Hiding Sensitive Frequent Itemsets by a Border-Based Approach

Xingzhi Sun
IBM China Research Lab, Beijing, China
sunxingz@cn.ibm.com

Philip S. Yu
IBM Watson Research Center, Hawthorne, NY, USA
psyu@us.ibm.com

Nowadays, sharing data among organizations is often required during the business collaboration. Data mining technology has enabled efficient extraction of knowledge from large databases. This, however, increases risks of disclosing the sensitive knowledge when the database is released to other parties. To address this privacy issue, one may sanitize the original database so that the sensitive knowledge is hidden. The challenge is to minimize the side effect on the quality of the sanitized database so that non-sensitive knowledge can still be mined.

In this paper, we study such a problem in the context of hiding sensitive frequent itemsets by judiciously modifying the transactions in the database. Unlike previous work, we consider the quality of the sanitized database especially on preserving the non-sensitive frequent itemsets. To preserve the non-sensitive frequent itemsets, we propose a border-based approach to efficiently evaluate the impact of any modification to the database during the hiding process. The quality of database can be well maintained by greedily selecting the modifications with minimal side effect. Experiments results are also reported to show the effectiveness of the proposed approach.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Application - Data Mining

General Terms: Privacy preserving data mining, frequent itemset

Additional Key Words and Phrases: Border

1. INTRODUCTION

Information sharing may require an organization to release its data to public or to allow another party to access it. However, some sensitive information, which is secret to the organization, needs to be hidden before the data is released. Data mining technology enables people to efficiently extract unknown knowledge from a large amount of data. This, however, extends the sensitive information from sensitive raw data (e.g., individual identifier, income, and type of disease identifier) to the sensitive knowledge (e.g., sales pattern of particular product). The reason of this extension is because the data to be released may carry some sensitive knowledge, which can be discovered by certain data mining algorithms.

In [Clifton and Marks 1996], a scenario that releasing data with sensitive knowledge can be a threat to the business of a company or organization is analyzed as follows. Consider a supermarket company S. During the negotiation with a paper company C_1, S reaches an agreement for a reduced price on C_1's products and as a repayment, allows C_1 to access its database to find sales pattern of its merchandise. After mining S's database, C_1 finds that people who buy product P_1 often purchase a paper product of another company C_2. C_1 now runs a coupon promotion "50 cents off product P_1 when people buy the paper product of company C_1". This heavily cuts into the sales of the paper product of company C_2, which then increases the prices to S because of the low sales figure of its product. During company S's next negotiation with C_1, C_1 is reluctant to offer S the discounted price because of the reduced competition. Then, company S starts to lose business to its competitors (who were able to negotiate a better deal with C_2).

In this paper, we give another motivating example of hiding sensitive knowledge during information sharing. In Australia, the supermarket COLES and K-MART share the same customer bonus card, by which the customer id can be identified during the transactions. To facilitate business cooperation (note that the products sold in these two supermarkets are not much overlapped), two supermarkets may integrate their transaction datasets and analyze the "inter-associations" between their products. However, before releasing the dataset to the other party, each supermarket wants to hide sensitive frequent itemsets/association rules of its own products.

Generally, there is a need to prevent confidential or sensitive knowledge (contained in a database) to be discovered by another party even though the information sharing is necessary. In [Verykios et al. 2004], the problem of hiding sensitive knowledge has been considered as an important issue of privacy preserving data mining. To preserve data privacy in terms of knowledge, one can modify the original database in some way so that the sensitive knowledge is excluded from the mining result.

One of the most important data mining techniques is to find the frequent itemsets from a transaction database. The task is: given a transaction database D and a threshold σ, to find all itemsets that are contained by at least σ transactions in D. In this paper, we study the problem of hiding frequent itemsets from a transaction database. Considering the given scenario, if the company S can make some changes on the supermarket database to hide the frequent itemset containing product P_1 and the paper product of company C_2, the sensitive association between them can be protected from company C_1.

To hide the sensitive frequent itemsets, the original database D needs to be modified into D', called *result database*. This modification can be regarded as a process of deliberately injecting noises into D, which unavoidably impairs the quality of D'. Considering the goal of information sharing, releasing a garbage database is meaningless. Therefore, during the process of hiding sensitive frequent itemsets, the quality of the database needs to be preserved so that the impact on the non-sensitive frequent itemsets is minimized.

The quality of result database D' can be evaluated from two different angles. The *raw quality* of D' can be measured by how many items has been changed during the transformation from D to D'. The *aggregated quality* can be evaluated by how well the non-sensitive frequent itemsets in D are preserved in D'. In this paper, with the focus on the second angle, we study the following research problem: to hide sensitive frequent itemsets from a database and meanwhile, maintain the aggregated quality of the result database.

Previous work [Atallah et al. 1999; Saygin et al. 2001; Dasseni et al. 2001; Saygin et al. 2002; Oliveira and Zaiane 2002; 2003a; 2003b; Oliveira et al. 2004] on hiding knowledge has limited concern on minimizing the side effect on the aggregated quality of the result database. Basically, in their work, some simple heuristics are applied to preserve the quality of the database. In this paper, we use the *border* [Mannila and Toivonen 1997] of the non-sensitive frequent itemsets to track the impact of altering transactions. As all itemsets form a lattice, the elements on the border are the boundary to the infrequent itemsets. During the hiding process, instead of considering every non-sensitive frequent itemset, we focus on preserving the quality of the border, which can well reflect the aggregated quality of the result database. According to this idea, a border-based approach is proposed to efficiently evaluate the impact of any modification to the database during the hiding process. The quality of database can be well maintained by greedily selecting the modifications with minimal side effect.

The contribution of this paper is as follows. First, we introduce the border concept to the problem of preserving non-sensitive frequent itemsets. Furthermore, during the hiding process, while previous work only follows some heuristics (rather than really evaluating the impact of each change), our approach ensures that any modification on the database is controlled according to the impact on the result database. An efficient algorithm is devised to identify the candidate change that has minimal impact on the border. Specifically, for each sensitive frequent itemset X, the algorithm determines the optimal deletion candidate (T, x), i.e., the item x and the transaction T from which item x is deleted. For each item $x \in X$, we provide an efficient way to estimate the upper and lower bounds on the impact (to the border) of deleting x. With these bounds, the item which may potentially minimize the border impact can be straightforwardly selected. We then determine from which transactions to delete that item.

The rest of this paper is organized as follows. In Section 2, we give the background of hiding sensitive frequent itemsets and formulate our research problem. A border-based approach is proposed in Section 3. Section 4 gives algorithms to hide sensitive itemsets. The experiment results are shown in Section 5. Section 6 reviews the related work. Finally, we conclude this paper in Section 7.

2. HIDING SENSITIVE FREQUENT ITEMSETS

In this section, we further discuss the problem of hiding sensitive frequent itemsets and then give its formal definition.

Our work is based on the concept of frequent itemset, which is defined as follows.

Let $I = \{i_1, \ldots, i_m\}$ be a set of *items.* An *itemset* is a subset of I. A *transaction* T is a pair (tid, X), where *tid* is a unique identifier of a transaction and X is an itemset. A transaction (tid, X) is said to *contain* an itemset Y iff $X \supseteq Y$. A *database* D is a set of transactions. Given a database D, the *support* of an itemset X, denoted as $Supp(X)$, is the number of transactions in D that contain X. For a given threshold σ, X is said to be *σ-frequent* if $Supp\,(X) \geq \sigma$.

Tid	Items
1	*abcde*
2	*acd*
3	*abdfg*
4	*bcde*
5	*abd*
6	*bcdfh*
7	*abcg*
8	*acde*
9	*acdh*

(a) Database

Frequent itemset : Support
$abd : 3, acd : 4, bcd : 3, cde : 3$
$ab : 4, ac : 5, ad : 6, bc : 4, bd : 5, cd : 6, ce : 3, de : 3$
$a : 7, b : 6, c : 7, d : 8, e : 3$

(b) All frequent itemsets

Expected Frequent itemsets on D′
acd, cde
$ab, ac, ad, bd, cd, ce, de$
a, b, c, d, e

(c) Non-sensitive frequent itemsets

Figure 1. An example.

To give more intuitive explanation of our hiding problem, we give the following example. A transaction database D is given in Figure 1(a). Let the support threshold σ be 3. Figure 1(b) shows all σ-frequent itemsets[1] (with their support) in D. Among those frequent itemsets, *abd*, *bcd*, and *bc* are sensitive itemsets, which cannot be seen by another party. The question is how to transform D into the result database D' in a sensible way such that 1) the sensitive frequent itemsets become infrequent in D' and 2) the aggregated quality of D' is maintained.

2.1 The Way to Change

We first look at how to hide a sensitive frequent itemset X. According to the above definitions, X is hidden if $Supp(X)$ drops below the threshold σ. For any transaction that contains X, once an item $x \in X$ is changed, the record does not contain X anymore and $Supp(X)$ decreases by 1.

Considering a single change of an item, we may have three different approaches: deletion, perturbation (i.e., changing to another item) and blocking (i.e., changing to unknown). Perturbation could introduce new frequent itemsets into result database D', which damages its aggregated data quality. So, in our problem, either deletion or blocking can be a candidate approach. For simplicity, we select deletion during the discussions of the rest of the paper. To avoid unnecessary changes and maintain the quality of the result database, for a sensitive frequent itemset X, we delete an item from a transaction in the database D only if it contributes to the decrease of $Supp(X)$. Also, we stop further deletion once X becomes infrequent, i.e., $Supp(X) = \sigma - 1$.

[1]The itemset is denoted as a list of items alphabetically and the set brackets are omitted

2.2 Aggregated Quality of the Result Database

Since the challenge of hiding frequent items lies in minimizing the side effect on the result database D', we now analyze how to evaluate the aggregated data quality of D'.

Given the set L of σ-frequent itemsets in D and the set $\triangle L$ of sensitive frequent itemsets, we can determine the set L_r of *non-sensitive frequent itemset*s. Note that in general, L_r is not equal to $L - \triangle L$. According to the Apriori property [Agrawal and Srikant 1994], if we hide a sensitive frequent itemset X, any super-itemset of X should be hidden from L as well. So, L_r can be computed by removing each sensitive frequent itemset and its supper-itemset from L. In our running example, $\triangle L = \{abd, bcd, bc\}$, So, L_r should be the set of itemsets shown in Figure 1(c).

Considering the problem of preserving non-sensitive frequent itemsets, L_r can be regarded as the *expected set* of frequent itemsets in D'. Let L' be the actual set of frequent itemset of D' (under the same threshold). The aggregated quality of D' can be evaluated by comparing the similarity between L' and L_r, which is discussed from the following two aspects.

—Itemsets in L' and L_r. Intuitively, to evaluate two sets of itemsets L_r and L', $|L_r - L'|$ and $|L' - L_r|$ can be used as indicators of their similarity. To our problem, because deleting an item from a transaction can only reduce the support of itemsets, no new frequent itemset can be introduced after the hiding (i.e., $|L' - L_r| = 0$). Naturally, we use $|L_r - L'|$ as an indicator of the quality of D'. If $|L_r - L'| = 0$, we have the optimal hiding result in terms of the remaining frequent itemsets with $L_r = L'$.

—The support of itemsets in L' and L_r. In the result of frequent itemsets mining, each itemset is affiliated with a support, indicating its frequency. Hiding $\triangle L$ may decrease the support of different itemsets in L_r to different levels. In this case, the maintenance of relative frequency of itemsets in L_r could reflect the quality of D'. Let us consider the following example. Suppose that we want to hide a set $\triangle L$ of sensitive itemsets in a given database D under the threshold 200. Let abc and bcd be two non-sensitive frequent itemsets with $Supp(abc) = 500$ and $Supp(bcd) = 400$ in D. Assume that there are two result databases D' and D'' corresponding to different hiding processes. In D', $Supp'(abc) = 330$ and $Supp'(bcd) = 390$, while in D'', $Supp''(abc) = 400$ and $Supp''(bcd) = 320$. Apparently, the quality of result database D'' is better than D' as the relative frequency of itemsets is maintained much better.

One reason that we should not underestimate the importance of the second aspect is that the frequent itemset mining is sensitive to threshold. Completely ignoring the relative frequency of itemsets in L_r may cause some misleading mining result of D' under anther threshold. Therefore, to maintain the quality of D' during the hiding process, we have the following two considerations: 1) trying to keep itemsets in L_r frequent and 2) maintaining relative frequency among L_r during the sanitization process.

2.3 Formal Problem Definition

After giving the standards to evaluate the aggregated quality of the result database, the problem studied in this paper can be formulated as follows: Given a threshold σ and a database D, let L be the complete set of σ-frequent itemsets in D. In addition, let $\triangle L \subset L$ be the set of frequent itemsets that need to be hidden. Note that L and $\triangle L$ determine the set L_r of non-sensitive frequent itemsets based on the Apriori property. The problem is to transform D into D' by deleting some items from the transactions such that for the set L' of σ-frequent itemset in D', the following **condition**s are satisfied: 1) $L' \cap \triangle L = \phi$, 2) $|L_r - L'|$ is minimized, and 3) the relative frequency of remaining frequent itemsets is maintained.

3. A BORDER-BASED APPROACH

In this section, we propose a novel border-based approach to address the hiding problem. We first give the basic idea and the rationale of our approach in Section 3.1, followed by technical details in Sections 3.2 and 3.3.

3.1 Basic Idea

The way we are performing transformation guaranties that condition 1) must be satisfied. The challenge is how to deal with conditions 2) and 3). In work [Atallah et al. 1999], the authors have proven that optimal hiding (i.e., hiding $\triangle L$ but minimizing $|L_r - L'|$) is NP-hard. Due to complexity of the problem, previous work provides only some heuristics to avoid degrading the quality of the database. However, without actually tracking the impact to the result database during the hiding process, the relative frequency among itemsets in L_r are not maintained and considerable amount of itemsets in L_r are over-hidden. To rectify this shortcoming, we use the strategy of explicitly monitoring the frequent itemsets in L_r and their support throughout the entire hiding process.

The number of frequent itemsets in L_r is often very large. It is difficult and inefficient to evaluate the impact of deleting an item from a transaction on all frequent itemsets in L_r. Because of the Apriori property, we can focus on the *border* of L_r, which is the set of maximal frequent itemsets in L_r.

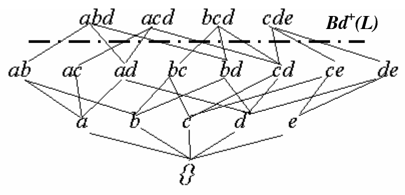


Figure 2. Lattice and border 1.

Formally, given a set of itemsets U the **upper border** (respectively, **lower border**) of U, denoted as $Bd^+(U)$ (respectively, $Bd^-(U)$), is a subset of U such that 1) $Bd^+(U)$ (respectively, $Bd^-(U)$) is an antichain collection of sets[2]

[2]A collection S of sets is an antichain if for any $X, Y \in S$, both $X \subsetneq Y$ and $Y \subsetneq X$ hold.

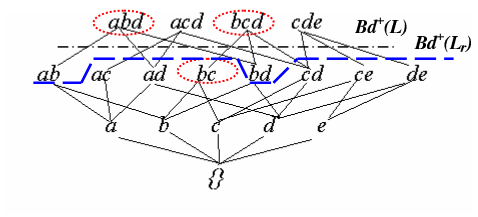


Figure 3. Lattice and border 2.

and 2) $\forall X \in U$, there exists at least one itemset $Y \in Bd^+(U)$ (respectively, $Y \in Bd^-(U)$) holding $X \subseteq Y$ (respectively, $X \supseteq Y$). An itemset in the upper border or lower border is called a **border element**. In the running example, we have $Bd^+(L) = \{abd, acd, bcd, cde\}$ and $Bd^-(\triangle L) = \{abd, bc\}$. The itemset bc is a border element of $Bd^-(\triangle L)$. Note that due to the Apriori property, we only need to hide the lower border $Bd^-(\triangle L)$ rather than every itemset in $\triangle L$. Figure 2 gives a graphic representation of the upper border of L in the itemset lattice.

Using the border notion, the number of itemsets needed to be considered is reduced dramatically. So, in stead of evaluating every itemset in L_r during the hiding process, we can focus on $Bd^+(L_r)$. Figure 3 shows $Bd^+(L_r)$ in the itemset lattice of our example (note that the sensitive frequent itemsets are circled). Clearly, $Bd^+(L_r)$ can be determined by L and $Bd^-(\triangle L)$. A straightforward approach for computing $Bd^+(L_r)$ is as follows. First, for each $X \in Bd^-(\triangle L)$, we remove X and all its supper-itemset from L. We then find the upper border of the remaining itemsets. Now let us examine why the border of L_r is helpful to maintain the aggregated quality of the result database, i.e., taking care of the conditions 2) and 3).

—Condition 2) is to reduce the over-hiding of any non-sensitive frequent itemsets. According to the Apriority property, if all itemsets on the border of L_r remain frequent, every itemset in L_r is frequent. So, concentrating on the border $Bd^+(L_r)$ during the hiding process is effective in avoiding the over-hiding problem. Note that border representation of L_r is not completely lossless in terms of condition 2). This is because when an itemset on the border becomes infrequent, according to requirement 2), we should consider the new border instead of sticking with $Bd^+(L_r)$. However, we could regard $Bd^+(L_r)$ as a close approximation in most cases.

—To satisfy condition 3), keeping the relative frequency among border elements should be helpful. First the support of border elements is relatively low and the relative frequency among them is sensitive to the sanitization. Intuitively, focusing on the most sensitive part of the frequent itemsets can effectively avoid the significant change on the relative frequency. In addition, because of the Apriori property, the support of the border could also reflect the support of the other frequent itemsets to some degree.

From the above discussion, we can see that introducing border representation is a sensible and effective approximation for our hiding problem.

The basic idea of our border-based approach is as follows. Each border element

B in $Bd^+(L_r)$ is assigned a weight, showing its vulnerability of being affected by item deletion. The weight of B is dynamically computed based on its current support during the hiding process. Let us first consider a single sensitive itemset X. When we try to reduce $Supp(X)$, for each possible item (from a transaction) to be deleted, we can evaluate its impact on the border based on the weights of the border elements that will be affected. Each time the candidate item with a minimal impact on the border $Bd^+(L_r)$ is deleted until $Supp(X)$ drops to $\sigma - 1$. Next we consider a set of sensitive itemsets. Note that the border of non-sensitive frequent itemsets $Bd^+(L_r)$ is determined by the sensitive frequent itemsets and our hiding approach is based on the impact on the entire border. So, hiding any single sensitive itemset will not destroy the border unconsciously. In this case, we can order the sensitive frequent itemsets in an appropriate way and hide one of them at a time.

The challenge to our approach is twofold. The first one is on how to quantify the impact of an item deletion in term of the aggregated data quality of D'. The second issue is on how can we find the item with minimal impact on $Bd^+(L_r)$ efficiently. We will discuss the first challenge in the rest of this section and address the second issue in Section 4.

3.2 Hiding One Itemset with Minimal Impact on Border

For brevity, we use border Bd^+ and Bd^- to denote $Bd^+(L_r)$ and $Bd^-(\triangle L)$ in the rest of this paper. In this section, we analyze the problem of hiding a given itemset with a minimal effect on Bd^+. It is clear that to hide an itemset X below the threshold σ, we need to delete $Supp(X) - \sigma + 1$ items from different transactions. Because there are too many choices to make deletions, it is hard to select $Supp(X) - \sigma + 1$ items such that the total impact of them on Bd^+ is minimal. Accordingly, we adopt a greedy technique, deleting one item (from a transaction) with minimal impact on Bd^+ at a time.

In the following parts, we discuss the search space of hiding an itemset in Section 3.2.1. In Section 3.2.2, we discuss how to quantify the impact of deleting an item.

3.2.1 *Search Space of Hiding an Itemset.* Given a frequent itemset X, let $\Lambda(X)$ be the set of transactions that contain X. For each transaction $T \in \Lambda(X)$, if any item $x \in X$ is changed, T will no longer contain X and $Supp(X)$ will decrease by 1. The set C of **hiding candidates** of itemset X is defined as $\{(T,x)|T \in \Lambda(X) \wedge x \in X\}$. Clearly, the total number of hiding candidates of X is $|\Lambda(X)| * |X|$. To hide the itemset X below the threshold σ, we need to change $Supp(X) - \sigma + 1$ of hiding candidates of X such that any two of them are not within the same transaction. To maintain the border Bd^+, the greedy approach reduces the support of X by 1 at each step, i.e., delete one hiding candidate at a time. Note that once a hiding candidate (T_0, x_0) is deleted, i.e., x_0 is deleted from transaction T_0, the new set of C' hiding candidate is $C - \{(T,x)|T = T_0\}$.

3.2.2 *Quantify the Impact of Hiding Candidates on the Border.* Note that deleting a hiding candidate may affect from zero to multiple border elements. Our goal is to select the appropriate hiding candidate at each step such that deleting this hiding candidate causes minimal impact on the border. Intuitively, each border element should have a weight reflecting the vulnerability of further change. The impact of deleting an item could be evaluated by the sum of weights of the affected border elements.

We examine the weight of a border element first. Remember that we have two considerations for the border, i.e., 1) to keep the existing border and 2) to maintain the relative frequency for border elements. To take care of these two considerations, we give the following definition for the weight of a border element. The larger the weight of a border element B is, the more vulnerable B is to further change, therefore, the lower priority of having B affected.

Definition 3.1. Given a database D and a border element $B \in Bd^+$, let $Supp\,(B)$ be the support of B in D. In addition, let $\widetilde{D}$ be the database during the process of transformation and $\widetilde{Supp}\,(B)$ be the support of B in $\widetilde{D}$ (note that at the beginning of transformation, $\widetilde{D} = D$ and $\widetilde{Supp}\,(B) = Supp\,(B)$). The **weight** of border element B is defined as:

$$w(B) = \begin{cases} \frac{Supp(B) - \widetilde{Supp}(B) + 1}{\widetilde{Supp}(B) - \sigma}, & \widetilde{Supp}\,(B) \geq \sigma + 1 \\ \leftthreetimes + \sigma - \widetilde{Supp}\,(B), & 0 \leq \widetilde{Supp}\,(B) \leq \sigma \end{cases}$$

From the definition, we can see the following points: 1) For a border element B, when the current support of B, $\widetilde{Supp}\,(B)$, is greater than the threshold σ, $w(B)$ is no more than 1. When $\widetilde{Supp}\,(B)$ equals to σ, $w(B)$ is assigned a large integer $\leftthreetimes$, where $\infty > \leftthreetimes > |Bd^+|$. The intuition behind this is: if the border element B is about to be infrequent, a large value is assigned to $w(B)$, indicating low priority of being affected. If B is already over-hidden ($\widetilde{Supp}\,(B) < \sigma$), B should also be avoided for further change. In that case, $w(B)$ is decided by $\leftthreetimes$ and the amount of $\widetilde{Supp}\,(B)$ less than σ. 2) If $\widetilde{Supp}\,(B) > \sigma + 1$, with the decrease of $\widetilde{Supp}\,(B)$, $w(B)$ increases under the rate of $\frac{1}{\widetilde{Supp}(B) - \sigma}$. This reflects the consideration of checking the risk of destroying the border element and maintaining the relative frequency among itemsets on the border. For example, consider the case for two border elements B_1 and B_2, where $Supp\,(B_1) = 30$, $Supp\,(B_2) = 15$ and $\sigma = 10$. When we need to hide a sensitive itemset by affecting the support of B_1 and B_2, we can see that B_1 has higher priority of being affected until its support is down to 26. Then B_2 starts to decrease its support by 1, followed by another 4 changes on $Supp\,(B_1)$ if necessary. 3) After $\widetilde{Supp}\,(B)$ drops below the threshold, the weight is increased at rate 1 with the decrease of $\widetilde{Supp}\,(B)$ (note that this rate is no less than the rate of $\frac{1}{\widetilde{Supp}(B) - \sigma}$). Considering two border elements with their current support below the threshold, they will have the same increase rate on their weights (i.e., their support difference in original database D is ignored). The reason is that we want to keep the support of every disappeared border element close to the threshold.

In our running example, the weight of each border element in database D is shown in Figure 5 (with value on the left most column).

After defining the weight of the border element, we next discuss the impact on the border caused by deleting a hiding candidate. Hiding a frequent itemset may not potentially affect all border elements. For example, hiding ab may decrease the support of border element acd, say by deleting (T_1, a). However, it will not decrease the support of border element *cde* for certain. Apparently, only the border element that has intersection with the sensitive frequent itemset may be affected during the hiding process. Given a sensitive frequent itemset X and a border Bd^+, we define the potentially **affected border** of X, denoted as $Bd^+|_X$, as the set of border elements of Bd^+, which may potentially be affected by hiding X. Formally, we have $Bd^+|_X = \{B_i | B_i \in Bd^+ \wedge B_i \cap X \neq \phi\}$. Clearly, for evaluating the impact of hiding X on Bd^+, only $Bd^+|_X$ needs to be considered.

For a hiding candidate u of sensitive frequent itemset X, we can determine the set S_u of border elements that will be affected by deleting u (note that S_u is a subset of $Bd^+|_X$). The impact of deleting u on the border should be the sum of the weights of border elements in S_u. Formally, let $Bd^+|_X$ be $\{B_1, \ldots, B_n\}$ and a lexicographical order can be imposed among $B_1, \ldots,$ and B_n. Given a hiding candidate u of sensitive frequent itemset X, we have a **relevance bit vector** $b_1 b_2 \cdots b_n$ such that $b_i = 1$ if u is a hiding candidate of B_i (i.e., deleting u will decrease $Supp(B_i)$), otherwise $b_i = 0$. The relevance bit vector of u shows which border element B_i will be affected if deleting u. In our running example, for sensitive itemset *abd*, $Bd^+|_{abd} = \{ab, bd, acd, cde\}$. The relevance bit vector of hiding candidate (T_1, a) and (T_3, b) are 1010 and 1100 respectively.

Definition 3.2. Given a hiding candidate u of a sensitive itemset X, $w(B_i)$ for each $B_i \in Bd^+|_X$, and the relevance bit vector $b_1 b_2 \cdots b_n$ of u, the **impact** of u on Bd^+, denoted as $I(u)$, is defined as: $I(u) = \sum b_i * w(B_i)$.

The value of $I(u)$ is the sum of the weights of border elements that will be affected by deleting u. In our running example, if we first delete (T_1, a) to reduce $Supp(abd)$, the impact $I((T_1, a))$ is computed as $w(ab) + w(acd) = 1 + 1 = 2$.

According to definition 3.2, at each step, we can compute the impact for each hiding candidate and select the one with minimal impact to delete. Note that after the deletion of one hiding candidate, the set of hiding candidates of X shrinks and the weights of affected border elements increase.

3.3 The Order of Hiding Itemsets in Bd^-

In this part, we discuss the appropriate order of hiding frequent itemsets in Bd^-. The reason why we should consider the order of itemsets in Bd^- is as follows. If there exists two itemsets $X, Y \in Bd^-$, where $Bd^+|_X \cap Bd^+|_Y \neq \phi$. Hiding X may change the weight of border element $B \in Bd^+|_X \cap Bd^+|_Y$, therefore, affect the process of hiding Y. In general, different orders may lead to different results. Let us consider the following example. Suppose that *abcd* and *de* are two sensitive itemsets and $bcd \in Bd^+$. *bcd* is directly related to hiding *abcd* and indirectly related to *de* (because *abcd* is a super-itemset of *bcd* but *de* is not). If $Supp(abcd)$ and $Supp(bcd)$ are close, hiding *abcd* may have the risk of over-hiding *bcd*. Note that, hiding *de* may also affect the support of *bcd*. However, this may be regarded as a side effect on maintaining *bcd* because they are less correlated. To keep the border element *bcd* frequent, we want to maintain that every decrease of its support contributes to the hiding of *abcd*. In this case, we need to consider *abcd* first to avoid any side effect on the vulnerable border element *bcd*. In general, the longer border element is vulnerable to be over-hidden. In this case, for any two sensitive frequent itemsets, we consider the longer sensitive itemset first. For two sensitive itemsets with the same length, we hide the one with less support first for the same reason.

4. ALGORITHM

In this section, we give the border-based algorithms to efficiently hide sensitive frequent itemsets. The main algorithm shows the structure of our approach. The candidate selection algorithm performs a key task of efficiently identifying and deleting the candidate with minimal impact on Bd^+. For each sensitive frequent itemset X, the candidate selection algorithm determines the optimal hiding candidate $((T, x))$. We provide an efficient way to estimate the upper and lower bounds on the border impact of deleting each item x, $x \in X$. With these bounds, the item which may potentially minimize the border impact can be easily selected. We then determine from which transactions to delete that item. Finally, we demonstrate how the algorithms work on our running example.

4.1 Main Algorithm

```
**Main Algorithm**
**Input:** A database $D$, the set $L$ of $\sigma$-frequent itemset in
$D$ and the set of sensitive itemsets $\triangle L$
**Output:** $D'$ so that aggregated data quality is maintained
**Method:**
Compute $Bd^-$ and $Bd^+$;
Sort itemsets in $Bd^-$ in descending order of length and
ascending order of support;
**for each** $X \in Bd^-$ **do**
  Compute $Bd^+|_X$ and $w(B_j)$ where $B_j \in Bd^+|_X$;
  Initialize $C$ ($C$ is the set of hiding candidates of $X$);
  **for**$(i = 0; i < Supp(X) - \sigma + 1; i++)$ **do**
  /* Candidate selection algorithm*/
    Find $u_i = (T_i, x_i)$ such that $I(u_i) = Min\{I(u) | u \in C\}$;
    Update $C = C - \{(T, x) | T = T_i\}$;
    Update $w(B_j)$ where $B_j \in Bd^+|_X$;
Update database $D$;
Output $D' = D$;
```

Figure 4. Main algorithm.

Figure 4 shows the main algorithm of hiding sensitive frequent itemsets, which is a summary of the approach we described before. The key step is to efficiently find a hiding candidate with minimal impact on the border, as will be shown in the next section. After selecting a candidate, we need to update the hiding candidate set and the weights of the border elements, respectively. Note that for each sensitive frequent itemset, we update the database once after selecting all hiding candidates (to be deleted).

4.2 Candidate Selection Algorithm

The candidate selection algorithm gives the core of the border-based approach, which is to efficiently find the hiding candidate with minimal impact on the border.

Remember that at each step, $\left\{(T, x)|T \in \widetilde{\Lambda(X)} \wedge x \in X\right\}$ is the search space of hiding candidates, where $\widetilde{\Lambda(X)}$ is the set of transactions in the current database $\widetilde{D}$ that contains X. For a large database and a long sensitive itemset X, it is costly to evaluate the impact of every hiding candidate in $\widetilde{C}$. In this section, we propose some heuristics to speed up this search.

4.2.1 *The Approach for Speeding up the Search.* To efficiently select one hiding candidate with minimal impact on the border, our strategy is to quickly select the item x first and then decide a transaction $T \in \widetilde{\Lambda(X)}$.

To find an item x that may bring the minimal impact, we first estimate the possible impact of deleting a hiding candidate with item x. Based on the estimation, we select an item with possible minimal impact on the border.

Let us look at the running example. For the sensitive frequent itemset *abd*, its potentially affected border is $\{ab, bd, acd, cde\}$.

To hide *abd*, deleting a hiding candidate with item d will definitely affect the border element *bd*, but may possibly affect border elements *acd* and *cde* (depending on the transaction). For instance, if d is deleted from T_1, both *acd* and *cde* will be affected, but if d is deleted from T_3, neither *acd* nor *cde* will be affected. In general, for a sensitive frequent itemset X, when deleting the hiding candidate with $x \in X$, we can find some border elements that *must* be affected and some border elements that *could* be affected. According to this observation, for any item $x \in X$, we can use an interval to estimate the possible impact (to the border) of deleting the hiding candidate with x.

Formally, given a sensitive itemset X, the affected border $Bd^+|_X$ can be *partitioned* into **direct border** and **indirect border**, denoted as $Bd^+|_X^1$ and $Bd^+|_X^2$ respectively, such that $\forall Y \in Bd^+|_X^1$, $Y \subset X$. For example, given a sensitive itemset *abd*, $\{ab, bd\}$ is its direct border and $\{acd, cde\}$ is its indirect border. Let $u = (T, x)$ be a hiding candidate of a sensitive itemset X. For any direct border element $Y \in Bd^+|_X^1$ and $x \in Y$, deleting u must decrease the support of border element Y. For an indirect border element $Z \in Bd^+|_X^2$ and $x \in Z$, the support of Z decreases iff $T \supseteq X \cup Z$. From the above example, for $u_1 = (T_3, a)$, deleting u_1 will affect all direct border elements $\{ab, bd\}$, but none of the indirect border elements. On the other hand, for $u_2 = (T_1, a)$, deleting u_2 will affect all direct and indirect border elements.

Given a sensitive frequent itemset X and an item $x \in X$, we can use an interval $i(x) = [I_l, I_r]$, called **impact interval**, to represent possible range of the impact of changing a hiding candidate with item x. $i(x)$ can be interpreted as: changing a hiding candidate with item x will cause at least I_l impact on the border for sure and with the risk of I_r impact in the worst case.

At the first iteration, for any $x \in X$, $i(x).I_l = \sum w(Y_i)$ where $Y_i \in Bd^+|_X^1 \wedge x \in Y_i$ and $i(x).I_r = \sum w(Z_i)$ where $Z_i \in Bd^+ \wedge x \in Z_i$. The **left bound** I_l is the sum of the weights of all relevant direct border. The **right bound** I_r is the sum of the weights of all relevant border element of Bd^+.

Having known the impact interval of each item, to show the priority of being changed, we define partial order $\succeq$ on items based on the following principle: for any two items $x_1, x_2 \in X$, if $i(x_1).I_r < \leftthreetimes \wedge i(x_2).I_r \geq \leftthreetimes$, $x_1 \succeq x_2$; on the contrary, if $i(x_2).I_r < \leftthreetimes \wedge i(x_1).I_r \geq \leftthreetimes$, $x_2 \succeq x_1$; otherwise, $x_1 \succeq x_2 \Leftrightarrow i(x_1).I_l < i(x_2).I_l$ or $i(x_1).I_l = i(x_2).I_l \wedge i(x_1).I_r \leq i(x_2).I_r$

The intuition behind this ordering is: for any two items, if we know one has risk of damaging the border but the other does not, we will change the no risk one to guarantee that the border is intact. Otherwise we always select the item with less possible impact on the border. Let us consider our running example. The impact interval for item a, b, and d are initialized as $[1,2]$, $\left[\frac{3}{2}, \frac{3}{2}\right]$, and $\left[\frac{1}{2}, \leftthreetimes + \frac{3}{2}\right]$, respectively. We have the order $a \succeq b \succeq d$. In general, based on the order $\succeq$, we can select an item x. Now we show how to determine the transaction T.

After finding an item x, we calculate the impact of deleting x for each transaction and find the one with the minimal impact. To hide a sensitive frequent itemset X, we use a bit map representation to reduce the size of $\Lambda(X)$. We maintain $|\Lambda(X)|$ bit vectors, each of which corresponds to a transaction $T \in \Lambda(X)$. The length of each bit vector is $|Bd^+|_X|$. For a bit vector of the transaction $T \in \Lambda(X)$, $b'_i = 1$ iff T contains B_i, where $B_i \in Bd^+|_X$. In Figure 5, we can see bit vectors for T_1, T_3, T_5 are $1111, 1100, 1100$.[3] Given a hiding candidate $u = (T, x)$ of sensitive frequent itemset X and the bit vector $b'_1, \ldots b'_{|Bd^+|_X|}$ of T, the relevance bit vector $b_1, \ldots b_{|Bd^+|_X|}$ of u can be computed as $b_i = b'_i \wedge (x \in B_i)$ for $i = 1, \ldots,$ and $|Bd^+|_X|$. In our example, $Bd^+|_{abd} = \{ab, bd, acd, cde\}$ and the relevance bit vector of (T_1, a) is computed as 1010, which is the same as the result in our previous discussion. For each transaction $T \in \Lambda(X)$, the impact of deleting (T, x) can be computed by the formula given in definition 3.2. After scanning the bit map once (in a worst case), we can find the hiding candidate with minimal impact on the border. Several techniques can be used for improve the performance of scanning bit vectors. For example, if we find a transaction T such that the impact $I((T, x)) = i(x).I_l$, there is no need to perform the rest of the scan. Also, we can order the transaction based on its bit vector and current weight of the border elements. A simple example is that 1100 is always evaluated prior to 1110 because changing the former transaction always affects fewer border elements than changing the latter one. We do not give details here due to space limitation.

	$X \in Bd^-$	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
	abd	1		$1 \xrightarrow{a} 0^{[1]}$		$1 \xrightarrow{d} 0^{[2]}$				
	bc	1			$1 \xrightarrow{b} 0^{[3]}$		$1 \xrightarrow{c} 0^{[4]}$	1		
$w(B_i)$	$B_i \in Bd^+$									
$1 \rightarrow \leftthreetimes^{[1]}$	ab	1		$1 \rightarrow 0$		1		1		
$\frac{1}{2} \rightarrow 1^{[2]} \rightarrow \leftthreetimes^{[3]}$	bd	1		1	$1 \rightarrow 0$	$1 \rightarrow 0$	1			
1	acd	1	1						1	1
$\leftthreetimes$	cde	1			1				1	

Figure 5. Algorithm demo.

4.2.2 *Update Impact Intervals.* Once a hiding candidate is deleted, for any affected border element Y, its weight needs to be updated from $w(Y)$ to $w'(Y)$.

[3]For completeness, we show bit vectors for all transactions. However, for a sensitive frequent itemset X, only $T \in \Lambda(X)$ needs to be considered.

Accordingly, the impact interval $i(x)$ of each item x also needs to be updated to $i'(x)$.

Given a set of new weights for the border elements of Bd^+, we can always compute $i'(x).I_l = \sum w'(Y_i)$ where $Y_i \in Bd^+|_X^1 \wedge x \in Y_i$. However, to make the impact estimation more accurate, we use the following way to tighten the left bound.

THEOREM 4.1. *Let $A'(x)$ be computed as $\sum w'(Y_i)$ where $Y_i \in Bd^+|_X^1$ and $x \in Y_i$. For any item x, if there exists a* perviously deleted *hiding candidate u_0 with item x such that the impact $I(u_0)$ is greater than the estimated left bound I_l (at that stage), for the next step, the new value $i'(x).I_l$ should be be updated as $Max\{V_1, V_2\}$, where $V_1 = A'(X) + Min\{w'(Z_i)\,|Z_i \in Bd^+|_X^2 \wedge x \in Z_i\}$ and $V_2 = A'(x) + I(u_0) - A(x)$. Otherwise, the new value for $i'(x).I_l$, is updated as $A'(x)$.*

PROOF. According to our approach of selecting hiding candidate, $i(x).I_l$ is initialized as the sum of weights of elements in the direct border. Also, we always select the one with minimal impact on the border. So, the first occurrence of a deleted hiding candidate u_0 with $I(u_0) > i(x).I_l$ indicates that, from now on, changing item x will at least affect an indirect border element $Z_i \in Bd^+|_X^2$. In this case, $A'(X) + Min\{w'(Z_i)\,|Z_i \in Bd^+|_X^2 \wedge x \in Z_i\}$ gives the sum of the compulsory impact on the direct border and the minimal impact on the indirect border. Also, because the weight of a border element and the impact $I(u)$ increase monotonically during the iterations, the impact on the indirect border is at least $I(u_0) - A(x)$. With the compulsory impact $A'(x)$ on the direct border, we have the value V_2. According to the definition of the left bound, the value should be $Max\{V_1, V_2\}$. Before the first time that $I(u) > i(x).I_l$ happens, we can only guarantee the impact on the direct border with value $A'(X)$. □

For $i'(x).I_r$, according to our definition, it is always updated as $\sum w'(Y_i)$ where $Y_i \in Bd \wedge x \in Y_i$.

4.3 Algorithm Demo

Now we demonstrate how our approach works in the running example.

From our previous discussion, Bd^- and Bd^+ are $\{abd, bc\}$ and $\{ab, bd, acd, cde\}$, respectively. Specifically, we want to make $Sup(abd) = 1$ and $Sup(bc) = 2$. (Sometimes, it is better to decrease the support of sensitive itemsets to different levels. Otherwise it may cause suspicions when many itemsets are with the same support $\sigma - 1$. The proposed algorithm is applicable to either case.)

We consider *abd* first as it is longer than *bc*. The direct border and indirect border of *abd* are $\{ab, bd\}$ and $\{acd, cde\}$, respectively. The initial weight of each border element is shown on Figure 5 (which is the value on the left most column). The impact intervals for items a, b, and d are $[1, 2]$, $\left[\frac{3}{2}, \frac{3}{2}\right]$, and $\left[\frac{1}{2}, \leftthreetimes + \frac{3}{2}\right]$, respectively. According to the order $\succeq$, item a is selected. $\Lambda(abd) = \{T_1, T_3, T_5\}$. The impact of each tuple is calculated as follows: $I((a, T_1)) = w(ab) + w(acd) = 2, I((T_3, a)) = I((T_5, a)) = w(ab) = 1$. T_3 is selected and hiding candidate (T_3, a) is deleted. Then, we update $w(ab)$ to $\leftthreetimes$ and $\Lambda(abd)$ to $\{T_1, T_5\}$. For the next iteration, the impact intervals for item a, b, and d are $[\leftthreetimes, \leftthreetimes + 1]$, $\left[\leftthreetimes + \frac{1}{2}, \leftthreetimes + \frac{1}{2}\right]$, and $\left[\frac{1}{2}, \leftthreetimes + \frac{3}{2}\right]$, respectively. Item d is selected. Because $I((T_1, d)) = \leftthreetimes + \frac{3}{2}$ and $I((T_5, d)) = \frac{1}{2}$, we select T_5 and delete (T_5, d).

Now we consider the second sensitive frequent itemset bc. The direct border and indirect border of bc are ϕ and $\{ab, bd, acd, cde\}$, respectively. Both impact intervals for b and c are $[0, \leftthreetimes + 1]$. We can select either b or c. It is ideal to select c as it will cost zero impact on the border. Suppose that item b is selected. The impact for the hiding candidate for transactions T_1, T_4, T_6, and T_7 are $\leftthreetimes + 1, 1, 1$, and $\leftthreetimes$, respectively. We can select T_4. Note that $I\left((T_4, b)\right) = 1 > i(b).I_l = 0$. So we have $i'(b).I_l = Min\{w'(ab), w'(bd)\}$. The intervals of b, and c are updated to $[\leftthreetimes, 2\leftthreetimes]$ and $[0, \leftthreetimes + 1]$, respectively. Finally, we delete (T_6, c) with no impact on the border.

5. EXPERIMENT RESULTS

In this section, we evaluate the effectiveness and the efficiency of our border-based approach by comparing it with a heuristic-based approach, which is referred as Algorithm 2.b in [Verykios et al. 2004]. The heuristic in Algorithm 2.b for selecting a hiding candidate is straightforward. Give a sensitive frequent itemset, for all the transactions containing this itemset, Algorithm 2.b first identifies the transaction with the shortest length. In such a transaction, the candidate item with the maximal support value is deleted to decrease the support of the sensitive itemset. This approach hides the frequent sensitive itemsets efficiently and meanwhile demonstrates good effectiveness on minimizing the side effect on the result database. In the rest part of this section, we denote it as the *heuristic approach.*

Table I. Characteristics of datasets.

Dataset	$\|T\|$	$\|I\|$	$\|L\|$	$\|D\|$	N	Size in Megabytes
T10I6L1.5K	10	6	1.5K	100K	1K	5.8
T10I6L1K	10	6	1K	100K	1K	5.8
T20I8L2K	20	8	2K	100K	1K	10.44

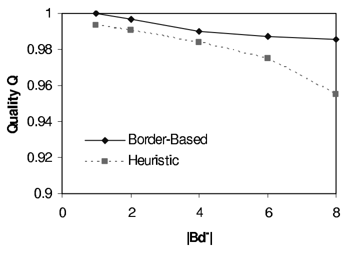
We evaluate our border-based approach on three synthetic datasets, which are created by IBM synthetic data generator [Agrawal and Srikant 1994]. The characteristics of datasets are given in Table I (The description of each parameter can be found in [Agrawal and Srikant 1994]).

5.1 Effectiveness Evaluation

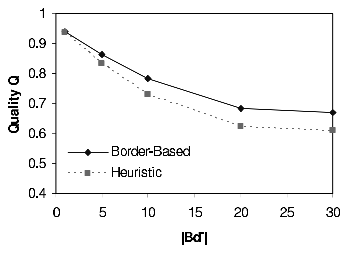
Recall that our goal is to maintain the aggregated quality of the result dataset D'. It is natural to compare the set of non-sensitive frequent itemset L_r with the set L' of frequent itemsets in D'. As our approach does not introduce new frequent itemsets in D', the *quality* of the result dataset D' could be measured as: $Q = \frac{|L'|}{|L_r|}$. Apparently, the percentage of over-hidden non-sensitive frequent itemsets is $1 - Q$.

For each given dataset, we evaluate the quality Q of the result dataset under different sets of sensitive frequent itemsets. Now we first look at the characteristics of the set $\triangle L$ of sensitive frequent itemsets. As discussed before, for $\triangle L$, we only need to consider its lower border Bd^- during the hiding process. Thus, given a set $\triangle L$ of sensitive frequent itemset in a dataset D, we define the following three characteristics of $\triangle L$ in terms of Bd^-:

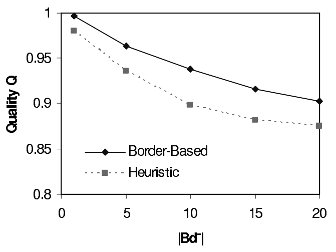
(1) Number of sensitive itemsets in Bd^-, denoted as $|Bd^-|$.



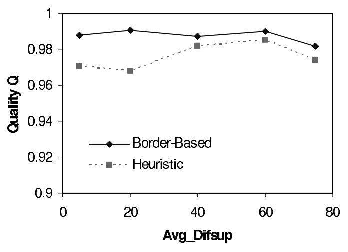
(a) T10I6L1.5K. $\frac{\sigma}{100K}$: 0.6%, Avg_len : 3, Avg_Difsup : 10 ∼ 12



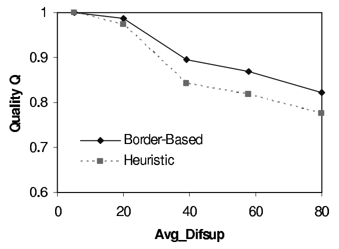
(b) T10I6L1K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 3, Avg_Difsup : 49 ∼ 52



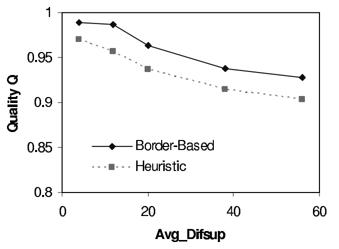
(c) T20I8L2K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 5, Avg_Difsup : 20 ∼ 24



(d) T10I6L1.5K. $\frac{\sigma}{100K}$: 0.6%, Avg_len : 3, $|Bd^-|$: 4



(e) T10I6L1K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 3, $|Bd^-|$: 5



(f) T20I8L2K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 4, $|Bd^-|$: 5

Figure 6. Effectiveness evaluation.

(2) Average support difference, formally defined as:
$Avg\_Difsup = \frac{\sum(Supp(X_i)-\sigma+1)}{|Bd^-|}$, where $X_i \in Bd^-$ and σ is the support threshold.

(3) Average length of itemsets in Bd^-, defined as:
$Avg\_len = \frac{\sum len(X_i)}{|Bd^-|}$, where $X_i \in Bd^-$ and $len(X_i)$ returns the length of X_i.

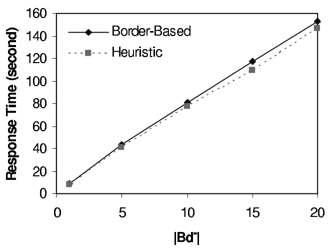
For example, if Bd^- is $\{a : 10, bc : 8, def : 6\}$ and the support threshold σ is 5, we have $|Bd^-| = 3$, $Avg\_Difsup = 4$, and $Avg\_len = 2$.

In our experiments, for each dataset, we evaluate the quality Q of result dataset in terms of the size of the lower border ($|Bd^-|$) and the average support difference (Avg_Difsup) of $\triangle L$. Figure 6 gives the complete results of effectiveness evaluation by comparing with the heuristic approach. Figure 6(a)˜6(c) show the impact of $|Bd^{-}|$ on the quality of the result dataset. Let us take Figure 6(a) as an example. The corresponding experiments are performed on dataset T10I6L1.5K with the percentage of support threshold (i.e., $\frac{\sigma}{100K}$) always set as 0.6%. We intentionally create multiple sets of sensitive frequent itemsets such that in each set, the average length is 3 and the average support difference is controlled within the range from 10 to 12. In this case, we can evaluate how $|Bd^{-}|$ impacts the quality Q. This result shows that the quality of the result dataset is well maintained with over 98% of non-sensitive frequent itemsets preserved. In general, the quality of the result dataset decreases with the increase of $|Bd^{-}|$. In Figure 6(d)˜6(f), the impact of Avg_Difsup on Q is shown on the condition that $|Bd^{-}|$ and Avg_len are constant in each dataset. It is also clear that the quality Q degrades with the rise of Avg_Difsup (the reason is that the increase on Avg_Difsup requires to delete more hiding candidates, thus, leads to more impact on Q).

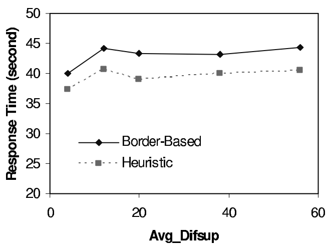
Based on the experiment results, we have the following observations. First, our approach outperforms the heuristic approach in terms of the quality of the result dataset (i.e., protecting more non-sensitive frequent itemsets from being over-hidden). In all these figures, the maximum improvement by the border-based approach is around 5%, i.e., up to an additional 5% of the non-sensitive frequent items could be oven-hidden by the heuristic approach. Note that $|L_r|$ is often a large number (it is roughly ranged from 1000 to 2000 in our experiments). So, little difference in the percentage indicates more significant difference in the actual number. Also, the over-hidden non-sensitive frequent itemsets are close to the border, which often carry more significant information than the itemsets at the lower level of the lattice. In general, our border-based approach achieves the considerable improvement, which also proves the correctness of our consideration on maintaining the border. In addition, we observe that the characteristics of databases have impact on the effectiveness of our approach. Let us consider datasets T10I6L1.5K and T10I6L1K. According to the meaning of parameter $|L|$ (i.e., the number of maximal potentially frequent itemsets), frequent itemsets in T10I6L1K are more strongly correlated than those in T10I6L1.5K. Considering the quality of the result dataset, the experiment result on the "denser" database T10I6L1K is not as good as that of T10I6L1.5K. This is because the following two reasons. First, in a strongly correlated dataset, over-hidden cases are very likely to occur. Some of them are unable to avoid. For example, if $Supp(abd) = Supp(ab) = Supp(ad)$, hiding abd will unavoidably hide either ab or ad. So, the actual number of non-sensitive frequent itemsets in the optimal situation is often less (sometime, far less) than $|L_r|$. Thus, the actual quality of the result dataset should be better than what is displayed in the figures. Secondly, considering our approach, if a border element is hidden during the hiding process, it is ideal to immediately replace the old border by the new one. However, finding a new border for every such case is very time-consuming, so, we only recompute the border after one sensitive frequent itemset is hidden. This approximation leads some tradeoff between the effectiveness and the efficiency.

5.2 Efficiency Evaluation

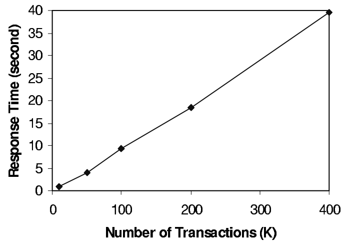
The efficiency of our approach is studied in terms of the response time. All experiments are performed on a PC with an Intel Pentium III 500MHz CPU and 256M



(a) T20I8L2K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 5, Avg_Difsup : 20~24



(b) T20I8L2K. $\frac{\sigma}{100K}$: 0.8%, Avg_len : 4, $|Bd^-|$: 5



(c) T20I8L2K (D400K). $\frac{\sigma}{|D|}$: 0.8%, $|Bd^-|$: 1

Figure 7. Efficiency evaluation.

main memory, running Microsoft Windows XP.

Figure 7 shows the efficiency of the border-based approach on the basis of the dataset T20I8L2K. Particularly, Figure 7(a) and Figure 7(b) depict the performance of hiding frequent itemsets in terms of $|Bd^-|$ and Avg_Difsup respectively. From both figures, we can see that while the heuristic approach takes less time than our border-based approach, their performance curves are very close. This is because the most time-consuming step of hiding sensitive frequent itemsets lies in the dataset scan. Both approaches require the same number of dataset scan, i.e., $|Bd^-|$. Although our border-based approach is more complex in the step of selecting hiding candidates, the heuristics introduced in Section 4.2 offers an innovative algorithm which effectively reduces the computational cost. Note that to improve the performance, the work in [Oliveira and Zaiane 2003a] uses a inverted file to reduce the number of dataset scans. This technology can be applied in our border-based approach as well.

Further, we look at the scalability of our border-based approach. Figure 7(c) shows the response time of hiding one sensitive itemset with respect to the number of transactions in the dataset. We can see that our approach is linearly scalable.

6. RELATED WORK

Privacy preserving data mining [Verykios et al. 2004] has become a popular research direction recently. The goal of this research includes two aspects. First, sensitive raw data should be protected from the original database, in order to preserve the individual privacy. Second, sensitive knowledge, which can be extracted by data mining algorithms should be trimmed out from the database. Our research in this paper targets on the second aspect.

The problem of hiding frequent itemsets (or association rules) was firstly studied in [Atallah et al. 1999] by Atallah et al. In this work, finding the optimal sanitization solution to hide sensitive frequent itemsets was proved as a NP-hard problem. Also, a heuristic-based solution was proposed to exclude sensitive frequent itemsets by deleting items from the transactions in the database. The subsequent work [Dasseni et al. 2001] extended the sanitization of sensitive frequent itemsets to the sanitization of association rules. The work prevented association rules from being discovered by either hiding corresponding frequent itemsets or reducing their confidence below the threshold. The work provided some heuristics to select the items to be deleted, with the consideration of minimizing the side effect under the assumption that sensitive frequent itemsets were disjoint. The later work [Saygin et al. 2001; Saygin et al. 2002] further discussed the problem of hiding association rules by changing items to “unknown” instead of deleting them.

Also, substantial work [Oliveira and Zaiane 2002; 2003a; 2003b; Oliveira et al. 2004] has been done in this area by Oliveira and Zaiane. Generally, their work focused on designing a variety of heuristics to minimize the side effect of hiding sensitive frequent itemsets. Particularly, in [Oliveira and Zaiane 2002], the Item Grouping Algorithm (IGA) grouped sensitive association rules in clusters of rules sharing the same itemsets. The shared items were removed to reduce the impact on the result database. In [Oliveira and Zaiane 2003b], a sliding window was applied to scan a group of transactions at a time and sanitized the sensitive rules presented in such transactions. In recent work [Oliveira et al. 2004], they considered the attacks against sensitive knowledge and proposed a Downright Sanitizing Algorithm (DSA) to hide sensitive rules while blocking inference channels by selectively sanitizing their supersets and subsets at the same time.

In summary, the challenge of hiding sensitive itemsets (or association rules) is to minimize the side effect on the result database. In previous work, a variety of approaches have been proposed based on different heuristics. However, during the hiding process, none of them really evaluates the impact of each modification on the database.

A preliminary version of this paper appeared in [Sun and Yu 2005]. In this paper, we extend the work in following ways. First, we comprehensively discuss the problem of hiding frequent itemsets. Second, we present the more detailed algorithms with examples and proof. Importantly, we introduce the algorithm for updating the impact interval in Section 4.2.2.

7. CONCLUSIONS

In this paper, we have studied the problem of hiding sensitive frequent itemsets, with a focus on maintaining the aggregated quality of the result database. The originality and contributions of our work include the following aspects: 1) We considered the aggregated quality not only based on the number of non-sensitive frequent itemsets preserved in the result database, but also in terms of their relative frequency. 2) Most importantly, to minimize the side effect on the result database, we provided the first efforts to evaluate the impact of any modification to the database during the hiding process. Thus, the quality of database can be well maintained by controlling modifications according to the impact on the result database. 3) We applied the border as an appropriate representation of the set of non-sensitive frequent itemsets in the context of frequent itemset hiding problem. A border-based approach was proposed to efficiently select the modification with minimal side effect. 4) We study the performance of the proposed approach and the results were superior to the previous work in effectiveness, at the expense of a small degradation in efficiency.

REFERENCES

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB*. 487–499.

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., and Verykios, V. 1999. Disclosure limitation of sensitive rules. In *Proc. of KDEX'99*. 45–52.

Clifton, C. and Marks, D. 1996. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*. Montreal, Canada, 15–19.

Dasseni, E., Verykios, V. S., Elmagarmid, A. K., and Bertino, E. 2001. Hiding association rules by using confidence and support. In *Proc. of the 4th Information Hiding Worshop*. 369–383.

Mannila, H. and Toivonen, H. 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery 1,* 3, 241–258.

Oliveira, S. and Zaiane, O. 2002. Privacy preserving frequent itemset mining. In *Proc. ICDM Workshop on Privacy, Security, and Data Mining*. 43–54.

Oliveira, S. and Zaiane, O. 2003. Algorithms for balancing privacy and knowledge discovery in association rule mining. In *7th Proc. of the IDEAS*. 54–63.

Oliveira, S. and Zaiane, O. 2003. Protecting sensitive knowledge by data sanitization. In *Proc. of the 3rd ICDM*. 613–616.

Oliveira, S., Zaiane, O., and Saygin, Y. 2004. Secure association rule sharing. In *Proc. of the 8th PAKDD*. 74–85.

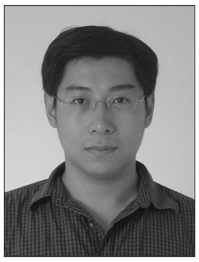
Saygin, Y., Verykios, V. S., and Clifton, C. 2001. Using unknowns to prevent discovery of association rules. *ACM SIGMOD Record 30*, 45–54.

Saygin, Y., Verykios, V. S., and Elmagarmid, A. K. 2002. Privacy preserving association rule mining. In *Proc. of RIDE*.

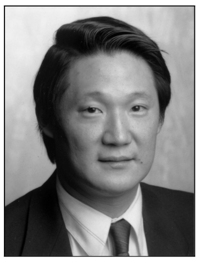
Sun, X. and Yu, P. S. 2005. A border-based approach for hiding sensitive frequent itemsets. In *Proc. of the 5th ICDM*. 426–433.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. 2004. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record 33*, 50–57.

Verykios, V. S., Elmagarmid, A. K., Bertino, E., Saygin, Y., and Dasseni, E. 2004. Association rule hiding. *TKDE 16*, 434–447.



Xingzhi Sun received the BSc degree in electronic engineering from Shanghai Jiao Tong University in 2000, and the PhD degree in computer science from the University of Queensland in 2005. He is currently a researcher at IBM China Research Lab. His main research interests are data mining and ontology data management.



Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is with the IBM Thomas J. Watson Research Center and currently manager of the Software Tools and Techniques group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. Dr. Yu has published more than 500 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and a Fellow of the IEEE. He is associate editors of ACM Transactions on the Internet Technology and ACM Transactions on Knowledge Discovery from Data. He is on the steering committee of IEEE Conference on Data Mining and was a member of the IEEE Data Engineering steering committee. He was the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2001-2004), an editor, advisory board member and also a guest co-editor of the special issue on mining of databases. He had also served as an associate editor of Knowledge and Information Systems. In addition to serving as program committee member on various conferences, he was the program chair or co-chairs of the IEEE Workshop of Scalable Stream Processing Systems (SSPS'07), the IEEE Workshop on Mining Evolving and Streaming Data (2006), the 2006 joint conferences of the 8th IEEE Conference on E-Commerce Technology (CEC' 06) and the 3rd IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE' 06), the 11th IEEE Intl. Conference on Data Engineering, the 6th Pacific Area Conference on Knowledge Discovery and Data Mining, the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, the 2nd IEEE Intl. Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the 2nd IEEE Intl. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair or co-chairs of the 2006 ACM Conference on Information and Knowledge Management, the 14th IEEE Intl. Conference on Data Engineering, and the 2nd IEEE Intl. Conference on Data Mining. He has received several IBM honors including 2 IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, 2 Research Division Awards and the 88th plateau of Invention Achievement Awards. He received a Research Contributions Award from IEEE Intl. Conference on Data Mining in 2003 and also an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor.