Evaluating the Privacy Implications of Frequent Itemset Disclosure

Edoardo Serra, Jaideep Vaidya, Haritha Akella, Ashish Sharma

Evaluating the Privacy Implications of Frequent Itemset Disclosure*

Edoardo Serra[1], Jaideep Vaidya[2], Haritha Akella[1], and Ashish Sharma[1]

[1]CS Department, Boise State University, USA
[2]MSIS Department, Rutgers University, USA

Abstract. Frequent itemset mining is a fundamental data analytics task. In many cases, due to privacy concerns, only the frequent itemsets are released instead of the underlying data. However, it is not clear how to evaluate the privacy implications of the disclosure of the frequent itemsets. Towards this, in this paper, we define the k-distant-IFM-solutions problem, which aims to find k transaction datasets whose pair distance is maximized. The degree of difference between the reconstructed datasets provides a way to evaluate the privacy risk. Since the problem is NP-hard, we propose a 2-approximate solution as well as faster heuristics, and evaluate them on real data.

Keywords: Inverse Frequent itemset mining, column generation

1 Introduction and Related Work

Frequent itemset mining [1] is a crucial data mining task which has numerous applications in knowledge discovery such as recommendation, classification, etc. Many efficient implementations exist, [5], all of which assume that the underlying database is accessible to the data miner. However, often privacy concerns prohibit the direct release of data. Since frequent itemsets can serve as a good proxy for the underlying data and still enable different kinds of analysis, often they are released instead. Prior work has examined whether it is possible to find the original dataset from the frequent itemsets, defined as the Inverse frequent set mining (IFM) problem and studied from several different perspectives[9,7,6,4]. IFM aims to find a transaction dataset D that satisfies a given set of itemset support constraints (i.e., the support or frequency of an itemset should be contained within the specified numeric interval). Wang and Wu[13] also introduced the `ApproSUPP` problem, where they asked whether it is possible to satisfy the various support constraints in an approximate fashion and presented an ILP formulation along with heuristic strategies. Several alternative heuristics have

* This work was supported by Idaho Global Entrepreneurial Mission (IGEM) program Grant 131G106011 (Precision Ag - Increasing Crop), the National Science Foundation Grant CNS-1422501 and the National Institutes of Health Award R01GM118574. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

also been proposed[14,10] However, while IFM provides a measure of the degree of difficulty in inverting a set of support constraints into *a* dataset producing these, there is no notion of how different that dataset is from the original that needs to be protected.

In this paper, we precisely tackle this problem. We formulate a new problem called the k-distant-IFM-solutions which combines the IFM problem with elements of K-anonymity [11] and L-diversity [8]. Specifically, the problem consists in finding k IFM solutions (transaction datasets) whose pair distance is maximized. This ensures that we have at least k different solutions to the IFM problem that are all potentially quite different. Since any of these could be the source, and are as different as possible from each other, this gives a minimum bound on the degree of privacy afforded by the frequent itemsets. We show that the problem is NP-hard, and give a 2-approximation based on the greedy strategy. However, given the complexity of the underlying problem, we also develop a heuristic based on an ILP formulation (see [12]) that is quite efficient. Thus, our work is orthogonal to all of the prior work, since it considers the problem of finding multiple datasets meeting the given set of support constraints, and provides a better estimate of the risk of disclosure through the frequent itemsets.

2 Problem Statement

Let $\mathcal{I}$ be a set of items. An itemset I is a subset of $\mathcal{I}$. A transaction Dataset $\mathcal{D}$ is a pair $(T_\mathcal{D}, \#_\mathcal{D})$, where $T_\mathcal{D}$ is a set of transactions (i.e. itemsets) contained in $\mathcal{D}$ and $\#_\mathcal{D} : 2^\mathcal{I} \rightarrow N$ is a function assigning to each transaction a number of duplicates such that if $t \in T_\mathcal{D}$ then $\#_\mathcal{D}(t) > 0$, otherwise $\#_\mathcal{D}(t) = 0$.

Example 1. Let $\mathcal{I} = \{a, b, c\}$. Following is an example of a transaction database where $\#_\mathcal{D}(t) = 0$ for each transaction t that is not present in $T_\mathcal{D}$ and $\#_\mathcal{D}(t) > 0$ for transactions that are present in $T_\mathcal{D}$.

D
{a,b}
{a}
{a,b}
{a,b}
{a,b,c}
{a,b,c}

$\Rightarrow$

$T_\mathcal{D}$	$\#_\mathcal{D}$
{a}	1
{a,b}	3
{a,b,c}	2

Given an itemset I, the support of I w.r.t. $\mathcal{D}$ is $support(I, \mathcal{D}) = \sum_{t \in T_\mathcal{D}, I \subseteq t} \#_\mathcal{D}(t)$ and its frequency is $frequency(I, \mathcal{D}) = \frac{support(I,\mathcal{D})}{|\mathcal{D}|}$. Given a dataset $\mathcal{D}$, the *frequent itemset mining* problem aims to find all of the itemsets whose frequency is greater than a given threshold.

In our paper, we assume that instead of releasing the actual dataset, only a set of itemsets is released along with their frequencies due to privacy concerns. However, in this case, the problem that we study is the extent to which it is possible to retrieve the original dataset $\mathcal{D}$. This is related to the *Inverse Frequent itemset Mining (IFM)* problem, which aims to find a dataset such that the frequencies of a set of given itemsets for that dataset are in a specific range

interval. IFM is formally defined as follows:

The IFM problem Given a set of items $\mathcal{I}$, two integer numbers s_l, s_u, a set of support constraints S of the form (I, l, u) where I is an itemset on $\mathcal{I}$ and $l, u \in \mathcal{N}$ with $l \leq u$. The IFM problem, denoted as $IFM(\mathcal{I}, s_l, s_u, S)$, consists in finding a dataset $\mathcal{D}$ such that $s_l \leq |\mathcal{D}| \leq s_u$ and $\forall (I, l, u) \in S : l \leq support(l, \mathcal{D}) \leq u$.

Given a set of support constraints S, IFM provides information about degree of difficulty of generating a dataset that produces those frequent itemsets. However, note that while the solution to IFM enables malicious users to find *a* dataset that also meets the same support constraints, it does not say anything about whether this is the real dataset or how different it is from the real dataset. Indeed, given a set of support constraints S, more than one dataset solution can exist for the IFM problem. While this increases uncertainty in terms of the actual dataset, it may not significantly increase privacy since all of the datasets might be quite similar, thus actually reducing privacy.

Thus, to enable evaluation of the privacy risk associated with frequent itemset disclosure, we formalize a new problem called *k-distant-IFM-solutions*, i.e. find k IFM solutions whose pair distance is maximized. If we can find a sufficient number of solutions that are quite different from each other, then it significantly increases the degree of uncertainty and thus privacy. We can also take into consideration the problem of finding subset of support constraints or a perturbed version that can maximize the pair distance among all of the k IFM solutions.

2.1 K-distant-IFM-solutions problem

We first define the distance between two datasets, and then formalize the actual problem. While Jaccard or Hamming distance is a good metric to measure the distance between two individual transactions, they cannot directly be used to measure the difference among the collection of transactions. In our problem, the number of duplicate transactions has a significant meaning and therefore we chose to define our own metric that extends the Hamming distance for collection of transactions. Furthermore, we use the edit distance constraint to ensure that the different datasets obtained are sufficiently apart from each other based on our distance metric. Consider a case where the dataset $\mathcal{D}_1 = \{\{a, b, c\}, \{a, b, c\}, \{a, b, f\}\}$ and $\mathcal{D}_2 = \{\{a, b, c, h\}, \{a, b, c, h\}, \{a, b, f, h\}\}$. Since there are no transactions in common, $(\mathcal{D}_1, \mathcal{D}_2) = 6$ is the maximum distance that can be obtained.However, these datasets are exactly the same except for the item h. The edit distance constraint addresses this issue.

Given two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ over $\mathcal{I}$, we define the function $dist(\mathcal{D}_1, \mathcal{D}_2)$ between $\mathcal{D}_1$ and $\mathcal{D}_2$ as

$$dist(\mathcal{D}_1, \mathcal{D}_2) = \sum_{t \in T_{\mathcal{D}_1} \cup T_{\mathcal{D}_2}} |\#_{\mathcal{D}_1}(t) - \#_{\mathcal{D}_2}(t)|$$

This distance is a metric, but we omit the proof due to lack of space.

Algorithm 1 Greedy Algorithm

1: **procedure** GREEDYALGORITHM($\mathcal{I}, S$)
2: $SD = \emptyset$;
3: Choose $\mathcal{D}^* \in SOL(\mathcal{I}, S)$
4: **while** ($|SD| \leq k$) **do**
5: $SD = SD \cup \{\mathcal{D}^*\}$;
6: Choose $\mathcal{D}^* \in \arg\max_{\mathcal{D} \in SOL(\mathcal{I},S) \setminus SD} pairDist(SD \cup \mathcal{D})$;
7: **end while**
8: **return** SD;
9: **end procedure**

2.2 k-distant-IFM-solutions

Given a set of items $\mathcal{I}$, a positive integer number k, two integer numbers s_l, s_u, a set of support constraints S of the following form (I, l, u) where I is an itemset on $\mathcal{I}$ and $l, u \in \mathcal{R}$. The k-distant-IFM-solutions problem consists of finding a set of k datasets $SD = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ such that for each $\mathcal{D} \in SD$, $\mathcal{D}$ is a solution of $IFM(\mathcal{I}, s_l, s_u, S)$, and the pair distance $pairDist(SD) = \sum_{\mathcal{D}_j, \mathcal{D}_i \in SD, i>j} dist(\mathcal{D}_i, \mathcal{D}_j)$ is maximized.

Theorem 1. *The k-distant-IFM-solutions problem is* NP*-hard.*

Since finding even one solution of $IFM(\mathcal{I}, s_l, s_u, S)$ is NP-hard (as shown in [4]), finding k solutions is also NP-hard.

3 Proposed Approach

We now discuss how this problem can be solved. Let us assume $SOL(\mathcal{I}, S)$ is the set of all datasets that are the solution of $IFM(\mathcal{I}, S)$.

Thus, the k-distant-IFM-solutions can be formalized as

$$SD^* \in \underset{SD \subseteq SOL(\mathcal{I},S), |SD|=k}{\arg\max} pairDist(SD)$$

For this problem, Borodin et al. [3] show that if the function *dist* is a metric, then the Greedy Algorithm (Algorithm 1) gives a 2-approximate solution. However, we still need to specify how steps 3 and 6 of Algorithm 1 will be executed, i.e. how to choose $\mathcal{D}^* \in SOL(\mathcal{I}, S)$ (for step 3) and $\mathcal{D}^* \in \arg\max_{\mathcal{D} \in SOL(\mathcal{I},S) \setminus SD} pairDist(SD \cup \mathcal{D})$ (for step 6). Step 3 simply requires finding a solution for IFM, which is well understood. For the sake of simplicity and efficiency, we simply choose the first feasible solution instead of choosing a solution randomly. We denote the problem in Step 6 as the *Maximum Distant Dataset* and now show how to solve it.

3.1 Maximum Distant Dataset

The goal of maximization is to maximize the difference between the created dataset and existing datasets. Thus, we would like to find a dataset $\mathcal{D}$ that maximizes $\sum_{\mathcal{D}' \in S\mathcal{D}} dist(\mathcal{D}', \mathcal{D})$, which is equivalent to maximizing $pairDist(SD \cup \mathcal{D})$. In order to solve the Maximum Distant Dataset we provide an ILP formulation. This formulation is based on three kinds of variables:

- a real variable x_t, for each possible transaction $t \subseteq \mathcal{I}$, modeling the number of duplicates $\#(t)$ for each transaction t in the dataset that we have to generate (we relax the assumption that number of duplicates is an integer number); Effectively, the variable x_t gives the support count of the transaction t in the newly created dataset.
- a real variable $y_t^{\mathcal{D}}$, for each $\mathcal{D} \in S\mathcal{D}$ and $t \in T_{\mathcal{D}}$, modeling the values $|\#_{\mathcal{D}}(t) - x_t|$; Note that for all transactions t present in the existing datasets, $|\#\mathcal{D}(t) - x_t|$ gives the absolute difference in support for such transactions in each dataset $\mathcal{D}$. For the transactions t that are not present in the existing datasets, x_t directly gives the support of such transactions in the new dataset.
- a binary variable $z_t^{\mathcal{D}}$, for each $\mathcal{D} \in S\mathcal{D}$ and $t \in T_{\mathcal{D}}$, that is used to emulate the absolute value $|\#_{\mathcal{D}}(t) - x_t|$.

Now, the formulation is as follows:

$$maximize \sum_{\mathcal{D} \in S\mathcal{D}} \Big(\sum_{t \in T_{\mathcal{D}}} y_t^{\mathcal{D}} + \sum_{t \subseteq \mathcal{I}, t \notin T_{\mathcal{D}}} x_t \Big) \tag{1}$$

$$\sum_{t \subseteq \mathcal{I}, I \subseteq t} x_t \geq l \qquad (I, l, \_) \in S \tag{2}$$

$$\sum_{t \subseteq \mathcal{I}, I \subseteq t} x_t \leq u \qquad (I, \_, u) \in S \tag{3}$$

$$\sum_{t \subseteq \mathcal{I}} x_t \geq s_l \tag{4}$$

$$\sum_{t \subseteq \mathcal{I}} x_t \leq s_u \tag{5}$$

$$\#_{\mathcal{D}}(t) - x_t \leq y_t^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{6}$$

$$-\#_{\mathcal{D}}(t) + x_t \leq y_t^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{7}$$

$$\#_{\mathcal{D}}(t) - x_t + 2 * k_t * (1 - z_t^{\mathcal{D}}) \geq y_t^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{8}$$

$$-\#_{\mathcal{D}}(t) + x_t + 2 * k_t * z_t^{\mathcal{D}} \geq y_t^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{9}$$

$$x_t \geq 0 \qquad t \subseteq \mathcal{I} \tag{10}$$

$$y_t^{\mathcal{D}} \geq 0 \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{11}$$

$$z_t^{\mathcal{D}} \in \{0, 1\} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \tag{12}$$

Where, $k_t = \min(s_u, \min_{(I,\_,u) \in S, I \subseteq t} u)$.

As can be seen, we have two groups of constraints. The first group of constraints from 2 to 5 defines the minimum support, the maximum support, the minimum size and the maximum size, respectively. The second group constraints from 6 to 9 contribute in modeling the absolute value $|\#_{\mathcal{D}}(t) - x_t|$ is equal to $y_t^{\mathcal{D}}$. More specifically, constraints 6 and 7 impose that $|\#_{\mathcal{D}}(t) - x_t| \leq y_t^{\mathcal{D}}$ ensuring that the variable y_t^D is an upper bound of the absolute difference between x_t and $\#_{\mathcal{D}}(t)$. Constraints 8 and 9 impose that only one condition between $\#_{\mathcal{D}}(t) - x_t$ and $x_t - \#_{\mathcal{D}}(t)$ has to be greater than or equal to $y_t^{\mathcal{D}}$. Latter two constraints ensure that y_t^D is also the lower bound of the absolute difference between x_t and $\#_{\mathcal{D}}(t)$. The constraints from 6 to 9, together, ensure that $y_t^D = \#_{\mathcal{D}}(t) - x_t$. However, note that only one of the constraints between 8 and 9 can be met. z_t^D is the decision variable activating one of these two constraings, while k_t^D is the smallest constant that is large enough to ensure that these constraints are met. Finally, the maximization function (expression 1) maximizes the degree of difference in support for transactions present, which exactly models the maximization of distance metric defined in 2.1.

Usually, adding more constraints to an integer linear program reduces the search space by improving the bound obtained by the linear formulation and consequently reduces the computation time. Therefore, we define additional constraints and variables imposing that the value $y_t^{\mathcal{D}} = |\#_{\mathcal{D}}(t) - x_t|$ is $\max(\#_{\mathcal{D}}(t), x_t) - \min(\#_{\mathcal{D}}(t), x_t)$. The real variables $y_{t,max}^{\mathcal{D}}$ and $y_{t,min}^{\mathcal{D}}$ model $\max(\#_{\mathcal{D}}(t), x_t)$ and $\min(\#_{\mathcal{D}}(t), x_t)$, respectively. The revised ILP is given below. Note that in this case, the integer variables and the constraints are polynomial in the description of $S\mathcal{D}$ and S, respectively. The main issue is represented by the exponential number of real variables x_t due to all the possible transactions $t \subseteq \mathcal{I}$. Thus, these linear programs cannot really be directly solved. However, we can use an alternative technique called the branch and price algorithm (see [2]).

$$\#_{\mathcal{D}}(t) \geq y_{t,min}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (13)$$

$$x_t \geq y_{t,min}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (14)$$

$$\#_{\mathcal{D}}(t) \leq y_{t,max}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (15)$$

$$x_t \leq y_{t,max}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (16)$$

$$\#_{\mathcal{D}}(t)(1 - z_t^{\mathcal{D}}) \leq y_{t,min}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (17)$$

$$x_t - k_t * (1 - z_t^{\mathcal{D}}) \leq y_{t,min}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (18)$$

$$(\#_{\mathcal{D}}(t) - k_t) * z_t^{\mathcal{D}} + k_t \geq y_{t,max}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (19)$$

$$x_t + k_t * z_t^{\mathcal{D}} \geq y_{t,max}^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (20)$$

$$y_{t,max}^{\mathcal{D}} - y_{t,min}^{\mathcal{D}} = y_t^{\mathcal{D}} \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (21)$$

$$y_{t,max}^{\mathcal{D}} + y_{t,min}^{\mathcal{D}} = \#_{\mathcal{D}}(t) + x_t \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (22)$$

$$y_{t,min}^{\mathcal{D}} \geq 0 \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (23)$$

$$y_{t,max}^{\mathcal{D}} \geq 0 \qquad \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}} \quad (24)$$

Algorithm 2 HeuristicSolver

procedure HeuristicSolver($(S\mathcal{D}, S, s_l, s_u)$)
  Generate integer linear program P according $S\mathcal{D}, S, s_l$ and s_u where the set of variables with prefix x is only equal to $\{x_t | \mathcal{D} \in S\mathcal{D}, t \in T_{\mathcal{D}}\}$;
  Relax in P the binary constraints the variables with prefix z;
  Solve the program P;
  Find a new transaction t (if it exists) by solving the price problem;
  while (t exists) **do**
    Add the variable x_t in P;
    Solve the program P;
    Find a new transaction t (if it exists) by solving the price problem;
  end while
  Add in P the binary constraints on the the variables with prefix z;
  Solve the program P;
  Obtain from the solution of P the dataset $\mathcal{D}$;
  return $\mathcal{D}$;
end procedure

3.2 Heuristic Solver

The branch and price algorithm is a branch and bound algorithm that at each branch solves the relaxed problem (i.e. the linear one) by using column generation techniques. Given that the number of variables is huge, column generation techniques make the problem tractable. Instead of working on the entire set of variables, the column generation technique starts with a prefixed number (in our case the variables related to all the transactions in $S\mathcal{D}$) and at each iteration it generates a new variable or column (a new transaction) whose reduced cost is negative [4][7]. In order to generate a new variable a new problem called price problem has to be solved. The price problem consists of finding a new column with negative reduce cost, which is strictly related to the simplex algorithm [12] and how it works. Note that the specific price problem changes based on the underlying LP or ILP formulation. In prior work[7] the problem in Definition 2 has already been formalized as a linear program whose constraints are the constraints from 2 to 5. [7] also solves it with a column generation techniques and its price problem. In our problem we start by considering all the variables referring to all the transactions in $S\mathcal{D}$. Then the new variables or columns that we have to generate are those not involved in the constraints from 6 to 22. The main idea is to use the column generation techniques to solve the relaxation formulation where all the binary variables are substituted with real variables restricted to $[0, 1]$. Then, use all the columns generated in the column generation algorithm to solve the ILP version. Algorithm 2 gives the details.

Price problem The pricing problem consists in finding a new transaction different from all the previous transactions whose reduced cost is negative. It is known that the reduced cost of a column can be expressed as a linear combination of the dual variable associated to each constraint of the linear program

(see [6]). Let dsl, dsu, dl_I and du_I (where $(I, \_, \_) \in S$) be the dual variables associated to the constraints of the kinds 4, 5, 2 and 3. The reduced cost of a transaction t is $rc(t) = 1 + dsl + dsu + \sum_{I \subseteq t, (I,\_,\_) \in S} (dl_I + du_I)$. Given the set of current datasets generated $S\mathcal{D}$, the set of all the different transactions present in $S\mathcal{D}$ is defined as $tr(S\mathcal{D}) = \bigcup_{\mathcal{D} \in S\mathcal{D}} T_{\mathcal{D}}$. Now, we show an integer linear program solving the price problem. A generic transaction is a set of items then we can represent this transaction by using $|\mathcal{I}|$ binary variable $\{q_i | i \in \mathcal{I}\}$ s.t. if the item i is contained in t then $q_i = 1$ or $q_i = 0$ otherwise. In order to model the reduced cost function, it is essential to know which of the itemset in S are contained in the new transaction. Therefore, we define a set of binary variables $\{h_I | (I, \_, \_) \in S\}$ s.t. if the itemset I (with $(I, \_, \_) \in S$) is contained in the new transaction then $h_I = 1$ or $h_I = 0$ otherwise.

The objective function represents the reduced cost of the new transaction. The first two constraints 26 and 27, impose that whether the transaction represented by the set of variables $\{q_i | i \in \mathcal{I}\}$ contains an itemset I, the variable h_I is equal to 1 or 0 otherwise. The third constraint 28 imposes that the edit distance between each transaction in $tr(S\mathcal{D})$ and the new one has to be greater than or equal to the constant $minED$. $minED$ can be one if we only want that the current transaction should be different by each other, but can be more than one to enforce that all the transactions in all the K datasets generated are very different. This parameter is very important in order to produce datasets different not only in terms of number of duplicates, but also in terms of transaction structure. The last constraint 29 imposes that the transaction is not an empty set. Thus, the following integer linear program finds a new transaction s.t. its reduced cost is minimized. Note that after solving this ILP program we have to check if the reduced cost is negative, and only continue to iterate if so. Otherwise, the heuristic solver stops because there does not exist any transaction with negative reduced cost.

$$minimize \quad 1 + dsl + dsu + \sum_{(I,\_,\_) \in S} h_I \cdot (dl_I + du_I) \tag{25}$$

$$h_I \leq q_i \qquad (I, \_, \_) \in S, i \in I \tag{26}$$

$$\sum_{i \in I} q_i \leq |I| - 1 + h_I \qquad (I, \_, \_) \in S \tag{27}$$

$$\sum_{i \in t} (1 - q_i) + \sum_{i \in \mathcal{I} \setminus t} q_i \geq minED \qquad t \in tr(S\mathcal{D}) \tag{28}$$

$$\sum_{i \in \mathcal{I}} q_i \geq 1 \tag{29}$$

$$q_i \in \{0, 1\} \qquad i \in \mathcal{I} \tag{30}$$

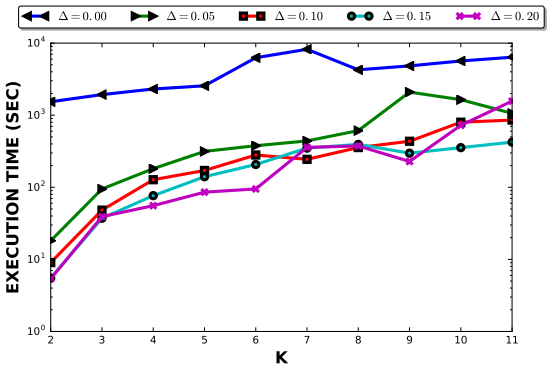
$$h_I \in \{0, 1\} \qquad (I, \_, \_) \in S \tag{31}$$

Table 1: Dataset Description

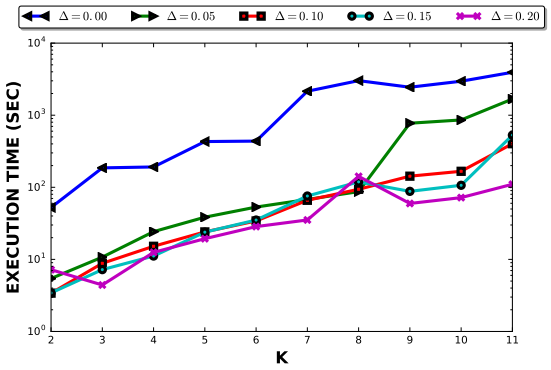
Dataset Name	Real Dataset Size	Distinct Items	Avg. trans. Size	Max trans. Size	s_l	s_u
BMS WebView1	59602	497	2.5	267	49602	69602
BMS WebView2	77512	3340	4.6	161	67512	87512
T10I4D100K	100000	870	10	300	90000	110000

4 Experimental Evaluation

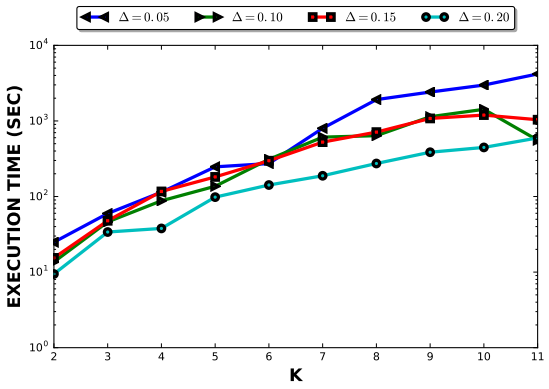
We now discuss the experimental evaluation. Three datasets – 2 real datasets (BMS-Webview-1, BMS-Webview-2) and a synthetic one (T10I4D100K) – were used to conduct experiments. The dataset parameters are given in Table 1.



(a) BMS Webview1: $\rho = 0.9\%$, MinED = 1

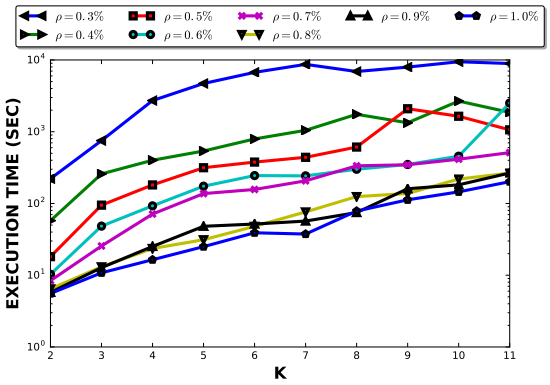


(b) BMS Webview2: $\rho = 0.9\%$, MinED = 1

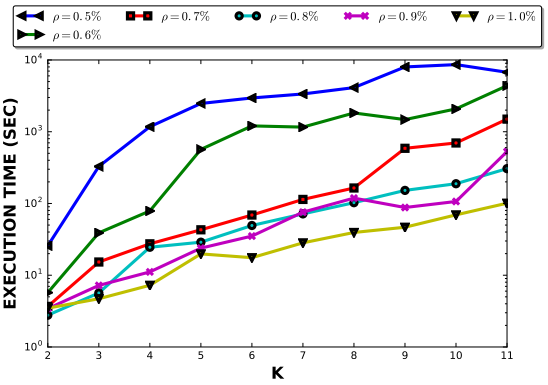


(c) T10I4D100K: $\rho = 0.9\%$, MinED = 1

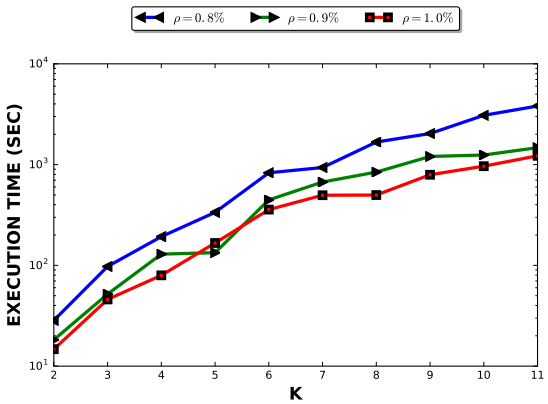
Fig. 1: Varying k and interval threshold Δ



(a) BMS WebView1: $\Delta = 0.05$, MinED = 1



(b) BMS WebView2: $\Delta = 0.15$, MinED = 1



(c) T10I4D100K: $\Delta = 0.15$, MinED = 20

Fig. 2: Varying Support Threshold

Each instance of our problem is represented by several parameters: sizemax (s_u), sizemin (s_l), set of items ($\mathcal{I}$), set of support constraints S (at different levels of support), support values (ρ), k (number of different datasets to be generated), edit distance (minED). Sizemax and Sizemin were obtained for each dataset by adding and subtracting 10000 from the size of the original dataset, respectively (as noted in Table 1). $minED$ was set to 1, 10, 20, 30, and K was varied from 2 to 11 (inclusive).

In order to generate the support constraints, of the form (I, l, u), we compute the set of the frequent itemsets from each datasets where the minimum support value δ was varied in the the range 0.2%, 0.3%, 0.4%..., 0.9%, 1%. The lower and the upper bound threshold for each frequent itemset I were obtained by using the following formulas $l = support(I, \mathcal{D}) * (1 - \Delta)$ and $u = support(I, \mathcal{D}) * (1 + \Delta)$

where the interval threshold (Δ) was set to values 0.0, 0.05, 0.1, 0.15, 0.2 . Thus, when $\Delta = 0.0$, we have that $l = u = support(I, \mathcal{D})$.

All experiments were carried out on machines with CentOS7 (x86-64) Operating System, 2 Xeon Processors (E5-2620 v3 @ 2.40GHz), and 256GB RAM. We report the execution time as well as the $pairDist$ calculated for each dataset.

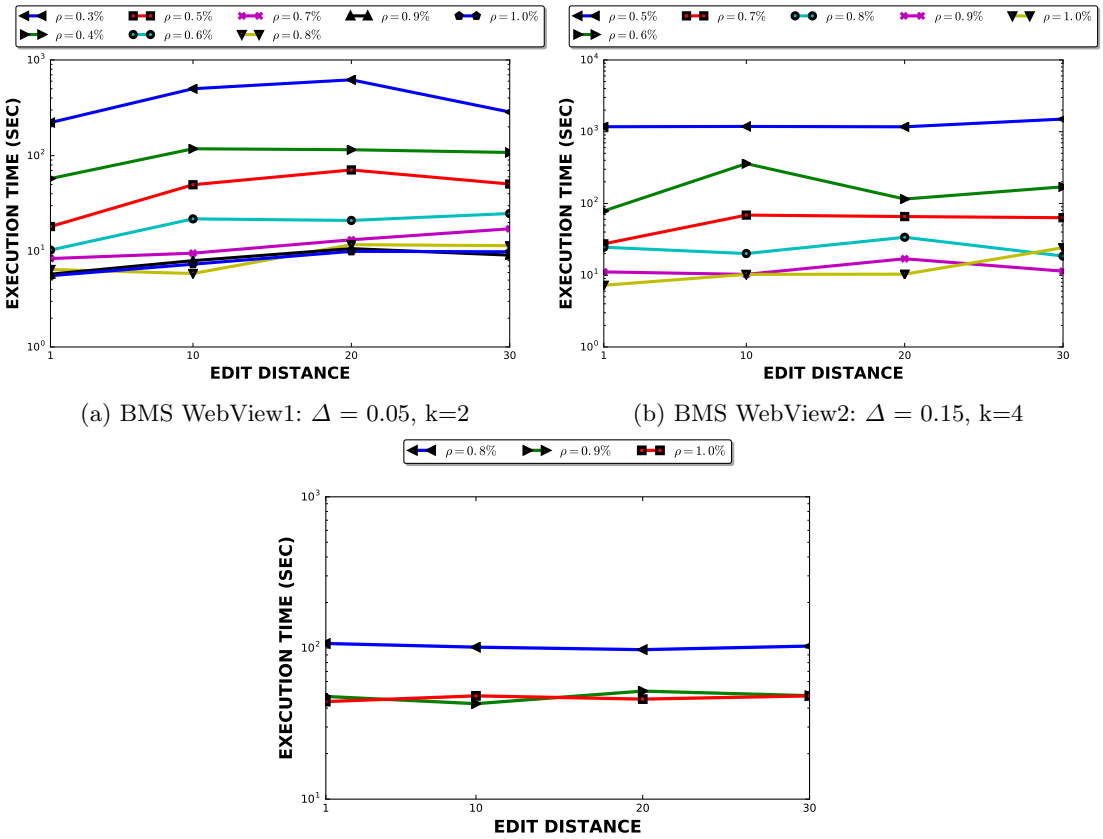
Varying Δ: We first observe the impact of varying itemset interval threshold Δ on execution time. k (the number of datasets to create) has been varied from 2-11. It was observed that the execution time was almost constant with varying interval threshold Δ values in the interval {0.0, 0.05, 0.1, 0.15, 0.2} for all the three datasets except for $\Delta = 0.0$ for which execution time increased. Figures 1a through 1c represent the impact of k-anonymity values and interval threshold values on execution time for the three datasets. The results show that as we increase the value of delta, the flexibility allowed to the solver also increases and it quickly finds a feasible solution.

Varying support threshold values: We next observe the impact of varying k along with varying support threshold values on the execution time for solving a k-distant-IFM-solution. Figures 2a through 2c show the impact of varying k on execution time. It can be noted that while the time required is different for the three datasets for the different support threshold values, it does not change much with respect to k. For BMS Webview-2 and T10I4D100K datasets with varying ρ%, similar trend is observed. In [7] (which models IFM with linear programs), increasing ρ% decreases the execution time. However, for integer linear formulations with more constraints, search space is decreased and it is easier to find a solution. Therefore, increase in ρ%, increases the execution time.

Due to the significant computational resources required and the large number of experiments to be carried out, we were only able to carry out experiments for a few values of support for BMS Webview2 dataset. But we did check to make sure that the overall behavior is the same. For T10I4D100K dataset, lower values of support lead to a huge number of frequent itemsets. Therefore, we limited the experiments to higher values of support. Also, the behavior of execution time with respect to k is clear even when k is limited to 11, which was sufficient reason not to go beyond 11 for k as these operations are computationally expensive.

Varying edit-distance values: We next observe the impact of varying edit-distance values in the range $[1, 10, 20, 30]$. Figures 3a - 3c show the impact on execution time for the three datasets. We can generally observe that time does not significantly change by varying the edit distance.

Pairwise average distance varying k, ρ and Δ: Finally, we observe the effect of varying k, ρ and Δ on the pairwise distance. Firstly, for varying support constraints ρ and k, figures 4a through 4c show the impact of k w.r.t. $distance/((k * (k-1))/2)$ values for the three datasets. Here, no specific trend can be observed. However, if we consider that our approach is based on 2-approximation algorithm, these trends can be considered constant within an approximation range. Similarly, if we consider the case where we vary ρ, it can be observed that as the support increases, the average distance decreases and



(a) BMS WebView1: $\Delta = 0.05$, k=2

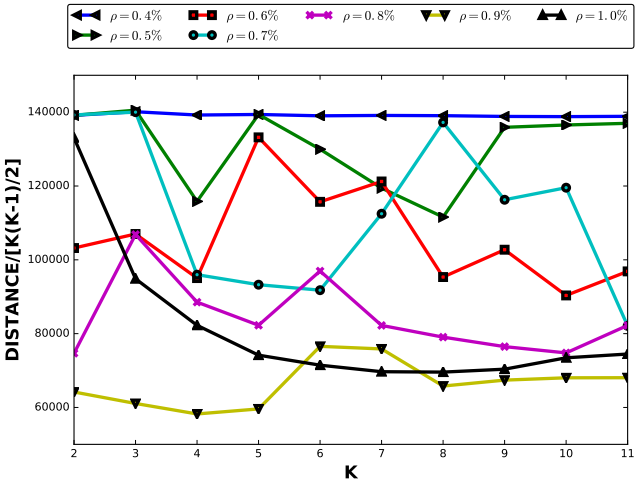
(b) BMS WebView2: $\Delta = 0.15$, k=4

(c) T10I4D100K: $\Delta = 0.15$, k=3

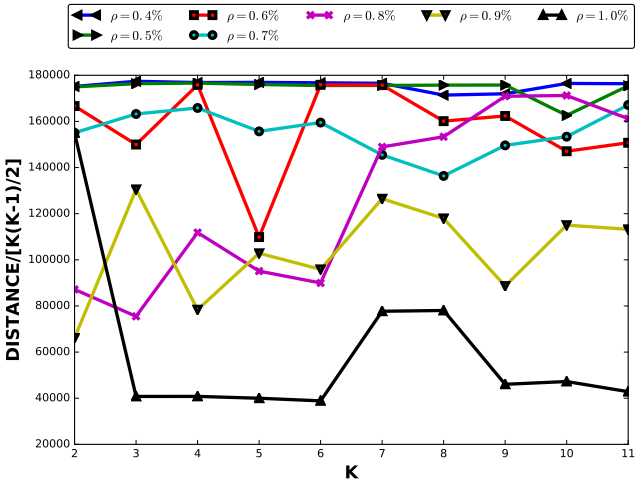
Fig. 3: Varying edit-distance for different Δ and k

vice-versa (of course within the approximation range). This shows that as the amount of information about the distribution of the itemsets disclosed increases, the privacy risk increases (several transaction databases nearby each other).

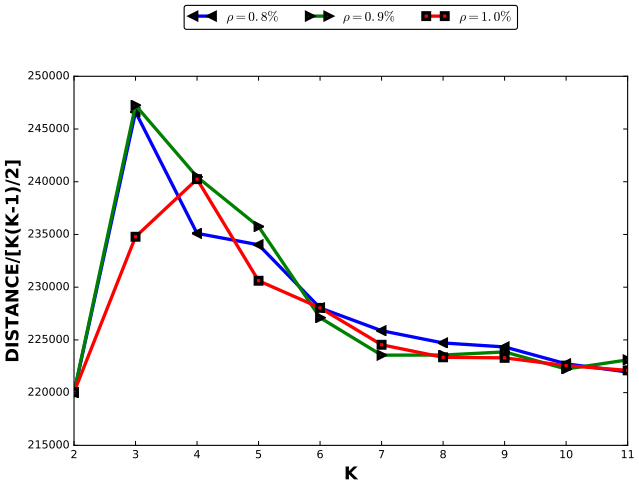
Secondly, for varying interval threshold Δ, figures 4d through 4f show the impact of k values w.r.t. $distance/((k*(k-1))/2)$ for the three datasets. We can observe that as interval Δ decreases, the average distance also decreases. Essentially, increasing the support interval size for each itemset increases the uncertainty of the itemset distribution and thus decreases the privacy risk. Additionally, note that in fig 4c and fig 4f, there is a peak in the plots between $k = 2$ and $k = 3$. This is because Algorithm 1 in line 3 initialize the SD with an arbitrary transaction database which is not chosen in a way that would maximize the distance of the future transaction database candidates.



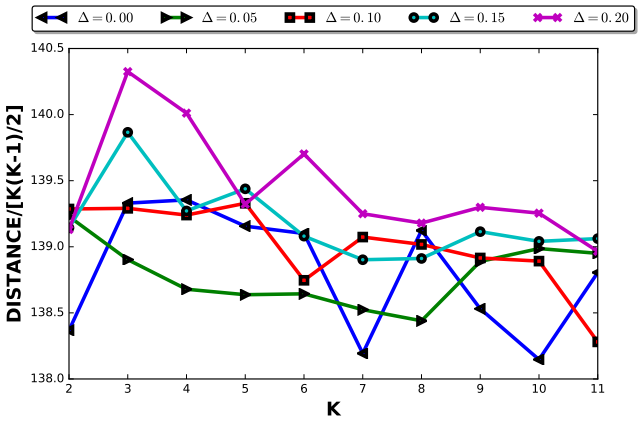
(a) BMS WebView1: $\Delta = 0.15$, MinED = 10



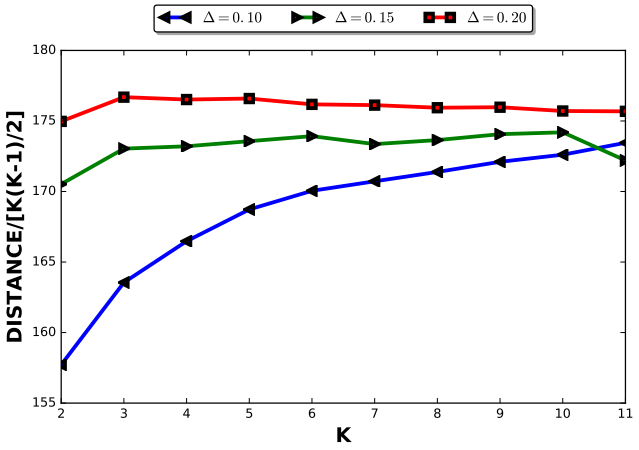
(b) BMS WebView2: $\Delta = 0.20$, MinED = 30



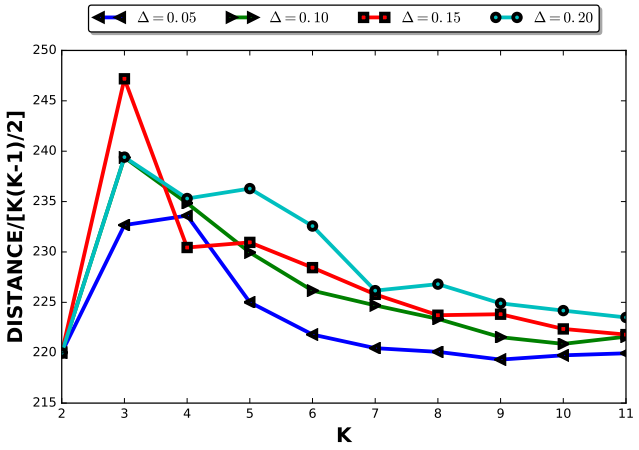
(c) T10I4D100K: $\Delta = 0.15$, MinED = 1



(d) BMS WebView1: $\rho = 0.4\%$, MinED = 20



(e) BMS WebView2: $\rho = 0.5\%$, MinED = 1



(f) T10I4D100K: $\rho = 0.8\%$, MinED = 10

Fig. 4: Average Distance w.r.t varying ρ and Δ

5 Conclusion

In this paper we define the K-distant-IFM-solutions problem, that enables evaluation of the frequent itemset disclosure risk and propose a solution for it. The experimental evaluation shows that the proposed approach is effective. In our future work, we plan to develop methodologies that are able to perturb the support of the itemsets disclosed in order to minimize the disclosure risk. In addition, we plan to extend these techniques to work with sequence mining as well – where we consider sequences rather than itemsets.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. ACM SIGMOD Record 22(2), 207–216 (1993)
2. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329 (1996)
3. Borodin, A., Lee, H.C., Ye, Y.: Max-sum diversification, monotone submodular functions and dynamic updates. In: Proceedings of the 31st Symposium on Principles of Database Systems. pp. 155–166. PODS '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2213556.2213580
4. Calders, T.: Itemset frequency satisfiability: Complexity and axiomatization. Theor. Comput. Sci. 394(1-2), 84–111 (2008)
5. Goethals, B., Zaki, M.J.: Fimi03: Workshop on frequent itemset mining implementations. In: Third IEEE International Conference on Data Mining Workshop on Frequent Itemset Mining Implementations. pp. 1–13 (2003)
6. Guzzo, A., Moccia, L., Saccà, D., Serra, E.: Solving inverse frequent itemset mining with infrequency constraints via large-scale linear programs. TKDD 7(4), 18 (2013), http://doi.acm.org/10.1145/2541268.2541271
7. Guzzo, A., Saccà, D., Serra, E.: An effective approach to inverse frequent set mining. In: Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on. pp. 806–811 (Dec 2009)
8. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 3 (2007)
9. Mielikainen, T.: On inverse frequent set mining. In: Society, I.C. (ed.) Proc. of 2nd Workshop on Privacy Preserving Data Mining (PPDM). pp. 18–23 (2003)
10. Ramesh, G., Maniatty, W., Zaki, M.: Feasible itemeset distributions in data mining: theory and application. In: Proc. 28th International Conference on Very Large Data Bases. pp. 682–693 (2002)
11. Samarati, P.: Protecting respondents identities in microdata release. Knowledge and Data Engineering, IEEE Transactions on 13(6), 1010–1027 (2001)
12. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons, Inc., New York, NY, USA (1986)
13. Wang, Y., Wu, X.: Approximate inverse frequent itemset mining: Privacy, complexity, and approximation. In: ICDM. pp. 482–489 (2005)
14. Wu, X., Wu, Y., Wang, Y., Li, Y.: Privacy-aware market basket data set generation: An feasible approach for inverse frequent set mining. In: Proc. 5th SIAM International Conference on Data Mining (2005)