Approximate Inverse Frequent Itemset Mining: Privacy, Complexity, and Approximation

Yongge Wang
SIS Department
UNC Charlotte

Xintao Wu
CS Department
UNC Charlotte

Abstract

In order to generate synthetic basket datasets for better benchmark testing, it is important to integrate characteristics from real-life databases into the synthetic basket datasets. The characteristics that could be used for this purpose include the frequent itemsets and association rules. The problem of generating synthetic basket datasets from frequent itemsets is generally referred to as inverse frequent itemset mining. In this paper, we show that the problem of approximate inverse frequent itemset mining is **NP***-complete. Then we propose and analyze an approximate algorithm for approximate inverse frequent itemset mining, and discuss privacy issues related to the synthetic basket dataset. In particular, we propose an approximate algorithm to determine the privacy leakage in a synthetic basket dataset.*

Keywords: data mining, privacy, complexity

1 Introduction

Since the seminal paper [2], association rule and frequent itemset mining received a lot of attention. By comparing five well-known association rule algorithms using three real-world data sets and the artificial data set from IBM Almaden, Zheng et al. [29] found out that the algorithm performance on the artificial data sets are very different from their performance on real-world data sets. Thus 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. Privacy preserving association rule mining has been a very active research topic in the last few years. In this area, researchers are interested in topics such as preserving the database privacy while mining useful association rules from it. There have been two different approaches to this problem. The first is to disturb the data before delivery for mining so that real values are obscured while preserving statistics on the collection. Some recent work [8, 1, 9, 22, 6, 3, 17, 23, 7] investigates the tradeoff between private information leakage and accuracy of mining results. One problem related to the perturbation based approach is that it can not always fully preserve individual's privacy while achieving precision of mining results [14]. The second approach is distributed privacy preserving association rule mining [25, 13] based on secure multiparty computation[28]. Though this approach can fully preserve privacy, it works only for distributed environment and needs sophisticated protocols (secure multi-party computation based), which makes it infeasible for our scenario.

One potential approach [27] to address this problem is to generate synthetic basket datasets for benchmarking purpose by integrating characteristics from real-world basket datasets that may have influence on the software performance. The frequent sets and their supports (defined as the number of transactions in the basket dataset that contain the items) can be considered to be a reasonable summary of the real-world data set. As observed by Calders [5], association rules for basket dataset can be described by frequent itemsets. Thus it is sufficient to consider frequent itemsets only. Ramesh et al. [21] recently investigated the relation between the distribution of discovered frequent set and the performance of association rule mining. 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.

Informally speaking, in this approach, one first mines frequent itemsets and their corresponding supports from the real-world basket datasets. These frequent itemset support constraints are used to generate the synthetic (mock) dataset which could be used for benchmarking. For this approach, private information should be deleted from the frequent itemset support constraints or from the mock database. The authors of [16, 5] investigate the problem whether there exists a data set that is consistent with the given frequent itemsets and frequencies and show that this problem is **NP**-complete. The frequency of each frequent itemset can be 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 this scenario is far beyond hundreds or thousands (e.g., 2^t, where t is the number of items). Hence it is impractical to apply linear programming techniques directly.

The authors of [21] proposed a method to generate 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 transaction databases generated is much larger than that of original database while the number of items generated is much smaller. We believe the numbers of items and transactions are two important parameters as they may significantly affect the performance of association rule mining algorithms.

Instead of using the exact inverse frequent itemset mining approach, we propose an approach to construct transaction databases which have the same size as the original transaction database and which are approximately consistent with the given frequent itemset constraints. These approximate transaction databases are sufficient for benchmarking purposes. In this paper, we consider the complexity for this approximation problem and study the privacy issues.

We first introduce some terminologies. $\mathcal{I}$ is the finite set of items. A transaction over $\mathcal{I}$ is defined as a pair (tid, I) where I is a subset of $\mathcal{I}$ and tid is a natural number, called the transaction identifier. A transaction database $\mathcal{D}$ over $\mathcal{I}$ is a finite set of transactions over $\mathcal{I}$. For an item set $I \subseteq \mathcal{I}$ and a transaction (tid, J), we say that (tid, J) contains I if $I \subseteq J$. The support of an itemset I in a transaction database $\mathcal{D}$ over $\mathcal{I}$ is defined as the number of transactions T in $\mathcal{D}$ that contains I, and is denoted $support(I, \mathcal{D})$. The frequency of an itemset I in a transaction database $\mathcal{D}$ over $\mathcal{I}$ is defined as

$$freq(I, \mathcal{D}) =_{def} \frac{support(I, \mathcal{D})}{|\mathcal{D}|}.$$

Calders [4, 5] defined the following problems that are related to the inverse frequent itemset mining.

FREQSAT
Instance: An item set $\mathcal{I}$ and a sequence (I_1, f_1), (I_2, f_2), $\cdots$, (I_m, f_m), where $I_i \subseteq \mathcal{I}$ are itemsets and $0 \le f_i \le 1$ are nonnegative rational numbers, for all $0 \le i \le m$.
Question: Does there exist a transaction database $\mathcal{D}$ over $\mathcal{I}$ such that $freq(I_i, \mathcal{D}) = f_i$ for all $0 \le i \le m$?

FFREQSAT (Fixed size FREQSAT)
Instance: An integer n, an item set $\mathcal{I}$, and a sequence (I_1, f_1), (I_2, f_2), $\cdots$, (I_m, f_m), where $I_i \subseteq \mathcal{I}$ are itemsets and $0 \le f_i \le 1$ are nonnegative rational numbers, for all $0 \le i \le m$.
Question: Does there exist a transaction database $\mathcal{D}$ over $\mathcal{I}$ such that $\mathcal{D}$ contains n transactions and $freq(I_i, \mathcal{D}) = f_i$ for all $0 \le i \le m$?

FSUPPSAT
Instance: An integer n, an item set $\mathcal{I}$, and a sequence (I_1, s_1), (I_2, s_2), $\cdots$, (I_m, s_m), where $I_i \subseteq \mathcal{I}$ are itemsets and $s_i \ge 0$ are nonnegative integers, for all $0 \le i \le m$.
Question: Does there exist a transaction database $\mathcal{D}$ over $\mathcal{I}$ such that $\mathcal{D}$ contains n transactions and $support(I_i, \mathcal{D}) = s_i$ for all $0 \le i \le m$?

Obviously, the problem FSUPPSAT is equivalent to the problem FFREQSAT. Calders [4] showed that FREQSAT is **NP**-complete and the problem FSUPPSAT is equivalent to the Intersection Pattern problem IP: given an $n \times n$ matrix C with integer entries, do there exist sets $S_1, \ldots, S_n$ such that $|S_i \cap S_j| = C[i, j]$? Though it is known that IP is **NP**-hard, it is an open problem whether IP belongs to **NP**. Thus it is an open problem whether FSUPPSAT and FFREQSAT belong to **NP**.

In this paper, we will consider the problem of generating transaction databases that approximately satisfy the given frequent itemset support constraints. Section 2 discusses the computational complexity of approximating transaction databases. Section 3 proposes an algorithm to approximately generate an approximate transaction database. Finally, Section 4 discusses privacy issues.

2 Approximations

Though it is an interesting problem to study whether there exists a size n transaction database that satisfies a set of given frequency constraints, it is sufficient for benchmarking purpose to construct a transaction database that is approximately at the size of n and that approximately satisfies the set of given frequency constraints. Thus we define the following problem.

ApproSUPPSAT
Instance: An integer n, an item set $\mathcal{I}$, and a sequence (I_1, s_1), (I_2, s_2), $\cdots$, (I_m, s_m), where $I_i \subseteq \mathcal{I}$ are itemsets and $s_i \ge 0$ are nonnegative integers, for all $1 \le i \le m$.
Question: Does there exist a transaction database $\mathcal{D}$ of n' transactions over $\mathcal{I}$ such that $|n - n'| = O(m)$ and $|support(I_i, \mathcal{D}) - s_i| = O(m)$ for all $0 \le i \le m$?

Note that in the above definition, the approximation errors are based on the parameter m instead of n since for most applications, m is small and n is bigger. Indeed, n could be at the exponential order of m. For performance testing purpose, it is not meaningful to use n as the parameter in these situations. It also straightforward to show that the problem ApproSUPPSAT is equivalent to the following problem: given an integer n, an item set $\mathcal{I}$, and a sequence $(I_1, s_1), (I_2, s_2), \cdots, (I_m, s_m)$, decide whether there exists a transaction database $\mathcal{D}$ over $\mathcal{I}$ with n transactions and $0 \leq support(I_i, \mathcal{D}) - s_i = O(m)$ for all $0 \leq i \leq m$.

In the following we show that ApproSUPPSAT is **NP**-complete. Note that for the non-approximate version FSUPPSAT of this problem, we do not know whether it is in **NP**.

Lemma 2.1 *ApproSUPPSAT* $\in \mathbf{NP}$.

Proof. Since the size of the transaction database is n which might be exponential in the size of the instance input description, it is not possible to guess a transaction database in polynomial time and check whether it satisfies the constraints. In the following, we use other techniques to show that the problem is in **NP**. Let $\mathcal{I}$ be the collection of item sets and (I_1, s_1), (I_2, s_2), $\cdots$, (I_m, s_m) be the sequence of support constraints. Assume that $|\mathcal{I}| = t$. Let $J_0, J_1, \cdots, J_{2^t-1}$ be an enumeration of the 2^t subsets of $\mathcal{I}$ (in particular, let $J_0 = \emptyset$ and $J_{2^t-1} = \mathcal{I}$), and $X_0, X_1, \ldots,$ X_{2^t-1} be 2^t variables corresponding to these itemsets.

Assume that a transaction database $\mathcal{D}$ with $n' = n + O(m)$ transactions contains X_i itemset J_i for each $0 \leq i < 2^t$ and $\mathcal{D}$ approximately satisfies the support constraints (I_1, s_1), (I_2, s_2), $\cdots$, (I_m, s_m). Then there exists an integer k such that the following equations (1) hold for some integer values $X_0, \ldots, X_{2^t-1}$, Z_0, $\ldots$, Z_m. Similarly, if there is an integer k and an integer solution to the equations (1), then there is a transaction database $\mathcal{D}$ with $n' = n + O(m)$ transactions that approximately satisfies the support constraints (I_1, s_1), $\ldots$, (I_m, s_m).

$$\begin{array}{rcl} X_0, \ldots, X_{2^t-1} & \geq & 0 \\ |Z_0|, |Z_1|, \ldots, |Z_m| & \leq & km \\ \sum_{i=0}^{2^t} X_i + Z_0 & = & n \\ \sum_{I_1 \subseteq J_i} X_i + Z_1 & = & s_1 \\ \cdots & & \\ \sum_{I_m \subseteq J_i} X_i + Z_m & = & s_m \end{array} \tag{1}$$

where k is a large enough integer. In another word, if the given instance of the ApproSUPPSAT problem is satisfiable, then the equations (1) have an integer solution. That is, the solution space for the equation (1) is a non-empty convex polyhedron. A simple argument[1] could then be used to show that there is an extreme point $(X_0^0, \ldots, X_{2^t-1}^0)$ (not necessarily an integer point) on this convex polyhedron that satisfies the following property:

- There are at most $m + 1$ non-zero values among the variables X_0^0, $\ldots$, $X_{2^t-1}^0$, Z_0, $\ldots$, Z_m.

Let $Y_i = [X_i^0]$ be the closest integer to X_i^0 for $0 \leq i < 2^t$ and $\mathcal{D}^Y$ be the transaction database that contains Y_i copies of the itemset J_i for each $0 \leq i \leq 2^t$. Then $\mathcal{D}^Y$ contains $n + O(m)$ transactions and $|support(I_i, \mathcal{D}) - s_i| = O(m)$ for all $0 \leq i \leq m$.

In another word, the given instance of the ApproSUPPSAT problem is satisfiable if and only if there exist itemsets $J_1, \ldots, J_{m+1}$ and an integer sequence $x_1, \ldots, x_{m+1}$ such that the transaction database $\mathcal{D}$ consisting of x_i copies of itemset J_i for each $i \leq m$ witnesses the satisfiability. Thus ApproSUPPSAT $\in \mathbf{NP}$ which completes the proof of Lemma. Q.E.D.

Lemma 2.2 *ApproSUPPSAT is* **NP***-hard.*

Proof. The proof is based on an amplification of the reduction in the **NP**-hardness proof for FREQSAT in [4] which is alike the one given for 2SAT in [11]. In the following, we reduce the **NP**-complete problem 3-colorability to ApproSUPPSAT. Given a graph $G = (V, E)$, G is 3-colorable if there exists a 3-coloring function $c : V \to \{R, G, B\}$ such that for each edge (u, v) in E we have $c(u) \neq c(v)$.

For the graph $G = (V, E)$, we construct an instance $\mathcal{A}(G)$ of ApproSUPPSAT as follows. Let $m = 6|V| + 3|E|$, and $n = k_0 m^2$ for some large k_0 (note that we need $k_0 > k$ for the constant k we will discuss later). Let the itemset $I = \{R_v, G_v, B_v : v \in V\}$ and the m support constraints are defined as follows. For each vertex $v \in V$:

$$\begin{array}{l} support(\{R_v\}) = [\frac{n}{3}], support(\{G_v\}) = [\frac{n}{3}], \\ support(\{B_v\}) = [\frac{n}{3}], \\ support(\{R_v, G_v\}) = 0, support(\{R_v, B_v\}) = 0, \\ support(\{G_v, B_v\}) = 0. \end{array}$$

For each edge $(u, v) \in E$:

$$\begin{array}{l} support(\{R_u, R_v\}) = 0, support(\{G_u, G_v\}) = 0, \\ support(\{B_u, B_v\}) = 0. \end{array}$$

In the following, we show that there is a transaction database $\mathcal{D}$ satisfying this ApproSUPPSAT problem if and only if G is 3-colorable.

[1]Similar argument has been used to prove the fundamental theorem of linear optimization in linear programming. See, e.g., [10, 18].

Suppose that c is a 3-coloring of G. Let T be a transaction defined by letting $T_1 = \{C_v : v \in V\}$ where

$$C_v =_{def} \begin{cases} R_v & \text{if } c(v) = R; \\ G_v & \text{if } c(v) = G; \\ B_v & \text{if } c(v) = B. \end{cases}$$

Let transactions T_2 and T_3 be defined by colorings c' and c'' resulting from cyclically rearranging the colors R, G, B in the coloring c. Let the transaction database $\mathcal{D}$ consist of $[\frac{n}{3}]$ copies of each of the transaction T_1, T_2, and T_3 (we may need to add one or two additional copies of T_1 if $3[\frac{n}{3}] \neq n$). Then $\mathcal{D}$ satisfies the ApproSUPPSAT problem $\mathcal{A}(G)$.

Suppose $\mathcal{D}$ is a transaction database satisfying the ApproSUPPSAT problem $\mathcal{A}(G)$. We will show that there is a transaction T in $\mathcal{D}$ from which a 3-coloring of G could be constructed. Let $\mathcal{I}_1$ be the collection of itemsets defined as

$$\begin{aligned} \mathcal{I}_1 = \{&\{R_v, G_v\}, \{R_v, B_v\}, \{G_v, B_v\} : v \in V\} \cup \\ &\{\{R_u, R_v\}, \{G_u, G_v\}, \{B_u, B_v\} : (u, v) \in E\}. \end{aligned}$$

That is, $\mathcal{I}_1$ is the collection of itemset that should have 0 support according to the support constraints. Since $\mathcal{D}$ satisfies $\mathcal{A}(G)$, for each $I' \in \mathcal{I}_1$, $support(I', \mathcal{D}) = 0$ is approximately satisfied. Thus there is a constant $k_1 > 0$ such that at most $k_1 m \times |\mathcal{I}_1| = 3k_1 m(|V| + |E|)$ transactions in $\mathcal{D}$ contain an itemset in $\mathcal{I}_1$. Let $\mathcal{D}_1$ be the transaction database obtained from $\mathcal{D}$ by deleting all transactions that contain itemsets from $\mathcal{I}_1$. Then $\mathcal{D}_1$ contains at least $n - 3k_1 m(|V| + |E|)$ transactions.

For each vertex $v \in V$, we say that a transaction (tid, J) in $\mathcal{D}$ does not contain v if J does not contain any items from $\{R_v, G_v, B_v\}$. Since $\mathcal{D}$ satisfies $\mathcal{A}(G)$, for each $v \in V$, approximately one third of the transactions contain R_v (G_v, B_v, respectively). Thus there is a constant $k_2 > 0$ such that at most $3k_2 m \times |V|$ transactions in $\mathcal{D}$ do not contain some vertex $v \in V$. In another word, there are at least $n - 3k_2 m \times |V|$ transactions J in $\mathcal{D}$ such that J contains v for all $v \in V$.

Let $\mathcal{D}_2$ be the transaction database obtained from $\mathcal{D}_1$ by deleting all transactions J such that J does not contain some vertex $v \in V$. The above analysis shows that $\mathcal{D}_2$ contains at least $n - 3k_1 m(|V| + |E|) - 3k_2 m|V|$ transactions. Let $k = \max\{k_1, k_2\}$. Then we have

$$\begin{aligned} |\mathcal{D}_2| &\geq n - 3km(|V| + |E|) - 3km|V| \\ &= n - km(6|V| + 3|E|) \\ &= n - km^2 \\ &= 3 \cdot k_0 m^2 - km^2 \end{aligned}$$

By the assumption of k_0 at the beginning of this proof, we have $|\mathcal{D}_2| \geq 1$. For any transaction J in $\mathcal{D}_2$, we can define a coloring c for G by letting

$$c(v) = \begin{cases} R & \text{if } J \text{ contains } R_v \\ G & \text{if } J \text{ contains } G_v \\ B & \text{if } J \text{ contains } B_v \end{cases}$$

By the definition of $\mathcal{D}_2$, the coloring c is defined unambiguously. That is, G is 3-colorable.

This completes the proof for **NP**-hardness of ApproSUPPSAT. Q.E.D.

Theorem 2.3 *ApproSUPPSAT is* **NP***-complete.*

Proof. This follows from Lemma 2.1 and Lemma 2.2. Q.E.D.

We showed that the problem ApproSUPPSAT is **NP**-hard. In the proof of Lemma 2.2, we use the fact that the number n of transactions of the target basket database is larger than the multiplication of the number m of support constraints and the approximate error $O(m)$ (that is, n is in the order of $O(m^2)$). In practice, the number n may not be larger than km^2. Then one may wonder whether the problem is still **NP**-complete. If n is very small, for example, at the order of $O(m)$, then obviously, the problem ApproSUPPSAT becomes trivial since one can just construct the transaction database as the collection of n copies of the itemset $\mathcal{I}$ (that is, the entire set of items). This is not a very interesting case since if n is at the order of m, one certainly does not want the approximate error to be at the order of n also. A reasonable problem could be that one defines a constant number γ to replace the approximate error $O(m)$. Then the proof in Lemma 2.2 shows that the problem ApproSUPPSAT with approximate error γ (instead of $O(m)$) is still **NP**-complete if $n > \gamma m$. Tighter bounds could be achieved if weighted approximate errors for different support constraints are given.

3 Generating approximate transaction databases

In this section, we design and analyze a linear program based algorithm to approximate the **NP**-complete problem ApproSUPPSAT. Let $\mathcal{I} = \{e_1, \ldots, e_t\}$ be the collection of items, n be the number of transactions in the desired database $\mathcal{D}$, and (I_1, s_1), (I_2, s_2), $\cdots$, (I_m, s_m) be the sequence of support constraints. According to the proof of Lemma 2.1, if this instance of ApproSUPPSAT is solvable, then there is a transaction database $\mathcal{D}$, consisting of at most $m + 1$ itemsets $J_1, \ldots, J_{m+1}$, that satisfies these constraints. Let $X_1, \ldots, X_{m+1}$ be variables representing the numbers of duplicated copies of these itemsets in $\mathcal{D}$ respectively. That is, $\mathcal{D}$ contains X_i copies of J_i for each i.

For all $i \le m$ and $j \le m+1$, let $x_{i,j}$ and $y_{i,j}$ be variables with the property that $x_{i,j} = X_j \times y_{i,j}$ and

$$y_{i,j} = \begin{cases} 1 & \text{if } I_i \subseteq J_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Then we have $support(I_i, \mathcal{D}) = x_{i,1} + \cdots + x_{i,m+1}$ and the above given ApproSUPPSAT instance could be formulated as the following question.

$$\text{minimize } z_1 + z_2 + \cdots + z_m \quad (3)$$

subject to

$$\begin{cases} X_1 + X_2 + \cdots + X_{m+1} = n, \\ s_i + z_i = x_{i,1} + \cdots + x_{i,m+1}, \\ y_{i,j} = 1 \text{ if } I_i \subseteq J_j \text{ and } y_{i,j} = 0 \text{ otherwise,} \\ x_{i,j} = X_j \times y_{i,j}, \\ z_i,\ X_j \text{ are nonnegative integers,} \end{cases} \quad (4)$$

for $i \le m$ and $j \le m+1$.

The condition set (4) contains the nonlinear equation $x_{i,j} = X_j \times y_{i,j}$ and the nonlinear condition specified in (2). Thus in order to approximate the given ApproSUPPSAT instance using linear program techniques, we need to convert these conditions to linear conditions.

We first use characteristic arrays of variables to denote the unknown itemsets $J_1, \ldots, J_{m+1}$. For any itemset $I \subseteq \mathcal{I}$, let the t-ary array $\chi(I) \in \{0,1\}^t$ be the characteristic array of I. That is, the i-th component $\chi(I)[i] = 1$ if and only if $e_i \in I$. Let $\chi(J_1) = (u_{1,1}, \ldots, u_{1,t})$, $\ldots$, $\chi(J_{m+1}) = (u_{m+1,1}, \ldots, u_{m+1,t})$ be a collection of $(m+1)t$ variables taking values from $\{0,1\}$, representing the characteristic arrays of $J_1, \ldots, J_{m+1}$ respectively.

In order to convert the condition specified in (2) to linear conditions. we first use inner product constraints to represent the condition $I_i \subseteq J_j$. For two characteristic arrays χ_1 and χ_2, their inner product is defined as $\chi_1 \cdot \chi_2 = \chi_1[1] \cdot \chi_2[1] + \cdots + \chi_1[t] \cdot \chi_2[t]$. It is straightforward to show that for two itemsets $I, J \subseteq \mathcal{I}$, we have $\chi(I) \cdot \chi(J) \le \min\{|I|, |J|\}$ and $\chi(I) \cdot \chi(J) = |I|$ if and only if $I \subseteq J$.

Now the following conditions in (5) will guarantee that the condition in (2) is satisfied.

$$\begin{cases} |I_i| \cdot y_{i,j} \le \chi(J_j) \cdot \chi(I_i) \le y_{i,j} + |I_i| - 1 \\ y_{i,j},\ u_{j,k} \in \{0,1\} \end{cases} \quad (5)$$

for all $i \le m$, $j \le m+1$, and $k \le t$. The geometric interpretation of this condition is as follows. If we consider $(\chi(J_j) \cdot \chi(I_i), y_{i,j})$ as a point in the 2-dimensional space (x, y) shown in Figure 1, then $|I_i| y \le x$ defines points below the line passing the points $(0,0)$ and $(|I_i|, 1)$, and $x \le y + |I_i| - 1$ defines the points above the line passing through the points $(|I_i| - 1, 0)$ and $(|I_i|, 1)$. Thus $y_{i,j} = 1$ if and only if $\chi(J_j) \cdot \chi(I_i) = |I_i|$. That is, $y_{i,j} = 1$ if and only if $I_i \subseteq J_j$.

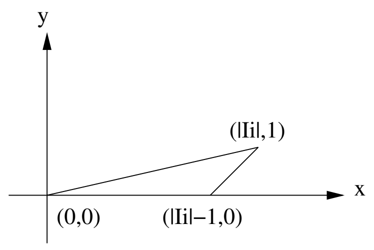

Figure 1: Triangle

The nonlinear equations $x_{i,j} = X_j \times y_{i,j}$ can be converted to the following conditions consisting of inequalities.

$$\begin{cases} x_{i,j} - n y_{i,j} \le 0, \\ X_j \ge x_{i,j}, \\ n y_{i,j} + X_j - x_{i,j} \le n, \\ x_{i,j} \ge 0, \\ y_{i,j} \in \{0,1\}, \end{cases} \quad (6)$$

for all $i \le m$ and $j \le m+1$. The constant n is used in the inequalities due to the fact that $X_j \le n$ for all $j \le m+1$. The geometric interpretation for the above inequalities is described in the following. If we consider $(x_{i,j}, y_{i,j}, X_j)$ as a point in a 3-dimensional space (x, y, X) shown in Figure 2, then

1. $x - ny = 0$ defines the plane passing through points $(0,0,0)$, $(0,0,n)$, and $(n,1,n)$; Thus $x_{i,j} - n y_{i,j} \le 0$ guarantees that $x_{i,j} = 0$ if $y_{i,j} = 0$.

2. $X \ge x$ defines the points above the plane passing through points $(0,0,0)$, $(0,1,0)$, and $(n,1,n)$. This condition together with the condition $y_{i,j} \in \{0,1\}$ guarantees that $x_{i,j} \le X_j$ when $y_{i,j} = 1$.

3. $ny + X - x \le n$ defines the points below the plane passing through points $(0,1,0)$, $(0,0,n)$, and $(n,1,n)$. This condition together with the condition $y_{i,j} \in \{0,1\}$ guarantees that $x_{i,j} \ge X_j$ when $y_{i,j} = 1$. Together with the condition 2, we have $x_{i,j} = X_j$ when $y_{i,j} = 1$.

Note: For the reason of convenience, we introduced the intermediate variables $y_{i,j}$. In order to improve the linear

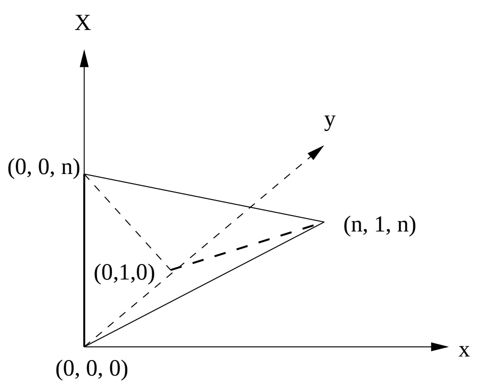

Figure 2: Tetrahedron

program performance, we may combine the conditions (5) and (6) to cancel the variables $y_{i,j}$.

Thus the integer programming formulation for the given AppроSUPPSAT instance is as follows.

$$\text{minimize } z_1 + z_2 + \cdots + z_m \tag{7}$$

subject to conditions (5), (6), and

$$\begin{cases} X_1 + X_2 + \cdots + X_{m+1} = n, \\ s_i + z_i = x_{i,1} + \cdots + x_{i,m+1}, \\ z_i,\ X_j \text{ are nonnegative integers,} \end{cases} \tag{8}$$

for $i \leq m$ and $j \leq m+1$. We first solve the linear relaxation of this integer program. That is, replace the second equation in the condition (5) by

$$0 \leq y_{i,j}, u_{j,k} \leq 1 \quad \text{for all } i \leq m, j \leq m+1, \text{ and } k \leq t$$

and replace the third equation in the condition (8) by

$$z_i,\ X_j \geq 0.$$

Let $o^* = \{(u^*_{j,k}, y^*_{i,j}, x^*_{i,j}, z^*_i, X^*_j) : i \leq m, j \leq m+1, k \leq t\}$ denote an optimal solution to this relaxed linear program. There are several ways to construct an integer solution $\bar{o}$ from o^*. Let $OPT(z; I)$ denote the optimal value of $z_1 + \cdots + z_m$ for a given ApproSUPPSAT instance I and $\overline{OPT}(z; I)$ be the corresponding value for the computed integer solution. For an approximation algorithm, one may prefer to compute a number α such that

$$\overline{OPT}(z; I) \leq \alpha OPT(z; I).$$

Theorem 2.3 shows that it is **NP**-hard to approximate the ApproSUPPSAT by an additive polynomial factor. Thus $\overline{OPT}(z; I)$ is not in the order of $O(m)$ in the worst case for any polynomial time approximation algorithms, and it is not very interesting to analyze the worst case for our algorithm.

In the Appendix, we present two simple naive rounding methods to get an integer solution $\bar{o}$ from o^*. We then present two improved randomized and derandomized rounding methods. The complexity analysis of the approximation algorithm are also given in the appendix.

4 Privacy issues

Wang, Wu, and Zheng [26] considered general information disclosure in the process of mock database generation. In this section, we discuss privacy disclosures in synthetic transaction databases. Confidential information in transaction databases may be specified as a collection of itemsets and their corresponding support (frequency) intervals. Let $\mathcal{P}$ be a set defined as follows.

$$\mathcal{P} = \{(I_i, s_i, S_i) : I_i \subseteq \mathcal{I}, i \leq l\}.$$

We say that a (synthetic) transaction database $\mathcal{D}$ does not disclose confidential information specified in $\mathcal{P}$ if one cannot infer that

$$s_i \leq \mathit{support}(I_i; \mathcal{D}) \leq S_i$$

for all $(I_i, s_i, S_i) \in \mathcal{P}$. Similarly, we say that a support constraint set $\mathcal{S} = \{(I'_1, s_1), \ldots, (I'_m, s_m)\}$ does not disclose confidential information specified in $\mathcal{P}$ if for each element $(I_i, s_i, S_i) \in \mathcal{P}$, there is a transaction database $\mathcal{D}_i$ that satisfies all support constraints in $\mathcal{S}$ and

$$\mathit{support}(I_i, \mathcal{D}_i) \notin [s_i, S_i].$$

For the synthetic transaction database generation, there are two scenarios for potential private information disclosure. In the first scenario, the database owner uses the following procedure to generate the synthetic transaction database:

1. use a software package to mine the real-world transaction database to get a set of itemset support (frequency) constraints;

2. use a software package based on our linear program methods to generate a synthetic transaction database $\mathcal{D}$ from the support (frequency) constraints;

3. release the synthetic transaction database $\mathcal{D}$ to the public.

In this scenario, the mined support (frequency) constraints are not released to the public and only the synthetic transaction database is released. In this case, it is straightforward to protect the confidential information specified in

$\mathcal{P}$. The database owner proceeds according to the above steps until step 3. Before releasing the synthetic transaction database $\mathcal{D}$, he can delete the confidential information as follows.

- For each $(I_i, s_i, S_i) \in \mathcal{P}$, chooses a random number $r_i \leq n$, where n is the total number of transactions. We distinguish the following two cases:
    1. If $u_i = support(I_i, \mathcal{D}) - r_i < 0$, then chooses a random series of $-u_i$ transactions t_j that do not contain the itemset I_i, and modify these transactions to contain the itemset I_i.
    2. If $u_i = support(I_i, \mathcal{D}) - r_i \geq 0$, then chooses a random series of u_i transactions t_j that contain the itemset I_i, and modify these transactions in a random way so that they do not contain the itemset I_i.

After the above process, the resulting transaction database contains no confidential information specified in $\mathcal{P}$ and the database owner is ready to release it.

In the second scenario, the database owner uses the following procedure to generate the synthetic transaction database:

1. use a software package to mine the real-world transaction database to get a set of itemset support (frequency) constraints;
2. release the support (frequency) constraints to the public;
3. a customer who has interest in a synthetic transaction database generates a synthetic transaction database $\mathcal{D}$ from the published support (frequency) constraints using a software package based on our linear program methods.

In this scenario, the mined support (frequency) constraints are released to the public directly. Thus the database owner wants to make sure that no confidential information specified in $\mathcal{P}$ is contained in these support (frequency) constraints. Without loss of generality, we assume that there is a single element (I, s, S) in $\mathcal{P}$ and the mined support constraints are $\mathcal{S} = \{(I_i, s_i) : i \leq m\}$. $\mathcal{S}$ contains the confidential information (I, s, S) if and only if for each transaction database $\mathcal{D}$ which is consistent with $\mathcal{S}$, we have $support(I; \mathcal{D}) \in [s, S]$. In another word, $\mathcal{S}$ does not contain the confidential information (I, s, S) if and only if there exists an integer s' with $s' < s$ or $S < s' < n$ such that $\mathcal{S} \cup \{(I, s')\}$ is consistent. That is, there is a transaction database $\mathcal{D}$ that satisfies all support constraints in $\mathcal{S} \cup \{(I, s')\}$. In the following, we show that there is even no efficient way to approximately decide whether a given support constraint set contains confidential information. We first define the problem formally.

ApproPrivacy
Instance: An integer n, an item set $\mathcal{I}$, a support constraint set $\mathcal{S} = \{(I'_1, s'_1), \cdots, (I'_m, s'_m)\}$, and a set $\mathcal{P} = \{(I_i, s_i, S_i) : I_i \subseteq \mathcal{I}, i \leq l\}$.
Question: For all transaction database $\mathcal{D}$ of n transactions over $\mathcal{I}$ with $|support(I'_i, \mathcal{D}) - s'_i| = O(m)$ for all $0 \leq i \leq m$, do we have $support(I_i, \mathcal{D}) \in [s_i, S_i]$ for all $i \leq l$? If the answer is yes, we write $\mathcal{S} \models_a \mathcal{P}$.

By Theorem 2.3, we have the following result. Similar **NP**-hardness results for exact frequency constraints inference have been obtained in [4, 5, 16].

Theorem 4.1 *ApproPrivacy is co***NP***-complete.*

Proof. $\mathcal{S} \not\models_a \mathcal{P}$ if and only if there is a transaction database $\mathcal{D}$ and an index $j \leq l$ such that $\mathcal{D}$ satisfies $\mathcal{S} \cup \{(I_j, support(I_j, \mathcal{D}) < s_i)\}$ or $\mathcal{D}$ satisfies $\mathcal{S} \cup \{(I_j, support(I_j, \mathcal{D}) > S_i)\}$ approximately. Thus the theorem follows from Theorem 2.3. Q.E.D.

Thus there is no efficient way for the database owner to decide whether a support constraint set $\mathcal{S}$ leaks confidential information specified in $\mathcal{P}$. In practice, however, we can use the linear program based approximation algorithms that we have discussed in Section 3 to compute the confidence level about private information leakage as follows.

1. Convert the condition $\mathcal{S} \cup \{(I, s') : s' < s \text{ or } S < s' \leq n\}$ to an integer linear program in the format of (8). Note that the condition "$s' < s$ or $S < s' \leq n$" is equivalent to the existential clause $\exists s' ((s' < s) \vee (S < s' \leq n))$. Thus it is straightforward to convert it to integer linear program conditions.
2. Let the confidence level be $c = \sum_{i=1}^{m} z_i$. The smaller c, the higher confidence. In the ideal case of $c = 0$, we have found an itemset transaction database $\mathcal{D}$ that witnesses that no confidential information specified by (I, s, S) is leaked in $\mathcal{S}$.

If the database owner thinks that the confidence value $c = \sum_{i=1}^{m} z_i$ obtained in the above procedure is too larger (thus confidence level is too low). He may use the following procedure to delete potential confidential information from the support constraint set.

1. Let i be the number that maximizes $\max_{(I_i, s_i) \in \mathcal{S}} |I \cap I_i|$.
2. Modify the value s_i to be a random value.

3. Approximately revise support constraint values in $\mathcal{S}$ to make it consistent. For example, to make it satisfy the monotonic rule. Since it is **NP**-hard to determine whether a support constraint set is consistent, we can only revise the set $\mathcal{S}$ to be approximately consistent.

It should be noted that after the above process, the resulting support constraint set may become inconsistent. Thus in the next round, the value $c = \sum_{i=1}^{m} z_i$ may be larger. If that happens, the larger value c does not interpret as the privacy confidence level. Instead, it should be interpreted as an indicator for inconsistency of the support constraint set. Thus the above privacy deletion procedure should only be carried out one time.

We should note that even if the confidence level is higher, (that is, $c = \sum_{i=1}^{m} z_i$ is small), there is still possibility that the confidential information specified by (I, s, S) is leaked in theory. That is, for each transaction database $\mathcal{D}$ that satisfies the constraints $\mathcal{S}$, we have $support(I, \mathcal{D}) \in [s, S]$. However, no one may be able to recover this information since it is **NP**-hard to infer this fact. Support constraint inference has been extensively studied by Calders in [4, 5].

It would be interesting to consider conditional privacy-preserving synthetic transaction database generations. That is, we say that no private information is leaked unless some hardness problems are solved efficiently. This is similar to the methodologies that are used in public key cryptography. For example, we believe that RSA encryption scheme is secure unless one can factorize large integers.

In our case, we may assume that it is hard on average to efficiently solve integer linear programs. Based on this assumption, we can say that unless integer linear programs could be solved efficiently on average, no privacy specified in $\mathcal{P}$ is leaked by $\mathcal{S}$ if the computed confidence level $c = \sum_{i=1}^{m} z_i$ is small.

5 Conclusions

In this paper, we discussed the general problems regarding privacy preserving synthetic transaction database generation for benchmark testing purpose. In particular, we showed that this problem is generally **NP**-hard. Approximation algorithms for both synthetic transaction database generation and privacy leakage confidence level approximation have been proposed. These approximation algorithms include solving a continuous variable linear program. According to [15], linear problems having hundreds of thousands of continuous variables are regularly solved. Thus if the support constraint set size is in the order of hundreds of thousands, then these approximation algorithms are efficient on regular Pentium-based computers. If more constraints are necessary, then more powerful computers are needed to generate synthetic transaction databases.

References

[1] D. Agrawal and C. Agrawal. On the design and quantification of privacy preserving data mining algorithms. In: *Proc. 20th Symposium on Principles of Database Systems (PODS)*, 2001.

[2] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of ACM SIGMOD International Conference on Management of Database*, pages 207–216, 1993.

[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. *Proc. IEEE Knowledge and Data Engineering Exchange Workshop*, pages 45–52, 1999.

[4] T. Calders. *Axiomatization and Deduction Rules for the Frequency of Itemsets*. PhD Thesis, Universiteit Antwerpen, 2003.

[5] T. Calders. Computational complexity of itemset frequency satisfiability. In: *Proc. 23rd ACM PODS 04*, pages 143–154, ACM Press, 2004.

[6] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In: *Proc. 4th International Information Hiding Workshop*, pages 369-383, 2001.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. *Proc. 22nd Symposium on Principles of Database Systems (PODS)*, pages 202–210, 2003.

[8] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In: *Proc. the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database System (PODS)*, pages 211–222, 2003.

[9] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In: *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, Edmonton, Canada, 2002.

[10] R. Fagin, J. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, **87**(1,2):78–128, 1990.

[11] G. Georgakopoulos, D. Kavvadias, and C. Papadimitriou. Probabilistic satisfiability. *J. of Complexity*, **4**:1–11, 1988.

[12] D. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.* **9**:256–278, 1974.

[13] M. Kantarcioglu and C. Clifton. Privacy preserving distributed mining of association rules on horizontally partitioned data. In: *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 24–31, 2002.

[14] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the Privacy Preserving Properties of Random Data Perturbation Techniques. In: *Proc. 3rd International Conference on Data Mining*, pages 99-106, 2003.

[15] Linear Programming Frequently Asked Questions. `http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html`

[16] T. Mielikäinen. On inverse frequent set mining. In: *Proc. of 2nd Workshop on Privacy Preserving Data Mining (PPDM)*, pages 18–23, IEEE Computer Society, 2003.

[17] S. Oliveira and O. Zaiane. Protecting sensitive knowledge by data sanitization. In: *Proc. 3rd IEEE International Conference on Data Mining*, pages 211–218, 2003.

[18] C. Potts. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Appl. Math.* **10**:155–164, 1985.

[19] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer problems. *Journal of Comp. Sys. Sci.*, **37**:130–43, 1988.

[20] P. Raghavan and C. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* **7**:365–374, 1987.

[21] G. Ramesh, W. Maniatty, and M. Zaki. Feasible itemset distributions in data mining: theory and application. In: *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 284–295, 2003.

[22] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In: *Proc. 28th International Conference on Very Large Data Bases*, pages 682–693, 2002.

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

[24] D. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Combinatorial Optimization*, pages 355–398. AMS Press, 1995.

[25] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In: *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[26] Y. Wang, X. Wu, and Y. Zheng. Privacy preserving data generation for database application performance testing. In: *Proc. 1st Int. Conf. on Trust and Privacy in Digital Business (TrustBus '04, together with DEXA)* , Lecture Notes in Computer Science 3184, pages 142-151, 2004, Springer-Verlag.

[27] X. Wu, Y. Wu, Y. Wang, and Y. Li. Privacy aware market basket data set generation: a feasible approach for inverse frequent set mining. In: *Proc. 5th SIAM International Conference on Data Mining*, 2005.

[28] A. Yao. How to generate and exchange secrets. In: *Proc. 27th IEEE FOCS*, pages 162–167, 1986.

[29] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proc. of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406. ACM Press, 2001.

6 Appendix: Rounding methods

Method 1: rounding $u^*_{j,k}$

Construct an integer solution $\bar{o} = (\bar{u}_{j,k}, \bar{y}_{i,j}, \bar{x}_{i,j}, \bar{z}_i, \bar{X}_j)$ by rounding $u^*_{j,k}$ to their closest integers, rounding X^*_j to their almost closest integers so that $\bar{X}_1 + \cdots + \bar{X}_{m+1} = n$, and computing $\bar{y}_{i,j}, \bar{x}_{i,j}$, and $\bar{z}_i$ according to their definitions. That is, for each $j \le m+1$ and $k \le t$ set

$$\bar{u}_{j,k} = \begin{cases} 1 & \text{if } u^*_{j,k} \ge 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

For the rounding of X^*_j, first round X^*_j to their closest integers $[X^*_j]$. Then randomly add/subtract 1's to/from these values according to the value of $\bar{X}_1 + \cdots + \bar{X}_{m+1} - n$ until $\bar{X}_1 + \cdots + \bar{X}_{m+1} = n$.

From the construction, it is clear that $\bar{o}$ is a feasible solution of the integer program. The rounding procedure will introduce the following errors to the optimal solution:

1. By rounding $\{u^*_{j,k} : i \le m, k \le t\}$, the values in $\{\chi(I_i) \cdot \chi(J_j) : i \le m, j \le m+1\}$ change. Thus the values in $\{\bar{y}_{i,j} : i \le m, j \le m+1\}$ will change. Thus the values in $\{\bar{x}_{i,j} : i \le m, j \le m+1\}$ will be different from the values in $\{x^*_{i,j} : i \le m, j \le m+1\}$.

2. By rounding $\{X^*_j : j \le m+1\}$, the values of $\{\bar{x}_{i,j} : i \le m, j \le m+1\}$ will change also.

Method 2: rounding $x^*_{i,j}$

Construct an integer solution $\bar{o} = (\bar{u}_{j,k}, \bar{y}_{i,j}, \bar{x}_{i,j}, \bar{z}_i, \bar{X}_j)$ by rounding $x^*_{i,j}$ to 0 or X^*_j and computing the other values according to their definitions or relationships. That is, first round X^*_j to their closest integers $[X^*_j]$. Then randomly add/subtract 1's to/from these values according to the value of $\bar{X}_1 + \cdots + \bar{X}_{m+1} - n$ until $\bar{X}_1 + \cdots + \bar{X}_{m+1} = n$. Now round $x^*_{i,j}$ as follows. Let

$$\bar{x}_{i,j} = \begin{cases} \bar{X}_j & \text{if } x^*_{i,j} \ge 0.5\bar{X}_j, \\ 0 & \text{otherwise.} \end{cases}$$

J_j's could be computed by setting

$$J_j = \cup_{\bar{x}_{i,j} = \bar{X}_j} I_i.$$

The values of $\bar{u}_{j,k}$ and $\bar{y}_{i,j}$ can be derived from J_j easily. We still need to further update the values of $\bar{x}_{i,j}$ by using the current values of $\bar{y}_{i,j}$ since we need to satisfy the requirements $x_{i,j} = X_j \times y_{i,j}$.

From the construction, it is clear that $\bar{o}$ is a feasible solution of the integer program. The rounding procedure will introduce the following errors to the optimal solution:

1. By rounding $\{x^*_{i,j} : i \le m, j \le m+1\}$, we need to update the values of $\bar{y}_{i,j}$, which again leads to the update of values of $\bar{x}_{i,j}$.

2. By rounding $\{X^*_j : j \le m+1\}$, the values in $\{\bar{x}_{i,j} : i \le m, j \le m+1\}$ will change also.

Method 3: randomized and derandomized rounding

For quite a few **NP**-hard problems that are reduced to integer programs, naive round methods remain to be the ones with best known performance guarantee. Our methods 1 and 2 are based on these naive rounding ideas. In last decades, randomization and derandomization methods (see, e.g., [24, 19]) have received a great deal of attention in algorithm design. In this paradigm for algorithm design, a randomized algorithm is first designed, then the algorithm is "derandomized" by simulating the role of the randomization in critical places in the algorithm. In this section, we will design a randomized and derandomized rounding approach to obtain an integer solution $\bar{o}$ from o^* with performance of at least the expectation. It is done by the method of conditional probabilities.

In rounding method 1, we round $u^*_{j,k}$ to its closest integer. In a random rounding [20], we set the value of $\bar{u}_{j,k}$ to 1 with probability $u^*_{j,k}$ and to 0 with probability $1 - u^*_{j,k}$ (independent of other indices).

In rounding method 2, we round $x^*_{i,j}$ to the closest value among 0 and $\bar{X}_j$. In a random rounding [20], we set the value of $\bar{x}_{i,j}$ to $\bar{X}_j$ with probability $\frac{x^*_{i,j}}{\bar{X}_j}$ and to 0 with probability $1 - \frac{x^*_{i,j}}{\bar{X}_j}$ (independent of other indices).

A random rounding approach produces integer solutions with an expected value z_0 for $\sum_{i=1}^m z_i$. An improved rounding approach (derandomized rounding) produces integer solutions with $\sum_{i=1}^m z_i$ guaranteed to be no larger than the expected value z_0. In the following, we illustrate our method for the random rounding based on the rounding methods 1 and 2.

Randomized and derandomized rounding of $x^*_{i,j}$. We determine the value of an additional variable in each step. Suppose that $\{\bar{x}_{i,j} : (i,j) \in I_0\}$ has already been determined, and we want to determine the value of $\bar{x}_{i_0,j_0}$ with $(i_0, j_0) \notin I_0$. We compute the conditional expectation for $\sum_{i=1}^m z_i$ of this partial assignment first with $\bar{x}_{i_0,j_0}$ set to zero, and then again with it set to $\bar{X}_{j_0}$. If we set $\bar{x}_{i_0,j_0}$ according to which of these values is smaller, then the conditional expectation at the end of this step is at most the conditional expectation at the end of the previous step. This implies that at the end of the rounding, we get at most the original expectation.

In the following, we show how to compute the conditional expectation. At the beginning of each step, assume that for all entries (i',j') in I_0, $\bar{x}_{i',j'}$ has been determined already and we want to determine the value of $\bar{x}_{i_0,j_0}$ for $(i_0,j_0) \notin I_0$ in this step.

In order to compute the conditional expectation of $\sum_{i=1}^m z_i$, we first compute the probability Prob$[I_i \subseteq J_j]$ for all $(i,j) \notin I_0$. For each $j \leq m+1$, let

$$J_j^0 = \bigcup_{I_{i'} \subseteq J_j, (i',j) \in I_0} I_{i'}$$

If $I_i \subseteq J_j^0$, then we have $I_i \subset J_j$ and Prob$[I_i \subseteq J_j] = 1$. Otherwise, continue with the following computation. By regarding $\frac{x^*_{i,j}}{\bar{X}_j}$ as the probability that $x^*_{i,j}$ takes the value $\bar{X}_j$, we know that with at least probability $\frac{x^*_{i,j}}{\bar{X}_j}$ we have $I_i \subseteq J_j$. However, the actual probability may be larger since other entries $I_{i'}$ with $I_i \cap I_{i'} \neq \emptyset$ may contribute items to J_j, which may lead to the inclusion of I_i in J_j. First we define the following sets.

$$\begin{aligned} L_{i,j} &= \{1,\ldots,i-1,i+1,\ldots m\} \setminus \{i' : (i',j) \in I_0\} \\ U_{i,j} &= \left\{ K \subseteq L_{i,j} : I_i \subseteq J_j^0 \cup \bigcup_{i' \in K} I_{i'} \right\}, \end{aligned}$$

and

$$U'_{i,j} = \{K \in U_{i,j} : \text{there is no } K' \in U_{i,j} \text{ such that } K' \subset K\}.$$

For each $K \in U'_{i,j}$, let

$$p(i,j,K) = \prod_{i' \in K} \frac{x^*_{i',j}}{\bar{X}_j}.$$

Then the probability Prob$[I_i \subseteq J_j]$ can be approximated as

$$\text{Prob}[I_i \subseteq J_j] = \frac{x^*_{i,j}}{\bar{X}_j} + \left(1 - \frac{x^*_{i,j}}{\bar{X}_j}\right) \sum_{K \in U'_{i,j}} p(i,j,K).$$

Note that we say that we approximate the probability Prob$[I_i \subseteq J_j]$ since in the computation, we assume that Prob$[I_{i'} \subseteq J_j] = \frac{x^*_{i',j}}{\bar{X}_j}$ for other i' which may not be true. If necessary, we can improve the approximation by iteration. That is, repeat the above procedure for several rounds and, in each round, use the approximated probabilities for Prob$[I_{i'} \subseteq J_j]$ from the previous round. If sufficient rounds are repeated, the probability will converge in the end.

Since we have the probabilities Prob$[I_i \subseteq J_j]$ for all $(i,j) \notin I_0$ now, it is straightforward to compute the conditional expectation of $E(\sum_{i=1}^m z_i) = \sum_{i=1}^m E(z_i)$. The expected value for z_i is

$$E(z_i) = E\left(\sum_{j=1}^{m+1} x_{i,j}\right) - s_i = \sum_{j=1}^{m+1} \bar{X}_j \cdot \text{Prob}[I_i \subseteq J_j] - s_i.$$

Randomized and derandomized rounding of $u^*_{j,k}$. We determine the value of an additional variable in each step. Suppose that $\{\bar{u}_{j,k} : (j,k) \in I_0\}$ has already been determined, and we want to determine the value of $\bar{u}_{j_0,k_0}$ with $(j_0,k_0) \notin I_0$. We compute the conditional expectation for $\sum_{i=1}^m z_i$ of this partial assignment first with $\bar{u}_{i_0,j_0}$ set to zero, and then again with it set to 1. If we set $\bar{u}_{j_0,k_0}$ according to which of these values is smaller, then the conditional expectation at the end of this step is at most the conditional expectation at the end of the previous step. This implies that at the end of the rounding, we get at most the original expectation.

According to our analysis in the randomized and derandomized rounding of $x^*_{i,j}$, it is sufficient to compute the probability Prob$[I_i \subseteq J_j]$ for all (i,j). Assume $\mathcal{I} = \{e_1,\ldots,e_t\}$ and $I_i = \{e_{i_1}, \ldots, e_{i_{|I_i|}}\}$. Set

$$\text{Prob}[I_i \subseteq J_j] = \hat{u}_{j,i_1} \times \cdots \times \hat{u}_{j,i_{|I_i|}}$$

where

$$\hat{u}_{j,i_s} = \begin{cases} \bar{u}_{j,i_s} & \text{if } (j,i_s) \in I_0, \\ u^*_{j,i_s} & \text{otherwise} \end{cases}$$

for $s \leq |I_i|$. Using Prob$[I_i \subseteq J_j]$, one can compute the conditional expectation of $\sum_{i=1}^m z_i$ as in the case for rounding of $x^*_{i,j}$.

Complexity analysis of the approximation algorithm

In the integer linear program formulation of our problem, we have $t(m+1)$ variables $u_{j,k}$, $m+1$ variables X_j, $m(m+1)$ variables $x_{i,j}$, $m(m+1)$ variables $y_{i,j}$, and m variables z_i. In total, we have $t(m+1) + 2m^2 + 4m + 1$ variables.

There are $(m+1)(2m+t)$ constraints in the condition (5), $4m(m+1)$ constraints in the condition (5), and $3m+2$ constraints in the condition (8). Thus we have $6m^2 + 9m + mt + t + 2$ constraints in total.

The rounding, randomized, and derandomized rounding algorithms could be finished in $O(tm^3)$ steps. Thus the major challenge is to solve the relaxed continuous variables linear program. According to [15], hundreds of thousands of continuous variables are regularly solved. Thus our approximation algorithm are efficient when m and t take reasonable values.