# Frequent Itemset Mining Under Differential Privacy with Multiple Minimum Supports

1st Yum-Luh Wang
*Graduate Institute of Electronics Engineering*
*National Taiwan University*
Taipei, Taiwan
r07943161@ntu.edu.tw

2nd Yao-Tung Tsou
*Department of Communications Engineering*
*Feng Chia University*
Taichung, Taiwan
yttsou@fcu.edu.tw

3rd Sy-Yen Kuo
*Graduate Institute of Electronics Engineering*
*National Taiwan University*
Taipei, Taiwan
sykuo@ntu.edu.tw

***Abstract*—Frequent itemset mining is an extensively studied research domain of data mining. The aim is to find interesting correlations between items in a transaction database. However, malicious user might gain sensitive information in the mining process. Differential privacy is the de facto standard when it comes to protecting data. Combining differential privacy with frequent itemset mining can provide strong privacy guarantee while generating statistical information from sensitive data. In existing studies, most algorithms are focusing on finding frequent patterns using one predefined threshold with the protect of differential privacy. However, using single threshold to extract itemsets creates "rare item problem", and setting respective support thresholds for each item is more adequate to reflect the nature of widely varied items. In this paper, we propose a novel FP-growth-like solution DPCFP++ to solve the rare item problem of frequent itemset mining while guarantee differential privacy at the same time. The experiments illustrate that our algorithm acheieves high utility in privately solving rare item problem.**

***Index Terms*—Differential Privacy, data mining, multiple minimum supports, frequent itemset mining**

## I. Introduction

In modern era, the development of information technology grows rapidly, more and more services aggregate data from individuals. With the evolution of machine learning and statistics, organizations utilize these enormous amount of data to improve the accuracy of their model, understand the past and predict the future. Frequent itemset mining is a popular research topic in data mining, and it was first introduced in 1993 by Agrawal et al. [1] The original purpose of frequent itemset mining was to analyze customer habits in a supermarket. They tried to find associations between different items that customers frequently bought together. According to the mined results, the management team of the supermarket can make business decisions to maximize profit including product rearrangement, bundle selling, etc. After decades of research, frequent itemset mining can also be extended to many other area besides market basket analysis. For example, text mining, disease diagnosis, and so on.

Majority of existing literatures endeavors on single threshold mining, that is, treat all the items equally and apply a single threshold to decide whether an itemset is frequent or not. However, setting one appropriate threshold for all items is a hard question. For example, consider two items: soda and television in a wholesale store. Soda is frequently bought while television is an uncommon purchase. If the threshold is set too high, we might consider TV as a less important item, but TV contributes thousand times more to the store revenue than the common purchased soda. On the other hand, if the threshold is too low, the computational cost becomes expensive, the mining result will cause combinational explosion and we will receive many meaningless itemsets. The above dilemma is the famous "rare item problem". In real life applications, some items tend to have more weights in natural than other items. Researchers [2] [3] [4] [5] have worked on this problem by allowing users to use "multiple minimum supports". Briefly speaking, rare itemset mining is a more advanced setting of frequent itemset mining, and it allows user to apply different threshold on each item .

The convenience which data mining brings comes with the price of privacy leakage. The more precise the model is, the more personal behavioral information reveals. How to de-identification is a mainstream topic in related research. A straightforward approach is to .....

Differential Privacy, first proposed by Dwork in [4], had come to rescue in 2006. Since then, many research based on differential privacy have been conducted. Tech giants such as FAANG(facebook, applem, amazon, netflix, google) have added features to their products with different kinds of differential privacy model. It guarantees that the outputs of a mechanism will be approximately the same when two input is almost identical, and thus protect privacy from explosure.

It guarantees that a user..... In this paper, we study the problem of how to perform frequent itemset mining (FIM) on transaction databases while satisfying differential privacy.

## II. Related Work

### A. Frequent Itemset Mining

There are mainly two types of frequent itemset mining algorithms. One is Apriori, and the other is FP-growth.

### B. Frequent Itemset Mining with Multiple Minimum Supports

### C. Differentially Private Frequent Itemset Mining

Frequent itemset mining, also known as frequent pattern mining, is The NoisyCut algorithm in [8] was proven violating differential privacy in

## III. Preliminaries and Problem Statement

In this section, we introduce and provide formal definitions of differential privacy and the rare item problem of frequent itemset mining. Finally, we give the problem statement in the last subsection.

### A. Differential Privacy

Differential privacy guarantees that the presence or absence of any individual will not significantly change the output of an algorithm. By this approach, one can only infer limited information about that particular individual from the output.

**Definition 1.** *($\varepsilon$-Differential Privacy):*
*Let $D$ and $D'$ denote two neighboring databases, which means they differ by at most one record. A mechanism $\mathcal{M}$ is $\varepsilon$-differential private if and only if for any $S \subseteq Range(\mathcal{M})$*

$$Pr[\mathcal{M}(\mathcal{D}) \in S] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(\mathcal{D}') \in S] \tag{1}$$

The parameter $\varepsilon$ is called privacy budget, it quantifies the degree of privacy. The larger the $\varepsilon$, the less private the result is. If the $\varepsilon$=0, the result is perfectly private.

There are a few mechanisms that acheieves differential privacy. Laplace mechanism is one of the most popular method. It injects properly calibrated noise into the original output according to the sensitivity. We give the definition of sensitivity as follows.

**Definition 2.** *(Sensitivity):*
*When answering numeric queries of neighboring databases, the sensitivity of a query $Q$ is*

$$\triangle_Q = \underset{D,D'}{max} \|Q(D) - Q(D')\|_1 \tag{2}$$

Sensitivity is used to measure the maximum possible change in the outputs over any two neighboring databases.

**Definition 3.** *(Laplace Mechanism):*
*For any function $Q : D \to R$, Laplace mechanism $\mathcal{M}$ satisfies $\varepsilon$-differential privacy by adding noise to the origin output:*

$$\mathcal{M}(D) = Q(D) + Lap(\frac{\triangle_Q}{\varepsilon}), \tag{3}$$

*where $Lap(\frac{\triangle_Q}{\varepsilon})$ is the noise drawn i.i.d from the Laplace distribution with scale $\frac{\triangle_Q}{\varepsilon}$.*

There are some qualitative properties of differential privacy. We will use following properties in our solution.

**Lemma 1.** *(Sequential Composition):*
*Given an algorithm $\mathcal{M}$ consisting of a sequence of procedures $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n\}$. If each procedure $\mathcal{M}_i$ satisfies $\varepsilon_i$-differential privacy, then $\mathcal{M}$ satisfies $\sum_{i=1}^{n} \varepsilon_i$-differential privacy.*

**Lemma 2.** *(Closure Under Post-Processing):*
*For any algorithm $\mathcal{A}$ and a $\varepsilon$-differentially private mechanism $\mathcal{M}$, the computaion $\mathcal{A} \circ \mathcal{M}(D)$ satisfies $\varepsilon$-differential privacy as long as $A$ do not access database directly.*

### B. Frequent Itemset Mining

Given a threshold $\lambda$ and a transaction database $D = \{t_1, t_2, t_3, \ldots, t_n\}$ consisting of $n$ transactions where each transaction is composed of sets of distinct items in the item universe $I = \{i_1, i_2, i_3, \ldots, i_m\}$, i.e. $t_j \subseteq I$, Frequent itemset mining, also known as frequent pattern mining, refers to discovering all set of patterns while the support $\sigma$ of which is greater than or equal to a user-specified threshold.

**Definition 4.** *(Itemset):*
*Any subset of the item universe $I$ is called an itemset.*

An itemset which contains k distinct items is called k-itemset. k is also called the length of the itemset or the cardinality of the itemset. We will use them in this work interchangebly.

**Definition 5.** *(Support and Threshold):*
*The number of transactions in the database containing an itemset is known as the support of that itemset.*

We denote support and threshold as $\sigma$ and $\lambda$, respectively. There are two ways to specify $\sigma$ and $\lambda$. One is the percentage way, and the other is by absolute number. It is simple to convert between them, just multiply or divide the number of total transactions $n$.

**Definition 6.** *(Frequent Itemset):*
*An itemset $X$ is frequent if and only if its support $\sigma(X)$ is greater than or equal to support threshold $\lambda$.*

This means that the presence of itemset $X$ in the database is statistically significant.

TABLE I: A transaction database

| TID | Items | TID | Items |
|---|---|---|---|
| 1 | a,b | 11 | a,f |
| 2 | a,d,f | 12 | b,e,f |
| 3 | b,c,d | 13 | c,d,e |
| 4 | c,e | 14 | a,b,f |
| 5 | a,b,e | 15 | a,b,e,f |
| 6 | a,c,g | 16 | a,b,c,e,f,h |
| 7 | a,c,e | 17 | a,b,h |
| 8 | e | 18 | c,e |
| 9 | a,b,c | 19 | c,d |
| 10 | b,c,d | 20 | a,b,d |

**Example 1.** Consider the database shown in Table I. There are $n = 20$ transactions in the database, and item universe

$I = \{a, b, c, d, e, f, g, h\}$. The itemset (a,b) is a 2-itemset. It occurs in the database 6 times, so the $\sigma$(a,b) is 6. Let's say the user-specified threshold is 2 and we can see (a,b) is a frequent itemset while (a,g) is not, for $\sigma$(a,g) = 1.

### C. Rare Item Problem

By the nature of items, some items appear more frequently than others. However, this sometimes causes meaningless itemsets. We have to find frequent itemsets without generating too many meaningless itemsets. To deal with rare item problem, Liu et al. [2] first tackled the problem with minimum item supports (MIS). MIS allows user to specify respective threshold for every item.

**Definition 7.** *(Minimum Item Support)*
*Minimum Item Support refers to each item's respective user-defined threshold.*

MIS can not only be set by user one by one, but also can be set adaptively. Liu et al. also proposed a equation for fast-setting MIS by items' supports. For an item $i_j$ with support $\sigma(j)$, the Minimum Item Support(MIS) of $j$ is

$$MIS(i_j) = max\{\beta * \sigma(i_j), \lambda\}, \tag{4}$$

where $\beta \in [0, 1]$. $\beta$ is a relevance parameter that controls how the MIS values for items should be related to their frequencies. Noted that when $\beta$=0, the MIS value equals $\lambda$, which is the same as the traditional setting of frequent itemset mining.

**Definition 8.** *(MIS of an Itemset)*
*The minimum item support of a k-itemset $X = \{i_1, i_2, i_3, \ldots, i_k\}$ is the smallest MIS value of items in $X$, i.e.*

$$MIS(X) = min\{MIS(i_j) | i_j \in X\}. \tag{5}$$

**Example 2.** Consider the database in Table I. If we set MIS(a)=15 and MIS(f)=5, then itemset (a) is not frequent since $\sigma$(a) = 12. However, itemset (af) is a frequent itemset since its support is 5 which is greater than MIS(af) = min{MIS(a),MIS(f)} = 5.

Example 2 shows that frequent itemset mining using multiple minimum supports is different from single threshold. Under single threshold condition, if an itemset is not frequent, its superset is not frequent either. This principle is called Apriori property, or downward closure property. However, Apriori property cannot be directly applied to our problem.

### D. Problem Statement

Based on above definitions, we give the formal statement of our problem. Our algorithm works in a centralized model. As illustrate in Figure. Given a transaction database, a threshold and a privacy budget, we want to find all frequent itemsets whose support is no less than the threshold using multiple minimum supports in a differentially private way.

### E. Main Challenges

We oberserve there are two major challenges in using multiple minimum supports to mine frequent itemsets under differential privacy.

1) *High Sensitivity*
   A straightforward approach is to count up each item's support, and direct apply Laplace noise to it. After that we can use traditional FIM algorithm to solve the problem. However, as we illustrate in the following example, this way is not practical.

   **Example 3.** Consider a relatively tiny database which just contains 100 different items. If there is a transaction containing all 100 items, and its neighboring database doesn't, then when we start counting supports, all 100 items' support will change. Hence the sensitivity is 100 which is large, and we need to apply a tremendous amount of privacy budget. Otherwise, the enormous noise we add would make the output of traditional FIM algorithm meaningless.
2) *Running Time*
   Most FIM algorithms under differential privacy is time-consuming. We want to develope an algorithm that is agile and scans database as less as possible. An FP-growth-like algorithm doesn't generate candidate itemsets and it significantly scans database less than Apriori-like algorithm does. As a result, we aim to build an algorithm based on FP-growth.

## IV. Proposed Method

### A. Overview

We now describe the framework of our proposed algorithm, which is called DPCFP, in the following paragraph. Motivated by the success in [4] [6] [8], we combine their advantage together in this paper. DPCFP consists of three stages, namely *TruncateDatabase, MISAssigning, CFP-Growth.* The basic idea is to differential privately get each item's support and MIS. Next, we sort the items according to the descending order of MIS and put them into MIS header table. Afterwards, we add transactions into MIS-tree by the same order one after another. Finally, we use a FP-growth-like algorithm to find frequent itemsets. Each step consumes different degrees of privacy budget, $\varepsilon_1$, $\varepsilon_2$,and $\varepsilon_3$, respectively. $\varepsilon_1$, $\varepsilon_2$,and $\varepsilon_3$ follows the sequential composition rule of differential privacy, and hence the sum of them is the amount of privacy budget that user allocated.

### B. TruncateDatabase

In this part, we truncate database to limit sensitivity to a acceptable level as [6] proposed. The rationale behind truncating is that most transactions in a database is short in general case. If there exist a few long transactions, then sensitivity will be affected remarkably. These few long transactions which have major impact on sensitivity but less importance on frequent itemsets is infuriating. To avoid this problem, we pose a limit

**Algorithm 1** TruncateDatabase

**Input:** database $D$; privacy budget $\varepsilon_1$;
**Output:** truncated database $D'$

1: $D' \leftarrow \emptyset$
2: Read $D$ to get total items $m$ and the total number of transactions $n$
3: $z$ = EstimateDistribution($D$, $\varepsilon_1$, $n$, $m$)
4: Let $l$ be the smallest integer such that $\sum_{i=1}^{l} z_i \geq 0.95$
5: **for** each transaction $t$ in $D$ **do**
6: add $t'$ = RandomTruncate($t$, $l$) to $D'$
7: **return** $D'$
8: **function** ESTIMATEDISTRIBUTION($D$,$\varepsilon$,$n$,$m$)
9: Let $z = [z_1, z_2, z_3, \ldots, z_m]$, where $z_i$ is the number of transactions with cardinality $i$ in $D$
10: $z' = z + [Lap_1, Lap_2, Lap_3, ..., Lap_m]$, where $Lap_i$ is drawn i.i.d. from Laplace noise $(\frac{1}{\varepsilon_1})$
11: **return** $\frac{z'}{n}$
12: **function** RANDOMTRUNCATE($t$,$l$)
13: $t'$ = Random Sample min($|t|$,$l$) item from t
14: **return** $t'$

to the length of transasctions. Of course, get rid of some items in transactions will create degrees of information loss.

In Algorithm 1, function *EstimateDistribution* estimates the noisy distribution of the database. The elements of vector $z$ are the number of 1-itemsets, 2-itemsets, ..., respectively. Calculating the distribution of the database has privacy leakage concern, so we use Laplace mechanism to perturb the result after $z$ is calculated. In line 4, $l$ is the maximal length parameter that controls the tradeoff between information loss and sensitivity reduction. 0.95 is just a experimental value that produces best results for our testing dataset. as the way which [6] proposed. Afterwards, function *RandomTruncate* randomly select $l$ items from the transactions that are longer than $l$ and keep the rest transactions the same. Finally, we put these new transactions into a new database $D'$.

**Theorem 1.** *Algorithm 1 satisfies $\varepsilon_1$-differential privacy.*

*Proof:* It is already proved in [6] that any $\varepsilon$-differential private algorithm on a local transformation of databases guarantees differential privacy. Since adding or removing one transaction can only affect $z$ in one element by one, the sensitivity is 1. Deploying Laplace noise $(\frac{1}{\varepsilon_1})$ to each element of $z$ satisfies satisfies $\varepsilon_1$-differential privacy. ■

**Example 4.** Let's continue the example in Table I. Say we set the maximal length constraint $l = 3$, so TID 15 and 16 will be truncated. We show the truncated database in Table II.

### C. MISAssigning

With the tranformed database, we can now count the noisy support and use it to fast assign MIS value to each item by (4).

TABLE II: Truncated database

| TID | Items | TID | Items |
|---|---|---|---|
| 1 | a,b | 11 | a,f |
| 2 | a,d,f | 12 | b,e,f |
| 3 | b,c,d | 13 | c,d,e |
| 4 | c,e | 14 | a,b,f |
| 5 | a,b,e | 15 | b,e,f |
| 6 | a,c,g | 16 | b,c,f |
| 7 | a,c,e | 17 | a,b,h |
| 8 | e | 18 | c,e |
| 9 | a,b,c | 19 | c,d |
| 10 | b,c,d | 20 | a,b,d |

**Algorithm 2** NoisySupportandMISTable

**Input:** database $D'$; privacy budget $\varepsilon_2$; truncated length $l$; relevance parameter $\beta$; threshold $\lambda$
**Output:** noisy supports $S$; MISTable $M$

1: $M \leftarrow \emptyset$;$S \leftarrow \emptyset$
2: **for** each $i$ in item universe $I$ **do**
3: Count i's support $i.\sigma$
4: Add $i.\hat{\sigma}$ = $i.\sigma + Lap(\frac{l}{\varepsilon_2})$ to $S$
5: Add $i.MIS$ = $max\{\beta * i.\hat{\sigma}, \lambda\}$ to $M$
6: **end for**
7: **return** $S$; $M$

In Algorithm 2, we first find the exact accumulated support sount. Line 4 and 5 compute and store the noisy support and MIS table.

**Theorem 2.** *Algorithm 2 satisfies $\varepsilon_2$-differential privacy.*

*Proof:* It straightforward to see the only part that accesses the database in Algorithm 2 is in line 3. Since we already set transaction length constraint to $l$, any addition or deletion of a transaction can at most increase or decrease the items' support by $l$. In this way, appling Laplace noise $(\frac{l}{\varepsilon_2})$is enough to garantee $\varepsilon_2$-differential privacy. The rest of algorithm only performs post-processing. ■

Next, we introduce the concept of MIS-tree and least minimum support(LMS). MIS-tree is a kind of prefix tree structure that database information while LMS can reduce the search space when we mining frequent patterns.

**Definition 9.** *(MIS-tree):*
*A MIS-tree consists of two parts. The tree itself and a header table.*

1) It consists of one root labeled as Null.
2) The header table

**Definition 10.** *(Least minimum support,LMS)*
*LMS refers to the lowest MIS value of all frequent itemsets.*

**Theorem 3.** *For any 1-itemsets $X$'s support is lower than LMS, it is not a frequent itemset, as well as its supersets.*

*Proof:* Let $X$ be a 1-itemset, and $X^+$ be its superset. According to Apriori property, $\sigma(X^+) \leq \sigma(X)$. If $X$'s support is lower than LMS ($\sigma(X)$ <LMS), it is not a frequent itemset

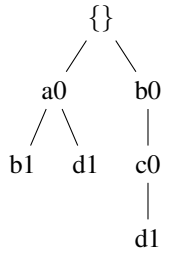



Fig. 1: A MIS-Tree

and MIS($X$)<LMS. Because $\sigma(X^+) \leq \sigma(X)$ <LMS and MIS($X^+$)$\leq$ MIS($X$), $X^+$ must be an infrequent itemset. ■

TABLE III: MIS Table

| item | support | threshold | MIS |
|---|---|---|---|
| a | 10 | 2 | 5 |
| b | 11 | 2 | 5 |
| c | 10 | 2 | 5 |
| d | 6 | 2 | 3 |
| e | 8 | 2 | 4 |
| f | 6 | 2 | 3 |
| g | 1 | 2 | 2 |
| h | 1 | 2 | 2 |

**Example 5.** Continue the previous example in Table II, we ommit the Laplace noise part for give a better understanding of the algorithm. We set $\beta = 0.45$ and threshold = 2 and using (4) to compute MIS. The result of the supports, the threshold and the approximate MIS results in Table III. Because item g and h are not frequent, and the smallest MIS in frequent 1-itemsets is 3, so LMS = 3.

**Algorithm 3** LMSPruning

**Input:** noisy supports $S$; MISTable $M$
**Output:** SortedMISTable $M$

Sort $M$ by descending MIS value.
**for** from the last item $i_j$ in $M$ to the first **do**
  **if** $S[i_j] < M[i_j]$ **then**
    Delete $i_j$ in $M$
  **else**
    LMS= $M[i_j]$
    Break
**end for**
**for** each item in $M$ **do**
  **if** $S[i_j] <$ LMS **then**
    Delete $i_j$ in $M$
**end for**
**return** $M$

Since items with support less than LMS cannot be frequent

**Algorithm 4** MIS-Tree

**Input:** database $D'$; noisy supports $S$; MISTable $M$; privacy budget $\varepsilon_3$
**Output:** MIS-tree $T$

$\hat{M}$ = LMSPruning($S, M$)
create $root \leftarrow \emptyset$ for MIS-tree $T$
**for** each transactions $t'$ in $D'$ **do**
  Sort $t'$ in $\hat{M}$ order
  AddTransaction($root, t', \varepsilon_3$)
  UpdateTree($root$)
**return** MIS-tree $T$
**function** ADDTRANSACTION($root, t'$)
  $Node$=$root$
  **for** each items $i_j$ in $t'$ **do**
    Check if $i_j$ is $Node$'s child
    **if** false **then**
      Create $NewNode$ $i_j$ under $Node$, initial with Laplace noise $Lap(\frac{\triangle}{\varepsilon_3})$
      $Node = NewNode$
    **else**
      $Node = i_j$
  $Node.\sigma$ ++
**function** UPDATETREE($node$)
  **for** each child in $node$'s children **do**
    UpdateTree(child)
    node.$\sigma$ += child.$\sigma$

**Algorithm 5** CFP-Growth

**Input:** MIS-tree $T$ **Output:** frequent itemsets $F$

$F \leftarrow \emptyset$
CFPGrowth()
**return** frequent itemsets $F$

## V. EXPERIMENT

### A. Metrics

We use F-score, a the widely used F-measure which is the harmonic mean of precision and recall.

### B. Experimental Results

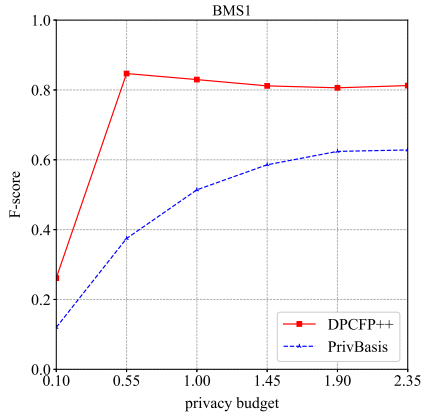
*1) :*

## REFERENCES

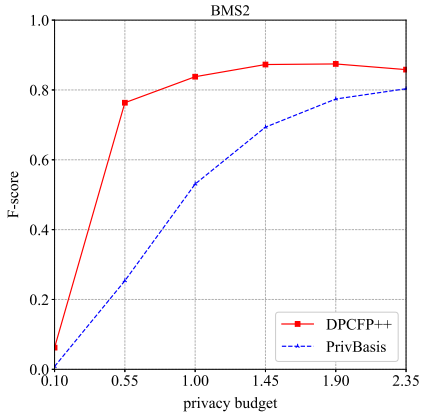
[1] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases," in Proceedings of ACM-SIGMOD, 1993.

TABLE IV: Dataset Characteristics

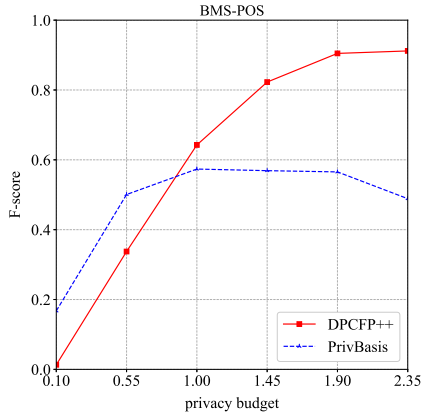
| Dataset | n | m | max$\|t\|$ | avg$\|t\|$ |
|---|---|---|---|---|
| retail | 88162 | 16470 | 76 | 10.3 |
| BMS1 | 59602 | 497 | 267 | 2.5 |
| BMS2 | 77512 | 3340 | 161 | 5.0 |
| BMS-POS | 515597 | 1657 | 164 | 6.5 |
| kosarak | 990002 | 41270 | 2498 | 8.1 |
| T10I4D100K | 100000 | 870 | 29 | 10.1 |



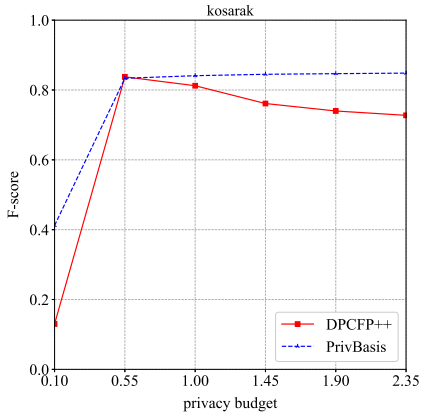

(a) BMS1



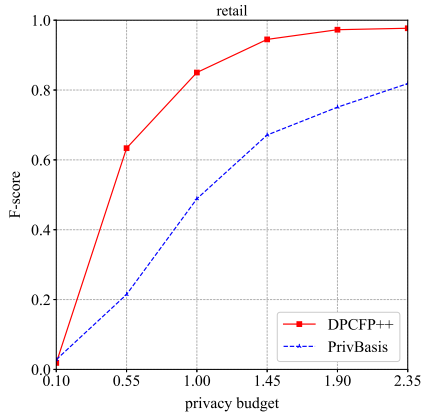

(b) BMS2



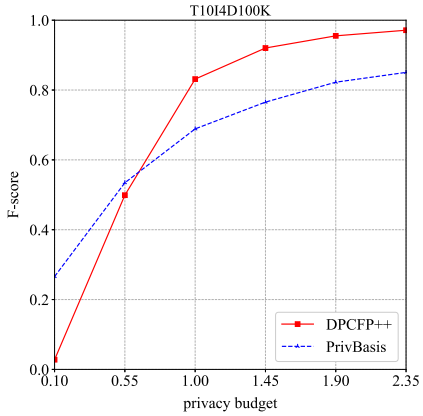

(c) BMS-POS




(d) Kosarak

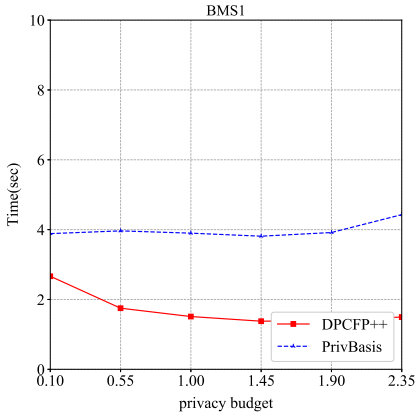



(e) Retail

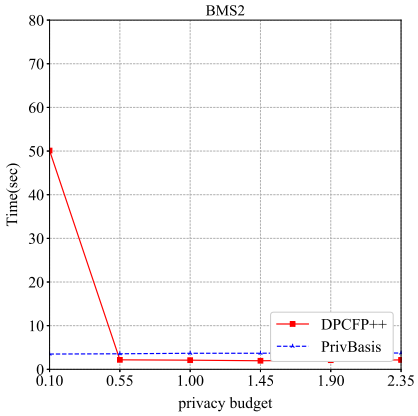



(f) T10I4D100K

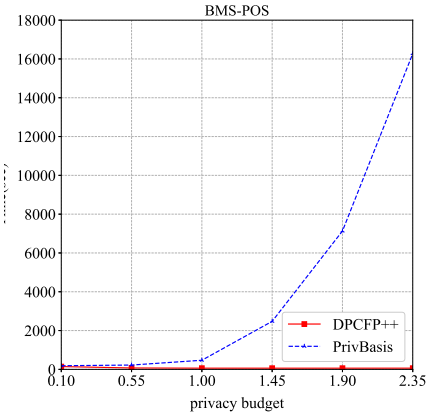
Fig. 2: F-score.



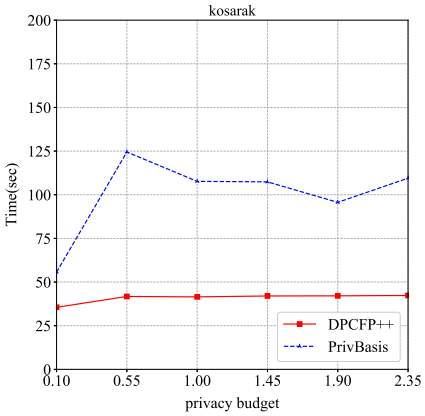

(a) BMS1



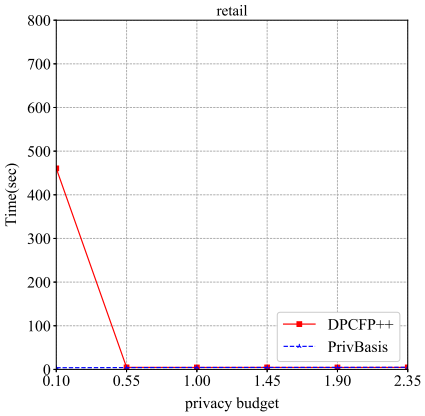

(b) BMS2



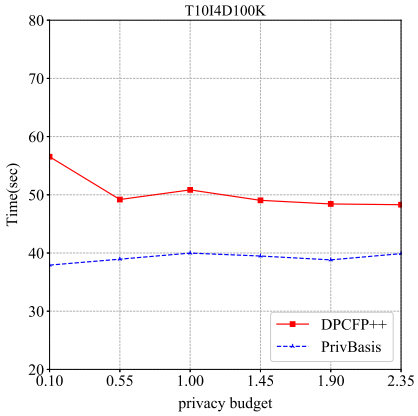

(c) BMS-POS




(d) Kosarak




(e) Retail




(f) T10I4D100K

Fig. 3: Runtime.

[2] B. Liu, W. Hsu, and Y. Ma. “Mining association rules with multiple minimum supports,” in Proceedings of ACM-SIGKDD, 1999.
[3] Y. Hu, and Y. Chen. “Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism,” in Decision Support System, 2006.
[4] R. Uday Kiran, and P. Krishna Reddy. “Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms,” in Proceedings of ACM-EDBT, 2011
[5] W. Gan, J. C. Lin, P. Fournier-Viger, H. Chao, and J. Zhan. “Mining of frequent patterns with multiple minimum supports,” in Engineering Applications of Artificial Intelligence, 2017
[6] C. Zeng, J. F. Naughton, and J. Cai, “On differentially private frequent itemset mining,” in VLDB, 2012.
[7] N. Li, W. H. Qardaji, D. Su, and J. Cao, “Privbasis: frequent itemset mining with differential privacy,” in PVLDB, 2012.
[8] J. Lee and C. W. Clifton, “Top-k frequent itemsets via differentially private fp-trees,” in KDD, 2014.
[9] C. Dwork. Differential privacy. In ICALP, 2006.
[10] Y. Chen and A. Machanavajjhala, “On the privacy properties of variants on the sparse vector technique,” CoRR, 2015.
[11] J. Zhang, X. Xiao, and X. Xie, “Privtree: A differentially private algorithm for hierarchical decompositions,” in SIGMOD, 2016.