Hiding Emerging Pattern with Local Recoding Generalization

Michael Wai-Kit Cheng

Abstract

*When data is published to public, it is vastly preferable to publish meaningful data and yet protect embedded sensitive patterns. This process is often referred to as privacy preserving data publishing (*PPDP*). In this paper, we investigate* PPDP *in the context of frequent itemsets mining – one of the fundamental data-mining concepts – and emerging patterns (*EP*s) – patterns that have a high classification power. More specifically, we want to hide all* EP*s embedded in data while publishing the data with a minimal distortion on its frequent itemsets. We propose a novel heuristic solution that captures the unique properties of hidding* EP*s and incorporates with local recoding generalization. To guide a bottom-up recoding generalization, a metric is derived based on (i) frequent-itemset distortion that quantifies the quality of published data and (ii) the degree of reduction in emerging patterns in each recoding generalization. We have implemented our proposed solution and experimentally verified its effectiveness using two benchmark datasets.*

1 Introduction

Organizations often publish or share their data to support subsequent data analysis for various reasons, for instance, advancing medical research [27] or raising customer satisfaction level [32]. In many cases, data may involve personal information or company's secrets which are not supposed to be disclosed in normal circumstances. Protecting the published data against privacy breaching is often of top priority. Nonetheless, people may still choose to relinquish certain privacy for a greater good, but only when they can be assured that the published data will be properly sanitized.

Studies on data sanitization can be dated back to the earlier work on statistical disclosure control [1]. For instance, it is a common requirement to remove personal identifiers from published data using mechanisms like masking (e.g., ID="123456(7)" $\rightarrow$ ID="123***(*)") and substitution (e.g., name="Donald" $\rightarrow$ name="YxfG1L") to ensure anonymization. However, the privacy-breaching challenge goes beyond that. It has been clear that combinations of quasi-identifiers, e.g., the demographic attributes, can allow sensitive information such as personal identity, facts, or forbidden patterns to be inferred, and thus disclosed [34]. Given that data publishing is still a must in many situations, there has been a stream of work called *privacy-preserving data publishing* (PPDP), emerged in recent years to flight against the privacy-breaching challenge [13]. The particular type of PPDP considered in this paper tries to sanitize the data to be published so that some specified sensitive patterns could not be discovered and meanwhile, certain statistical properties of the published data are preserved as far as possible to support subsequent analysis.

In this paper, we argue that emerging patterns embedded in data can carry important intelligence (e.g., newly emerging sales opportunities revealed in sales records) that data owners may want to hide before they publish them for further analysis. In particular, emerging patterns has known to have a high classification power where EP-based classifiers can often be more accurate than classifiers with `C5.0`, naive Bayes and CAEP. At the meantime, frequent itemset mining is a classical data mining technique and has already been well-supported by commercial data mining software. Published data may be evidently subjected to such an analysis. Therefore, we study a novel form of PPDP where the sensitive information to be protected is emerging patterns in datasets and the data analysis is frequent itemset mining.

More specifically, given two transactional datasets D_1 and D_2, a threshold of growth rate ρ and a threshold of support σ, we want to determine a sanitization operation G on both D_1 and D_2 such that the emerging patterns with a growth rate higher than ρ are eliminated and the distortion of σ-frequent itemsets is minimized. (The details of the problem formulation shall be given in Section 3.)

Example 1.1: Let us consider an example from a demographic dataset [36], which contains some census information of the United States. We divide the dataset into two: (i) people who earned more than \$50/year and (ii) people who do not. When we set ρ to be 35, we find that there are 8 emerging patterns. One emerging pattern in (ii) is (`never-married`, `own-child`). In other words, in [36], there are at least 35 times more people who have never married and have their own child in (ii) than (i). We remark that (`never-married`, `own-child`) is not a frequent itemset of the entire dataset when σ = 0.4.

The example shows that emerging patterns may reveal certain sensitive information from data. For applications like publishing census reports, sanitizing data might not be legitimate. But for applications related to business intelligence, protecting customer data and cooperations' interests become a necessity. We speculate that in many other scenarios, one may prefer to sanitize such sensitive information embedded in data before publishing. ■

Different data sanitization approaches have been proposed in the literature for PPDP [12] (as well as other privacy preservation related fields), which could be roughly categorized into recoding generalization [20, 5, 15, 19, 34, 28, 40, 35, 31] and perturbation [2, 8, 10, 18]. In this paper, recoding generalization is adopted. Recoding generalization has been extensively used in the context of k-anonymization which was originally proposed in [34]. Recoding generalization is (intuitively) a value-grouping process according to a given attribute generalization hierarchy. Since the attribute hierarchy is always published with the data, data recipients can correctly interpret every generalized values that appear in the sanitized data. Therefore, the sanitized data can be "blurred" but is not lost. This work focuses on exploring the possibilities of adopting recoding generalization in the context of hiding emerging patterns.

A sanitization is obviously unacceptable if a data mining technique would either (i) reveal sensitive information or (ii) produce a highly distorted result from the sanitized data. These two competing objectives make the problem technically intriguing.

We remark that hiding emerging patterns is more technically challenging than some related works on hiding frequent itemsets [30, 2]. The reason is that the *apriori anti-monotone* property of frequent itemsets does not hold for emerging patterns. Thus, the search space of emerging patterns cannot be pruned as effectively as that of frequent itemsets. Furthermore, recoding generalization may hide an emerging pattern as well as generating new emerging patterns. To the best of our knowledge, there has not been work on adopting recoding generalization techniques to hide emerging patterns.

The paper is organized as follows. In Section 2, we give a brief summary on the research on PPDP. In Section 3, we present the background and notations used and give the formal problem statement of this paper. In Section 4, we discuss the recoding generalization technique adopted to solve this problem. Section 5 gives an overview of our algorithm. Section 6 derives the heuristics that quantifies the quality of a recoding generalization. Section 7 discusses a number of optimizations related to solving this PPDP problem. In Section 8, we present an experimental evaluation that verifies the effectiveness of our proposed algorithm. Section 9 concludes this work and discusses future works.

2 Related Work

There have been recent research efforts on various aspects of PPDP. In this section, we highlight some relevant, admittedly non-exhaustive PPDP techniques.

Anonymity measure and recoding generalization. In an analysis of a demographic dataset with external information [35], it has been shown that personal identity can be recovered from a published dataset, even the identifiers have been removed. The notion of k-anonymity [35] was thus introduced. A published data is said to be k-anonymized if at least k individuals are linked with a particular record in the published data even if the data may be cross-referenced with external information. Each group of items with indistinguishable attribute values is called an equivalence class. To further improve k-anonymity, ℓ-diversity [25] was proposed to require also each sensitive value in each equivalence class to appear at most $1/\ell$ times.

Given a particular measure, recoding generalization [34, 40, 23] is commonly adopted to achieve anonymization. Recoding generalization has two advantages over the perturbation approach for data sanitization,. First, the dataset sanitized by recoding generalization is still semantically consistent with the original one. The information conveyed by a generalized dataset always contains the truth, even though it is "blurred". In contrast, perturbation techniques may generate fake information. These may not be acceptable in many real applications, e.g., a database with patient information. Second, since the attribute hierarchy is often published with the dataset, the generalized values in the published dataset can be always interpreted. We adopt recoding generalization, to protect emerging patterns.

The notion of information preservation. There has been a stream of work on k-anonymity which also makes an attempt to preserve as much information of the original dataset as possible [5, 16, 14]. For example, [5, 16] considers the preservation of classification accuracy during the sanitization process. [14] investigates the preservation of cluster structures of the data. In addition, there has been some recent work studying the trade-off effect between privacy and utility [24] in the context of PPDP. However, they are not designed to tackle some particular patterns.

Frequent itemset hiding. There has been previous work [30, 33, 29] on hiding frequent itemsets for PPDP. In particular, users specify a subset of frequent itemsets, namely sensitive frequent itemsets, that are not supposed to be disclosed to public. The main objective of this related work is to sanitize the data such that the sensitive frequent itemsets are removed from the sanitized data while non-sensitive frequent itemsets are retained as many as possible. In our study, we focus on hiding emerging patterns, which makes a different contribution to PPDP.

We remark that the previous work on frequent-itemset hiding used techniques like removing sensitive data, introducing random and unknown data. Recoding generalization has not been adopted in this stream of work.

Emerging patterns. Intuitively, EPs are some distinctive features from one class to the others. Much previous effort [17, 38, 11, 7] has been spent on using EPs for building classifiers. The results of these efforts have shown that the accuracy of EP-based classifiers can often be higher than those of traditional classifiers, such as, C5.0 and naive Bayes. Due to the high classification power, EP-based classifiers have been used for predicting the likelihood of diseases such as acute lymphoblastic leukemia [21] and discovering knowledge in gene expression data [22].

Although EPs have been found useful in classification, there have not been studies on EPs with PPDP. Previous work on EPs focuses on the mining efficiency. Mining EPs has been technically intriguing. First, the total number of EPs in large datasets can be huge. In the worst case, the total number of EPs is exponential to the total number of attributes in transactions. Second, there has not been a corresponding notion of the apriori anti-monotone property of frequent itemsets in EPs. As such, the search space pruning strategies of frequent-itemset mining cannot be adopted to EP mining. There have been a border-based approach [4], constraint-based approach [41] and jumping emerging patterns [3] to improve the efficiency of EP mining.

3 Background and Problem Statement

In the following, we present the definitions, notations used and the problem statement of the paper.

Let D be a transactional dataset and $I := \{i_1, i_2, ..., i_n\}$ be a finite set of distinct items in D. A transaction t may have a set of nominal attributes A and each attribute takes values from a set $V_i \subseteq I$, where the domain $\mathcal{D}$ is $V_1 \times V_2 \times ... \times V_{|A|}$. We make two simple remarks about these notations. (i) While we assume transactional data with nominal attributes, data of a continuous domain can be casted into nominal data, e.g., by defining ranges. (ii) One may consider a relation as a set of transactions of a fixed arity.

$Supp_D(X)$ denotes the support of an itemset $X \subseteq I$ in a dataset D, which can be computed as $\frac{|\{t \mid X \in t \wedge t \in D\}|}{|D|}$. Given a support threshold σ, X is said to be a *σ-frequent itemset* if $Supp_D(X) \geq \sigma$. The growth rate of an itemset is the ratio of its support in one dataset to that in the other.

Definition 3.1: [6] Given two datasets, namely D_1 and D_2, the *growth rate* of an itemset X, denoted as $GR(X, D_1, D_2)$, from D_1 to D_2 is defined as $GR(X, D_1, D_2)$ =

$$\begin{cases} 0 & \text{, if } Supp_{D_1} = 0 \text{ and } Supp_{D_2} = 0 \\ \infty & \text{, if } Supp_{D_1} = 0 \text{ and } Supp_{D_2} > 0 \\ \frac{Supp_{D_2}(X)}{Supp_{D_1}(X)} & \text{, otherwise.} \end{cases}$$ ■

Intuitively, given two datasets, emerging patterns (EPs) [6] are the itemsets whose support increases significantly from one dataset to another. The formal definition of EPs can be given as follows.

Definition 3.2: [6] Given a growth rate threshold ρ and two datasets $D_1$and D_2, an itemset X is said to be a *ρ-emerging pattern* (ρ-EP) from D_1 to D_2 if $GR(X, D_1, D_2) \geq \rho$. ■

An emerging pattern with a growth rate ∞ (i.e. itemset that appears in one dataset but not the other) is called a *jumping emerging pattern*.

For ease of presentation, we may skip σ, ρ, D_1 and D_2 of EPs when they are not essential to our discussions.

Example 3.1: Table 1 shows a hypothetical dataset D of Adult dataset [36]. More description of the dataset can be found in Section 8. We opt to present some nominal attributes of Adult for discussions. Each record (or transaction) represents a person. Let us consider two subsets of D: D_1 contains the people who are married and D_2 contains the people who never marry before. From Table 1, we find the following patterns, among many others.

- The pattern (MSE, manager) has a support 75% in D_1 and 20% in D_2. Therefore, the growth rate of (MSE, manager) from D_1 to D_2 is 3.75. When we set ρ to 3, (MSE, manager) is a ρ-emerging pattern in D_2. Obviously, when we set ρ to 4, (MSE, manager) is not a ρ-emerging pattern.
- High-school graduate (HS) has a support 0% in D_1 but 20% in D_2. Hence, its grwoth rate from D_2 to D_1 is infinite. (HS) is a jumping emerging pattern in D_1.
- Suppose that the threshold support for frequent itemset σ is set to 50%. Neither (MSE, manager) nor (HS) is a σ-frequent itemset of D. In comparison, (manager) is a σ-frequent itemset, whose support is 77.8%. ■

Using the above notations, we state the formal problem statement of this paper below (visualized in Figure 3).

Problem statement. Given two datasets (D_1, D_2), σ and ρ, we want to sanitize (D_1, D_2) to (D'_1, D'_2) such that no ρ-EPs can be mined from (D'_1, D'_2) while the distortion between σ-frequent itemsets of (D_1, D_2) and those of (D'_1, D'_2) is minimized. ■

4 Global vs Local Recoding Generalization

Our proposed algorithm for hiding emerging pattern is derived from local recoding generalization. In this section,

Table 1. A hypothetical subset of `Adult`

ID	Edu.	Martial	Occup.	Rel.	Race	Sex
1	BA	married	executive	wife	black	F
2	MSE	married	manager	husband	black	M
3	MSE	married	manager	wife	white	F
4	MSE	married	manager	husband	black	M
5	BA	never	manager	NA	white	M
6	MSE	never	manager	NA	white	F
7	HS	never	repair	NA	black	M
8	BA	never	manager	NA	white	M
9	BA	never	manager	NA	black	F

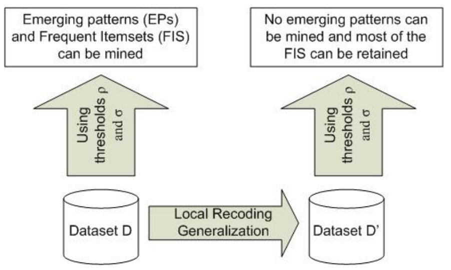

Emerging patterns (EPs) and Frequent Itemsets (FIS) can be mined
No emerging patterns can be mined and most of the FIS can be retained
Using thresholds ρ and σ
Using thresholds ρ and σ
Dataset D
Local Recoding Generalization
Dataset D'

Figure 1. The problem statement illustration

we give a brief overview on global and local recoding generalization, or *recoding* for simplicity, and its variations.

Recoding. In privacy-preserving data mining, recoding has been a popular technique for sanitizing datasets in order to achieve anonymization. The key idea of recoding is to modify a value into a more general value such that more tuples will share the same values and cannot be distinguished individually. Thus, anonymization can be achieved. In this work, we recode emerging patterns with some non-emerging values. As a result, the corresponding patterns become less emerging after recoding. Such a recoding process is repeated until the desired anonymization is reached.

Variations of recoding. In the following, we describe the there types of recoding. In this work, we adopt the last recoding technique.

1. Single-dimensional global recoding. Single-dimensional global recoding has been studied in [5, 15, 19, 34]. It performs recoding on the domain of an attribute in a dataset. It recodes a value of the domain to another (generalized) value. That is, if a particular value is recoded, the attribute of all the tuples containing that particular value will be recoded. It is evident that datasets are very often over-generalized by this recoding.

2. Multidimensional global recoding. Multidimensional global recoding was first proposed in [20]. It generalizes data at "cell" level. This recoding relies on the notion of *equivalence classes*. An equivalence class of attributes A is a set of tuples T where $\pi_A(T)$ is a singleton. That is, the tuples in T has the same value in attributes A. In multidimensional global recoding, the tuples in an equivalence class (of a set of attributes) are generalized together. By doing so, both original and generalized values may exist in the recoded dataset. When compared to the previous global recoding, multidimensional global recoding generalizes data in a finer granularity. Hence, this recoding over-generalizes data less often.

Example 4.1: Let us revisit the dataset in Table 1 and the emerging pattern (`MSE`, `manager`). The pattern is related to the attributes of education background (`Edu.`) and occupation (`Occup.`). Regarding (`Edu.`, `Occup.`), the equivalence classes in D_2 are $\{\{5, 8, 9\}, \{6\}, \{7\}\}$, where the numbers are the IDs. In multidimensional global recoding, we may recode the `Edu.` attribute of $\{2, 3, 4\}$ and $\{5, 8, 9\}$. For instance, we may recode `BA` and `MSE` into degree holder `Deg.` After such a recoding, all `BA`, `MSE` and `Deg.` appear in the recoded dataset. We remark that (`Deg.`, `manager`) is not an emerging pattern since its growth rate is 75%/60% = 1.25. ∎

3. Multidimensional local recoding. Local recoding has been studied in [9, 28, 40, 23] in the context of k-anonymity. It performs generalization at cell level as well. It also relies on equivalence classes. Its difference between the multidimensional global recoding is that it does not require the entire equivalence class to be recoded, as long as anonymity has been achieved. For instance, we may recode $\{8, 9\}$, as opposed to $\{5, 8, 9\}$, with $\{2, 3, 4\}$ in Example 4.1. The growth rate of (`Deg.`, `manager`) in the recoded dataset is 75%/40% = 1.875. Hence, (`Deg.`, `manager`) is not ρ-emerging when $\rho = 3$.

In this paper, we focus on this recoding. In the subsequent discussions, we use the term *local recoding* to refer to multidimensional local recoding.

5 Overview on hiding emerging patterns

In this section, we present an overall algorithm `hide-eps` (shown in Figure 2) to hide emerging patterns with a minimal distortion in frequent itemsets. Its details shall be discussed shortly. Then, we provide some details of the heuristics in `hide-epse` in Section 6.

Overall algorithm. We give the main ideas of Figure 2. First, we find the frequent itemsets to be preserved (Line 03) and the emerging patterns to be hidden (Line 04). In Line 05, we select an emerging pattern to hide. Next, we carry out a local recoding `local-recoding` (Line 06). This process is repeated until there is no more emerging pattern to hide (Lines 07-08).

The main tasks of the local recoding `local-recoding` are to compute (i) the equivalence classes of an emerging pattern e to generalize (Line 03) and (ii) the corresponding utility gain (details in Section 6) to guide us to choose an equivalence class for local recoding (Line 06). To avoid be-

```
Procedure hide-eps
Input: two datasets, $D_i$ and $D_j$, the threshold of
  the growth rate and frequent itemsets $\rho$ and $\sigma$,
  the heuristic parameters $p$ and $q$,
  a temperature $t_0$ and the cooling parameter $\alpha$
Output: transformed datasets ($D_i$, $D_j$)
01 initialize a hashtable $H$ to optimize
       computing equiv. classes; $t = t_0$
02 do
03     $F$ := mine-fis($D_i \cup D_j$, $\sigma$)
04     $E$ := mine-eps($D_i$, $D_j$, $\rho$)
05     $e$ := next-overlapping-ep($E$)
06     if $e$ is not null
           ($D_i$, $D_j$) := local-recoding($D_i$, $D_j$, $e$, $F$, $t$, $\alpha$)
07     if $t > 0.01$ then $t = \alpha \times t$
08 while $E \neq \emptyset$
09 return ($D_i$, $D_j$)

Procedure local-recoding
Input: two datasets, $D_i$ and $D_j$, an emerging pattern $e$
  a frequent itemset $F$, a temperature $t_0$
Output: transformed datasets ($D_i$, $D_j$)
01 let $D_i$ be the dataset where $e$ has a higher support
02 $t := t_0$;    $u = 0$ //initialization
03 compute equiv. classes $C$ of $D_j$ of the attr. of $e$
04 //compute the heuristic function of local recoding
05 denote $c_e$ be the equiv. class of $e$ in $D_i$
06 for each $c_k$ in $C$
       $H[c_e][c_k]$ := util_gain($G_{(c_e,c_k)}$, $E$) //heap
       $r[c_k] = exp^{\frac{H[c_e][c_k]}{t}}$
       $sum = sum + r[c_k]$
07 find $k$ s.t. $r[c_k]/sum <$ rand(1)$< r[c_{k+1}]/sum$
       $D_i$ := recode($D_i$, $c_e$, $c_k$)
       $D_j$ := recode($D_j$, $c_e$, $c_k$)
08 return ($D_i$, $D_j$)
```

Figure 2. Overall algorithm

ing trapped at some local sub-optima, we present our algorithm in a simulated-annealing style search (Lines 06-08).

Next, we provide the details of the overall algorithm.

The effect of local recoding on frequent itemsets. We first remark that global recoding may only merge some frequent itemsets, shown in Figure 3 (a). Hence, the change in frequent itemsets due to global recoding can be determined easily. However, local recoding may lead to a non-trivial change in frequent itemsets, shown in Figure 3 (b). The reason is that by the definition of local recoding, two tuples, even in the same equivalence class, are not always recoded together. This leads to a few possible effects on the frequent itemsets. First, since some of the values of the attributes A have been recoded to a generalized value, a frequent itemset may not be frequent after a local recoding. Second, multiple frequent itemsets may be recoded into one generalized frequent itemset. Third, a frequent itemset may not be affected by a local recoding. The first case is only possible

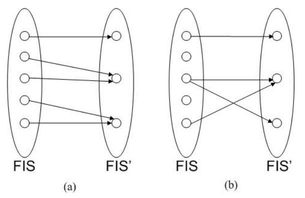


Figure 3. The relationship between F and F' in (a) global recoding (b) local recoding

with local recoding whereas the second and third cases may appear correspondingly in global recoding as well.

The above complication leads to Line 04 of hide-eps, which mines the current frequent itemsets in each iteration of local recoding.

The selection of emerging patterns for hiding. In the worst case, there are $O(|E| \times (|D_1| + |D_2|) \times \mathcal{D} \times \mathcal{H})$ possible local recodings on a set of emerging patterns E, where $\mathcal{D}$ is the total size of the domains of the attributes of E and $\mathcal{H}$ is the total height of the hierarchy of the attributes of E. In the worst case, local recoding allows tuple-wise recoding, which leads to the term $|D_1|$ + $|D_2|$. While there may be search algorithms to explore an optimal local recoding on E, we employ a simple and efficient solution to avoid exploring a large search space – we hide an emerging pattern, one-at-a-time (Line 05 of hide-eps, next-overlapping-ep), based on equivalence classes.

The pseudocode of next-overlapping-ep has been omitted, since it is straightforward but tedious. Here, we present its intuitions. Given a set of emerging patterns E, next-overlapping-ep determines the emerging pattern e in E such that it overlaps with the remaining emerging patterns the most. The intuition is that reducing the growth rate of e may indirectly reduce the growth rate of the overlapping emerging patterns as well. We verify this with some preliminary experiments that this approach consistently outperforms generalizing the most independent, the longest or the shortest emerging patterns first.

The search algorithm. The algorithm is presented in a simulated anneanling style in Figure 2. We have implemented a deterministic search to analyze our heuristics (Section 6).

6 Heuristics for Guiding Local Recoding

In this section, we present the formulation of the utility gain (util_gain) adopted to guide the local recoding process local-recoding, discussed in Figure 2. The main idea is that we consider a local recoding perferrable if it results in a small distortion in the frequent itemsets and at the same time it reduces the growth rate of an emerging pattern significantly. In the following two subsections 6.1 and 6.2, we present the details of the metrics for measuring (i) the distortion of frequent itemsets and (ii) the reduction in the growth rate of emerging patterns, respectively.

6.1 Metric for the distortion on frequent itemsets

Recoding is a process of grouping existing values to some new generalized values. After a recoding, some σ-frequent itemsets may not be frequent anymore and some may contain generalized values.

Example 6.1: Let us reconsider the recoding in Example 4.1. We set σ to be 4. Recoding `BA` and `MSE` into `Deg.` leads to a new frequent itemset (`Deg.`, `manager`), which contains both non-generalized and generalized values. (`MSE`, `manager`) is not frequent after the recoding. ∎

Therefore, we need a metric for measuring the distortion in the *σ-frequent itemsets* of D and D'. For presentation clarity, we will present our proposed metric for global recoding followed by its adaption for local recoding.

Distortion metric for single-dimensional global recoding. In global recoding, no frequent itemsets "disappear" after in a recoding (recall Figure 3 (a)). However, *some* frequent itemsets may appear in a generalized form. That is, a particular frequent itemset f in D will always have a corresponding itemset in the generalized frequent itemsets in D', in either the original or generalized form.

Inspired by the distortion metric proposed in [23], we propose a metric for measuring the *recoding distance* ($RDist$) between the original and generalized form of a tuple. Then, we define a metric called *value distance* (VD) which measures the distance between the original and generalized form of a single attribute value. We will use VD as a building block for the definition of distortion (RD). Since a recoding always assumes an attribute hierarchy, we may skip the hierarchy H when it is clear from the context.

Definition 6.1: *Recoding Distance (RDist):* Consider a recoding G which generalizes a set of attribute values V to a single generalized value v_g. The recoding distance of G is:

$$RDist(G) = |V|,$$

where $|V|$ is the number of leave nodes (non-generalized values) under v_g in the attribute hierarchy. ∎

Definition 6.2: *Value Distance (VD):* Let h be the height of the hierarchy of an attribute H, where level h is the most generalized level and level 0 is the most specific level. Consider a value v at level p is generalized to a value v' at level q. Let G_i denote the recoding which generalizes an attribute from level $i-1$ to i, where $0 < i \leq h$. The *value distance* between v and v' is:

$$VD(v, v') = \sum_{i=p}^{q} \frac{i \cdot RDist(G_i)}{h}.$$

∎

Value distance is mainly unfavorable to recoding (i) many values together into one single generalized value and (ii) a value into a generalized value that is close to the top of the hierarchy. This gives us a measure on the distortion of a value affected by a recoding.

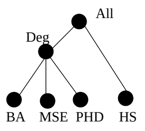


Figure 4. An attribute hierarchy of `Edu`.

Next, we extend VD to measure the distortion of a tuple due to recoding.

Definition 6.3: *Tuple Distance (TD):* Suppose a tuple $f = (v_1, v_2, \ldots, v_n)$ is generalized to $f' = (v'_1, v'_2, \ldots, v'_n)$. The tuple distance between f and f' is defined as:

$$TD(f, f') = \sum_{i=1}^{n} VD(v_i, v'_i).$$

∎

Finally, the distortion of between frequent itemsets due to a recoding is defined.

Definition 6.4: *Recoding Distortion (RD):* Let $F = \{f_1, f_2 \ldots f_n\}$ be a set of *σ-frequent itemsets* in D and $F' = \{f'_1, f'_2 \ldots f'_m\}$ be the set of *σ-frequent itemsets* in D', where $m \leq n$. The corresponding frequent itemset of f_i due to global recoding is denoted to be $f'_j = G(f_i)$. The recoding distance between F and F' is defined as:

$$RD(F, F') = \sum_{i=1}^{n} TD(f_i, G(f_i)).$$

∎

Example 6.2: Following up Example 4.1, we compute the (global) recoding distortion of generalizing (`MSE`, `manager`) to (`Deg.`, `manager`). Figure 6.1 shows the attribute hierarchy of `Edu`. The recoding distortion $RD(\{(\texttt{MSE}, \texttt{manager})\}, \{(\text{Deg.}, \texttt{manager})\})$, RD, can be computed as follows.

$$\begin{aligned} RD &= TD((\texttt{MSE}, \texttt{manager}), (\texttt{Deg.}, \texttt{manager})) \\ &= VD(\texttt{MSE}, \texttt{Deg.}) + VD(\texttt{manager}, \texttt{manager}) \\ &= \sum_{i=0}^{2} \frac{i \cdot RDist(G_i)}{h} = \frac{1 \times 3}{2} + \frac{2 \times 0}{2} = 1.5 \end{aligned}$$

∎

Distortion metric for local recoding. As discussed in Section 4, single-dimensional global recoding may often lead to over-generalization. Since we have adopted local recoding, we extend the recoding distortion to local recoding. Regarding local recoding, we remark that there are two unique challenges in calculating the recoding distance (see Figure 3 (b)).

(1) An itemset in F having no correspondence in F'. Local recoding allows part of the tuples that share the same attribute values to be generalized. Such recoding may generalize some supporting tuples of a frequent itemset which makes the itemset (in the original or generalized form) not frequent anymore. To address this, we measure the distortion of the disappeared frequent itemset to the most general

form. The reason behind this is that the frequent itemset can be trivially recovered when the entire dataset is generalized to the most general form.

Specifically, given a frequent itemset f in F, if we cannot find a corresponding frequent itemset in F', we may first create an artificial itemset, f_{max}, which generalizes each value in f to it most general form. Then, we can calculate the recoding distance (RD) between f and f_{max}.

Example 6.3: Reconsider the dataset in Table 1. Suppose we recode the `Edu.` attribute of Records 1 and 2 to `Deg`. When σ is 40%, {`BA`} and {`MSE`} were frequent itemsets (not minimal for illustration purpose) before recoding and there is no frequent itemset after recoding. ■

(2) An itemset in F having more than one corresponding itemsets in F'. As discussed, local recoding may generalize a frequent itemset f in F into more than one correspondences in F', denoted as F_f. In this case, we calculate the tuple distance of each of the corresponding itemsets in F_f and take the minimum tuple distance as the tuple distance of f. This is because the itemset with the minimum tuple distortion has been revealed in F' even there may be some other more distorted itemsets.

With the above considerations, we have the following recoding distance for local recoding:

Definition 6.5: *Recoding Distance for Local Recoding (RD_{local}):* Let $F = \{f_1, f_2 \ldots f_n\}$ be a set of *σ-frequent itemsets* in D and $F' = \{f'_1, f'_2 \ldots f'_m\}$ be the set of *σ-frequent itemsets* in D'. The corresponding frequent itemset(s) of f_i due to local recoding is denoted as $F_f = G(f_i)$. The recoding distance between F and F' is defined as:

$$RD_{local}(F, F') = \sum_{i=1}^{n} \frac{TD_{local}(f_i, G(f_i))}{TD_{local}(f_i, f_{max})}$$

where $TD_{local}(f_i, G(f_i))$ =

$$\begin{cases} \theta_q \times TD(f_i, G(f_i)), & \text{if } f \text{ has 1 correspondent in } F' \\ (1-\theta_q) \times TD(f_i, f_{max}), & \text{if } f \text{ has no correspondent in } F' \\ \theta_q \times min(TD(f_i, f_j)), & \\ \qquad where\ f_j \in G(f_i), & \text{otherwise,} \end{cases}$$

θ_q is a parameter that specifies the importance between the itemsets that are distorted and those disappeared due to G and $TD_{local}(f_i, f_{max})$ at the denominator is used to normalized RD_{local}. ■

Example 6.4: Following up the local recoding in Example 6.1, when σ is 30%, the frequent itemset {(`BA`)} corresponds to the frequent itemsets {(`BA`), (`Deg`)} in the recoded datasets. ■

6.2 Metric for the change in growth rate

The second component in our heuristics concerns the growth rate of the emerging patterns. Intuitively, we aim at a recoding that significantly reduces the growth rate of the emerging patterns, in order to hide them. Given an emerging pattern e and the result of a local recoding e', the reduction in growth rate due to the recoding can be easily defined as the growth rate of e minus the growth rate of e'. Then, the growth rate reduction of E due to a local recoding G, denoted as $RG_{local}(G, E)$, can be defined as the total reduction in growth rate of e in E divided by the total growth rate of e in E.

Putting all these together. Based on the derivations from Section 6.1 and Section 6.2, the utility gain due to a local recoding G on a set of emerging patterns E is defined as:

$$\texttt{util\_gain}(G, E) = \theta_p RG_{local}(G, E) - (1-\theta_p) RD_{local}(F, F')$$

This utility gain is used in our heuristic algorithm, presented in Figure 2. There are two parameters θ_p and θ_q specified by users.

It is worth mentioning that *jumping emerging patterns* are a special case in measuring the reduction in growth rate. Recall that jumping emerging patterns have an infinite growth rate. Hence, whenever a recoding that either hides jumping emerging patterns or converts them into non-jumping emerging patterns would yield an infinite reduction in growth rate. Our heuristic algorithm will tend to hide the jumping emerging patterns prior to the non-jumping ones.

7 Implementation Optimization

In this section, we discuss some implementation issues for optimizing the computation of the proposed algorithm.

1. The maintenance of the equivalence classes. Local recoding relies on equivalence classes (Lines 03, 06 and 07 in `local-recoding` in Figure 3). A local recoding would change the equivalence classes in the datasets slightly. We used a hashtable to keep track of the update of the classes caused by a recoding (Line 07). Since an emerging pattern may not be hidden by simply one recoding, we note that we may compute the equivalence class of attributes multiple times (Line 03). With the hashtable, Line 03 does not recompute existing equivalence classes.

2. An index for checking correspondents of F in F'. Given two sets of frequent itemsets F and F', we check the correspondents of an itemset in F to measure distortion between F and F' (`util_gain`). The core of this is to check whether a value v is a generalized version of another value in an attribute hierarchy. While an attribute hierarchy is often a small tree, these checks occur in every iteration of `hide-eps`. We apply an index [39] for computing the ancestor-descendant relationship of nodes in a tree. This significantly reduces the runtime of our algorithm.

3. A data structure for checking correspondents of E in E'. The other task of in computing `util_gain` is to keep track of the change of emerging patterns during local recoding, which is neccessary to measure the reduction in growth rate. Hence, we associate an emerging pattern of e to its records. By comparing the records of e and e', we obtain the correspondence between e and e'.

8 Experimental Evaluation

To verify the effectiveness of our proposed algorithms, we have conducted an experimental study of our algorithm on two benchmark datasets.

We have implemented both our algorithm in simulated annealing style in Figure 2 and a greedy-search version of this algorithm, denoted as `SA` and `Greedy`, respectively. The implementation is written in `JAVA SDK` 1.6. We have run our experiment on a desktop PC with a Quad `CPU` at 2.4`GHz` and 3.25`GB RAM`. The desktop PC runs a Windows XP operating system. We have used system calls to invoke the implementations from [41] and [26] to determine emerging patterns and frequent itemsets, respectively.

We have applied our implementation on two datasets, namely `Adult` [36] and `Mushroom` [37]. `Adult` is a census dataset which is used to predict whether a person's income exceeds \$50k/year. Each record in `Adult` contains eight attributes. We removed the records that contain missing values. `Adult` has two classes of people – people who earn more than \$50k/year (7508 records) and people who do not (22654 records). We used the attribute hierarchy presented in [15] for local recoding. `Mushroom` describes physical characteristics, classification and poisonous/edible properties of mushrooms. Our discussions will mainly focus on `Adult` as its results demonstrate the properties of our proposed algorithm better.

The effect of the parameters θ_p and θ_q. We performed a set of experiments to verify the effects on the parameters θ_p and θ_q on the heuristic algorithm. The performance metric used was given in distortion on the frequent itemsets / the number of missing frequent itemsets, unless otherwise specified. We used the `Adult` dataset and `Greedy` in this experiment. σ and ρ were set to 40% and 5, respectively. The frequent itemsets are:

(`Husband`, `Married-civ-spouse`, `Male`)
(`Married-civ-spouse`, `White`)
(`Married-civ-spouse`, `United-States`)
(`Male`, `Private`, `White`)
(`Male`, `Private`, `United-States`)
(`Male`, `White`, `United-States`)
(`Private`, `White`, `United-States`).

When we recode all attributes to `All` in the frequent itemsets, we obtain the maximum distortion of the frequent itemsets of `Adult` 623.1.

To illustrate the possible recoding in the frequent itemsets, we list the frequent itemsets after applying `Greedy`, where θ_p and θ_q were both set to 0.8:

Table 2. The effect of the parameters in `util_gain` on `Greedy`'s performance on `Adult`

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	0/0	50/1	50/1	50/1	50/1	NA
0.2	46.7/1	73.3/1	21.5/1	21.5/1	21.5/1	7.5/3
0.4	46.7/1	73.3/1	21.5/1	21.5/1	21.5/1	7.5/3
0.6	46.7/1	73.3/1	42.5/1	42.5/1	21.5/1	7.5/3
0.8	46.7/1	73.3/1	21.5/1	42.5/1	42.5/1	7.5/4
1.0	7.5/4	7.5/4	7.5/4	7.5/4	7.5/4	7.5/4

(`Relationship`, `United-States`)
(`Married`, `White`, `United-States`)
(`Male`, `Private`, `White`)
(`Male`, `Private`, `United-States`)
(`Male`, `White`, `United-States`)
(`Private`, `White`, `United-States`).

The distortion obtained is 21.5 (out of 623.1).

Next, we varied θ_p and θ_q and measured the distortion of frequent itemsets and runtime of our algorithm. The results are shown in Table 2 and Table 3.

We make a few observations from Table 2 and Table 3. Firstly, when θ_p is set to 0, the algorithm only concerns distortion (regardless of the corresponding reduction in growth rate) during local recodings. In such a case, the distortion in frequent itemsets of various θ_q's is in general large. We note that the corresponding runtime in Table 3 is also relatively high. We used `DNF` to denote "did not finish in 360 mins". The reason is that when θ_p is 0, the heuristics do not effectively reduce the growth rate and the search takes more recodings that are not relevant to hiding emerging patterns. This results in a relatively large distortion and runtime.

Secondly, when θ_p is 1, `util_gain` concerns only the reduction in growth rate. Table 3 shows that `util_gain` finished quickly. However, four frequent itemsets disappeared.

Thirdly, when we set θ_q to 1 and varied θ_p, we do not consider the frequent itemsets that disappear. Hence, such recodings lose relatively more frequent itemsets. Similarly, when we set θ_q to 0, we consider only the distortion of those that do not disappear. Since there is in fact one frequent itemset that disappears during recodings, overlooking this frequent itemset would lead to more distortion.

Fourthly, we obtained no distortion with no missing frequent itemset when θ_p was 1 and θ_q was 0. We speculate that this is an outlier (although a good outlier), as we benchmarked our the algorithm with a real dataset.

In all, we found that on the `Adult` dataset, `Greedy` may yield frequent itemsets with a distortion of 21.5 (out of 623.1) and one (out of seven) missing frequent itemset when θ_p is moderate (0.2 - 0.6) and θ_q is large (0.8).

Simulated annealing search. After verifying the effect of the parameters θ_p and θ_q on `Greedy`, we applied `SA` to `Adult`. We set the temperature of `SA` to be low with a high cooling rate (T = 10 and α = 0.4). Hence, `SA` initially has some chance to avoid local sub-optimas and then converges to `Greedy` quickly. `SA` is allowed to restart five times and

Table 3. The effect of the parameters in `util_gain` on `Greedy`'s runtime (min) on `Adult`

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	54	311	305	214	319	DNF
0.2	107	128	120	102	116	123
0.4	135	109	120	111	117	108
0.6	121	137	139	128	122	109
0.8	101	122	103	130	139	131
1.0	66	62	66	59	66	59

Table 4. The effect of the parameters in `util_gain` on `SA`'s performance on `Adult`, where α = 0.4 and T = 10

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	23.3/2	61.6/1	61.6/1	35/1	50/1	42.5/1
0.2	61.6/1/1	23.3/0	21.5/1	35/4	61.6/1	0/3
0.4	21.5/1	26.6/1	0/3	23.3/1	28.5/1	61.6/2
0.6	61.6/1	21.5/1	0/0	0/3	77/1	0/3
0.8	61.6/1	33.3/1	23.3/1	50/1	0/1	61.6/3
1.0	0/0	0/0	50/1	50/1	50/2	50/3

we report the best of the five runs. The results are shown in Table 4 and 5.

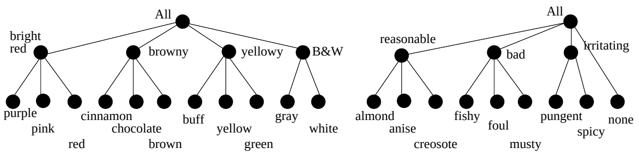
As expected, Table 4 shows that `SA` introduced a certain randomness in our results. `SA` also often performs well at a moderate θ_p and a large θ_q. Under many combinations of θ_p and θ_q, `SA` produces comparable or better results, for example when (p, q) are (0.6, 0.4) and (0.8, 0.8). `SA` may sometimes lose more frequent itemsets when compared to `Greedy`, for example (0.4, 0.4) and (0.6, 0.6).

Table 5 shows that `SA` did not finished more often. The randomness in simulated annealing may lead the search to a place where there are few effective recodings. We tackle this by restarting `SA` when it did not finish in 360 minutes.

Experiments with `Mushroom`. In addition to `Adult`, we tested `Greedy` on the `Mushroom` dataset. We divided `Mushroom` into two datasets: (i) edible mushrooms (3488 records) and (ii) poisonous mushrooms (2156 records). We considered the nominal attributes only. For nominal attributes, we derived our own attribute hierarchies. The attributes of a mushroom concern mostly color and smell. We show the hierarchies of these attributes in Figure 5. We have tested `Greedy` with various combinations of p and q. We set σ and ρ to be 40% and 5, respectively. We omitted the discussion of the runtime of `Greedy` on `Mushroom` since it is approximately 10 mins. The results are shown in Table 6.

Table 5. The effect of the parameters in `util_gain` on `SA`'s runtime (min) on `Adult`, where α = 0.4 and T = 10

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	713	472	366	629	738	417
0.2	953	644	935	DNF	523	1074
0.4	652	557	820	1012	871	893
0.6	611	DNF	881	849	638	954
0.8	408	677	1214	664	894	703
1.0	576	1084	781	940	DNF	DNF



(a) Color (b) Smell

Figure 5. Attribute hierarchies of `Mushroom`

Table 6. `Greedy`'s performance on `Mushroom`

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	92.7/0	92.7/0	92.7/0	92.7/0	92.7/0	92.7/0
0.2	92.7/0	74.7/0	74.7/0	74.7/0	74.7/0	74.7/0
0.4	92.7/0	74.7/0	74.7/0	74.7/0	74.7/0	74.7/0
0.6	92.7/0	74.7/0	74.7/0	74.7/0	74.7/0	55.3/1
0.8	92.7/0	74.7/0	92.7/0	53.3/1	0/1	55.3/1
1.0	92.7/0	92.7/0	92.7/0	NA	NA	74/0

Table 6 shows that our algorithm can often hide emerging patterns without losing any frequent itemsets. However, the distortion is often relatively severe. For example, we often obtained a distortion of 74.7 out of a maximum distortion of 116 in various combinations θ_p and θ_q. And the frequent itemsets in `Mushroom` are $\{$(`odor-none`, `veil-white`), (`stalk-color-above-ring-white`, `stalk-color-below-ring-white`, `veil-white`)$\}$. This may be due to the properties of the `Mushroom` dataset. (i) We found that the height of our attribute hierarchy of `color` and `smell` is rather small. When we recode attributes to hide emerging patterns, the recoded values often reach `All`. (ii) The attribute hierarchies carry certain semantics of the class, for example edible mushrooms often have a pale color. In order to hide emerging patterns, `All` is often used. For instance, recoding `purple`, and `red` into `brightred` may not notably reduce the growth rate of `eps`.

Similarly, we applied `SA` to `Mushroom`. The results are shown in Table 7. The results show that it is still possible to use `SA` to avoid some local sub-optimas. The best recoding leads to a distortion of 18 and no missing frequent itemsets.

9 Conclusions and Future Work

In this paper, we studied a particular case of `PPDP` where we hide emerging patterns of a dataset and at the same time preserve its frequent itemsets as far as possible. We have presented a heuristic local-recoding algorithm for this problem, where some metrics carefully derived for measuring the reduction of the growth rate of emerging patterns and the distortion of frequent itemsets are used to guide the re-

Table 7. `SA`'s performance on `Mushroom`, where α = 0.4 and T = 10

$\theta_p \backslash \theta_q$	0.0	0.2	0.4	0.6	0.8	1.0
0	55.3/0	55.3/0	55.3/0	55.3/0	55.3/0	55.3/0
0.2	55.3/0	55.3/0	55.3/0	36.7/0	55.3/0	92.7/0
0.4	55.3/1	55.3/0	55.3/0	55.3/0	92.7/0	92.7/0
0.6	55.3/0	55.3/0	55.3/0	55.3/0	55.3/0	92.7/0
0.8	55.3/0	18/0	18/0	36.7/0	55.3/0	55.3/0
1.0	55.3/0	55.3/0	92.7/0	55.3/0	74/0	55.3/0

coding process. We have implemented the proposed algorithm and tested it with two benchmark datasets. Our experimental results show that the algorithm is effective in hiding emerging patterns while minimizing the distortion on the frequent itemsets. To the best of our knowledge, this is the first work on hiding emerging patterns in a transactional dataset using recoding generalization.

Future work. While the proposed algorithm has shown to be effective in emerging pattern hiding, there remains a number of open research issues worth future research effort. Firstly, the heuristic algorithm hides emerging patterns in a greedy manner and one by one (`next-overlapping-ep`). It will be interesting to investigate if more advanced search strategies could be adopted to explore more effectively the *generalization space* to further minimize the frequent itemset distortion. Possible ideas include hiding a group of emerging patterns instead of hiding them one-at-a-time, and guiding the recoding process to avoid generating new emerging patterns as far as possible. Secondly, our algorithm invokes some existing implementations of frequent itemset and emerging pattern mining. It may be possible to further optimize our algorithm by incorporating our algorithm into the native implementations of those data mining techniques. Thirdly, since recoding approaches require attribute hierarchies, the frequent itemset distortion could be better measured by considering multilevel frequent itemsets. The corresponding change will be replacing Line 03 in `hide-eps`. The computational complexity for multilevel frequent itemset mining is high though.

References

[1] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of PODS*, 2001.

[3] J. Bailey, T. Manoukian, and K. Ramamohanarao. Fast algorithms for mining emerging patterns. In *Proc. of ECML/PKDD*, 2002.

[4] J. R. Bayardo. Efficiently mining long patterns from databases. In *Proc. of SIGMOD*, pages 85–93, 1998.

[5] R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of ICDE*, pages 217–228, 2005.

[6] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. of SIGKDD*, pages 43–52, 1999.

[7] G. Dong, X. Zhang, and L. Wong. CAEP: Classification by aggregating emerging patterns. In *Proc. of DS'99*, pages 30–42, 1999.

[8] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of SIGKDD*, 2003.

[9] Y. Du, T. Xia, Y. Tao, D. Zhang, and F. Zhu. On multidimensional k-anonymity with local recoding generalization. In *Proc. of ICDE*, pages 1422–1424, 2007.

[10] A. Evfimievski, R. Strikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of SIGKDD*, 2002.

[11] H. Fan and K. Ramamohanarao. A Bayesian approach to use emerging patterns for classification. In *Proc. of ADC*, pages 39–48, 2003.

[12] B. Fung, K. Wang, R. Chen, and P. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*, in press.

[13] B. Fung, K. Wang, A. Fu, and P. Yu. *Privacy-Preserving Data Publishing: Concepts and Techniques*. Chapman & Hall/CRC, 2010.

[14] B. Fung, K. Wang, L. Wang, and M. Debbabi. A framework for privacy-preserving cluster analysis. In *Proc. of ISI*, page 4651, 2008.

[15] B. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *Proc. of ICDE*, pages 205–216, 2005.

[16] B. Fung, K. Wang, and P. Yu. Anonymizing classification data for privacy preservation. *TKDE*, 10(5):711–725, 2007.

[17] H. F. K. Ramamohanarao. Pattern based classifiers. In *Proc. of WWW*, pages 71–83, 2007.

[18] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *KAIS*, 7(4):387–414, 2005.

[19] K. LeFevre, D. Dewitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of SIGMOD*, pages 49–60, 2005.

[20] K. LeFevre, D. Dewitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of ICDE*, page 25, 2006.

[21] J. Li, H. Liu, J. Downing, A.-J. Yeoh, and L. Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. In *Bioinformatics*, volume 19(1), pages 71–78, 2003.

[22] J. Li, H. Liu, S.-k. Ng, and L. Wong. Discovery of significant rules for classifying cancer ddiagnosis data. In *Bioinformatics*, volume 19(Suppl. 2), pages ii93–ii102, 2003.

[23] J. Li, R. Wong, A. Fu, and J. Pei. Anonymization by local recoding in data with attribute hierarchical taxonomies. *TKDE*, 20(9):1181–1194, 2008.

[24] T. Li and N. Li. On the tradeoff between privacy and utility in data publishing. In *Proc. of SIGKDD*, 2009.

[25] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Vẽnkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *TKDD*, 1(1):3, 2007.

[26] MAFIA. *Mining Maximal Frequent Itemsets*. *http://himalaya-tools.sourceforge.net/Mafia/*.

[27] K. Mandl, P. Szolovits, and I. Kohane. Public standards and patients' control: How to keep electronic medical records accessible but private. *British Medical Journal*, 322(7281):283–287, 2001.

[28] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of PODS*, pages 223–228, 2004.

[29] G. Moustakides and V. Verykios. A maxmin approach for hiding frequent itemsets. *DKE*, 65(1):75–79, 2008.

[30] S. Oliveira and O.R.Zaiane. Privacy preserving frequent itemset mining. In *Proc. of ICDM Workshop on Privacy, Security and Data Mining*, volume 14, pages 43–54, 2002.

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

[32] T. Server. Sharing real-time market research data to increase customer satisfaction. *Information Management Magazine*, July 2008.

[33] X. Sun and P. Yu. A border-based approach for hiding sensitive frequent itemsets. In *Proc. of ICDM*, pages 426–433, 2005.

[34] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *IJUFKS*, 10(5):571–588, 2002.

[35] L. Sweeney. k-anonymity: A model for protecting privacy. In *IJUFKS*, pages 557–570, 2002.

[36] UCI Machine Learning Repository. *Adult Data Set*. *http://archive.ics.uci.edu/ml/datasets/Adult*.

[37] UCI Machine Learning Repository. *Mushroom Data Set*. *http://archive.ics.uci.edu/ml/datasets/Mushroom*.

[38] Z. Wang, H. Fan, and K. Ramamohanarao. Exploiting maximal emerging patterns for classification. In *Proc. of AUS-AI*, pages 1062–1068, 2004.

[39] X. Wu, M. L. Lee, and W. Hsu. A prime number labeling scheme for dynamic ordered xml trees. In *Proc. of ICDE*, pages 66–78, 2004.

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

[41] X. Zhang, G. Dong, and K. Ramamohanarao. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *Proc. of SIGKDD*, pages 310–314, 2000.