T)|} \tag{8}$$

[2] We denote the probability of item t in T as $T(t)$, which is computed by $\frac{sup_T(t)}{|T|}$.

Algorithm 3: DNCSplitData($T, t_{\max}$)

```
1: if  $Cost(T) > t_{\max}$  then
2:     Split $T$ equally into $T_1$ , $T_2$
3:     DNCSplitData($T_1, t_{\max}$)
4:     DNCSplitData($T_2, t_{\max}$)
5: else
6:     PartialSuppressor($T, b_{\max}$)
7: end if
```

where N is the total number of items in T. In order to guarantee the consistency in data distribution between the original problem and the sub-problems, when splitting T equally into T_1 and T_2, we randomly assign half of the items in T to T_1 and the remaining items to T_2 .

3.4 Analysis of the Algorithm

In this section, we present some theoretical results along with their proofs in order to provide a comprehensive view of the problem and our algorithm.

Lemma 1 *If q is safe in both T_1 and T_2, then q is safe in $T = T_1 \cup T_2$.*

Proof For any item a,

$$q \text{ is safe in } T_1 \Rightarrow sup_{T_1}(q \cup \{a\}) \le \rho \cdot sup_{T_1}(q)$$
$$q \text{ is safe in } T_2 \Rightarrow sup_{T_2}(q \cup \{a\}) \le \rho \cdot sup_{T_2}(q)$$

So

$$sup_{T_1}(q \cup \{a\}) + sup_{T_2}(q \cup \{a\}) \le \rho \cdot sup_{T_1}(q) + \rho \cdot sup_{T_2}(q)$$

And

$$sup_T(q \cup \{a\}) = sup_{T_1}(q \cup \{a\}) + sup_{T_2}(q \cup \{a\})$$
$$sup_T(q) = \kappa_{T_1}(q) + sup_{T_2}(q)$$

Therefore,

$$\frac{sup_T(q \cup \{a\})}{sup_T(q)} \le \rho \ \Rightarrow q \text{ is safe in } T.$$

Theorem 1 PartialSuppressor *always terminates with a correct solution.*

Proof We first prove that if the algorithm terminates, the suppressed table is safe. Note that the algorithm can only terminate on Line 14 in Algorithm 1. Therefore, two conditions must be satisfied. First, the record cursor i should exceed the table size $|T|$. Second, the value $Safe$ must be **true** . $Safe$ is true if and only if there is no unsafe qids in the table, otherwise Line 2 will assign $Safe$ to **false** . If i exceeds $|T|$ and $Safe$ is **true** , the algorithm must have scanned the table at least once and hasn't found any unsafe *qid*s. Hence, the suppressed table is safe.

Then we prove that PARTIALSUPPRESSOR always terminates by measuring the number of items left (denoted l) in the table after each step of suppression. Initially,

$$l = l_0 = \sum_{i=1}^{|T|} |T[i]| \leq |D(T)||T|.$$

We state that for every invocation of SANITIZEBUFFER , Line 14 in Algorithm 2 is always executed at least once. So the value l strictly decreases by a positive integer when SANITIZEBUFFER is invoked. And before the table becomes safe, SANITIZEBUFFER will be invoked for every iteration of the loop in Algorithm 1. So l strictly decreases for each loop iteration in PARTIALSUPPRESSOR . Because l starts from a finite number which is at most

$$l_0 = \sum_{i=1}^{|T|} |T[i]|,$$

PARTIALSUPPRESSOR must terminate.

Now we prove that Line 14 in Algorithm 10 is always executed once SANITIZEBUFFER is invoked. Whenever SANITIZEBUFFER is invoked, it is guaranteed that there exists an unsafe *qid* $q \in B$ (see Line 9 in Algorithm 1). q is unsafe so that there always exists an item $e \in \mathcal{L}(q)$ such that $conf(q,e) > \rho$, i.e.

$$\frac{sup(q \cup \{e\})}{sup(q)} > \rho \Rightarrow sup(q \cup \{e\}) - \rho \cdot sup(q) > 0.$$

For k on Line 11 in Algorithm 2,

$$k = N_s(t, q \to e)$$

and

$$N_s(t, q \to e) \geq sup(q \cup \{e\}) - \lfloor \rho \cdot sup(q) \rfloor \geq 1$$

as is shown in Definition 1. Therefore, it is guaranteed that the number of deletions is at least 1 because the rule $q \to e$ is unsafe and there must be some deletions to make it safe. So $k \geq 1$ on Line 12 for the first time. Thus the condition is satisfied and Line 14 is executed.

Corollary 1 *The divide-and-conquer optimization* DNCSPLITDATA *is correct.*

Proof It follows directly from Lemma 1 and Theorem 1.

Table 2 Five Original Datasets

Dataset	Description	Recs	Dom. Size	Sensitive items	Non-Sens. items
BMS-POS (POS)	Point-of-sale data from a large electronics retailer	515597	1657	1183355	2183665
BMS-WebView (WV)	Click-stream data from e-commerce web site	77512	3340	137605	220673
Retail	Retail market basket data	88162	16470	340462	568114
Syn	Synthetic data with max record length = 50	493193	5000	828435	1242917

4 Experimental Results

We conducted a series of experiments on 4 main datasets in Table 2. BMS-POS and BMS-WebView are introduced in [?] and are commonly used for data mining. Retail is the retail market basket data [?]. Syn is the synthetic data in which each item is generated with equal probability and the max record length is 50. Figure 1 shows the record length distribution of the datasets. All the datasets exhibit a trend where the number of records decreases almost exponentially with the record length. We randomly designate 40% of the item types in each dataset as sensitive items and the rest as non-sensitive. To evaluate the performance of rule mining, we produce four additional datasets by truncating all records in the original datasets to 5 items only, and denote such datasets as "cutoff = 5".

We compare our algorithm with the global suppression algorithm (named Global) and generalization algorithm (named TDControl) of Cao *et al.* [?].[3] Our algorithm has two variants, *Dist* and *Mine*, which optimize for data distribution and rule mining, respectively. Experiments that failed to complete in 2 hours is marked as "N/A" or an empty place in the bar charts. We run all the experiments on Linux 2.6.34) with an Intel 16-core 2.4GHz CPU and 8GB RAM.

We design a metric, *info loss*, to evaluate how much information is lost after the partial compression.

$$\text{info loss} = \frac{Items(T) - Items(T')}{Items(T)}$$

Here, we define $Items(T)$ as the number of items in dataset T. The method we adopt to calculate information loss is reasonable, because $Items(T) - Items(T')$ represents the loss of raw data that can be used for statistical analysis or association rule mining. Obviously, *info loss* equals to 0 if there is no information loss.

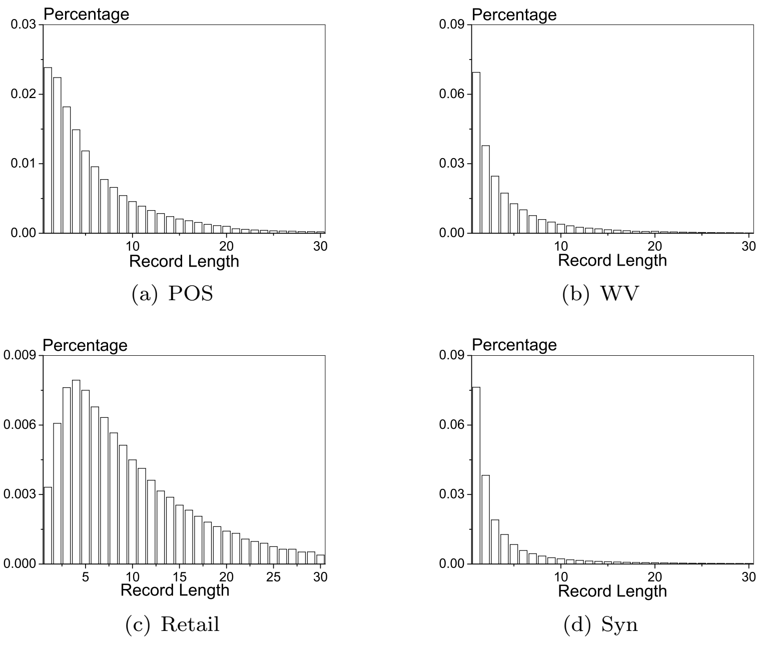


Fig. 1 Distribution in Record Length of Five Original Datasets

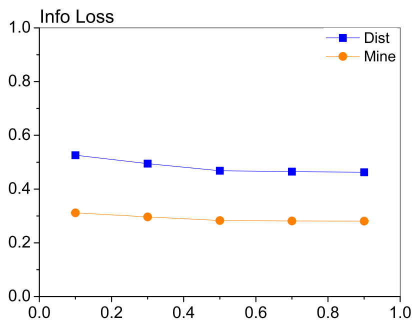
4.1 Variation of ρ

In this section, we gives the performance of our algorithm with regard to the variation of ρ. To determine the similarity between the item frequency distribution of original data and that of the anonymized data, we use the Kullback-Leibler divergence (also called relative entropy) as our evaluation metric. To prevent zero denominators, we modified Equation (3) to a symmetric form [**?**] defined as

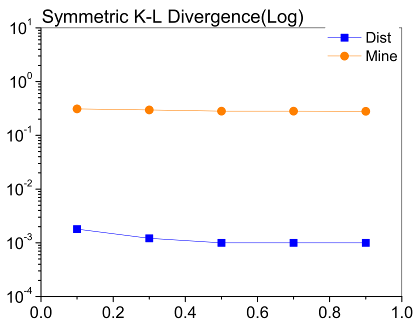
$$\mathcal{S}(H_1||H_2) = \frac{1}{2}KL(H_1||H_1 \cup H_2) + \frac{1}{2}KL(H_2||H_1 \cup H_2)$$

where $H_1 \cup H_2$ represents the union of distributions H_1 and H_2. We take WV as our test data. Figure 2(a) shows that the information loss decreases when ρ becomes larger. The reason is clear: as ρ grows, there are fewer unsafe qids in the data and fewer suppressions need to be executed and hence less information loss. Figure 2(b) shows that data distribution is better when ρ becomes larger. The reason should also be attributed to the fewer suppressions executed when ρ becomes larger. Fewer suppressions mean less disturbance to the original distribution. Figure 2(c) shows the relation between ρ and the Jaccard Similarity of $R(T)$ and $R(T')$. It is an interesting curve that initially rises slowly with ρ, but then drops drammatically. Our explanation is, when ρ is

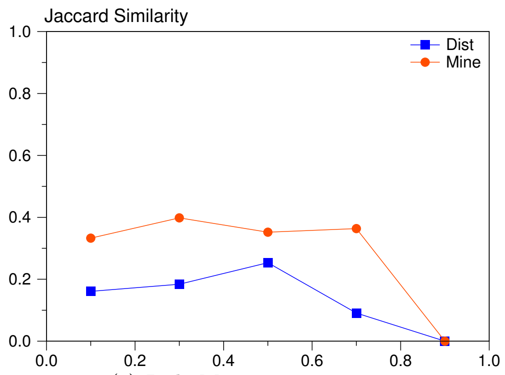
[3] The source code of these algorithms was directly obtained from Cao.



(a) Information Loss



(b) Data Distribution



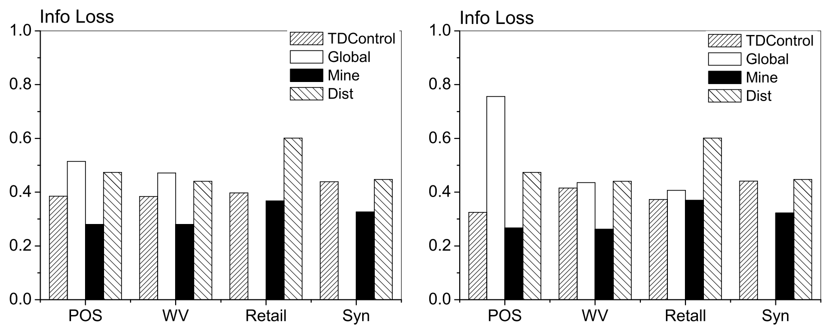
(c) Rule Mining

Fig. 2 Variation of ρ (from 0 to 1)

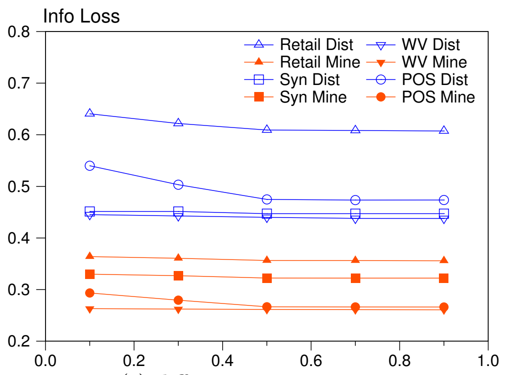
relatively small, Jaccard Similarity increases with ρ because fewer suppressions lead to fewer original rules deleted and fewer spurious rules added. When ρ becomes relatively large, the descendence of Jaccard Similarity should be attributed to the shrinkage of the sizes of $R(T)$ and $R(T')$. When ρ is relatively large, it is hard to mine association rules from dataset so the size of ruleset is small. And it is natural that when the sizes of both sets are comparatively small, it is difficult for them to be similar. When ρ is 0.9, we find no association rule in the suppressed dataset and only 1 rule in the original dataset. Therefore, the Jaccard Similarity is 0. [4]

In what follows, we first present results in data utility, then the performance of the algorithms, and then the effects of changing various parameters in our algorithm. Finally, we compare with a permutation method which utilizes a similar privacy model but with different optimization goals. Based on previous observations, we choose two representative values of ρ, 0.3 and 0.7, for the data utility experiments, and $\rho = 0.7$ for other experiments. Unless otherwise

[4] When we do rule mining, we adopt the A-priori algorithm and set the minimum support rate to be 0.05%, which is a reasonable value in practice.



(a) $\rho = 0.3$ (b) $\rho = 0.7$



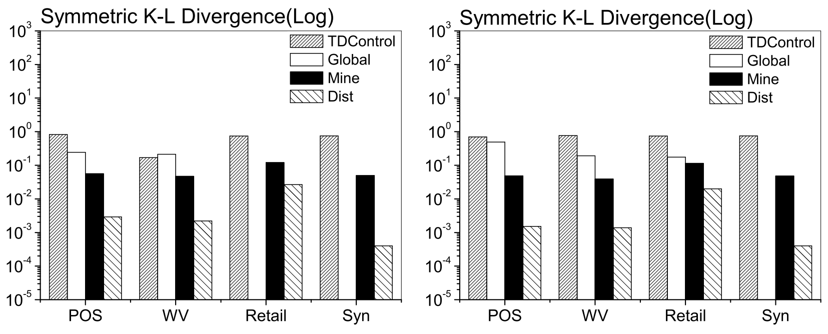
(c) different ρ

Fig. 3 Comparisons in Information Loss

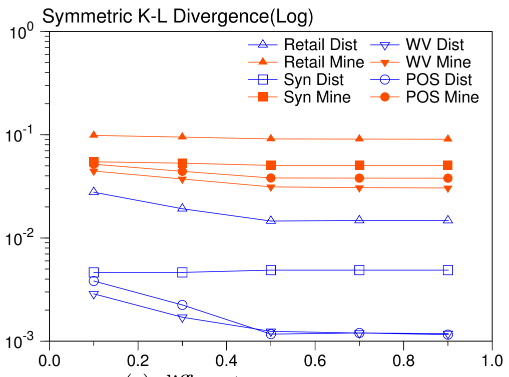
noted, we use the following default parameters: $b_{max} = 10^6$, $t_{max} = 500$. These default parameters are carefully set to provide better results and to ensure that the process can terminate in a reasonable amount of time. For example, we set $b_{max} = 10^6$ for it can provide a buffer big enough to hold all qids. If we use a smaller one, the execution time will be lengthened to be reloading of buffer space. If b_{max} is too large, it will result in waste of space. We set $t_{max} = 500$ since it is an appropriate time for the execution of each divide-and-conquer problem.

4.2 Data Utility

We compare the algorithms in terms of information loss, data distribution, and association rule mining. As TDControl generates rules in general form, we adopt a specialization method for the evaluation of its information loss, which we will explain at the end of section 4.2.



(a) $\rho = 0.3$ (b) $\rho = 0.7$



(c) different ρ

Fig. 4 Comparisons in Symmetric K-L Divergence

Figure 3 shows that *Mine* is uniformly better among the four techniques. It suppresses only about 26% items in POS and WV and about 35% items in Retail, while the other three techniques incur on average 10% more losses than *Mine* and up to 75% losses in the worst case. We notice that *Dist* has a generally worse performance than other techniques even though it tries to minimize the information loss at each iteration. The reason is that it also tries to retain the data distribution. Further, we argue that for applications that require data statistics, the distribution, that is, the summary information, is more useful than the details, hence losing some detailed information is acceptable. Note that Global and TDControl failed to complete in some datasets, because these methods don't scale very well.

Figure 4 shows that *Dist* outperforms the peers as its output has the highest resemblance to the original datasets. On the contrary, TDControl is the worst performer since generalization algorithm creates a lot of new items while suppressing too many item types globally. Since the symmetric relative entropy of *Dist* is very small, y-axis is in logrithmic scale to improve visibility. Therefore, the actual difference in K-L divergence is two to three orders of magnitude. The most common criticism of partial suppression is that it changes the support of good rules in the data and introduces spurious rules in rule mining. In this experiment, we test the algorithms on data sets with the max record length=5 (cutoff=5), and check the rules mined from the anonymized data with minimum support rate equal to 0.05% [5]. Figure 5(a),5(b) gives the performance of different anonymization methods. Both TDControl and Global perform badly in this category, with negligible number of original rules remaining after anonymization. Conversely, all of the partial suppression algorithms manage to retain most of the rules and the Jaccard Similarity reaches 80% in some datasets which shows our heuristic works very well. Specifically, *Mine* performs the best among partial algorithms.

The rules generated from TDControl are all in general form which is totally different from the original one. To enable comparison, we specialize the more general rules from the result of TDControl into rules of original level of abstraction in the generalization hierarchy. For example, we can specialize a rule {dairy product $\rightarrow$ grains} into: milk $\rightarrow$ wheat, milk $\rightarrow$ rice, yogurt $\rightarrow$ wheat, etc. Take WV as an example, there are 4 rules left in the result of TDControl when the ρ is 0.7 and the number becomes 28673 after specialization, which makes the results almost invisibly short on the figures.

Figure 5(c) shows the trend of Jaccard Similarity when ρ varies using partial suppression. We present the results of datasets POS(cutoff=5) and WV(cutoff=5) in the figure. Datasets Retail(cutoff=5) and Syn(cutoff=5) fail to complete within 2 hours, so we leave them out. We can see that Jaccard Similarity generally increases as ρ grows larger. But when ρ is relatively large, the Jaccard Similarity curve exhibits some anomaly. It is because the size of ruleset is small when ρ is relatively large, and Jaccard Similarity does not work very well to assess the similarity of two very small sets. The differences in datasets will have a significant impact on Jaccard Similarity when ρ is relatively large. For example, when $\rho = 0.9$, there is one rule left in the rulesets mined from original data and suppressed data by *POS-dist*, and these two rules happen to be the same. Consequently the Jaccard Similarity is as high as 1.

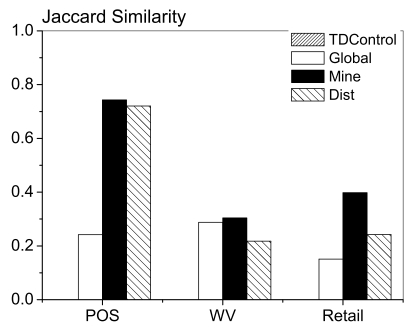
4.3 Performance

Next we evaluate the time performance and scalability of our algorithms.

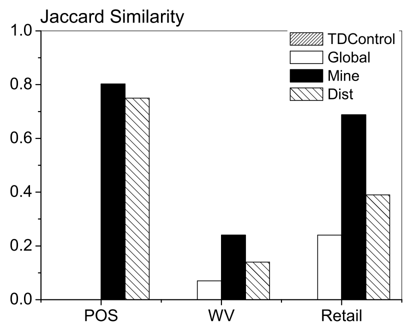
Table 3 Comparison in Time Performance ($\rho = 0.7$, $t_{max} = 300$)

Algorithm	POS	WV	Retail	Syn
TDControl	**183**	**30**	**156**	476
Global	1027	81	646	N/A
Dist	395	151	171	**130**
Mine	1554	478	256	132

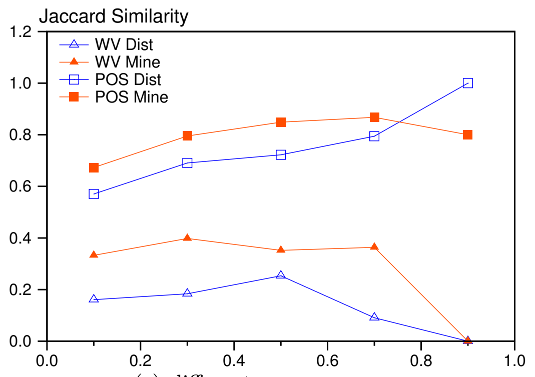
[5] We choose this support level just to reflect a practical scenario.



(a) $\rho = 0.3$



(b) $\rho = 0.7$

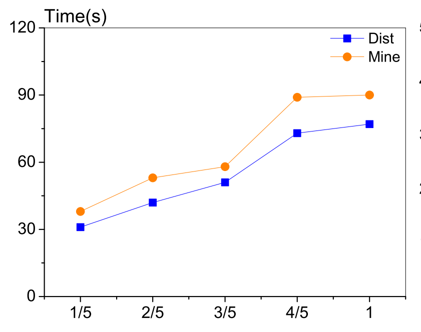


(c) different ρ

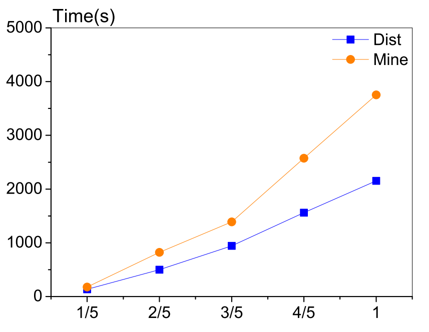
Fig. 5 Association Rules Mining with Support 0.05%

From Table 3, TDControl is the clear winner for two of the four datasets. *Mine* does not perform well in BMS-POS. The reason is that *Mine* incurs the least information loss among all the competiting methods. This means most of the original data remains unsuppressed. Given the large scale of BMS-POS, checking whether the dataset is safe in each iteration is therefore more time consuming than other methods or in other datasets. Results for Global are not available for Syn because it runs out of memory.

Next experiment illustrates the scalability of our algorithm w.r.t. data size. We choose *Retail* as our target dataset here because *Retail* has the maximum average record length of 10.6. We run partial algorithms on 1/5, 2/5 through 5/5 of *Retail* respectively. Figure 6 shows the time cost of our algorithm increases reasonably with the input data size with or without DnC. Furthermore, increased level of partitioning causes the algorithm to witness superlinear speedup in Figure 6(a). In particular, the dataset is automatically divided into 4, 8, 16 and 32 parts at 1/5, 2/5, 3/5 and the whole of the data, respectively.

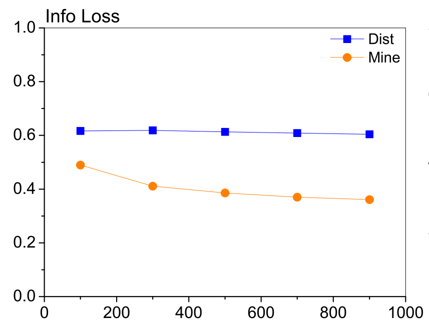


(a) With DnC

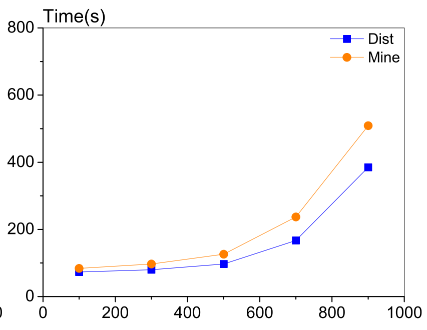


(b) Without DnC

Fig. 6 Scale-up with Input Data ($\rho = 0.7$)



(a) Information Loss



(b) Time Performance

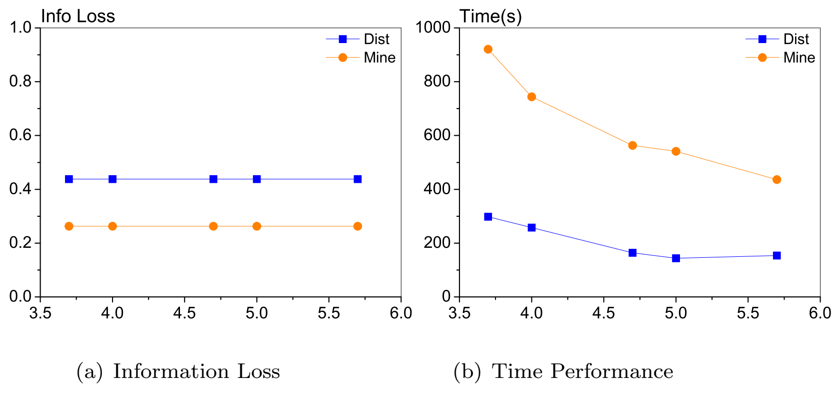
Fig. 7 Variation of t_{max} ($\rho = 0.7$)

4.4 Effects of Parameters on Performance

In this section, we study the effects of t_{max}, b_{max} on the quality of solution (in terms of information loss) and time performance.

4.4.1 Variation of t_{max}

We choose *Retail* as the target dataset again since *Retail* is the most time-consuming dataset that can terminate within acceptable time without DnC strategy. The value of t_{max} determines the size of a partition in DnC. Here, we evaluate how partitioning helps with time performance and its possible effects on suppression quality. Figure 7(a) shows the relationship between partitions and information loss. The lines of *Dist* is flat, indicating that increasing t_{max} doesn't cost us the quality of the solution. $Mine$ shows a slight descending tendency at first and then tends to be flat. We argue that a reasonable t_{max}



(a) Information Loss (b) Time Performance

Fig. 8 Variation of Buffer Size b_{max} ($\rho = 0.7$)

will not cause our result quality to deteriorate. On the other hand, Figure 7(b) shows that time cost increases dramatically with the increase of t_{max}. The reason is that partitioning decreases the cost of enumerating qids which is the most time-consuming part in our algorithm. Moreover, parallel processing is also a major reason for the acceleration.

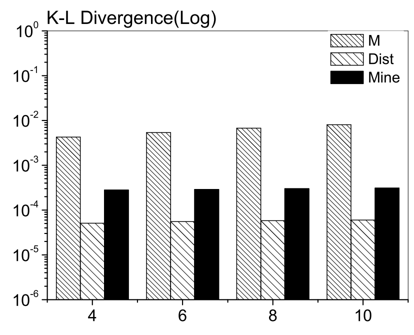
4.4.2 Variation of b_{max}

Next experiment (See Figure 8) illustrates the impact of varying b_{max} on performance. We choose WV as our target dataset this time since the number of distinct qids in WV is relatively small compared to other datasets and our algorithm can terminate even when we set a small b_{max}.

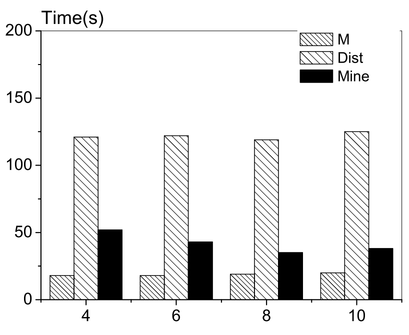
Note first that varying b_{max} has no effect on the information loss which indicates that this parameter is purely for performance tuning. At lower values, increasing b_{max} gives almost exponential savings in running time. But as b_{max} reaches a certain point, the speedup saturates, which suggests that given the fixed size of the data, when B is large enough to accommodate all qids at once after some iterations, further increase in b_{max} is not useful. The line for $Mine$ hasn't saturdated because $Mine$ suppresses fewer items and retains more qids, hence requires a much larger buffer.

4.5 A Comparison to Permutation Method

In this section, we compare our algorithms with a permutation method [**?**] which we call M. The privacy model of M states that the probability of associating any transaction $R \in T$ with any sensitive item $e \in D_S(T)$ is below $1/p$, where p is known as a privacy degree. This model is similar to ours when $\rho = 1/p$, which allows us to compare three variants of our algorithm against M where $p = 4, 6, 8, 10$ on dataset WV which was reported in [**?**]. Figure 9(a)



(a) K-L Divergence



(b) Time Performance

Fig. 9 Comparison with Permutation

shows the result on K-L divergence. All variants of our algorithm outperform M in preserving the data distribution. Figure 9(b) shows timing results. Even though M is faster, our algorithms terminate within acceptable time.

5 Related Work

This paper is an extended version of the best paper award winning work published at DASFAA 2014 [?]. In this paper, we formalize the problem more rigorously, give more details of the algorithm and prove a few of its interesting properties. Furthermore, additional experiments as well as in-depths analysis of the experimental results are added to this extended version. In this section, we will discuss a range of related work. We first present a number of well-known privacy models, then compare and contrast several general anonymization techniques, and finally show how our anonymizaton method can protect several kinds of unsual attacks.

5.1 Privacy Models

Privacy-preserving data publishing of relational tables has been well studied in the past decade since the original proposal of k-anonymity by Sweeney *et al.* [?]. Recently, privacy protection of set-valued data has received increasing interest. The original set-valued data privacy problem was defined in the context of association rule hiding [?,?,?], in which the data publisher wishes to "sanitize" the set-valued data (or *micro-data*) so that all sensitive or "bad" associate rules cannot be discovered while all (or most) "good" rules remain in the published data. Subsequently, a number of privacy models including (h, k, p)-coherence [?], k^m-anonymity [?], k-anonymity [?] and ρ-uncertainty [?] have been proposed. k^m-anonymity and k-anonymity are carried over directly from relational data privacy, while (h, k, p)-coherence and ρ-uncertainty protect the privacy by bounding the confidence and the support of any sensitive association rule inferrable from the data. This is also the privacy model this paper adopts.

5.1.1 The k-anonymity Model

Many datasets are published simply with key identifiers (e.g. name and social-security number) removed so the records are not related to specific people. However, some pseudo-identifiers (e.g. age and zip-code) can be combined to narrow down to or even identify a small number of individuals. In order to prevent identification, the k-anonymity model requires that every such combination in the dataset occurs at least k times so that every record is indistinguishable from at least $k-1$ other records.

5.1.2 The l-diversity Model

Kifer *et al.* [**?**] showed using two simple attacks that a k-anonymized dataset has some subtle but severe privacy problems, and proposed a novel and powerful privacy criterion called l-diversity that can defend against such attacks.

While k-anonymity is effective in preventing identification of a record, l-diversity focuses on maintaining the diversity of the sensitive attributes [**?**]. Therefore, the l-diversity model is defined as follows:

Definition 3 Let a q^*-block be a set of tuples such that its non-sensitive values generalize to q^*. A q^*-block is l-diverse if it contains l "well-represented" values for the sensitive attribute S. A table is l-diverse, if every q^*-block in it is l-diverse.

A number of different instantiations for this definition are discussed in [**?**], where the term "well-represented" is attached with different meanings.

5.1.3 The (h, k, p)-coherence Model

The (h, k, p)-coherence model by Xu *et al.* [**?**] requires that the attacker's prior knowledge to be no more than p public (non-sensitive) items, and any inferrable rule must be supported by at least k records while the confidence of such rules is at most $h\%$. They believe private items are essential for research and therefore only remove public items to satisfy the privacy model. They developed an efficient greedy algorithm using global suppression. In this paper, we do not restrict the size or the type of the background knowledge, and we use a partial suppression technique to achieve less information loss and also better retain the original data distribution.

5.1.4 The ρ-uncertainty Model

Cao *et al.* [?] proposed a similar ρ-uncertainty model which is used in this paper. They developed a global suppression method and a top-down generalization-driven global suppression method (known as TDControl) to eliminate all sensitive inferences with confidence above a threshold ρ. Their methods suffer from same woes discussed earlier for generalization and global suppression. Furthermore, TDControl assumes that data exhibits some monotonic property under a generalization hierarchy. This assumption is questionable. Experiments show that our algorithm significantly outperforms the two methods in preserving data distribution and useful inference rules, and in minimizing information losses.

5.1.5 Differential Privacy

Differential privacy [?] is targeted at a statistical database. A statistic is a quantity computed from a sample. The goal of differential privacy is to release statistical information without compromising the privacy of the individual respondents. It ensures that the removal or addition of a single database item does not (substantially) affect the outcome of any analysis. It follows that no risk is incurred by joining the database, providing a mathematically rigorous means of coping with the fact that distributional information may be divulged. However, in the scenario of a statistical database, users do not have exclusive access to all of the data set, but can only gain some statistics of the required data. Such mechanism is then less efficient and harder to deploy in real world than where the users can load the whole data set into memory and do arbitrary computations. Anonymization techniques of differential privacy add appropriately chosen random noise to produce response to the queries, which will lead to spurious rules when doing data mining.

5.2 Anonymization Techniques

A number of anonymization techniques were developed for these models. These generally fall in four categories[?]: *global/local generalization* [?,?,?], *global suppression* [?,?], *permutation* [?] and *perturbation* [?,?,?,?]. Next we briefly discuss the pros and cons of these anonymization techniques.

5.2.1 Generalization

Generalization replaces a specific value by a generalized value, e.g., "beer" by "drink", according to a generalization hierarchy [?]. Table 4 illustrates generalization by reusing the same dataset in Table 1, where both "beer" and "coffee" are generalized to "drink", according to some generalization hierarchy. While generalization preserves the correctness of the data, it compromises accuracy and preciseness. Worse still, association rule mining is impossible unless the data users have access to the same generalization taxonomy and they agree to the target level of generalization. For instance, if the users don't intend to mine rules involving "drink", then all generalizations to "drink" are useless.

Table 4 The Same Dataset in Table 1 and Generalization Anonymization Result

(a) Original Dataset

ID	**Transaction**
1	bread, **beer**, *condom*
2	**coffee**, fruits
3	**beer**, *condom*
4	**coffee**, fruits
5	flour, *condom*
6	bread, **coffee**
7	fruits, *condom*

(b) Generalization Result

ID	**Transaction**
1	bread, **drink**, *condom*
2	**drink**, fruits
3	**drink**, *condom*
4	**drink**, fruits
5	flour, *condom*
6	bread, **drink**
7	fruits, *condom*

5.2.2 Global Suppression

Global suppression is a technique that deletes all instances of some items so that the resulting dataset is safe. The advantage is that it preserves the support of existing rules that don't involve deleted items and hence retains these rules [**?**], and also it doesn't introduce additional/spurious association rules. The obvious disadvantage is that it can cause unnecessary information loss. In the past, partial suppression has not been attempted mainly due to its perceived side effects of changing the support of inference rules in the original data [**?**,**?**,**?**,**?**]. But our work shows that partial suppression introduces limited amount of new rules while preserving many more original ones than global suppression. Furthermore, it preserves the data distribution much better than other competing methods.

5.2.3 Permutation

Permutation was introduced by Xiao *et al.* [**?**] for relational data and was extended by Ghinita *et al.* [**?**] for transactional data. Ghinita *et al.* propose two novel anonymization techniques for sparse high-dimensional data by introducing two representations for transactional data. However the limitation is that the quasi-identifier is restricted to contain only *non-sensitive items*, which means they only consider associations between quasi-identifier and sensitive items, and not *among* sensitive items. Manolis *et al.* [**?**] introduced "disassociation" which also severs the links between values attributed to the same entity but does not set a clear distinction between sensitive and non-sensitive attributes. In this paper, we consider all kinds of associations and try best to retain them.

5.2.4 Perturbation

Perturbation is developed for statistical disclosure control [?]. Common perturbation methods include *additive noise*, *data swapping*, and *synthetic data generation*. Their common criticism is that they damage the data integrity by adding noises and spurious values, which makes the results of downstream analysis unreliable. Perturbation, however, is useful in non-deterministic privacy model such as differential privacy [?], as attempted by Chen *et al.* [?] in a probabilistic top-down partitioning algorithm based on a context-free taxonomy.

5.3 Adversarial Attacks

It is straight-forward to see that the anonymized data under our algorithm is immune from the *record linking attack* [?]. Furthermore, our technique can protect transactional data from *minimality attacks* [?] and *composition attacks* [?].

The minimality attack [?] is proposed for relational data. Assume an adversary knows the whole original quasi-identifier values as external data, also knows the privacy model and anonymization technique, by comparing the generalized version of quasi-identifier values with the original quasi-identifier values, the adversary can successfully predict some privacy. The minimality attack relies on the generalization anonymization technique, while our method uses suppression technique. Also for set-valued data it doesn't have fixed combination of items as quasi-identifiers, so it's unrealistic for an adversary to obtain the satisfactory external data.

The composition attack [?] is proposed for relational data by using the overlap population of multiple organizations' independent release of anonymized data, through intersection the privacy can still be breached in relational data. For example l-diversity [?] model can be violated by composition attack. The reason why composition attack can succeed is that quasi-identifier attribute values are generalized and sensitive attribute values are retained, when performing intersection the probability of a correlation between quasi-identifier and sensitive values will definitely increase. On the contrary, our partial suppression algorithm anonymizes set-valued data by randomly suppressing some sensitive items. In the same way to perform intersection, the probability that a sensitive item correlated with quasi-identifier items can both be higher or lower than before, which makes the composition attack not plausible. So in a summary our partial suppression technique is ideal to avoid composition attack depending on the randomized characteristic of suppression.

5.4 Improvements of the previous version

This paper is a improved version of our previous one[?]. In the last version, we stated our fundamental research purpose and the basic algorithms. However, there are some problems in that version, such as some vague definitions and inadequate experiments. In this version, we have proposed some specific explanations of the methods we use.

Firstly, we have proved some formulas from the definition in Section 3. Those proofs can show the validity of our algorithms. The significance of Regression is also stated to guarantee the correctness of the suppression. Particularly, we have added plenty of analysis of our algorithms in Section 3.4 in order to make our algorithms more pellucid. This part clearly presents why PartialSuppressor can successfully terminate with a correct solution, which outputs safe transaction records. Also it shows the reason why we use divide-and-conquer to speed up the process.

Secondly, we have clearly defined the evaluation, such as *info loss*, to make the experiments easier to understand. In section 4.1, Figure 1 is introduced to show the difference of data sets and the distribution of each one and we introduced some analysis to help better explain the results of different data sets. We reorganized the statement of experiments on variation of parameters such as ρ, t_{max} and b_{max} in following sections. This change makes it better to understand the functions of different variables.

Finally, we have discussed some privacy models this paper adopts. In this part, previous models, including their advantages and disadvantages are introduced. We also include the explanation of some anonymization techniques in this part to analysis the pros and cons of them. At last, some basic adversarial attacks are presented, in order to show the success of our partial suppression technique.

6 Conclusion

In this paper, we proposed a partial suppression framework for ρ-uncertainty privacy protection. Previous approaches such as global suppression generally delete more items to satisfy the same privacy model. We also developed two heuristics which produce anonymized data that is highly useful to data analytics applications. We compared the two heuristics with state-of-the-art anonymization techniques in different aspects such as information loss, data distribution preservation, and useful rule preservation, and the experiments showed that

- the first heuristic can effectively limit suprious rules while maximally preserving the useful rules mineable from the original data, and
- the second heuristic which minimizes the K-L divergence between the anonymized data and the original data helps preserve the data distribution, which is a feature largely ignored by the privacy community in the past.

We also studied empirically the effects of partial suppression parameters on the quality of solution (in terms of information loss) and time performance. This can be used as a general guidance of how to apply the presented framework to other kinds of data sets that are not covered in the experiments. Finally the divide-and-conquer strategy that we proposed in section 3.3 effectively controls the execution time with limited compromise in the solution quality.