**SCIENCE CHINA**
Information Sciences

• **Supplementary File** •

# An effective scheme for top-k frequent itemset mining under differential privacy conditions

Wenjuan Liang[1,2], Hong Chen[1], Jing Zhang[1*], Dan Zhao[1] & Cuiping Li[1]

[1] *Key Lab of Data Engineering and Knowledge Engineering of MOE, Renmin University of China, Beijing* 100872*, China;*
[2] *College of Computer and Information Engineering, Henan University, Kaifeng* 475001*, China*

## Appendix A Algorithm description

**Algorithm A1** Transaction Splitting by Count Estimation

```
Input: $\delta, \hat{F}, \hat{P}, t_i, l_{opt}, \lambda, \epsilon_1$
Output: a set of short transactions $R$
 1: $CS = \hat{F} \bigcup \hat{P}$;
 2: for $l$=3 to $\delta$ do
 3:   for each itemset $e$ of $t_i$ s.t. $|e| = l$ do
 4:     Calculate $C_{max}(e)$, $C_{min}(e)$ based on its $(l-1)$ frequent patterns in $CS, \epsilon_1$;
 5:     if $C_{max}(e) > \lambda|D|$ then
 6:       $CS = CS \cup \{(e, C_{max}(e), C_{min}(e))\}$.
 7:     end if
 8:   end for
 9: end for
10: $m = \lceil |t_i|/l_{opt} \rceil$. //the number of short transactions after splitting.
11: for $i$=1 to $m$ do
12:   $t_{temp} = \{\phi\}$; $j = |t_{temp}|$; //a short transaction being generated.
13:   while $j < l_{opt}$ do
14:     Update $f(e).weight$ of each $e$ in $CS$.
15:     Find $e_{max}$ with highest weight both in $CS$ and $t_i$.
16:     if $|t_{temp} \cup e_{max}| \leqslant l_{opt}$ then
17:       $t_{temp} = t_{temp} \cup e_{max}$; $t_i = t_i - \{e_{max}\}$;$j+= |e_{max}|$; Remove $e_{max}$ from $CS$;
18:     else
19:       add $t_{temp}$ to $R$; break;
20:     end if
21:   end while
22: end for
```

The splitting process can be seen Algorithm A1. First we privately estimate $CS$ of each long transaction $t_i$, and determine the number of short transactions after splitting (line 1-12). Next we construct an optimal short transaction $t_{temp}$ as follows (line 13-24): (1) $t_{temp}$ is initialized to $\{\phi\}$; $j$ denotes the length of $t_{temp}$. (2) If $j$ is less than $l_{opt}$, we first check wether $j$ is greater than 0, if yes, we need to update the support of each itemset in $CS$. Next we find $e_{max}$ with the highest weight that both contained in current $CS$ and $t_i$. (3) Join $e_{max}$ into $t_{temp}$, remove it from $CS$ and update $t_i$ and $j$. (4) Repeat (2)-(3)

* Corresponding author (email: Zhang-jing@ruc.edu.cn)

**Algorithm A2** Release based on weighted reservoir sampling and EM

```
Input: $D$, $\epsilon_2$, $\lambda$, $k$
Output: $\widehat{FI_k}$
1: $\epsilon' = \frac{\epsilon_2}{2k}$
2: for each $e$ in candidate frequent patterns of $D$ do
3:   if $C(e) < \lambda|D|$ then
4:     Eliminate e from candidate frequent patterns.
5:   else
6:     $C_r(e) = \frac{C(e)}{R_m(e)}$ //update the support based on equation (B2)
7:     $e.score = exp(\frac{\epsilon' C_r(e)}{2k})$. //calculate the score of e
8:     $r = Random()$. //generate a random number between $(0 \sim 1)$
9:     $e.sw = 1/r^{e.score}$. //calculate the sampling weight of e
10:    Finding the maximum weight $e_{max}$ in reservoir.
11:    if $|reservoir| < k$ or $e.sw > e_{max}$ then
12:      Add or replacing an element in reservoir with $(e, C_r(e), e.sw)$.
13:    end if
14:  end if
15: end for
16: $\widehat{FI_k} = \{e|(e, C_r(e)) \in reservoir\}$.
17: for each $e$ in $\widehat{FI_k}$ do
18:   $C_r(e) += Lap(2k/\epsilon'))$. //add LM noise to the support of $e$
19: end for
20: Return $\widehat{FI_k}$.
```

until $|t_{temp}|$ satisfies the length constraint. When one optimal short transaction is generated, we update $t_i$ and repeat the process of finding an optimal short transaction until all elements of $t_i$ are allocated. Detailed process of sampling top-k frequent patterns under differential privacy can be seen in Algorithm A2.

## Appendix B Information loss analysis of splitting

Splitting may cause information loss, because the support of some itemsets decreases after splitting. We approximate information loss by analyzing random splitting. Suppose the length of a long transaction $t$ is $l(l > l_{opt})$ and $t$ contains an $i$-itemset $X$. The length constraint on transactions is $l_{opt}$. From work [23] we can get the probability of an $i$-itemset $X$ remaining in the truncation transaction is:

$$Pr_{truncate(i,l)}(X) = \frac{\binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}} \tag{B1}$$

Based on Equation (B1), we analyze the probability that $X$ remains in $\lceil l/l_{opt} \rceil$ short transactions after splitting. After splitting $t$, there are $\lfloor l/l_{opt} \rfloor$ short transactions whose length is $l_{opt}$ and one short transaction whose length may be smaller than $l_{opt}$. Let $a = l - \lfloor l/l_{opt} \rfloor$ be the number of items in the short transaction with length smaller than $l_{opt}$.

If $a < i$, the probability of an $i$-itemset $X$ remaining in one of $\lfloor l/l_{opt} \rfloor$ short transactions is

$$\binom{\lfloor l/l_{opt} \rfloor}{1} \frac{\binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}} = \frac{\lfloor l/l_{opt} \rfloor \binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}}$$

If $a > i$, the probability that $X$ remains in the last short transaction whose length is smaller than $l_{opt}$ is $\frac{\binom{l-i}{a-i}}{\binom{l}{a}}$, so the total probability that $X$ remains in all short transactions of $t$ is $\frac{\lfloor l/l_{opt} \rfloor \binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}} + \frac{\binom{l-i}{a-i}}{\binom{l}{a}}$.

So after splitting, the probability that $X$ remains in $\lceil l/l_{opt} \rceil$ short transactions is

$$Pr_{split(i,l)}(X) = \begin{cases} \frac{\lfloor l/l_{opt} \rfloor \binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}} & \text{if } a < i \\ \frac{\lfloor l/l_{opt} \rfloor \binom{l-i}{l_{opt}-i}}{\binom{l}{l_{opt}}} + \frac{\binom{l-i}{a-i}}{\binom{l}{a}} & \text{if } a \leqslant i \end{cases}$$

We assume a uniform distribution among transactions with different cardinality containing the itemset $X$. Suppose the total number of transactions in the database is $n$. Let $f_k$ be the number of transactions of length $k$ containing $i$-itemset $X$. Let $\mu$ be the actual support of $x$, and $\mu'$ be the support of $X$ after splitting. $\mu'$ is a random variable. The expectation of $\mu'$ is

$$E(\mu') = \mu \cdot (\sum_{k=i}^{l_{opt}} \frac{f_k}{\sum_{j=1}^{n} f_j} + \sum_{k=l_{opt}+1}^{n} \frac{f_k}{\sum_{j=1}^{n} f_j} \cdot Pr_{split(i,l)}(X))$$

The remaining information rate of $i$-itemset $X$ after splitting is

$$R_m(X) = \sum_{k=i}^{l_{opt}} \frac{f_k}{\sum_{j=1}^{n} f_j} + \sum_{k=l_{opt}+1}^{n} \frac{f_k}{\sum_{j=1}^{n} f_j} \cdot Pr_{split(i,l)}(X)) \tag{B2}$$

## Appendix C Comparison with existing splitting method

Our splitting process is more efficient than existing splitting method. Existing splitting method PFP [25] is implemented by the following steps: (1)Scan the database for the first time to get the support of 2-itemset. (2)Based on the above statistics, construct an undirected weighted graph. (3)Identify the communities in the graph, and use them to construct the correlation tree (CR-Tree) to measure the correlation of all items. (4)Based on CR-Tree, scan the database for the second time to split long transactions. The reasons of inefficiency of PFP are as follows: (1)The number of database scanning is 2. (2)The communities mining is inefficient, since the number of items in the database is large. (3)CR-Tree contains the relations of all items in the database. When splitting each long transaction, they have to search the relations of all items in CR-Tree. The search space is too large. In fact, there is no need to use the relations of all items to split a long transaction. Our method can solve the above problem: (1)The number of database scanning is reduced to 1. From our experimental datasets, we can get that most transactions in the database are short; the statistics of short transactions can reflect the relations of all items. So we use the statistics of short transactions to do the count estimation. (2)For each long transaction, we only need to estimate the relations of its frequent items and do not need to find the relations of all items in the database. (3)When splitting a long transaction, the search space is its estimated frequent patterns instead of the relations of all items in CR-Tree, the search space is reduced.

## Appendix D Privacy Analysis

**Theorem 1.** Algorithm A2 satisfies $\epsilon_2$-differential privacy.

*Proof.* Let $(e_1, e_2, \cdots, e_k)$ represent the top-$k$ frequent patterns extracted from the database $D$ using the reservoir sampling, and $Pr(e_1, e_2, \cdots, e_k|D)$ represents the probability of extracting $k$ patterns from the database $D$. We can get

$$Pr(e_1, e_2, \cdots, e_k|D) = \prod_{i=1}^{k} \frac{exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}$$

$C(e_i, D)$ is the count of the pattern $e_i$ in database $D$, $\triangle C$ is the global sensitivity.

If we can prove $\frac{Pr(e_1,e_2,\cdots,e_k|D)}{Pr(e_1,e_2,\cdots,e_k|D')} \leqslant exp(\epsilon_2/2)$, which means that the probability ratio of sampling $k$ patterns from neighbor databases $D$ and $D'$ is no greater than $exp(\epsilon_2/2)$, we can get the result that the sampling process in Algorithm A2 satisfies $\epsilon_2/2$-differential privacy. The proof contains two steps: Firstly, we need to prove $\frac{Pr(e_i|D)}{Pr(e_i|D')} \leqslant exp(\epsilon')$, which means that the probability ratio of extracting one pattern from neighbor database $D$ and $D'$ is no greater than $exp(\epsilon')$. Secondly, we need to prove the probability ratio of sampling $k$ patterns is no greater than $\epsilon_2/2$. The proof of the first step is as follows.

$$\begin{aligned}
\frac{Pr(e_i|D)}{Pr(e_i|D')} &= \frac{\frac{exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}}{\frac{exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}} = \frac{exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})} \times \frac{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})} \\
&\leqslant exp(\frac{\epsilon'}{2}) \times exp(\frac{\epsilon'}{2}) \times \frac{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})} \\
&= exp(\epsilon')
\end{aligned}$$

The proof of the second step is as follows.

$$\frac{Pr(e_1,e_2,\cdots,e_k|D)}{Pr(e_1,e_2,\cdots,e_k|D')} = \frac{\prod_{i=1}^{k} \frac{exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}}{\prod_{i=1}^{k} \frac{exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}}$$
$$= \prod_{j=1}^{k} \frac{exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}{exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})} \times \frac{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D')}{2\triangle C})}{\sum_{e_i \in FI} exp(\frac{\epsilon' C(e_i,D)}{2\triangle C})}$$
$$= \prod_{j=1}^{k} \frac{Pr(e_i|D)}{Pr(e_i|D')} \leqslant \prod_{j=1}^{k} exp(\epsilon') = exp(k \cdot \epsilon') = exp(\epsilon_2/2)$$

According to the above proof, we can get the process of sampling k patterns based on EM satisfies $\epsilon_2/2$-differential privacy. Since the process of applying LM noise to top-k frequent patterns satisfies $\epsilon_2/2$-differential privacy, Algorithm A2 satisfies $\epsilon_2$-differential privacy.

**Theorem 2.** Our scheme satisfies $\epsilon$-differential privacy.

*Proof.* Our proposed scheme consists of Algorithm A1 and Algorithm A2. Algorithm A1 satisfies $\epsilon_1$ differential privacy. From Theorem 1, Algorithm A2 satisfies $\epsilon_2$ differential privacy. According to the sequential composition, our scheme satisfies $\epsilon = \epsilon_1 + \epsilon_2$ differential privacy.

## Appendix E Experiments

In this section, we verify data utility and efficiency of our proposed scheme on real datasets. We conduct all experiments on a PC with Intel(R) Core(TM) i7-3540M CPU (3.00Ghz) and 8G RAM.

**Comparison**. We compare our algorithm with two state-of-the-art algorithms in [25] and [26]. We use Diff-FIMCE to denote our algorithm, while PFP and PrivSuper denote the above two algorithms respectively. All algorithms are implemented in Java. Since algorithms involve randomization, we ran each algorithm ten times to obtain its average performance.

**Metrics**. To evaluate the performance of our algorithm, we employ **the running time** (the time period between input and output) to measure the efficiency, and employ the standard metrics F-score [23] to measure the utility of generated frequent itemsets.

**Datasets**. Real datasets we used in experiments are PUMSB [30], POS [31], BMS-WebView-1(WV1) [31] and BMS-WebView-2(WV2) [31]. Detailed information of datasets is shown in Table E1.

**Table E1** Detailed information of datasets

| **Dataset** | $\boldsymbol{|D|}$ | $\boldsymbol{|I|}$ | **Max**$\boldsymbol{|t|}$ | **Avg**$\boldsymbol{|t|}$ |
|---|---|---|---|---|
| PUMSB | 49046 | 2088 | 63 | 50 |
| POS | 515597 | 1657 | 164 | 6.5 |
| BMS-WebView-1(WV1) | 59602 | 497 | 267 | 2.5 |
| BMS-WebView-2(WV2) | 77512 | 3340 | 161 | 5.0 |

$|D|$ is the number of records of a dataset, $|I|$ is the number of distinct items, Max$|t|$ and Avg$|t|$ denote the maximal and the average record length respectively.

### Appendix E.1 Effect of $l_{opt}$

In our scheme, the optimal length $l_{opt}$ is set to the value that the percentage of the transactions with cardinality no greater than $l_{opt}$ is at least $\eta$ percentage. With the change of $\eta$, we observe the utility of the release result of our scheme on four datasets. The parameters used here are set as: $\lambda$ is 0.08 on POS, 0.7 on PUMSB, 0.01 on WV1 and 0.012 on WV2. $\alpha$ and $\gamma$ are 0.5. $\rho$ is 0.01. $\epsilon$ is 1. $\eta$ varies from 0.5 to 0.9. From Fig E1, we can see, when $\eta$ is small, the results on F-score are poor(0.6-0.72). This is because lots of information is lost due to the transformation of the database. Then, the utility is improved as $\eta$ increases from 0.5 to 0.8. When $\eta$ increases to 0.8, the F-Scores on four datasets achieve highest(0.86-0.93). When the constraint is larger than 0.8, the utility has fallen slightly(0.7-0.83). This is because more noise is required according to differential privacy. It offsets the gains obtained by preserving information, which decreases the quality of the results.

### Appendix E.2 Utility

In this experiment, we compare the utility of three algorithms. The parameters we used here are set as follows: $\lambda$ is set to 0.08 on POS and 0.7 on PUMSB, 0.01 on WV1 and 0.012 on WV2. $\gamma$ is 0.5, $\rho$ is set to 0.01, $\eta$ is set to 0.8. $\alpha$ varies from 0.1 to 0.9, $k$ varies from 10 to 200, $\epsilon$ varies from 0.5 to 1.5. With the change of $\alpha$, we observe the effect on the utility

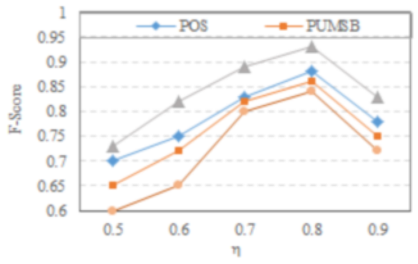

**Figure E1** Effect of $l_{opt}$(F-Score vs. $\eta$)

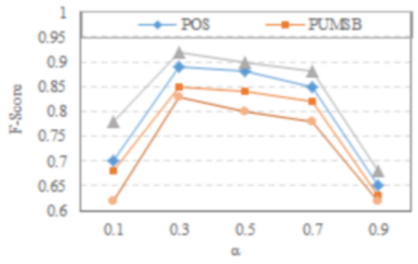

**Figure E2** Effect of $\alpha$ on utility

of our proposed scheme (Diff-FIMCE). With the change of $k$ and $\epsilon$, we observe the utility of the three algorithms on four datasets.

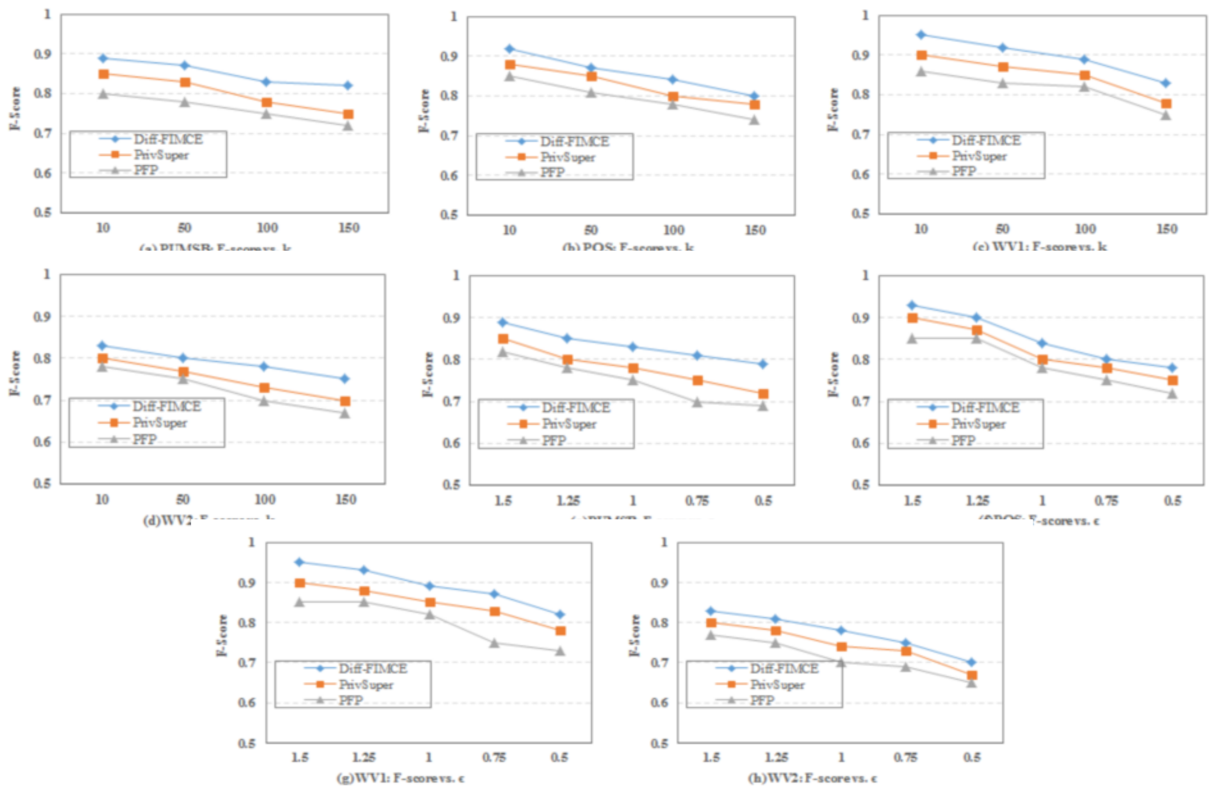

**Figure E3** Utility Evaluation

**Effect of $\alpha$ on utility.** We observe the utility of Diff-FIMCE on four datasets with the change of $\alpha$. Here we set $\epsilon = 1$ and $k = 100$. From Fig.E2, we can see, when $\alpha$ varies from 0.1 to 0.2, the results on F-score are relatively poor($< 0.75$). This is because the magnitude of noise added to transaction splitting is too large, which results in a larger information loss and affects the accuracy of the release result. When $\alpha$ varies from 0.3 to 0.7, the results on F-score are relatively good(0.83-0.93). When $\alpha$ varies from 0.7 to 0.9, the utility becomes lower($< 0.8$). The main reason is that when privately sampling top-k frequent itemsets based on EM, too much noise is added to the support of the patterns and affects the accuracy.

**Effect of k on utility.** We set $\epsilon = 1$, $\alpha = 0.5$ and observe the utility of Diff-FIMCE, PrivSuper and PFP with different values of $k$ on four datasets. From Fig.E3 (a)-(d), we can see that the F-Scores of Diff-FIMCE always higher than that of PrivSuper(+3-5%) on each dataset in all cases. This is because we split long transactions instead of random truncation, the information loss is much lower. The F-Scores of Diff-FIMCE always higher than that of PFP(+5-9%). Comparing with PFP, when splitting we consider only frequent patterns of this transaction instead of the relationship of all items, so it is much more accurate. PrivSuper performs better than PFP(+2-4%), this is because PrivSuper directly searches for maximal frequent itemsets, and subsequently adds their sub-itemsets to the results without additional privacy budget consumption. With the increase of $k$, the F-scores of the three algorithms are all reduced. This is because the amount of added noise is greater.

**Effect of $\epsilon$ on utility.** We set $k = 100$, $\alpha = 0.5$ and observe the utility of three algorithms with varying the privacy budget $\epsilon$ from 0.5 to 1.5 on four datasets. From Fig.E3 (e)-(h), we find that Diff-FIMCE always achieves better performance than the other two algorithms(+3-10%) for the same privacy level. The F-scores of PrivSuper are always higher that of PFP(+2-5%), this is because they only add noise to the maximal frequent itemsets and thus reduces the sensitivity of the private release. With the decrease of privacy budget, the values of three algorithms on F-score are decreasing. The reason is that more noise is added to the computation of frequent patterns. For the same privacy budget, the value of F-scores on POS and WV1 are higher than the values on PUMSB and WV2. This can be explained by less high support of itemsets in POS and WV1, which causes less information loss in transaction splitting.

## Appendix E.3 Efficiency

In this experiment, we compare the execution time of three algorithms. The parameters we used in this experiment are set as follows: The privacy budget $\epsilon$ is set to 1.0. $\alpha$ and $\gamma$ are 0.5. $\rho$ is set to 0.01, $\eta$ is set to 0.8. $\lambda$ is set to 0.08 on POS and 0.7 on PUMSB, 0.01 on WV1 and 0.012 on WV2. $k$ varies from 10 to 200. With the change of $k$ , we observe the efficiency of three algorithms on four datasets.

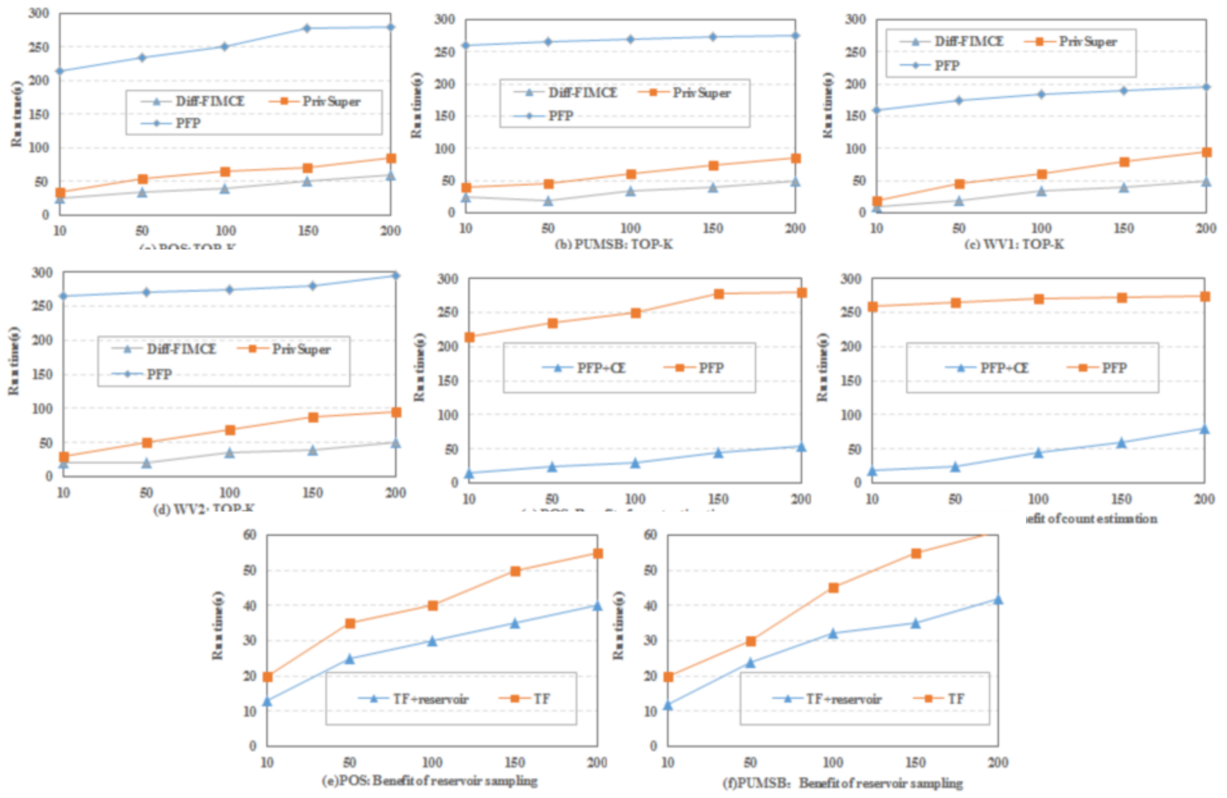

**Figure E4** Efficiency Evaluation

**Effect of k on efficiency.** From Fig.E4 (a)-(d), we can observe the running time of three algorithms on four datasets respectively with the increase of $k$. We can see that Diff-FIMCE performs better than PrivSuper, the running time of Diff-FIMCE is always lower than that of PrivSuper about 8-50s. This is because PrivSuper needs to perform the FIM algorithm twice, which consumes too much time. The first time is performing an no private FIM algorithm to get the statistics of the true top-k frequent itemsets, which will be used to save the privacy budget in the next privately release; the second time is performing a FIM algorithm in a differential privacy way by using the above statistics to get the noisy top-k frequent itemsets. From the figures, we can also see that Diff-FIMCE is more efficient than PFP. The running time of Diff-FIMCE is always lower than that of PFP about 145-255s. The reason for low efficiency of PFP is that it needs to find the correlation of all items in advance and uses the relation to guide the splitting. Their splitting method is too inefficient, it always consumes several hundreds of seconds on each dataset. We has improved the efficiency of the transaction splitting method, the core idea of our method is finding the relations of frequent items in each long transaction based on count estimation, which can reduce the preprocessing time and the number of database scanning. Moreover, we employ weighted reservoir sampling combing with EM to further improve the efficiency of the privately release. Therefore, Diff-FIMCE performs best. With the increase of $k$, the running time of three algorithms on four datasets are all increasing. This is because the size of candidate frequent item sets becomes larger, which consumes much more time to do the mining.
**Benefit of splitting based on count estimation.** To better understand the benefit of transaction splitting based on count estimation, we apply it to PFP by replacing its method of transaction splitting. Let PFP+CE denote the modified scheme. We compare the running time of PFP and PFP+CE with different values of k on PUMSB and POS. The parameters we used are set as: $\epsilon$ is 1.0, $\lambda$ is set to 0.08 on POS and 0.7 on PUMSB, 0.01 on WV1 and 0.012 on WV2, $\alpha$ and $\gamma$ are 0.5, $\rho$ is 0.01, $\eta$ is 0.8, $k$ varies from 10 to 200. From Fig.E4(e)-(f), we can see that the modified scheme PFP+CE performs better performance than PFP on each dataset, the running time of PFP+CE can save at least 200s comparing with PFP. The main reason is that we split long transactions based on count estimation instead of the correlation of all items in database,

which reduces the preprocessing time. PFP needs to find the correlation of all items in the preprocessing, which always consumes hundreds of seconds and results in lower efficiency. PFP+CE scans the database only once and needs a little preprocessing time, its efficiency is higher.

**Benefit of weighted reservoir sampling.** To better understand the benefit of weighted reservoir sampling, we apply it to TF [22] by replacing its sampling method of EM. TF [22] is a private top-k release scheme based on EM by considering the frequent patterns with length no greater than $l$. Let TF+reservoir denote the modified scheme. Next we compare the running time of TF and TF+reservoir with different values of k on on PUMSB and POS. The parameters we used are set as: $\epsilon$ is 1.0, $k$ varies from 10 to 200. To better observe the performance , we set $\lambda$ to 0.04 on POS and 0.5 on PUMSB. Since the more the candidate frequent patterns the higher the difference of the two methods. From Fig.E4(g)-(h), we can see that the modified scheme TF+reservoir performs better performance than TF on each dataset, its running time can be saved about 8-20s. TF privately releases top-k frequent patterns based on EM and its core strategy is weighted random sampling. It needs to traverse the frequent itemsets twice. TF+reservoir only needs to traverse the frequent itemsets only once, it is more efficient.

## Appendix F Related works

Recently, more and more works have begun addressing privacy preserving of different data mining tasks [4], such as classification [35], clustering [5,34] and frequent patterns mining. We focus on summarizing privacy preserving of frequent itemsets mining (PPFIM). These works can be divided into three categories:

[**Secure computation**] A number of schemes for PPFIM have been proposed based on secure computation. They focus on the privacy preservation of FIM in a distributed environment or in an outsourcing scenario. For example, in a distributed environment, the works in [6,7] employ secure computation of scalar products to find global frequent itemsets across vertically partitioned data. Clifton et al. [8] employ secure multi-party computation to find global frequent itemsets across horizontally partitioned data. When the task of FIM is outsourced, Wong et al. [9,10] propose a secure encryption scheme based on item mapping for privacy preservation in frequent itemset mining. Evfimievski et al. [11] design some random operations to protect the privacy of FIM. These works are different from our work, they all focus on avoiding privacy leakage during collaborative computations among multiple parties, while we focus on the release of frequent itemsets itself does not leak private information in a centralized scenario.

[**K-anonymity**] There are some schemes for FIM based on $k$-anonymity. $k$-anonymity means for any transaction $t_i$ in a dataset, and for any subset of $m$ items in $t_i$, there are at least $k-1$ other transactions with the same $m$ items. Atzori et al. [12] studies how to eliminate re-identification attacks for frequent patterns mining based on $k$-anonymity. Xu et al. [13] propose a $(h,k,p)$-coherence anonymity framework to release frequent itemsets. Hua et al. [14] propose a $k$-support anonymity scheme based on pseudo taxonomy for FIM. Several anonymity works for pubishing transactional data are also loosely related to our work. [15] presents a rule-based anonymity model for publishing transactional dataset. [16–18] propose some anonymity schemes with a reduced information loss and a lower computational complexity of the anonymization process. [19] studies the privacy breach caused by unsafe correlations in transactional data where individuals have multiple tuples in a dataset. [32] studies a new technique ensuring privacy in big data: K-anonymity without prior value of the threshold k. [33] proposes an efficient scheme based on K-anonymity. [36,37] study how to reduce side effects of hiding sensitive itemsets and make an optimizing based on GA-based algorithms. $K$-anonymity cannot provide sufficient privacy protection against adversaries with arbitrary prior knowledge. These schemes are vulnerable to many types of privacy attacks, for example the composition attack [20,38], foreground knowledge attack [21].

[**Differential privacy**] Differential privacy [2] can offer strong theoretical guarantees against attackers with arbitrary background knowledge. For this reason, several studies start to address this issue by differential privacy. Since the high dimensionality of long transactions, the sensitivity is very high. To reduce sensitivity, several methods are proposed. Bhaskar et al. [22] propose two kinds of differentially private FIM schemes based on LM [2] and EM [3] by considering candidate frequent patterns with length no greater than $l$. Zeng et al. [23] propose a differentially private FIM scheme, which contains a heuristic transaction truncation method to reduce sensitivity. Li et al. [24] propose to construct a graph to find the basis set, and then project long transactions to the basis set to reduce sensitivity. Su et al. [25] propose a method of transaction splitting to reduce sensitivity, first they find the relationship of all items, and then split transactions based on the above results. PriSuper [26] employs random sampling to truncate long transactions, and then uses SEM mechanism to release top-$k$ frequent patterns based on the maximum frequent itemsets found in advance. The above schemes cannot achieve both high utility and efficiency. The latest work [27] presents a scheme for frequent itemset mining based on local differential privacy.

**References**

1 Garofalakis M et al. Querying and mining data streams: you only get one look a tutorial. In: Proceedings of ACM SIGMOD International Conference, Madison, 2002. 635

2 C. Dwork, F. Mcsherry, K. Nissim, Calibrating noise to sensitivity in private data analysis. In: Proceedings of Theory of Cryptography Conference, New York, 2006. 265-284

3 F. McSherry and K. Talwar. Mechanism design via differential privacy. In: Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science, Providence, 2007. 94-103

4 Friedman A, Schuster A et al. Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, 2010. 493-502

5 Ling Chen, Ting Yu, Rada Chirkova. WaveCluster with Differential Privacy. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, 2015. 1011-1020

6 J. Vaidya et al. Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, 2002. 639-644

7 Nirali R. Nanavati* and Devesh C. Jinwala. A novel privacy-preserving scheme for collaborative frequent itemset mining across vertically partitioned data. Security and Communication Networks, 2015, 8(18):4407-4420

8 M. Kantarcioglu and C. Clifton et al. Privacy-preserving distributed mining of association rules on horizontally partitioned data. TKDE, 2004, 16(9):1026-1037

9 W. K.Wong, D.W. Cheung et al. Security in outsourcing of association rule mining. In: Proceedings of the 33rd International Conference on Very Large Data Bases, Austria, 2007.111-122

10 W. K.Wong, D.W. Cheung et al. An audit environment for outsourcing of frequent itemset mining. PVLDB, 2009, 2(1):1162-1172

11 A. Evfimievski, R. Srikant et al. Privacy preserving mining of association rules. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Canada, 2002.217-228

12 Maurizio Atzori, F. Bonchi et al. Anonymity preserving pattern discovery. VLDBJ, 2008, 17(4):703-727

13 Xu Y, Wang K, Fu A et al. Anonymizing Transaction Databases for Publication. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, 2008.767-775

14 Chih-Hua Tai,Philip S. Yu et al. k-Support Anonymity Based on Pseudo Taxonomy for Outsourcing of Frequent Itemset Mining. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, 2010.473-482

15 Grigorios Loukides, Utility-preserving transaction data anonymization with low information loss. Expert Syst.Appl. 2012, 39(10): 9764-9777

16 Grigorios Loukides. Efficient and flexible anonymization of transaction data. Knowl.Inf.Syst. ,2013, 36(1):153-210

17 Shyue-Liang Wang. On anonymizing transactions with sensitive items. Appl. Intell. 2014, 41(4):1043-1058

18 Jerry Chun. PTA: An Efficient System for Transaction Database Anonymization. IEEE Access, 4(2016):6467-6479

19 Bechara al Bouna. Anonymizing transactional datasets. Journal of Computer Security, 2015, 23(1):89-106

20 Wong R C W, Fu A et al. Can the utility of anonymized data be used for privacy breaches. TKDD, 2011, 5(3):1-24

21 C. Dwork, A. Roth et al. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 2014, 9(3-4):211-407

22 R. Bhaskar, S. Laxman et al. Discovering frequent patterns in sensitive data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, 2010. 503-512

23 C. Zeng, J. F. Naughton et al. On differentially private frequent itemset mining. PVLDB, 2012, 6(1):25-36

24 N. Li, W. H. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. PVLDB, 2012, 5(11): 1340-1351

25 Sen Su, Shengzhi Xu et al. Differentially Private Frequent Itemset Mining via Transaction Splitting. In: Proceedings of 32nd IEEE International Conference on Data Engineering, Helsinki, 2016. 1564-1565

26 Ning Wang, Xiaokui Xiao et al. PrivSuper: a Superset-First Approach to Frequent Itemset Mining under Differential Privacy. In: Proceedings of 33nd IEEE International Conference on Data Engineering, San Diego, 2017. 809-820

27 Tianhao Wang, Ninghui Li, Somesh Jha. Locally Differentially Private Frequent Itemset Mining. In: IEEE Symposium on Security and Privacy, San Francisco, 2018. 127-143

28 C. Hidber, Finding Recent Frequent Itemsets Adaptively over Online Data Streams. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, 2003. 487-492

29 Pavlos S. Efraimidis. Weighted random sampling with a reservoir. Inf.Process.Lett., 2006, 97(5):181-185

30 Frequent itemset mining dataset repository. http:// fimi.ua.ac.be/data.

31 Kohavi, R. KDD-Cup 2000 Organizers' Report: Peeling the Onion. SIGKDD Exploration, 2000, 2(2):86-93.

32 Zakariae El Ouazzani, Hanan El Bakkali. A new technique ensuring privacy in big data: K-anonymity without prior value of the threshold k. Procedia Computer Science, 2018, 127:52 C 59

33 Jerry Chun-Wei Lin, Qiankun Liu et al. PTA: An Efficient System for Transaction Database Anonymization. IEEE Access, 4(2016): 6467-6479

34 Fang Liu, Tong Li. A Clustering k-Anonymity Privacy-Preserving Method for Wearable IoT Devices. Security and Communication Networks, 2018, 4945152:1-8.

35 Ping Li, Jin Li et al. Privacy-preserving outsourced classification in cloud computing. Cluster Computing, 2018, 21(1): 277 C 286.

36 Chun-Wei Lin, Tzung-Pei Hong. Reducing Side Effects of Hiding Sensitive Itemsets in Privacy Preserving Data Mining. Scientific World Journal. 2014. 1-12

37 Chun-Wei Lin, Tzung-Pei Hong. The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion. Appl. Intell., 2015, 42(2): 210-230

38 A S M Touhidul Hasan, Qingshan Jiang. A New Approach to Privacy-Preserving Multiple Independent Data Publishing. Applied Sciences, 2018, 8(5):783