

Privacy-Aware Market Basket Data Set Generation: A Feasible Approach for Inverse Frequent Set Mining

Xintao Wu Ying Wu Yongge Wang
UNC at Charlotte
9201 Univ. City Blvd
Charlotte, NC 28223
{xwu,ywu,yonwang}@uncc.edu

Yingjiu Li
Singapore Management University
469 Bukit Timah Road
Singapore 259756
yjli@smu.edu.sg

Abstract

Association rule mining has received a lot of attention in the data mining community and several algorithms were proposed to improve the performance of association rule or frequent itemset mining. The IBM Almaden synthetic data generator has been commonly used for performance evaluation. One recent work shows that the data generated is not good enough for benchmarking as it has very different characteristics from real-world data sets. Hence there is a great need to use real-world data sets as benchmarks. However, organizations hesitate to provide their data due to privacy concerns. Recent work on privacy preserving association rule mining addresses this issue by modifying real data sets to hide sensitive or private rules. However, modifying individual values in real data may impact on other, non-sensitive rules. In this paper, we propose a feasible solution to the NP-complete problem of inverse frequent set mining. Since solving this problem by linear programming techniques is very computationally prohibitive, we apply graph-theoretical results to divide the original itemsets into components that preserve maximum likelihood estimation. We then use iterative proportional fitting method to each component. The technique is experimentally evaluated with two real data sets and one synthetic data set. The results show that our approach is effective and efficient for reconstructing market basket data set from a given set of frequent itemsets while preserving sensitive information.

1 Introduction

Since its introduction in [1], association rule mining has received a lot of attention in the data mining community. Several algorithms were proposed to improve performance of association rule or frequent itemset mining. The algorithms developed typically only show the performance advantage using synthetic data sets (e.g., the market basket data generator provided by IBM Almaden). Recently, Zheng et al. compared five well-known association rule algorithms using three real-world

data sets and the artificial data set from IBM Almaden [22]. One interesting result is that the artificial data sets from IBM Almaden have very different characteristics from the real-world data sets and hence there is a great need to use real-world data sets as benchmarks.

However, organizations hesitate to provide their real-world data sets as benchmarks due to the potential disclosure of private information or commercial secret. Some recent work [3, 6, 9, 10, 14, 16, 10, 17] on privacy preserving association rule mining considers how much information can be inferred or computed from large data made available through data mining algorithms and looks for ways to minimize the leakage of information. Most ideas are to modify the given database (e.g., randomization, hiding, replacing individual values with unknowns) so that the support of a given set of sensitive rules decreases below some minimum support thresholds. The problem here is modifying the individual values of original databases to limit the disclosure of sensitive rules may impact on other, non-sensitive frequent itemsets.

The frequent sets and their supports (defined as the number of transactions in the database that contain the items) can be considered to be a reasonable summary of the original data set. A natural problem is inverse frequent set mining, i.e., to find a binary data set that is compatible with frequent set mining results. The inverse frequent set mining is related to the questions of how well privacy is preserved in the frequent sets and how well the frequent sets characterize the original data set. The authors, in [13, 5], investigate the problem whether there exists a data set that is consistent with the given frequent itemsets and frequencies and show this problem is NP-Complete.

In this paper, we propose a feasible approach on inverse frequent itemset mining. Here the given frequent itemsets and their frequencies were discovered from the original one, hence there must exist at least one compat-

ible data set. Our focus here is how to generate one effectively and efficiently. The frequency of each frequent itemset is taken as a constraint over the original data set. The problem of inverse frequent set mining then can be translated to a linear constraint problem. Linear programming problems can be commonly solved today in hundreds or thousands of variables and constraints. However, the number of variables and constraints in our scenario is far beyond hundreds or thousands (e.g., 2^d , where d is the number of items). Hence it is impractical to apply linear programming techniques directly. In this paper, we propose a feasible approach using graphical decomposition techniques to reconstruct market basket data sets given a set of frequent itemsets and their supports.

The constructed market basket data set can be used as benchmarks for evaluating various different frequent itemset mining algorithms. Ramesh et al. recently investigated the relation between the distribution of discovered frequent set and the performance of association rule mining [15]. It suggests that the performance of association rule mining method using the original data set should be very similar to that using the synthetic one compatible with the same frequent set mining results.

Furthermore, our approach could achieve better privacy preserving as it only needs to access the summary information (i.e., frequencies of given frequent itemsets) instead of the individual values of the original data set. If frequencies of some frequent itemsets are considered confidential information by user, they can be removed directly from the released list. However, though those frequencies of the released frequent itemsets are not confidential themselves, attackers may still be able to derive some confidential or private information (e.g., frequencies of some confidential or sensitive itemsets) from the released information. We will also address this issue by using the *Frechet Bounds* to analyze the potential disclosure of our approach.

The rest of the paper is organized as follows. In Section 2 we review the related work. In Section 3 we formalize the problems and in Section 4 present our method in detail. Experimental results are discussed in Section 5. In Section 6 we draw conclusions and describe directions for future work.

2 Related Work

Privacy preserving data mining considers how much information can be inferred or computed from large data made available through data mining algorithms and looks for ways to minimize the leakage of information. Recent research has been focused on the information leakage in association rule mining. In [3, 6], the authors considered the problem of limiting disclosure of sensitive

rules, aiming at selectively hiding some frequent itemsets from large databases with as little impact on other, non-sensitive frequent itemsets as possible. The idea was to modify a given database so that the support of a given set of sensitive rules decreases below the minimum support value. Similarly, the authors in [17] presented a method for selectively replacing individual values with unknowns from a database to prevent the discovery of a set of rules, while minimizing the side effects on non-sensitive rules. The authors studied the impact of hiding strategies in the original data set by quantifying how much information is preserved after sanitizing a data set [14]. [16, 10] studied the problem of mining association rules from transactions in which the data has been randomized to preserve privacy of individual transactions. One problem is it may introduce some false association rules.

The authors, in [15], proposed a method to generate market basket data set for benchmarking when the length distributions of frequent and maximal frequent itemset collections are available. Though the generated synthetic data set preserves the length distributions of frequent patterns, one serious limitation is that the size of transactions generated is much larger than that of original database while the number of items generated is much smaller. We believe the sizes of items and transactions are two important parameters as they may significantly affect the performance of association rule mining algorithms. In our paper, we will apply graphical decomposition techniques to estimate the cell frequencies of the contingency table which is then used to construct market basket data set. Graphical modeling and decomposition techniques are well studied in statistics [12, 19] and were recently used for screening and interpreting multi-item associations in [20].

A number of researchers from statistics field have recently been working on the problem of determining upper and lower bounds on the cells of the cross-classification given a set of margins [8, 7]. Upper and lower bounds induced by some fixed set of marginal on the cell entries of a contingency table are of great importance in measuring the disclosure risk associated with the release of these marginal totals.

Wu et al. have proposed a general framework for privacy preserving database application testing by generating synthetic data sets based on some a-priori knowledge about the production databases [21]. The general a-priori knowledge such as statistics and rules can also be taken as constraints of the underlying data records. The problem investigated in this paper can be thought as a simplified problem where data set here is binary one and constraints are frequencies of given frequent itemsets. However, the techniques developed

in [21] are infeasible here as the number of items are much larger than the number of attributes in general data sets.

3 Problem Formulation

For market basket data, we define each transaction, such as list of items purchased, as a subset of all possible items.

DEFINITION 3.1. Let $\mathcal{I} = \{I_1, \dots, I_d\}$ be a set of d boolean variables called items. Then a set of transactions $\mathcal{T} = \{t_1, \dots, t_N\}$ is a collection of N d -tuples from $\{True, False\}^d$ which represent a collection of value assignments to the d items. The support of an itemset s over \mathcal{I} , denoted $support(s, \mathcal{T})$, is defined as the number of the transactions that contain s . The frequency of an itemset s , denoted $freq(s, \mathcal{T})$, is defined as $support(s, \mathcal{T})/N$.

The well known frequent itemset mining problem is defined as:

PROBLEM 1. Frequent Itemsets Mining. Given a transaction database \mathcal{T} over \mathcal{I} and a threshold $\tau \in [0, 1]$, find all frequent itemsets FS such that $freq(FS, \mathcal{T}) \geq \tau$.

In our paper, we focus on the inverse frequent itemset mining defined as:

PROBLEM 2. Inverse Frequent Itemsets Mining. Given a finite set $\mathcal{FS} = \{FS_1, \dots, FS_n\}$ together with their frequencies $\{freq(FS_1), \dots, freq(FS_n)\}$ discovered from the original database D , construct a transaction database \hat{D} such that

- 1) \hat{D} satisfies \mathcal{FS} and their frequencies and
- 2) D and \hat{D} are over the same set \mathcal{I} and have the same number of transactions.

DEFINITION 3.2. We define CT_s as a k -dimensional contingency table for itemset $s \subseteq \mathcal{I}$ where $|s| = k$. Each dimension has two values (True or False) and each cell contains the number (or the frequency) of transactions located in that cell.

The market basket data set in Definition 3.1 can be easily transformed into one (and only) d -dimensional contingency table, and vice versa. Table 1 shows a market basket data set with 9 transactions and three items and its 3-dimensional contingency table. Note the contingency table shown in Table 1(c) is equivalent to that shown in Table 1(b) when the number of transactions is fixed. The $cell(ABC)$ contains the number (or frequency) of those transactions which buy item ABC together (i.e., t8 and t9) while $cell(\bar{A}BC)$

contains the number (or frequency) of those transactions which buy BC but do not buy A (i.e., t6). Hence we can construct its contingency table by one scan of the original basket data set if the contingency table is fitted in memory. On the other hand, we can construct exactly one market basket data set by scanning each cell value of the contingency table¹. For example, from $cell(ABC) = 2$, we generate two transactions which contain item AB but not C.

As market basket data set is equivalent to its contingency table, Problem 2 can be mapped to the contingency table reconstruction problem, i.e., construct a feasible contingency table $CT_{\mathcal{I}}$ which satisfies the given marginal frequencies $\{freq(FS_1), \dots, freq(FS_n)\}$. Here each given frequent itemset and its frequency can be taken as a constraint or a marginal as shown in Example 1.

EXAMPLE 1. For the data set shown in Table 1, the frequent itemsets are A, B, C, AB, AC and their frequencies are $6/9, 7/9, 5/9, 4/9, 4/9$ respectively when support threshold is set as $4/9$. Each frequency can be taken as a constraint:

$$\begin{aligned}
 freq(ABC) + freq(\bar{A}\bar{B}\bar{C}) + freq(AB\bar{C}) + freq(\bar{A}BC) &= \frac{6}{9} \\
 freq(ABC) + freq(\bar{A}B\bar{C}) + freq(AB\bar{C}) + freq(\bar{A}BC) &= \frac{7}{9} \\
 freq(ABC) + freq(\bar{A}\bar{B}C) + freq(\bar{A}BC) + freq(\bar{A}BC) &= \frac{5}{9} \\
 freq(ABC) + freq(AB\bar{C}) &= \frac{4}{9} \\
 freq(ABC) + freq(\bar{A}BC) &= \frac{4}{9}
 \end{aligned}
 \tag{3.1}$$

It is straightforward to see we can apply linear programming techniques to compute frequencies of cells (e.g., $freq(ABC)$, $freq(\bar{A}\bar{B}\bar{C})$, etc.) and the generate market basket data using frequencies computed. However, the number of variables and constraints in our scenario is too large to be handled by the current linear programming packages.

4 Our Method

Figure 1 illustrates our method. Generally it involves grouping frequent itemsets into disjoint clusters, decomposing each cluster into components that are mutually independent, deriving or computing the cell frequency of

¹Here we assume transactions in market basket data are unordered.

Table 1: An example of data set with three items

TID	List of items
t1	A,B
t2	B
t3	B
t4	A,B
t5	A,C
t6	B,C
t7	A,C
t8	A,B,C
t9	A,B,C

(a) market basket data

	\tilde{B}		B	
	A	A	A	A
\tilde{C}	0	0	2	2
C	0	2	1	2

(b) contingency table

	\tilde{B}		B	
	A	A	A	A
\tilde{C}	0	0	2/9	2/9
C	0	2/9	1/9	2/9

(c) contingency table with cell value as frequency

the contingency table built from each component, computing the cell frequency of the contingency table for each cluster, and generating transactions by merging the contingency table of each cluster.

First we decompose d items into m disjoint item clusters $\mathcal{S} = \{s_1, \dots, s_m\}$, where $|s_i| = k_i$, $s_i \cap s_j = \phi$ and $\bigcup_{i=1, \dots, m} s_i = \mathcal{I}$, according to the given frequency sets \mathcal{FS} (line 2). it works as follows. We take each item as one node and add an edge between two nodes if both nodes are contained in some frequent itemset. After processing all frequent itemsets, each disconnected subgraph forms one cluster. For example, if the released frequency sets are $\{AB\}$, $\{BC\}$, $\{DE\}$, $\{BF\}$ for 6 items, we merge to get two clusters $\{ABCF\}$ and $\{DE\}$. As item clusters here are exclusive disjoint, we can generate \hat{D}_s independently (line 3-9) and join them to get \hat{D} finally (line 10). Here each \hat{D}_s is a vertical partition where each row contains those items in s .

If the frequency of item cluster s is already contained in the given frequency set, we can generate contingency table directly without further processing (line 4 and 5). For example, assume $s = ABC$ and $freq(ABC)$ is known, then $freq(AB)$ and $freq(AC)$ must also be available since the frequency of any set $s' \subset s$ is available when the frequency s is available. Hence we can compute $fq(ABC\bar{C}) = freq(AB) - freq(ABC)$, $fq(A\bar{B}C) = freq(AC) - freq(ABC)$ and so on. If s is not contained in the given frequency sets, we apply a divide and conquer approach and decompose s into components recursively (line 11-21). During this process, we apply graphical decomposition techniques to decompose the independence graph G_s into subgraphs while keeping the maximum likelihood estimation unchanged. We leave the discussion of this part in Section 4.2. When the component can not be further decom-

posed and it is not included in the frequent itemsets, we apply Iterative Proportional Fitting (IPF) method to estimate its contingency table. We present how to use IPF to generate contingency table in Section 4.1.

4.1 Iterative Proportional Fitting In this section we briefly review the iterative proportional fitting method. The IPF has been well studied in the statistical literature. It is an iterative algorithm that converges to the maximum likelihood estimation. In its simplest form, the algorithm provides a method of adjusting one two-way contingency table to conform to the margins. It begins by scaling the rows of the first table to have the correct row margins, then it scales the resulting table to have the correct column margins, then it scales the resulting table to have the correct row margins, and so on, iterating through the cycle of rows and columns, until convergence is reached.

In our scenario, the underlying contingency table is d dimensions (d is the number of items contained in s). IPF starts the reconstruction by initializing each cell c with frequency $1/2^d$ and computes its marginals specified by each constraint. At each following iteration, IPF loops over all constraints and all cells c involved in each constraint, and adjusts the values of the cells according to the formula:

$$\hat{f}q(c)^{(t+1)} = \hat{f}q(c)^{(t)} \frac{freq(FS_i)}{\hat{freq}^{(t)}(FS_i)}$$

here we denote as $\hat{f}q(c)^{(t)}$ the estimate of the value of cell c during the t -th iteration of the algorithm and denote as $\hat{freq}^{(t)}(FS_i)$ the frequency of marginal cell FS_i which is computed from estimated values of cells ($\hat{f}q(c)^{(t)}$) in the t -th iteration of the algorithm.

InverseFS($FS, freq(FS), d, N$)
input FS , a given set of frequent itemsets
 $freq(FS)$, frequencies of given FS
 d , number of items
 N , number of transactions
output \hat{D} , generated data set
BEGIN
1 $\hat{D} = \phi$
2 $\mathcal{S} = \text{GenItemCluster}(FS)$
3 For each cluster $s \in \mathcal{S}$ do
4 If $s \in \mathcal{FS}$
5 $CT_s = \text{GenContingencyTable}(s)$
6 Else
7 $G_s = \text{GenIndependenceGraph}(s)$
8 $CT_s = \text{Decompose}(s, G_s)$
9 Generate data set \hat{D}_s from CT_s
10 $\hat{D} = \hat{D} \cup \hat{D}_s$
END

Decompose (s, G_s)
input s , a given itemset
 G_s , independence graph of s
output CT_s , contingency tables of s
BEGIN
11 If $s \in \mathcal{FS}$
12 $CT_s = \text{GenContingencyTable}(s)$
13 Else
14 If s is irreducible
15 $CT_s = \text{IPF}(s)$
16 Else
17 there exists a partition of G_s
18 into $(G_{s_a}, G_{s_b}, G_{s_c})$ such that
19 $s_a \perp s_b \mid s_c$ and G_{s_c} is a clique
20 Decompose($s_{a \cup c}, G_{s_{a \cup c}}$)
21 Decompose($s_{b \cup c}, G_{s_{b \cup c}}$)
22 Decompose(s_c, G_{s_c})
23 $CT_s = \frac{CT_{s_a \cup s_c} \times CT_{s_b \cup s_c}}{CT_{s_c}}$
END

Figure 1: Inverse Frequent Set Generation Algorithm

The estimates converge in a monotonically decreasing fashion, and we commonly choose to terminate the iterations when the change in each cell estimate becomes smaller than some user specified threshold value.

As we stated in the introduction, we cannot apply IPF over the very large contingency table that contains many items. Usually the contingency table with many items tends to be sparse, which affects the accuracy of IPF method. Besides, even if the contingency table is dense, the complexity of IPF algorithm is generally iterative oriented and thus computationally expensive for large tables. In next section, we discuss how to decompose into subsets and apply IPF only on those irreducible components without losing any significant information.

4.2 Graphical Decomposition The graphical decomposition involves two steps: 1) building one *independence graph* for each cluster; 2) applying graph-theoretical results to decompose the graph into irreducible components.

4.2.1 Building Independence Graph from Frequent Itemsets The independence graph is defined by making every vertex of the graph correspond to a discrete random variable, and the edges denoting the dependency of the two variables linked. A missing edge in the graph represents the conditional independence of the two variables associated with the two vertices. To build the independence graph, we need to test conditional independence for every pair of variables, controlling for the other variables in the same cluster. There are several approaches to test conditional independence (See [2]). In our paper, we build the independence graph by applying the Cochran-Mantel-Hasenzel test.

Here we first assume the contingency table of s is known and describe how the Cochran-Mantel-Hasenzel test works. For any pair of two items I_i, I_j from item set $s \subseteq I$ ($|s| = k$), we derive one partial 2×2 contingency table (stratum) for each possible value from set $s \setminus \{I_i, I_j\}$. Hence we can have L ($L = 2^{k-2}$) strata. For each stratum l , we need to compute the marginal totals $\{n_{\cdot 0}^{(l)}, n_{\cdot 1}^{(l)}, n_{0 \cdot}^{(l)}, n_{1 \cdot}^{(l)}\}$, where a dot “.” denotes a sum along that dimension (e.g., $n_{\cdot 0}^{(l)} = n_{00}^{(l)} + n_{10}^{(l)}$). Table 2(a) shows the stratum form for item variable A and B while Table 2(b) shows one stratum ($C = 1$). Equation 4.2 shows the summary statistics where $m_{11}^{(l)}$ and $V(n_{11}^{(l)})$ is mean and variance respectively.

$$m_{11}^{(l)} = E(n_{11}^{(l)}) = \frac{n_{1 \cdot}^{(l)} n_{\cdot 1}^{(l)}}{n_{\cdot \cdot}^{(l)}}$$

Table 2: A 2×2 contingency table for variable A and B

	B	\tilde{B}	
A	n_{11}	n_{10}	$n_{1.}$
\tilde{A}	n_{01}	n_{00}	$n_{0.}$
	$n_{.1}$	$n_{.0}$	$n_{..}$

(a) stratum form

	B	\tilde{B}	
A	2	2	4
\tilde{A}	1	0	1
	3	2	5

(b) example of
one stratum
C=1

$$V(n_{11}^{(l)}) = \frac{n_{1.}^{(l)} n_{0.}^{(l)} n_{.1}^{(l)} n_{.0}^{(l)}}{n_{..}^{(l)} n_{..}^{(l)} (n_{..}^{(l)} - 1)}$$

$$(4.2) \quad M^2 = \frac{(|\sum n_{11}^{(l)} - \sum m_{11}^{(l)}| - 0.5)^2}{\sum V(n_{11}^{(l)})}$$

The summary statistics M^2 has approximately a chi-squared distribution with d.f. = 1 under the null hypothesis of conditional independence. Hence, if $M^2 > P_\alpha$, we can reject the null hypothesis of conditional independence and include the edge of I_i and I_j in the interaction graph.

However, the contingency table of s is unknown in our scenario. We apply a modified summary statistics version. From frequency itemsets FS , extract $FS^s = \{FS_{i_1}, \dots, FS_{i_k}\}$ such that $s \subset FS_{i_k}$. For each FS_{i_k} , we build its stratum and compute its summary statistics. If all summary statistics are larger than P_α , we can reject the null hypothesis of conditional independence and include the edge of I_i and I_j in the interaction graph.

4.2.2 Decomposing Independence Graph

THEOREM 4.1. *Graph Factorization ([12, 19]). If there exists a decomposition of independence graph G_s into $(G_{s_a}, G_{s_b}, G_{s_c})$ such that 1) $s_a \perp s_b \mid s_c$ (i.e., the variables in S_a are conditionally independent of those in S_b given the variables in S_c) and neither G_{s_a} nor G_{s_b} empty; and 2) the subgraph G_{s_c} is a clique², the joint density of s admits the factorization*

$$f_s = \frac{f_{s_a \cup s_c} f_{s_b \cup s_c}}{f_{s_c}}$$

²A clique is a subset of vertices which induce a complete subgraph for which the addition of any further vertex renders the induced subgraph incomplete. A graph is complete if all vertices are joined with undirected edges. In other words, the clique is maximally complete.

The theory may be interpreted by the following way: if two disjoint subsets of vertices s_a and s_b are separated by a subset s_c in the sense that all paths from s_a to s_b go through s_c , then the variables in s_a are conditionally independent of those in s_b given the variables in s_c . In other words, the maximum likelihood estimations (MLEs) for the parameters of the model can easily be derived by combining the estimates of the models on the lower dimensional tables represented by the simpler subgraphs. The subgraphs $G_{s_a \cup s_c}$ and $G_{s_b \cup s_c}$ may be further decomposed into subgraphs recursively until the graph is decomposed into basic, irreducible components. Hence, applying a divide-and-conquer approach based on the decompositions will make the procedure applicable to much larger tables.

EXAMPLE 2. *To clarify the concepts and the results presented so far, we use an example shown in Figure 2. The graph in Figure 2(a) has 7 vertices and 11 edges. The edge $\{A, C\}$ is a separator for $\{A, C, G\}$ and $\{A, B, C, D, E, F\}$. The former is a triangle, hence cannot be further decomposed. Similarly, $\{B, D\}$ separates $\{A, B, C, D\}$ and $\{B, D, E, F\}$. Both are a prime subgraph, therefore we have finished the decomposition. From Theorem 4.1, we have*

$$f_{ABCDEFG} = \frac{f_{ACG} f_{ABCDEF}}{f_{AC}} = \frac{f_{ACG} f_{ABCD} f_{BDEF}}{f_{AC} f_{BD}}$$

. The cell values in the original 7-dimensional contingency table (i.e., $ABCDEFG$) can be computed from the low dimensional contingency tables explicitly.

In each step, we search the clique separators of a graph. If there is no clique to be found, it means we get irreducible components and we apply IPF to estimate. Please note if the frequency of any component is already contained in the given frequency set, we can generate its contingency table directly without further decomposition (line 11-12 in Figure 1). An algorithm with a complexity of $O(ne + n^2)$ to find the clique separators of a graph or to find the vertex-sets of the irreducible components of the graphs was presented in [18], where n is the number of vertices and e is the number of edges. We have implemented the decomposition algorithm in our system. Note that decomposition step is determined by the size of independence graph (i.e., the number of variables n and the number of edges e). Our implementation is able to handle clusters with larger number of variables³.

³A widely used graph decomposition package is CoCo [4]. However, it can not decompose a graph with more than 128 variables.

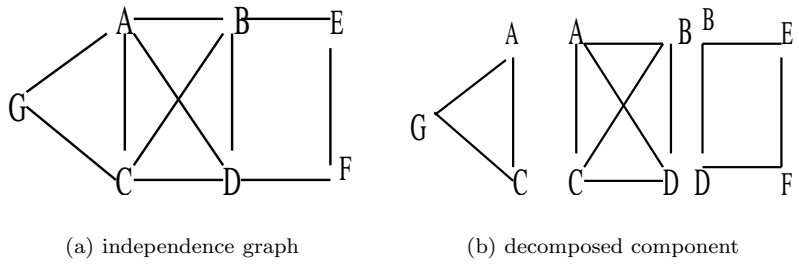


Figure 2: Composition of independence graph

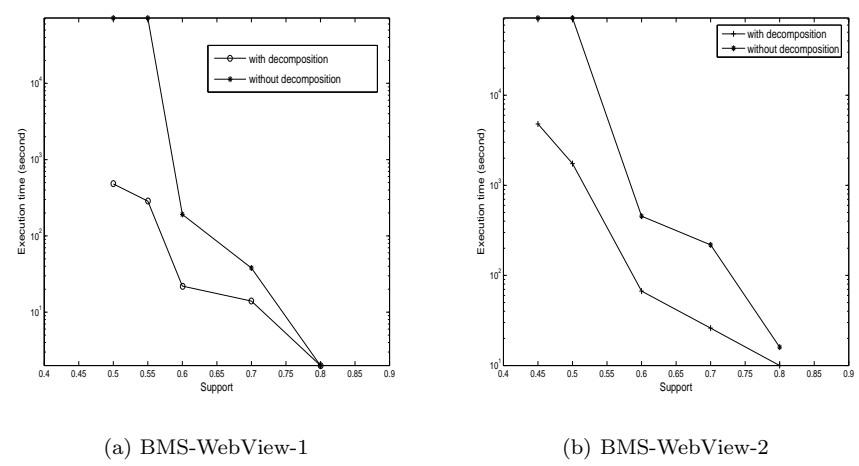


Figure 3: Performance vs. Varying Supports over BMS-WebView1 and BMS-WebView2

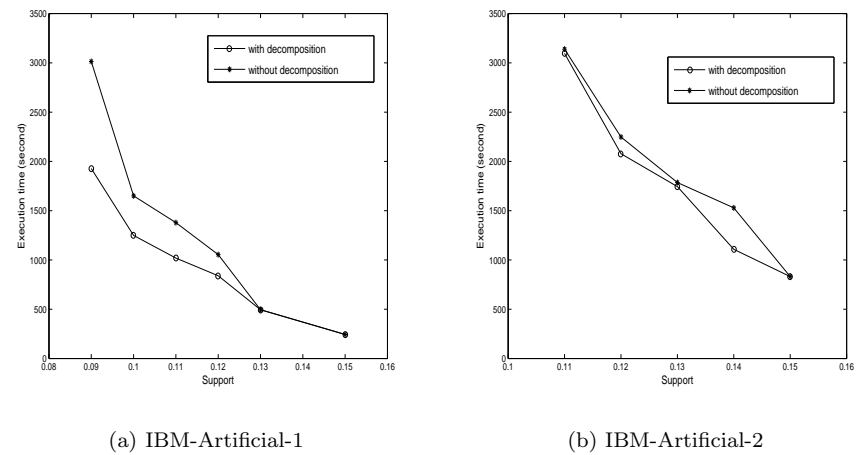


Figure 4: Performance vs. Varying Supports over two IBM Artificial Datasets

4.3 Privacy Aware Generation We have presented how our approach can effectively generate contingency table given the marginal frequencies. It is no wonder there may exist many data sets which satisfy the given frequent itemsets. IPF method itself generates a data set which converges to the maximum likelihood estimation. All the information contained in the generated synthetic data set is from the frequencies of the released frequent itemsets. Though the released frequent itemsets are not considered as confidential information, attackers may still be able to derive or estimate frequencies of some confidential sets.

For example, in the table which records the number of patients visiting physicians to receive treatments, the frequencies on Patient-Doctor and Doctor-Treatment are not sensitive and can be released. However, the Patient-Treatment information is sensitive, and so, confidential. Disclosure detection addresses to what extent attackers can infer cell entry values for the Patient-Treatment table. This entailment of frequent itemsets problem is defined as:

PROBLEM 3. *Entailment of Frequent Itemsets.* Given a finite set $\mathcal{FS} = \{FS_1, \dots, FS_n\}$ together with their frequencies $\{\text{freq}(FS_1), \dots, \text{freq}(FS_n)\}$ discovered from the original database D , plus a target private itemset ps with its original frequency $\text{freq}(ps)$, compute the bound of $\text{freq}(ps)$ entailed from \mathcal{FS} .

If the induced upper and lower bounds are too tight or too close to the actual sensitive value in a cell entry, the information associated with the transactions classified in that cell may be disclosed. The bound or feasibility interval can be obtained by solving the corresponding linear programming problem generally. The problem of determining sharp upper and lower bounds for the cell entries subject to some linear constraints expressed in this form is known to be NP-hard. Recently, Dobra and Fienberg have developed generalized Frechet Bounds for reducible log-linear models with any number of minimal sufficient statistics and have shown the upper and lower bounds can be expressed as explicit functions of marginal totals.

THEOREM 4.2. *Generalized Frechet Bounds for Reducible Models*⁴ [8]. Assume that the released set of marginals is the set of minimum sufficient statistics of a reducible loglinear model. Then the upper bounds for the cell entries in the initial table are the minimum of upper bounds of relevant components, while the lower

bounds are the maximum of zero, or sum of the lower bounds of relevant components minus the separators.

EXAMPLE 3. Using the example shown in Example 2, we have

$$f_{ABCDEFGF} = \frac{f_{ACG}f_{ABCDEF}}{f_{AC}} = \frac{f_{ACG}f_{ABCD}f_{BDEF}}{f_{AC}f_{BD}}$$

where we have three components $\{ACG\}$, $\{ABCD\}$, $\{BDEF\}$ and two separators $\{AC\}$, $\{BD\}$. Using Theorem 4.2, we can compute the bounds of cell values in the original 7-dimensional contingency table (i.e., $ABCDEFGF$) from the low dimensional contingency tables explicitly. If $\{ACG\}$, $\{ABCD\}$ and $\{BDEF\}$ are contained in the released frequency itemsets, we see the upper bounds for the cell entries in $\{ABCDEFGF\}$ induced by $\{ACG\}$, $\{ABCD\}$ and $\{BDEF\}$ are the minimum of the corresponding entries in the fixed marginals, while the lower bounds are the sum of the same entries minus the sum of corresponding entries in the marginals associated with the separators, i.e., $\{AC\}$ and $\{BD\}$. If not all components are contained in the released frequency itemsets, e.g., $\{ABCD\}$ is not contained but its subsets $\{AB\}$, $\{AC\}$, $\{AD\}$, $\{BC\}$, $\{BD\}$, and $\{CD\}$ are contained in the released frequency itemsets, we calculate bounds for the cell entries in the marginal $\{ABCD\}$ given the margins $\{AB\}$, $\{AC\}$, $\{AD\}$, $\{BC\}$, $\{BD\}$, and $\{CD\}$ and then calculate bounds for the cell entries in the marginal $\{ABCDEFGF\}$ given the marginals $\{ACG\}$, $\{BDEF\}$ and the computed marginals $\{ABCD\}$.

In our scenario, customers may specify a list of private itemsets \mathcal{PS} together with the set of released frequent itemsets \mathcal{FS} . Each ps in \mathcal{PS} may be associated with a secure interval which customers do not want attackers to derive the estimation of ps in this interval. Though we can apply Frechet Bounds to compute the upper bound and lower bound of ps given the released \mathcal{FS} , however, we do not know which frequencies should be removed from release list when the computed bounds are contained in its secure interval. Hence our aim here is to find $\hat{\mathcal{FS}}$ such that 1) $\hat{\mathcal{FS}} \subseteq \mathcal{FS}$ and 2) no ps in \mathcal{PS} can be entailed from $\hat{\mathcal{FS}}$ within a given bound. Our heuristic method is sketched as follows. For each private itemset $ps \in \mathcal{PS}$, decompose ps and compute its Frechet Bounds from the current \mathcal{FS} . If its bound exceeds the tolerated threshold, remove its largest component from \mathcal{FS} and recompute the bound until the bound fits in the threshold. We repeat this process for all private itemset and the reduced \mathcal{FS} can be released as $\hat{\mathcal{FS}}$ finally.

⁴An independence graph that is not necessarily decomposable, but still admits a proper decomposition, is called reducible. Our approach assumes reducible models implicitly.

Table 3: IBM Artificial data sets parameter

	ntrans	nitens	tlen	npats	patlen	corr	conf
IBM-Artificial-1	1M	100,000	10	10,000	4	0.25	0.75
IBM-Artificial-2	1M	100,000	12	10,000	6	0.25	0.75

Table 4: Data set characteristics

	trans.	items	max. trans. size	avg. trans. size
BMS-WebView-1	59,602	497	267	2.5
BMS-WebView-2	77,512	3,340	161	5.0

Table 5: The number of clusters and the maximum cluster size at different minimum support levels on four data sets

data set	support(%)	frequent itemset	clusters	max. cluster size
BMS-WebView-1	0.08	9,934	7	231
	0.1	3,648	4	205
	0.2	530	8	86
	0.4	105	5	39
	0.6	32	3	17
	0.8	17	3	9
BMS-WebView-2	0.08	39,500	66	231
	0.1	22,161	52	140
	0.2	3,110	20	77
	0.4	443	14	27
	0.6	130	7	15
IBM-Artificial-1	0.08	17362	2305	19
	0.09	11324	1970	13
	0.10	9080	1629	12
	0.11	6190	1381	12
	0.12	4561	1098	12
	0.13	3342	938	10
	0.15	1222	608	6
IBM-Artificial-2	0.10	15637	1919	18
	0.11	8788	1587	13
	0.12	5521	1235	13
	0.13	3935	1011	12
	0.14	3197	793	12
	0.15	2866	646	11

5 Empirical Evaluation

In this section we show the experiment results with both IBM synthetic data sets and two real data sets. IBM-Artificial-1 and IBM-Artificial-2 are generated using IBM Almaden association rule synthetic data generator. The parameters used are shown in Table 3. Please note all parameters of IBM-Artificial-1 data set are default setting. We also used two real data sets, *BMS-WebView-1*, *BMS-WebView-2* in our experiments. Each transaction in these data sets is a web session consisting of all the product detail pages viewed in that session. These two data sets can be found from KDD-CUP 2000 [11] and were used for performance evaluation of various association rule mining algorithms in [22]. Table 4 characterizes these data sets in terms of the number of transactions, the number of distinct items, the maximum transaction size, and the average transaction size. Our focus here is to show the feasibility of our approach. For each original data set, we first run Apriori algorithm using different support thresholds and extract a set of frequent itemsets (together with their frequencies) for each support threshold value. We then apply our approach to generate a data set using each set of frequent itemsets (together with their frequencies). In section 5.1, we show efficiency of our method by comparing against a method that does not use graphical decomposition. In section 5.2, we show effectiveness of our approach from two aspects. First, mining results (i.e., frequent itemsets) from original and generated one should be similar for certain thresholds. Second, the performance of mining algorithms (e.g., Apriori) on two sets should be similar. Please note we did not introduce any private itemsets in this experiment. Comparing with existing privacy preserving association rule mining approaches when private itemsets are introduced will be our future work. The experiments were conducted in a DELL Dimension 8100, with one 1.7G processor, and 640 Mbytes of RAM.

5.1 Performance Analysis For each data set, we run Apriori algorithm to generate frequent itemsets. Our first step is to merge frequent itemsets into disjoint clusters. Table 5 shows the number of frequent itemsets, the number of disjoint clusters, the average cluster size, and the maximum cluster size by varying support values on each data set. We observe that the items are grouped into a reasonable number of disjoint clusters for all four data sets. Each disjoint cluster contains a subset of items varying dozens to hundreds in most cases. This validates our strategy that we can generate transactions for each cluster and merge them later.

We believe that most real world market basket datasets display this characteristics, i.e., disjoint item

clusters exist. However, we did encounter one real data set, BMS-POS [11], with all items forming one cluster. This data set contains several years of point-of-sale data from an electronics retailer. The transaction in this data set is a customer’s purchase transaction consisting of all the product categories purchased at one time. The reason that all itemsets form one cluster is the BMS-POS data set was transformed from the original point-of-sale data by grouping products into category. Each item thus represents a category, rather than an individual product.

Though each disjoint cluster contains much fewer items, it may still be infeasible or impractical to apply IPF or other linear programming techniques directly on each cluster. So our following steps are to decompose each cluster into components, apply IPF to estimate cell frequencies of contingency table corresponding to each component, and finally calculate cell frequencies of contingency table corresponding to the original cluster. As the complexity of graphical decomposition is $O(ne+n^2)$ where n is the number of vertices and e is the number of edges, it is significantly lower than the complexity of IPF. Hence applying this divide and conquer strategy can significantly improve the performance when there are large clusters which can be decomposed.

Figure 3 shows the execution time of our method of inverse frequent itemset mining with graphical decomposition on BMS-WebView data sets. We also include the execution time of method without graphical decomposition in Figure 3. Under all support values, our method is significantly better (2 or 3 orders) than method without decomposition. For example, under support threshold 0.55, our method uses 286 seconds to generate data while the method without decomposition needs more than 20 hours.

Figure 4 shows the comparison on IBM-Artificial data sets. Though our method is still with better performance, the difference is not as significant as BMS-WebView datasets. The reason is that there are very few clusters which contain more than 10 items. In fact, more than 90 % clusters are single or two items clusters in these two data sets. When the number of items contained in one cluster is less than 10, the time of applying IPF method for each component is trivial. So graphical decomposition does not improve performance significantly as there are very few clusters involving decomposition. This experiment also validates the result discovered in [22], i.e., the artificial data sets generated using IBM Almaden data generator have very different characteristics from real-world data sets.

5.2 Accuracy Analysis As we discussed before, there may exist many data sets which satisfy the given

Table 6: Similarity of mining results on original vs. generated

BMS-Web	support	Jaccard	Dice	Overlap
View-1 s = 0.7	0.3	0.367	0.537	0.964
	0.4	0.507	0.673	0.986
	0.5	0.689	0.816	0.985
	0.6	0.817	0.899	0.985
	0.7	0.940	0.969	0.992
	0.8	0.883	0.938	0.934
	0.9	0.893	0.944	0.954
	1.0	0.887	0.940	0.959
View-2 s = 0.6	0.6	0.696	0.768	1
	0.7	0.708	0.739	0.964
	0.8	0.710	0.830	0.928
	0.9	0.722	0.838	0.976
	1.0	0.701	0.824	0.910
	1.1	0.704	0.826	0.962
	1.2	0.702	0.825	1

frequent itemsets. During data generation, using IPF method simply can generate a data set which converges to the maximum likelihood estimation and graphical decomposition does not lose any information. Hence we can expect to get the same frequent itemsets (and also same frequency values) when we mine from generated data with support values greater or equal to support threshold used to extract frequent itemsets. However, our method introduce errors when we apply graphical decomposition. Recall statistics we used to build independence graph is not complete as we can not compute statistics over all stratum (we can not access the original data set).

$$\begin{aligned}
 D_{jaccard}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{|\mathcal{FS}^0 \cap \mathcal{FS}^f|}{|\mathcal{FS}^0 \cup \mathcal{FS}^f|} \\
 D_{dice}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{2 \times |\mathcal{FS}^0 \cap \mathcal{FS}^f|}{|\mathcal{FS}^0| + |\mathcal{FS}^f|} \\
 D_{overlap}(\mathcal{FS}^0, \mathcal{FS}^f) &= \frac{|\mathcal{FS}^0 \cap \mathcal{FS}^f|}{\min(|\mathcal{FS}^0|, |\mathcal{FS}^f|)}
 \end{aligned}
 \tag{5.3}$$

We use Jaccard, dice and overlap coefficients to measure the similarity between frequent itemsets (\mathcal{FS}^0) mined from original data and frequent itemsets (\mathcal{FS}^f) mined from data set generated using our approach. Results on two IBM-BMS-View data sets are shown in

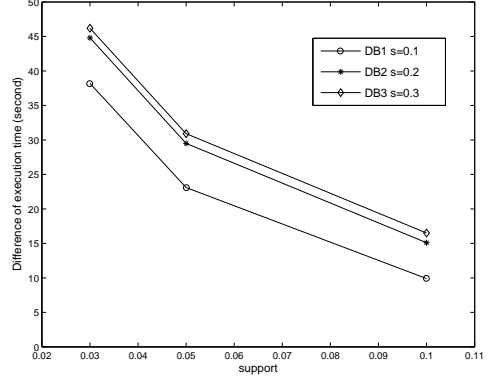


Figure 5: Performance of the Apriori method on three data sets generated using different support values (s=0.1, 0.2, 0.3) from IBM-Artificial-1

Table 6. For each data, we only show result between original and one data set generated due to space limitation. We can see \mathcal{FS}^0 and \mathcal{FS}^f are very similar in terms of three measures when support threshold values used for mining are greater than or equal to support threshold used for data generation. On the other hand, when support threshold values used for mining are less than that used for data generation, \mathcal{FS}^0 and \mathcal{FS}^f are dissimilar as we expected.

Figure 5 shows difference of performance of Apriori mining algorithm when running on original IBM-Artificial-1 and three data sets generated by our approach using support $s = 0.1, 0.2$ and 0.3 respectively. As we expected, the execution time on the first data set DB1 generated using $s = 0.1$ has the smallest difference with that on original one while the execution time on the third data set DB3 generated using $s = 0.3$ has the largest difference. Please note we used IBM-Artificial data instead of BMS-Webview data sets for this experiment because those two real data sets are relatively small.

6 Conclusions and Future Work

In this paper we presented a feasible solution to the NP-Complete problem of inverse frequent set mining. The approach can effectively and efficiently generate a synthetic market basket data set from the released frequent itemsets and their supports. We also presented a heuristic method to screen out confidential frequent itemsets from frequent itemsets used for data generation. The generated synthetic data set can preserve most frequent itemset patterns as the original one without disclosing confidential information. Hence it can be used for benchmarking frequent item set mining algo-

rithms.

There are some aspects of this work that merit further research. Among them, we are trying to figure out better solution for the entailment of frequent itemsets problem, i.e., given a set of frequency constraints plus a set of private itemsets, how to screen those frequency constraints to get the best list of released frequency constraints while guaranteeing no precise information of private itemsets can be derived.

Another aspect that merits further research is how to generate market basket data when only frequency bounds of frequent itemsets are available. In some scenario, customers would only provide several sequences of frequent itemsets with different support thresholds, rather than exact frequency of each frequent itemset. Finally, we will compare our approach with current privacy preserving association rule mining approaches and investigate the tradeoff between privacy and performance for each approach.

7 Acknowledgments

This work was supported in part by U.S. National Science Foundation CCR-0310974. The authors would like to thank Christian Borgelt for providing his implementation of the Apriori algorithm. We would also like to thank IBM Almaden Quest group for providing the market basket data generator.

References

- [1] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Database*, pages 207–216, 1993.
- [2] A. Agresti. *Categorical data analysis*. Wiley, 1990.
- [3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop*, pages 45–52, Nov 1999.
- [4] J. Badsberg. An environment for graphical models. Ph.D. Thesis, Aalborg University, Demark, 1995.
- [5] T. Calders. Computational complexity of itemset frequency satisfiability. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database System*, 2004.
- [6] E. Dasseni, V. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Information Hiding Workshop*, pages 369–383. Pittsburg,PA, April 2001.
- [7] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. *PNAS*, 97(22):11885–11892, 2000.
- [8] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. *Statistical Journal of the United Nations ECE*, 18:363–371, 2001.
- [9] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd Symposium on Principles of Database Systems*, pages 211–222, 2003.
- [10] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. Edmonton, Canada, July 2002.
- [11] KDDCUP2000. <http://www.ecn.purdue.edu/KDDCUP>.
- [12] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
- [13] T. Mielikainen. On inverse frequent set mining. In *Proceedings of the 2nd Workshop on Privacy Preserving Data Mining*, Nov 2003.
- [14] S. Oliveira and O. Zaiane. Protecting sensitive knowledge by data sanitization. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 211–218. Melbourne, Florida, Nov 2003.
- [15] G. Ramesh, W. Maniatty, and M. Zaki. Feasible itemset distributions in data mining: theory and application. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 284–295. San Diego, CA, June 2003.
- [16] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 682–693, August 2002.
- [17] Y. Saygin, V. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *Sigmod Record*, 30(4):45–54, Dec 2001.
- [18] R. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.
- [19] J. Whittaker. *Graphical Models in Applied Mathematical Multivariate Statistics*. Wiley, 1990.
- [20] X. Wu, D. Barbará, and Y. Ye. Screening and interpreting multi-item associations based on log-linear modeling. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 276–285. Washinton D.C, August 2003.
- [21] X. Wu, Y. Wang, and Y. Zheng. Privacy preserving database application testing. In *Proceedings of the ACM Workshop on Privacy in Electronic Society*, pages 118–128, 2003.
- [22] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406. San Francisco, CA, August 2001.