A Fast Algorithm for Mining Share-Frequent Itemsets

Yu-Chiang Li[1], Jieh-Shan Yeh[2], and Chin-Chen Chang[1,3]

[1] Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi 621, Taiwan
{lyc, ccc}@cs.ccu.edu.tw
[2] Department of Computer Science and Information Management,
Providence University, Taichung 433, Taiwan
jsyeh@pu.edu.tw
[3] Department of Computer Science and Information Engineering,
Feng Chia University, Taichung 407, Taiwan

Abstract. Itemset share has been proposed as a measure of the importance of itemsets for mining association rules. The value of the itemset share can provide useful information such as total profit or total customer purchased quantity associated with an itemset in database. The discovery of share-frequent itemsets does not have the downward closure property. Existing algorithms for discovering share-frequent itemsets are inefficient or do not find all share-frequent itemsets. Therefore, this study proposes a novel Fast Share Measure (FSM) algorithm to efficiently generate all share-frequent itemsets. Instead of the downward closure property, FSM satisfies the level closure property. Simulation results reveal that the performance of the FSM algorithm is superior to the ZSP algorithm two to three orders of magnitude between 0.2% and 2% minimum share thresholds.

1 Introduction

Recent developments in information science have a surprisingly rapid accumulation of data. Accordingly, efficiently managing massive bodies of data, rapidly discovering useful information, and making effective decisions based on data are crucial [10]. Newly developed data mining or knowledge discovery techniques have made routine the once impossible task of gathering hidden but potentially useful information from data in a large database or in a data warehouse. Such techniques have been widely applied in numerous areas, and have come to represent an important field of research.

Mining association rules is the main task of various data mining techniques. Agrawal *et al.* first introduced the problem, and developed an Apriori algorithm to generate all significant association rules for the retail organization in the context of bar code data analysis [2, 3]. The mining of association rules includes two-step process (1) finding all *frequent itemsets*, and (2) using these frequent itemsets to derive the association rules. Restated, the corresponding association rules can be straightforwardly derived from the frequent itemsets. Therefore, the first step is critical in mining associations. As the amount of data increase, the design of efficient algorithm becomes increasingly urgent. Various methods have been proposed to speed up the mining process, such as Apriori and subsequent Apriori-like algorithms [2, 3, 7, 8, 16] and pattern-growth methods [1, 11, 12, 15, 17].

Given a database of customer transactions, the goal of data analysis is to discover the buying patterns of customers. Such information hints that how to group the products in store layout or product packets to promote these goods. Each product is regarded as an *item*. An *itemset* is a group of items bought together in a transaction. The *support* value of an itemsets is the typical measure to address the importance of an itemset in a transaction database [2]. An itemset is referred as frequent itemsets when the occurrence count of the itemsets in a database is above a threshold value. However, the support measure only considers the number of transactions in which the itemset was purchased. The exact purchased number of products is not analyzed. Therefore, the support count method does not measure in terms of the profit or cost of an itemset. In 1997, Carter *et al.* presented a share-confidence framework to provide useful information about numerical values associated with transaction items and addressed the problem of mining characterized association rules from itemsets [9]. Recently, several searches about share measure have been proposed to efficiently extract share-frequent (SH-frequent) itemsets with infrequent subsets [4, 5, 6, 13, 14].

An SH-frequent itemset usually includes some infrequent subsets. Consequently, the downward closure property cannot be applied to discover all share-frequent itemsets. Existing algorithms are either inefficient or do not discover complete share-frequent itemsets. Accordingly, this study proposes an efficient Fast Share Measure (FSM) algorithm to discover all SH-Frequent itemset. Instead of the downward closure property, FSM employs the level closure property to rapidly reduce the number of candidate itemsets. The inequality of level closure property guarantees all supersets of the pruned itemsets must be infrequent. This study focuses on the technique to discover all SH-frequent itemsets efficiently.

The rest of this paper is organized as follows. Section 2 introduces the review of support-confidence and share-confidence frameworks. Section 3 explains the level closure property and the proposed fast share measure (FSM) algorithm. FSM applies the level closure property to efficiently prune useless candidates. Section 4 provides experimental results and evaluates the performance of the proposed algorithm. Finally, we conclude in Section 5 with a summary of our work.

2 Reviews of Support and Share Measures

2.1 The Support-Confidence Framework

In 1993, Agrawal *et al.* first presented a model to define the problem of mining association rules [2, 3]. Given a transaction database, the mining of association rules is to discover the important rules that apply to items. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let X be a set of items $X \subseteq I$, which is called an itemset. Let $DB = \{T_1, T_2, \ldots, T_n\}$ be the transaction database, where each transaction $T \in DB$, $T \subseteq I$, $1 \le q \le n$. Each transaction is associated with a unique identifier, called *TID*. An itemset X is contained in T if and only if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \phi$ (For example, $I=\{ABCDE\}$, $X=\{AE\}$, $Y=\{BD\}$). An association rule $X \Rightarrow Y$ has two characteristic values, called *support* and *confidence*. The support of an itemset X, denoted as support(X), is the percentage of transactions in *DB* containing X. If the itemset $X \bigcup Y$ appears in $s\%$ transactions of *DB*, the support of the rule $X \Rightarrow Y$ is $s\%$. This is taken to be the probability, $Pr(X \bigcup Y)$. The rule $X \Rightarrow Y$ has confidence $c\%$ in *DB* if $c\%$ is the percentage of transactions in *DB* containing X that also contain Y. This is taken to the conditional probability, $Pr(Y|X)$. The mathematical expression of confidence is confidence($X \Rightarrow Y$) = support($X \bigcup Y$)/support(X). The problem of mining association rules is to discover all rules whose support and confidence satisfy the user-specified minimum support (*minSup*) and minimum confidence (*minConf*) requirements, respectively. An itemset is called a *frequent* itemset when its support is greater than or equal to the *minSup* threshold.

Given a user-specified *minSup*, Apriori employs the characteristic of the downward closure to discover the frequent itemsets by filtering some infrequent itemsets beforehand. The downward closure property is that any subset of a frequent itemset must be frequent; otherwise the itemset is infrequent. The process makes multiple passes over the database. In each pass, Apriori collects a candidate set of frequent itemsets. The algorithm scans the entire transaction database to count the number of the occurrences of each candidate k-itemset (which is an itemset with k items), and then determines the frequent itemsets. Candidate k-itemsets are established from two arbitrary frequent (k-1)-itemsets, whose first k - 2 items are identical. If $k \geq 3$, Apriori applies the downward closure property to reduce the number of candidates. The process is repeated until no candidate can be generated.

Example 2.1. Consider the example database with eight transactions in Table 1 and the minimum support threshold is 36%. Let C_k be the set of candidate k-itemsets and F_k be the set of frequent k-itemsets. In the first pass, Apriori scans the database to count the support value of each item of C_1. In Figure 1, four 1-itemsets $\{B\}$, $\{C\}$, $\{D\}$ and $\{E\}$ satisfy the minimum support requirement and are added to F_1. Then, each frequent 1-itemset joins with each other to form C_2. In the second pass, Apriori scan the database second time to examine which itemsets of C_2 are frequent. C_3 is generated from F_2 as follows. Figure 1 displays two frequent itemsets of F_2 with the same first item, such as $\{BC\}$ and $\{BD\}$. Then, Apriori checks the 2-itemset $\{CD\}$, which is a subset of $\{BCD\}$ to determine whether $\{CD\}$ is frequent. If $\{CD\}$ is not frequent, then $\{BCD\}$ must be infrequent. Since $\{CD\}$ is in F_2, all the subsets of $\{BCD\}$ are frequent. Hence, $\{BCD\}$ is a candidate 3-itemset. The algorithm stops when no candidate 4-itemset can be generated from F_3. In each pass, Apriori scans the database once. Consequently, Apriori scans the database k times.

2.2 The Share-Confidence Framework

In 1997, Hilderman *et al.* first introduced the share-confidence framework, which is an alternative measure of the importance of itemsets [9]. The local measure value of an itemset X is the total count of each distinct item in the itemset in each transaction, which contains X. The share value is of an itemset X is known as the ratio of the local measure value to the total measure value in *DB*. Each item has a numerical attribute in each transaction. The value of the numerical attribute of an item i_p in a transaction T_q is called the *transaction measure value*, denoted as $tmv(i_p, T_q)$. For Table 1 example, *tmv*(*F*, *T02*) = 4. The type of the numerical attribute can be an integer type, such as the purchased quantity of customers in a transaction, or a real type such as profit margin, unit cost, or total revenue. The other notations and definitions of share measure are described as follows [6].

Table 1. Example of a transaction database with counting

TID	Transaction	Count
T01	{A, B, C, D, E, G, H}	{1, 1, 1, 1, 1, 1, 1}
T02	{F, H}	{4, 3}
T03	{B, C, D}	{4, 3, 3}
T04	{C, E}	{4, 1}
T05	{B, D}	{3, 2}
T06	{B, C, D}	{3, 2, 1}
T07	{B, C, D, E}	{3, 4, 1, 2}
T08	{A, F, G}	{4, 1, 1}

Pass 1

Scan *DB* ==>

C_1	Support
{A}	2
{B}	5
{C}	5
{D}	5
{E}	3
{F}	2
{G}	2
{H}	2

==>

F_1	Support
{B}	5
{C}	5
{D}	5
{E}	3

Pass 2

$F_1 \infty F_1$ ==>

C_2
{BC}
{BD}
{BE}
{CD}
{CE}
{DE}

Scan *DB* ==>

C_2	Support
{BC}	4
{BD}	5
{BE}	2
{CD}	4
{CE}	3
{DE}	2

==>

F_2	Support
{BC}	4
{BD}	5
{CD}	4

Pass 3

$F_2 \infty F_2$ & Prune ==>

C_3
{BCD}

Scan *DB* ==>

C_3	Support
{BCD}	4

==>

F_3	Support
{BCD}	4

Fig. 1. Application of Apriori algorithm

Definition 2.1. Each *k*-itemset $X \subseteq I$ has an associated transaction set $db_X = \{T_q \in DB \mid X \subseteq T_q\}$. In other words, db_X is a set of transactions containing itemset *X*.

Definition 2.2. The *global measure value* $gmv(i_p)$ of an item i_p is the sum of $tmv(i_p, T_q)$, where $T_q \in DB$. In other words, $gmv(i_p) = \sum_{T_q \in DB} tmv(i_p, T_q)$.

Definition 2.3. The *total measure value TMV* of all items is the sum of the *global measure value* of each item i_p. In other words, $TMV = \sum_{p=1}^{m} gmv(i_p)$, where m is the number of all distinct items.

Definition 2.4. The *local measure value* $lmv(i_p, X)$ of an item i_p in an itemset X is the sum of the transaction measure values of the item i_p in all transactions that contain X. In other words, $lmv(i_p, X) = \sum_{T_q \in db_x} tmv(i_p, T_q)$. Similarly, the local measure value $lmv(X)$ of an itemset X is the sum of the local measure values of each item i_p in X. In other words $lmv(X) = \sum_{i_p \in X} lmv(i_p, X)$.

Definition 2.5. The *item share* of an item i_p *in* X, denoted as $SH(i_p, X)$, is the ratio of the local measure value of i_p to the total measure value. In other words, $SH(i_p, X) = \frac{lmv(i_p, X)}{TMV}$. Similarly, the *itemset share* of an itemset X, denoted as $SH(X)$, is the ratio of the local measure value of X to the total measure value. In other words, $SH(X) = \frac{lmv(X)}{TMV}$.

Definition 2.6. A k-itemset X is *share-frequent* (SH-frequent) if $SH(X)$ is greater than a pre-defined minimum threshold (*minShare*) $s\%$.

Example 2.2. Consider the same transaction database as in Table 1 with a minimum share threshold of 36%. As shown in Table1, the column Count lists the corresponding count of each item in a transaction. The global measure value and the item share of each item are listed in Table 2, where $TMV = 56$. The local measure value of $\{B\}$ in the itemset $\{B, C, D\}$ is $lmv(B, \{BCD\}) = 1 + 4 + 3 + 3 = 11$. $SH(\{BCD\}) = lmv(\{BCD\})/TMV = (lmv(B, \{BCD\}) + lmv(C, \{BCD\}) + lmv(D, \{BCD\}))/56$. Then, $SH(\{BCD\}) = (11 + 10 + 6)/56 = 0.482 > 36\%$. Therefore, $\{B, C, D\}$ is SH-frequent. Table 3 enumerates all SH-frequent itemsets.

Table 2. Occurrence count (global measure value) and itemset share of each 1-itemset

Item	A	B	C	D	E	F	G	H	Total
$gmv(i_p)$	5	14	14	8	4	5	2	4	56
$SH(i_p)$	8.9%	25%	25%	14.3%	7.1%	8.9%	3.6%	7.1%	100%

Table 3. All SH-frequent itemsets of the sample database

SH-frequent itemset	BC	BD	BCD
$lmv(X)$	21	22	27
$SH(X)$	37.5%	39.3%	48.2%

3 Fast Share Measure (FSM) Algorithm

The Apriori-like algorithms employ the downward closure property to discover efficiently frequent itemsets based on the support measure. All (k-1)-itemsets of a candidate k-itemset are frequent itemsets, otherwise, the k-itemset can be pruned. Therefore, the characteristic of downward closure can be used to reduce the number of candidates and speed up the process. However, an SH-frequent itemset could include some infrequent subsets. It does not satisfy the downward closure property. Obviously, the exhaustive search method can find all SH-frequent itemsets, but has an exponential running time. Barber and Hamilton presented the ZP (zero pruning) algorithm and the ZSP (zero subset pruning) algorithm to improve the performance [6]. However, the two algorithms only prune the candidate itemsets whose local measure values are exactly zero. There is no efficient algorithm to discover all SH-frequent itemsets up to now. Consequently, this study develops a fast share measure (FSM) algorithm to find all SH-frequent itemsets efficiently.

3.1 Level Closure Property

The notations and definitions of FSM are described as follows.

Definition 3.1. The maximum length of all transactions in DB is denoted as ML. That is, $ML = \max(|T_q| \mid T_q \in DB)$.

Definition 3.2. Let MV be the *maximum transaction measure value* of all items in DB. That is, $MV = \max(tmv(i_p, T_q) \mid i_p \in T_q, \text{ and } T_q \in DB)$.

Definition 3.3. Let X be an itemset, which is a subset of X', the local measure value of X on X', denoted as $lmv(X, X')$, is the sum of the local measure values of each item i_p in X' in DB, where i_p in X. That is, $lmv(X, X') = \sum_{i_p \in X} lmv(i_p, X')$.

If $X = X'$, then $lmv(X, X') = lmv(X)$. The local measure values of itemsets have some characteristics, which are described as follows.

Lemma 3.1. Let X, X' and X'' be itemsets, where $X \subseteq X' \subseteq X''$, then
(1) $lmv(X, X'') \leq lmv(X', X'')$. Especially, when $X' = X''$, $lmv(X, X') \leq lmv(X')$.
(2) $lmv(X, X') \geq lmv(X, X'')$. Especially, when $X = X'$, $lmv(X) \geq lmv(X, X'')$.

Proof.
(1) Since $X \subseteq X'$, for arbitrary item i_p in X, i_p is also in X'. $lmv(X, X'') = \sum_{i_p \in X} lmv(i_p, X'') \leq \sum_{i_p \in X'} lmv(i_p, X'') = lmv(X', X'')$.

(2) Since $X' \subseteq X''$, $db_{X'} \supseteq db_{X''}$. For arbitrary item i_p in X, $lmv(i_p, X') \geq lmv(i_p, X'')$. Therefore, $lmv(X, X') \geq lmv(X, X'')$. Q.E.D

Lemma 3.2. Let X be a *k-itemset*, then $|db_X| \times k \leq lmv(X) \leq |db_X| \times k \times MV$.

Proof. By Definition 2.4, $lmv(X) = \sum_{i_p \in X} lmv(i_p, X) = \sum_{i_p \in X} \sum_{T_q \in db_x} tmv(i_p, T_q)$. For each item i_p and transaction T_q, $1 \leq tmv(i_p, T_q) \leq MV$. Therefore, $|db_X| \times k \leq lmv(X) \leq |db_X| \times k \times MV$. Q.E.D

Theorem 3.1. Given a *minShare* and a *k*-itemset *X*, if $lmv(X) + (lmv(X)/k) \times MV < minShare \times TMV$, all supersets of *X* with length $k + 1$ are infrequent.

Proof. For arbitrary superset *X'* of *X* with length $k + 1$, says $X' = X \cup \{i_p\}$. By Definition 3.3, $lmv(X') = lmv(X', X') = lmv(X, X') + lmv(i_p, X')$. First, by Lemma 3.1, we have $lmv(X, X') \leq lmv(X)$. Second, by Lemma 3.2 on *X'*, $lmv(i_p, X') \leq |db_{X'}| \times MV$. Since $db_{X'}$ is a subset of db_X, $|db_{X'}| \leq |db_X|$. So, $lmv(i_p, X') \leq |db_X| \times MV \leq (lmv(X)/\mathrm{k}) \times MV$, by Lemma 3.1 on *X*. Now, we have $lmv(X') \leq lmv(X) + (lmv(X)/\mathrm{k}) \times MV$. If the inequality $lmv(X) + (lmv(X)/k) \times MV < minShare \times TMV$ holds, $lmv(X') < minShare \times TMV$. That is, $SH(X') = lmv(X')/TMV < minShare$. *X'* is infrequent. Theorem is proofed. Q.E.D

Theorem 3.2. Given a *minShare*, a *k*-itemset *X* and a positive integer *k'*, if $lmv(X) + (lmv(X)/k) \times MV \times k' < minShare \times TMV$, all supersets of *X* with length less than or equal to $k + k'$ are infrequent.

Proof. Let *X'* be an arbitrary superset of *X* with length $k + i$, where $1 \leq i \leq k'$. Let $Y = X' - X$. Clearly, the size of *Y* is *i*. With the same argument in Theorem 3.1, we have

(1) $lmv(X') = lmv(X', X') = lmv(X, X') + lmv(Y, X')$.
(2) $lmv(X, X') \leq lmv(X)$.
(3) $lmv(Y, X') \leq |db_{X'}| \times i \times MV \leq (lmv(X)/k) \times MV \times i \leq (lmv(X)/k) \times MV \times k'$.

So, $lmv(X') \leq lmv(X) + (lmv(X)/k) \times MV \times k'$. If the inequality $lmv(X) + (lmv(X)/k) \times MV \times k' < minShare \times TMV$ holds, $lmv(X') < minShare \times TMV$. That is, $SH(X') = lmv(X') / TMV < minShare$. *X'* is infrequent. Q.E.D

Corollary 3.1. Given a *minShare* and a *k*-itemset *X*, if $lmv(X) + (lmv(X)/k) \times MV \times (ML - k) < minShare \times TMV$, all supersets of *X* are infrequent.

Proof. Since the maximum length of all transactions in *DB* is *ML*, $lmv(X') = 0$ for any superpset *X'* of *X* with length greater than *ML*. Definitely, *X'* is infrequent. For arbitrary superset *X'* of *X* with length less than or equal to *ML*, if the inequality $lmv(X) + (lmv(X)/k) \times MV \times (ML - k) < minShare \times TMV$ holds, by Theorem 3.2, *X'* is infrequent. Corollary is proofed. Q.E.D

Definition 3.4. The characteristic of Theorem 3.1, Theorem 3.2 and Corollary 3.1 is called the *level closure property*. For a given integer *k'*, let *CF* be a critical function, defined as $CF(X) = lmv(X) + (lmv(X)/k) \times MV \times L$, where $L = \min\{ML\text{-}k, k'\}$.

Theorem 3.2 guarantees if $CF(X) < minShare \times TMV$ holds, no superset of *X* with length $\leq k + k'$ is SH-frequent. The level closure property can be applied to prune candidates whose supersets are not SH-frequent with length $\leq k + k'$, but it cannot ensure the SH-frequency of the supersets with length greater than $k + k'$. Accordingly, Corollary 3.1 modifies the level closure property of Theorem 3.2 and assures that all supersets of *X* are not SH-frequent if $CF(X) < minShare \times TMV$.

3.2 Fast Share Measure (FSM) Algorithm

The FSM algorithm is a level-wise and a multiple passes algorithm. In the k-th pass, let C_k be the candidate set, RC_k be the remainder candidates after checking the critical function CF, and F_k be the SH-frequent set. Like Apriori, each single item is a candidate. In the first pass, FSM scans the database to count the local measure value of each item. Each candidate 1-itemset X is pruned when $CF(X) < minShare \times TMV$. In next each pass, FSM joins arbitrary two candidates in $RC_{k\text{-}1}$, whose first k-2 items are identical. The k subsets with length (k - 1) of each k-itemset in C_k are in $RC_{k\text{-}1}$; otherwise the k-itemset can be pruned. After C_k is produced, delete the $RC_{k\text{-}1}$. Next, for each itemset X in C_k, if the itemset share $lmv(X)/TMV$ is higher than $minShare$, X is added to F_k; if $CF(X)$ is greater than $minShare$, the superset of X could be SH-frequesnt, so X is added to $RC_{k.}$ The process is repeated until no candidate can be generated.

The pseudo code of FSM is described as follows.

Algorithm. FSM(*k'*)

Input: (1) *DB*: a transaction database with counts, (2) *minShare*: minimum share threshold, and (3) *k'*: the parameter of critical function (Definition 3.4)

Output: All SH-frequent itemsets

Procedure:

```
1.  $k$:=1; $F_1$:=$\phi$; $C_1$:=$I$;
2.  foreach $T \in DB$ { // scan DB
3.      count the local measure value of each item; }
4.  foreach $i_p \in C_1$   {
5.      if $lmv(i_p) \geq minShare \times TMV$ {
6.          $F_1$:= $F_1+i_p$; }
7.      elseif $CF(i_p) < minShare \times TMV$ {
8.          $C_1$:= $C_1-i_p$;   }
9.  }
10. $RC_1$:=$C_1$;
11. for $k$:=2 to $h$ {
12.     foreach $X_p$, $X_q \in RC_{k-1}$ {
13.         $C_k$ :=Apriori-join($X_p$, $X_q$); }
14.     foreach $T \in DB$ { // scan DB
15.         count each candidate's local measure value; }
16.     foreach $X \in C_k$     {
17.         if $lmv(X) \geq minShare \times TMV$ {
18.             $F_k$:= $F_k+X$; }
19.         elseif $CF(X) < minShare \times TMV$ {
20.             $C_k$:= $C_k-X$;   }   }
21.     $RC_k$:= $C_k$;
22. }
23. return $F$
```

4 Experimental Results

The performance of FSM was compared with that of ZSP using a 1.5GHz Pentium IV PC with 1GB of main memory, running Windows XP Professional. All algorithms were coded using Visual C++ 6.0, and applied to process the synthetic dataset. The whole SH-frequent itemsets were output to main memory to reduce the effect of disk writing.

The IBM synthetic dataset was generated using a synthetic data generator [18]. The VC++ version of the data generator was obtained from [19]. Table 4 lists the parameters of the synthetic data generation program.

Table 4. Parameters

x	Mean size of the transactions
y	Mean size of the maximal potentially frequent itemsets
z	Number of transactions in *DB*
n	Number of items

The notation Tx.Iy.Dz.Nn denotes a dataset with given parameters x, y, z and n. To simulate the characteristic of the count in each item in each transaction, the count of each item in each transaction is randomly generated between 1 to m, with the proportion of 1 equal 50%. The notation of the dataset becomes Tx.Iy.Dz.Nn.Sm.

Figures 2 plots the performance curves associated with the two algorithms applied to T4.I2.D100k.N50.S10. The x-axis represents the several distinct *minShare* thresholds between 0.2% and 2%, and the y-axis represents the running time. Note that Fig. 2 uses a logarithmic scale for y-axis. *FSM*(1), *FSM*(2), *FSM*(3) and *FSM*(*ML*-1) are special cases of the FSM algorithm with different parameter k', respectively. The lower *minShare* threshold results in the longer running time of FSM. In the low *minShare* (0.2%) scenario, *FSM*(*ML*-1) outperforms the ZSP algorithm two orders of magnitude. Contract to the high *minShare* (2%) scenario, *FSM*(*ML* – 1) outperforms ZSP more than three orders of magnitude. *FSM*(*ML* – 1) always outperforms ZSP and discovers all SH-frequent itemsets. In Fig. 2, *FSM*(1) is always the fastest. Although it could loss some SH-frequent itemsets while the parameter k' of FSM is set less than ML – 1, the output set of SH-frequent itemsets is identical with that of ZSP using T4.I2.D100k.N50.S10 with *minShare* = 0.8% as listed in Table 5. The number of C_k, RC_k and F_k and the total running time of ZSP and FSM algorithms are also listed in Table 5. ZSP only prunes the itemsets with $SH(X) = 0$. Therefore, ZSP terminates the process at pass ML. The value of ML of T4.I2.D100k.N50.S10 is 14. Contrast to *FSM*(1), *FSM*(2), *FSM*(3) and *FSM*(*ML*-1), their processes terminate at pass 5, 6, 6 and 6, respectively. In a very low *minShare* (0.005%) scenario, *FSM*(1), *FSM*(2) and *FSM*(3) lose some SH-frequent itemsets using the dataset. In the scenario, *FSM*(*ML* - 1) discovers all SH-frequent itemset.

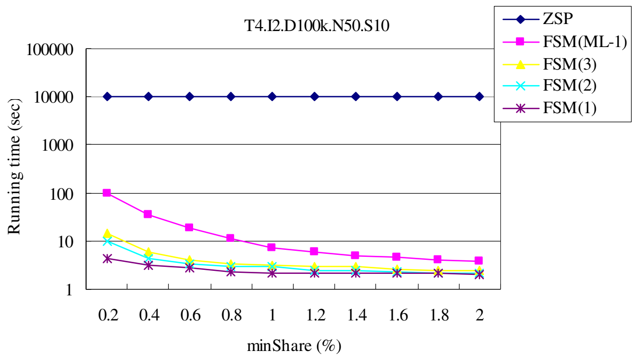

Fig. 2. Comparison of running time using T4.I2.D100k.N50.S10

Table 5. Comparison of the number of candidate set and SH-frequent set (F_k) in each pass using T4.I2.D100k.N50.S10 with *minShare* = 0.8% (*ML*=14)

Method / Pass (k)		ZSP	FSM(1)	FSM(2)	FSM(3)	FSM(*ML*-1)
k=1	C_k	50	50	50	50	50
	RC_k	50	49	49	49	50
	F_k	32	32	32	32	32
k=2	C_k	1225	1176	1176	1176	1225
	RC_k	1219	570	754	845	1085
	F_k	119	119	119	119	119
k=3	C_k	19327	4256	7062	8865	14886
	RC_k	17217	868	1685	2410	5951
	F_k	65	65	65	65	65
k=4	C_k	165077	1725	3233	5568	24243
	RC_k	107397	232	644	1236	6117
	F_k	9	9	9	9	9
k=5	C_k	406374	81	258	717	6309
	RC_k	266776	5	40	109	1199
	F_k	0	0	0	0	0
k=6	C_k	369341	0	1	4	287
	RC_k	310096	0	0	0	37
	F_k	0	0	0	0	0
$k \geq 7$	C_k	365975	0	0	0	0
	RC_k	359471	0	0	0	0
	F_k	0	0	0	0	0
Time(sec)		10349.9	2.30	2.98	3.31	11.24

Figure 3 presents the scalability with the number of transactions of *DB*. The *x*-axis represents the several distinct *DB* sizes between 100k and 1000k, and the *y*-axis represents the running time. Figure 3 uses a logarithmic scale for *y*-axis. Consider *minShare* = 0.8%, the running time linearly increases with the growth of the *DB* size. The running time of ZSP exceeds 10^5 seconds when $|DB| \geq$ 600k.

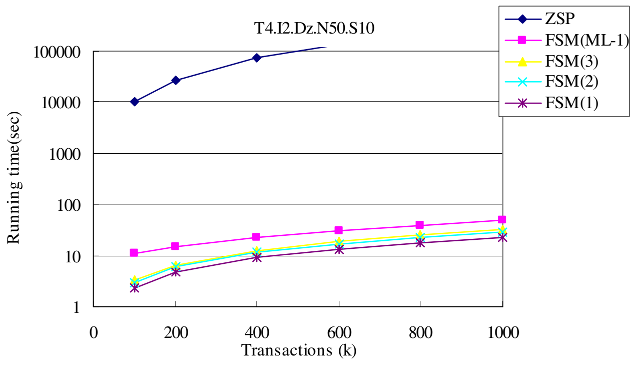

Fig. 3. Scalability with the transaction number of *DB*

5 Conclusions

Data mining techniques have been applied extensively across many areas, and data mining has become an important research field. Mining frequent itemsets in a transaction database plays an important role for mining association rules. Itemset share has been proposed to measure the importance of itemsets for mining association rules. Developing an efficient approach for discovering complete SH-frequent itemsets is very useful in solving numerous mining problems. However, share-frequent itemsets do not satisfy the downward closure property. To solve the problem and develop an efficient method for fast generating all SH-frequent itemsets, this study proposes the level closure property. The inequality of level closure property guarantees all supersets of the pruned itemsets must be infrequent. Consequently, the developed FSM algorithm, which implements the level closure property, can efficiently decrease the number of itemsets to be counted. Experiments indicate that FSM outperforms ZSP several orders of magnitude. In the future, the authors will consider the development of superior algorithms to improve the performance of discovering all SH-frequent itemsets.

References

1. R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad: A Tree Projection Algorithm for Generation of Frequent Itemsets. Journal of Parallel and Distributed Computing **61** (2001) 350-361
2. R. Agrawal, T. Imielinski, and A. Swami: Mining Association Rules between Sets of Items in Large Databases. In: Proc. 1993 ACM SIGMOD Intl. Conf. on Management of Data, Washington, D.C., (1993) 207-216
3. R. Agrawal and R. Srikant: Fast Algorithms for Mining Association Rules. In Proc. 20th Intl. Conf. on Very Large Data Bases, Santiago, Chile (1994) 487-499
4. B. Barber and H. J. Hamilton: Algorithms for Mining Share Frequent Itemsets Containing Infrequent Subsets. In: D. A. Zighed, H. J. Komorowski, J. M. Zytkow (eds.): Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Sciences, Vol. 1910. Springer-Verlag, Berlin Heidelberg New York (2000) 316-324

5. B. Barber and H. J. Hamilton: Parametric Algorithm for Mining Share Frequent Itemsets. Journal of Intelligent Information Systems **16** (2001) 277-293
6. B. Barber and H. J. Hamilton: Extracting Share Frequent Itemsets with Infrequent Subsets. Data Mining and Knowledge Discovery **7** (2003) 153-185
7. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur: Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: Proc. 1997 ACM SIGMOD Intl. Conf. on Management of Data, Tucson, AZ (1997) 255-264
8. F. Berzal, J. C. Cubero, N. Marín, and J. M. Serrano: TBAR: An Efficient Method for Association Rule Mining in Relational Databases. Data & Knowledge Engineering **37** (2001) 47-64
9. C. L. Carter, H. J. Hamilton, and N. Cercone: Share Based Measures for Itemsets. In: H. J. Komorowski, J. M. Zytkow (eds.): Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science, Vol. 1263. Springer-Verlag, Berlin Heidelberg New York (1997) 14-24
10. M. S. Chen, J. Han, and P. S. Yu: Data Mining: An Overview from a Database Perspective. IEEE Trans. Knowledge Data Engineering **8** (1996) 866-883
11. G. Grahne and J. Zhu: Efficient using Prefix-Tree in Mining Frequent Itemsets. In: Proc. IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Melbourne, FL (2003)
12. J. Han, J. Pei, Y. Yin, and R. Mao: Mining Frequent Patterns without Candidate Generation: A Frequent Pattern Tree Approach. Data Mining and Knowledge Discovery **8** (2004):53-87
13. R. J. Hilderman: Predicting Itemset Sales Profiles with Share Measures and Repeat-Buying Theory. In: J. Liu, Y. M. Cheung, H. Yin (eds.): Intelligent Data Engineering and Automated Learning. Lecture Notes in Computer Science, Vol. 2690. Springer-Verlag, Berlin Heidelberg New York (2003) 789-795
14. R. J. Hilderman, C. L. Carter, H. J. Hamilton, and N. Cercone: Mining Association Rules from Market Basket Data using Share Measures and Characterized Itemsets," Intl. Journal of Artificial Intelligence Tools **7** (1998) 189-220
15. J. Liu, Y. Pan, K. Wang, and J. Han: Mining Frequent Item Sets by Opportunistic Projection. In: Proc. 8th ACM-SIGKDD Intl. Conf. on Knowledge Discovery and Dada Mining, Alberta, Canada (2002) 229-238
16. J. S. Park, M. S. Chen, and P. S. Yu: An Effective Hash-Based Algorithm for Mining Association Rules. In: Proc. 1995 ACM-SIGMOD Intl. Conf. on Management of Data, San Jose, CA (1995) 175-186
17. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang: H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. In: Proc. 2001 IEEE Intl. Conf. on Data Mining, San Jose, CA (2001) 441-448
18. http://alme1.almaden.ibm.com/software/quest/Resources/datasets/syndata.html
19. http://www.cse.cuhk.edu.hk/~kdd/data/IBM_VC++.zip