ESWA_FT-PatternGrowth

ABSTRACT

fault tolerant(FT) frequent 相比与普通的频繁项集模式挖掘更昂贵

Apriori系列算法会指数级的生成大量的候选项级，包含数据库里没有的，并且会多次扫描数据库区计算支持度

提出一种新颖的算法利用频繁模式增长方法挖掘频繁项级，采用分而治之的思想，将事物数据库递归的投影到一组小的投影事物数据库中，通过探索局部频繁项级来挖掘容错频繁项级

FT-patternGrowth将书屋数据库存储在一个高压缩的更小数据结构FT-tree中，直接在树中计数，无需多次扫描数据库，提高算法的处理速度

Introduction

普通的频繁项级挖掘方法在项目集部分缺失时会面临阈值设置问题，阈值过大挖掘过少，阈值过小会冗余

FT-PatternGrowth　引入容错因子δ

an itemset X with length greater than (δ+1) is a FT frequent itemsets if it has support of al laset T number of FT-transactions

A transactions t is a FT-transaction of T under FT factor if it contains at least(|X|−δ) number of items of X

min_sup^{δ}　T is the support of X which must be greater or eqoals to the minimum itemset support

$item\_sup^{\delta}$　Each individual item i of X must apper in at least m number of FT-transactions of X　The m is the minimum item support under fault tolerant factor δ

Table 1
A sample transactional database. Items are removed from the transactions that have items support less than 3.

TID	Items	(Ordered) Frequent Items
10	b, c, e, f	e, f, c, b
20	a, b, e, f	e, f, a, b
30	e, a, f	f, e, a
40	e, g	e
50	a, b, c, e, h	e, a, c, b
60	a, e, f	e, f, a
70	d, c, f	f, c,
80	d, i, h	d
90	d, j	d

这里给出容错频繁挖掘的定义，假设min_supδ=3，item_supδ=2,允许一个不匹配，δ=1　FT-Apriori算法在阈值较小时处理时间长

项目集abcef具有项目集支持为3，事务10,30,50包含五个项目中的4个，并且每个单独项目出现在三个事务中的两个中

FT-Apriori
- 主要基于Apriori的候选生成和测试，数据处理量大
- 采用自下而上的搜索机制，在挖掘项集X之前先枚举了所有子集，这种机制限制了算法在合理的时限内挖掘完整的项目集
- 在挖掘时会多次扫描数据库

FT-patternGrowth
- 采用分而治之的思想，将事务数据库递归的投影到一个较小的投影数据库中，只探索局部频繁项来挖掘，减少数据库中不存在的候选项集
- 消除了apriori算法的候选生成测试和多次扫描数据库
- 第一次扫描数据库计算出所有长度为1的频繁项集，第二次扫描建立频繁模式树

Related work

Applications of mining FT frequent itemsets
- 从图数据库中挖掘频繁子图
- 使用频繁子图进行图像分类，图像的频繁近似子图被用于分类特征
- 确定和不确定数据集的频繁模式挖掘算法
- 利用频繁项集挖掘生物数据库中的项集的算法，比例项集的可容忍的错误数量与项集的长度成正比

FT frequent itemset mining algorithm
- FT-Apriori　基于apriori算法，应用自下而上的完全搜索空间，应用向下封闭属性来剪枝即发现长度为K的子集不频繁则抛弃其所有超集
- VB-FT-Mine　避免数据库重复扫描，只需扫描一次，为每个项目生成位向量，采用深度优先模式生成候选项集，使用位运算快速计算项集支持度，但是会生成大量的候选项集
- pattern growth　construct more than ont FP-trees for each itemset to mine its supersets
  - 在δ下，建立（δ+1）个FP-trees
  - 在X和δ下，建立2^|X|个FP-trees　此两种方法都不能充分利用FP-tree的计数特性

挖掘比例项集的概念与传统的项集相似，但比例项集的容错因子与项集的长度成正比

Fault tolerant(FT) frequent itemset mining:Problem statement

$I = i_1, \ldots, i_m, X \in I$　$|X|$

$A \quad transaction \quad T = (tid, t) \quad t = t_1, \ldots, t_n$

$T \in TDB$

$X \quad is \quad frequence \quad sup(X) \geq min\_sup$

$number \quad of \quad FT - transaction \geq min\_sup^{\delta}$

$\begin{cases} at \quad least \quad (|X| - \delta) \quad in \quad t \\ |X| \geq (\delta + 1) \end{cases}$

容错频繁项集

$m = item\_sup^{\delta}$

Each individual item i of X must appear in at least m number of FT-transaction of X

MIning fault tolerant(FT) frequent itemsets using pattern growth:Design and construction

Mining FT frequent itemsets from FT-Fp-tree

bcad

bcadf

bcadfe

什么也不是根据上表，第一，三个容错因子变为了3，第二个d=0<item_sup

Algorithm 2: Procedure for constructing FT-FP-Tree of itemset.
Data: Conditional patterns of itemset X
Result: Construct FT-FP-Tree of X
Procedure FT-FP-Tree(*T*, *X*)
for *each conditional pattern* c *in* C **do**
  convert c into FT-conditional pattern c^x;
  if c^x *contains at-least* $(|X| - \alpha)$ *items of* X **then**
    $C^x = C^x \cup c^x$;
construct FT-FP-Tree of X using C^x;
if *FT-FP-Tree of* X *!= ∅* **then**
  FT-FP-Growth(*FT-FP-Tree of X, X*);

Mining FT frequent itemsets of length more than (δ+1)

构建一个FT-FP-tree，和FP-tree树类似，里面包含了X的所有的条件模式，A FT-conditional pattern of itemset (X) contains four segments.

The first segment contains items that can generate itemsets contains

The second segment contains support of FT-conditional pattern

The third segment contais number of missing items of X in FT-conditionan pattern

The fourth segment contains item support of each item of X,which is useful for counting support of items

FT-CP-table at the end of each branch of FT-FP-tree

Table 3
FT-conditional patterns of itemset {bcad}.

FT-Conditional Patterns Discovered from FT-FP-tree of itemset {bca}	FT-Conditional Patterns used for Constructing FT-FP-tree of {bcad}
$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$	$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 2, a\ 1, c\ 1, b\ 0\}\}$
$\{\{af\}, \{sup: 2\}, \{\delta: 1\}, \{ac\ 1, c\ 1, b\ 2\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 2\}\}$ Ignored because it has (δ: 2) and item d is missing in the pattern
$\{\{af\}, \{sup: 1\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$	$\{\{af\}, \{sup: 1\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 0\}\}$
$\{\{e\}, \{sup: 3\}, \{\delta: 0\}, \{ac\ 3, c\ 1, b\ 1\}\}$	$\{\{e\}, \{sup: 3\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 1\}\}$

- $\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$,
- $\{\{af\}, \{sup: 2\}, \{\delta: 1\}, \{ac\ 1, c\ 1, b\ 2\}\}$,
- $\{\{af\}, \{sup: 1\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$, and
- $\{\{e\}, \{sup: 3\}, \{\delta: 0\}, \{ac\ 3, c\ 1, b\ 1\}\}$.

The conditional pattern $\{\{af\}, \{\delta: 2\}\}$ is ignored because it contains FT factor $\delta = 2$ and item d is missing from the pattern, this makes the FT factor equals to $\delta = 3$. All other conditional patterns qualify FT factor $\delta = 2$ which make the support count of itemset {bcad} equals to 5. Table 3 shows the FT-conditional patterns of {dacb}. The supported items are collected from FT-conditional patterns of {bcad} which makes the support of itemset $\{af, 2\}$, $\{ac\ 3\}$, and $\{b\ 3\}$. All items qualify $item\_sup^{\delta} = 2$. Thus {bcad} is a FT frequent itemset of length four. The Fig. 7 shows the FT-FP-tree of {bcad}.

Table 4
FT-conditional patterns of itemset {bcadf}.

FT-Conditional Patterns Discovered from FT-FP-tree of itemset {bcad}	FT-Conditional Patterns used for Constructing FT-FP-tree of {bcadf}
$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 2, a\ 1, c\ 1, b\ 0\}\}$	$\{\{\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, a\ 2, c\ 1, b\ 0\}\}$
$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 2\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, a\ 0, a\ 1, c\ 1, b\ 2\}\}$
$\{\{af\}, \{sup: 1\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 0\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, af\ 0, a\ 1, c\ 1, b\ 2\}\}$
$\{\{af\}, \{sup: 1\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 1\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, af\ 0, a\ 1, c\ 1, b\ 2\}\}$

bcadf

什么也不是根据上表，第一，三个容错因子变为了3，第二个d=0<item_sup

bcadfe

Fig. 1. FP-tree after inserting first transaction.

Fig. 4. FP-tree after inserting second transaction.

Fig. 5. Complete FP-tree after inserting all transactions.

每一交易都会被映射到FP-tree的分支，公共部分会被合并，两次扫描数据库

第一次计算每个项目的sup,大于小于阈值的会被删除，每条交易中被保留的项目则会根据支持频率重新排序

第二次扫描数据库，会构建FP-tree树，逐条插入，有相同前缀的会被插入到同一条分支中

项集X是一个频繁项集。算法会产生项a,b,c的conditoinal patterns。算法会忽略其他项的conditional patterns，这是因为如果FP-tree的一个分支不包含X中的任何项，那么该分支的FT因子则变为（δ=3），不符合（δ=2），算法会以频繁度递增的顺序枚举X中的项。对于项集X，FP-tree会生成如下conditional patterns集合

$$\begin{cases} b :< efab : 1 >, < efcb : 1 >, < eacb : 1 > \\ c :< efc : 1 >, < eac : 1 >, < fdc : 1 > \end{cases}$$

依次找出b,c的条件模式树,若条件模式CB是之前已经发现的条件模式的子集，则将CB的支持度从CA中减去，若CB减为0，则CB被忽略，c中efc和eac为子集被减去，则c中条件模式只剩下fdc

$$a :< efa : 2 >, < ea : 1 >, < fda : 1 >$$

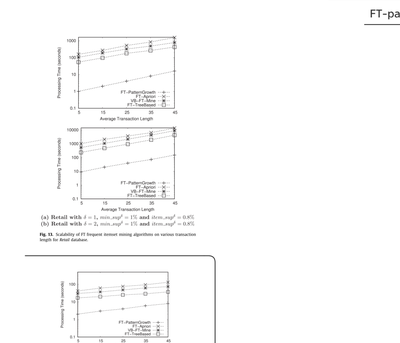
继续找出a的条件模式, 继续和之前已经找到的比较，剩余<efa:1>,<fda:1>

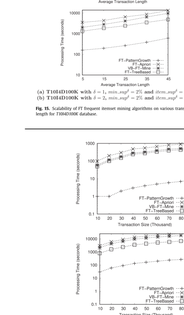
$$\begin{cases} b :< efab : 1 >, < efcb : 1 >, < eacb : 1 > \\ c :< fdc : 1 > \\ a :< efa : 1 >, < fda : 1 > \end{cases}$$

综上，剩余条件模式为,因为同时满足min_sup和item_sup所以X是长度为3的FT frequent

$$X = (bca)$$

Mining FT frequent itemsets of length equals to (δ+1)


Algorithm 1: Procedure for Mining Fault Tolerant Frequent Itemsets.
Data: Transactional database, min_sup^{α}, $item\_sup^{\beta}$, Fault Tolerant factor (δ)
Result: All Fault Tolerant (FT) frequent itemsets
Procedure FT-FP-Growth(*Tree*, *α*)
$\beta = \alpha$;
if $\alpha == \varnothing$ **then**
  for *each itemset X of length* $= (\delta + 1)$ **do**
    for *each items i of X* **do**
      $C = C \cup$ *i's conditional patterns from Tree*;
      if *itemset* $-$ *support*$(X) >= min\_sup^{\alpha}$ **then**
        output X as FT frequent itemset;
        $\beta = X$;
        FT-FP-Tree(*C*, *β*);
else
  if *Tree contains a single path P* **then**
    for *each items* P_i *in the P* **do**
      $X = \alpha \cup P_i$;
      if *itemset* $-$ *support*$(X) >= min\_sup^{\alpha}$ *and each item i in X has item_support*$(i) >= item\_sup^{\beta}$ **then**
        output X as a FT frequent itemset;
  else
    for *each header item* h_i *in the header table of Tree* **do**
      generate itemset $X = \alpha \cup h_i$;
      for *each path P of Tree* **do**
        $C = C \cup$ generate conditional pattern from P_i;
      if *itemset* $-$ *support*(X) *in* $C >= min\_sup^{\alpha}$ *and each item i of X in C has item_support*$(i) >= item\_sup^{\beta}$ **then**
        output X as a FT frequent itemset;
        $\beta = X$;
        FT-FP-Tree(*C*, *β*);



The first segment contains items that can generate itemsets contains

The second segment contains support of FT-conditional pattern

The third segment contais number of missing items of X in FT-conditionan pattern

The fourth segment contains item support of each item of X,which is useful for counting support of items


FT-CP-Table head pointer
e 4
f 5
d 2
Fig. 6. FT-FP-tree of itemset {bca}.

FT-CP-table at the end of each branch of FT-FP-tree

Table 2
FT-conditional patterns of itemset {bcad}.

FT-Conditional Patterns Discovered from FT-FP-tree of itemset {bca}	FT-Conditional Patterns used for Constructing FT-FP-tree of {bcad}
$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$	$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 2, a\ 1, c\ 1, b\ 0\}\}$
$\{\{af\}, \{sup: 2\}, \{\delta: 1\}, \{ac\ 1, c\ 1, b\ 2\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 2\}\}$ Ignored because it has (δ: 2) and item d is missing in the pattern
$\{\{af\}, \{sup: 1\}, \{\delta: 2\}, \{ac\ 1, c\ 1, b\ 0\}\}$	$\{\{af\}, \{sup: 1\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 1\}\}$
$\{\{e\}, \{sup: 3\}, \{\delta: 0\}, \{ac\ 3, c\ 1, b\ 1\}\}$	$\{\{e\}, \{sup: 3\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 1\}\}$

bcad

Table 4
FT-conditional patterns of itemset {bcadf}.

FT-Conditional Patterns Discovered from FT-FP-tree of itemset {bcad}	FT-Conditional Patterns used for Constructing FT-FP-tree of {bcadf}
$\{\{\emptyset\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 2, a\ 1, c\ 1, b\ 0\}\}$	$\{\{\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, a\ 2, c\ 1, b\ 0\}\}$
$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 2\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, af\ 0, a\ 1, c\ 1, b\ 2\}\}$
$\{\{af\}, \{sup: 1\}, \{\delta: 2\}, \{af\ 0, a\ 1, c\ 1, b\ 0\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, af\ 0, a\ 1, c\ 1, b\ 2\}\}$
$\{\{af\}, \{sup: 1\}, \{\delta: 1\}, \{af\ 0, a\ 1, c\ 1, b\ 1\}\}$	$\{\{af\}, \{sup: 2\}, \{\delta: 2\}, \{f\ 2, af\ 0, a\ 1, c\ 1, b\ 2\}\}$

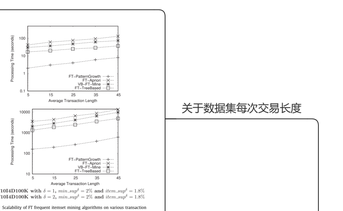
bcadf

什么也不是根据上表，第一，三个容错因子变为了3，第二个d=0<item_sup

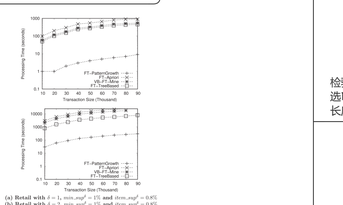
bcadfe

Algorithm 2: Procedure for constructing FT-FP-Tree of itemset.
Data: Conditional patterns of itemset X
Result: Construct FT-FP-Tree of X
Procedure FT-FP-Tree(*C*, *X*)
for *each conditional pattern* c *in* C **do**
  convert c into FT-conditional pattern c^x;
  if c^x *contains at-least* $(|X| - \alpha)$ *items of* X **then**
    $C^x = C^x \cup c^x$;
construct FT-FP-Tree of X using C^x;
if *FT-FP-Tree of* X *!= ∅* **then**
  FT-FP-Growth(*FT-FP-Tree of X, X*);

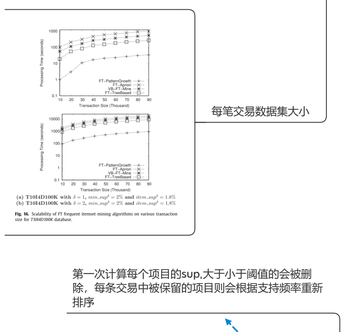
(a) **Retail with** $\delta = 1$, $min\_sup^{\delta} = 1\%$ **and** $item\_sup^{\delta} = 0.8\%$
(b) **Retail with** $\delta = 2$, $min\_sup^{\delta} = 1\%$ **and** $item\_sup^{\delta} = 0.8\%$
Fig. 13. Scalability of FT frequent itemset mining algorithms on various transaction length for *Retail* database.



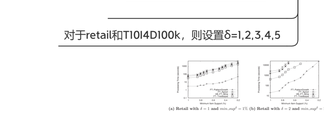
(a) **T10I4D100K with** $\delta = 1$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
(b) **T10I4D100K with** $\delta = 2$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
Fig. 15. Scalability of FT frequent itemset mining algorithms on various transaction length for *T10I4D100K* database.



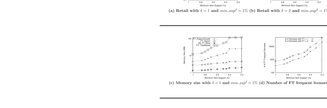
(a) **Retail with** $\delta = 1$, $min\_sup^{\delta} = 1\%$ **and** $item\_sup^{\delta} = 0.8\%$
(b) **Retail with** $\delta = 2$, $min\_sup^{\delta} = 1\%$ **and** $item\_sup^{\delta} = 0.8\%$
Fig. 14. Scalability of FT frequent itemset mining algorithms on various transaction size for *Retail* database.



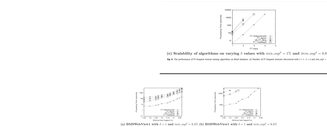
(a) **T10I4D100K with** $\delta = 1$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
(b) **T10I4D100K with** $\delta = 2$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
Fig. 16. Scalability of FT frequent itemset mining algorithms on various transaction size for *T10I4D100K* database.



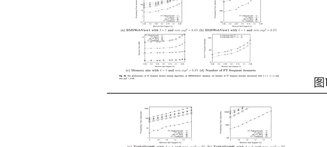
(a) **FoodMart with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.06\%$ (b) **FoodMart with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.06\%$
(c) **Memory size with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.06\%$ (d) **Number of FT frequent itemsets**
Fig. 17. The performance of FT-frequent itemset mining algorithms on FoodMart database. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 0.06%.



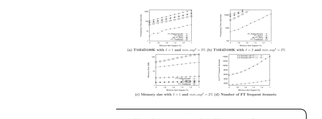
(a) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (b) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$
(c) **Memory size with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (d) **Number of FT frequent itemsets**
Fig. 18. The performance of FT-frequent itemset mining algorithms on different datatsets. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 0.4%.



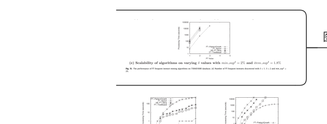
(a) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (b) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$
(c) **Memory size with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (d) **Number of FT frequent itemsets**
Fig. 19. The performance of FT-frequent itemset mining algorithms on different datatsets. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 0.4%.



(a) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (b) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$
(c) **Memory size with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (d) **Number of FT frequent itemsets**
Fig. 20. The performance of FT-frequent itemset mining algorithms on FoodMart database. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 0.4%.



(a) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (b) **T10I4D100K with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$
(c) **Memory size with** $\delta = 1$ **and** $min\_sup^{\delta} = 0.4\%$ (d) **Number of FT frequent itemsets**
Fig. 21. The performance of FT-frequent itemset mining algorithms on FoodMart database. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 0.4%.



(a) **T10I4D100K with** $\delta = 1$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
(b) **T10I4D100K with** $\delta = 2$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
Fig. 22. The performance of FT-frequent itemset mining algorithms on *T10I4D100K* database. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 2%.

(a) **T10I4D100K with** $\delta = 1$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
(b) **T10I4D100K with** $\delta = 2$, $min\_sup^{\delta} = 2\%$ **and** $item\_sup^{\delta} = 1.8\%$
Fig. 23. The performance of FT-frequent itemset mining algorithms on *T10I4D100K* database. (a) Number of FT-frequent itemsets discovered with $\delta = 1$, $\delta = 2$ and min_sup = 2%.