# ภาคผนวก

1. R. Amornchewin and W. Kreesuradej, "Incremental association rule mining using promising frequent itemset algorithm", in Proceeding 6th International Conference on Information, Communications and Signal Processing, Dec. 10-13 2007, pp.1-5.
2. R. Amornchewin and W. Kreesuradej, "Probability-based Incremental association rule discovery algorithm", (will be published) in Proceeding of The 2008 International Symposium on Computer Science and its Applications, Oct. 13-15 2008.

# Probability-based incremental association rule discovery algorithm

Ratchadaporn Amornchewin
*Faculty of Information Technology*
*King Mongkut's Institute of Technology*
*Ladkrabang*
*Bangkok, 10520 Thailand*
ramornchewin@yahoo.com

Worapoj Kreesuradej
*Faculty of Information Technology*
*King Mongkut's Institute of Technology*
*Ladkrabang*
*Bangkok, 10520 Thailand*
worapoj@it.kmitl.ac.th

## Abstract

*In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the maintenance of association rules for dynamic databases is an important problem. In this paper, probability-based incremental association rule discovery algorithm is proposed to deal with this problem. The proposed algorithm uses the principle of Bernoulli trials to find expected frequent itemsets. This can reduce a number of times to scan an original database. This paper also proposes a new updating and pruning algorithm that guarantee to find all frequent itemsets of an updated database efficiently. The simulation results show that the proposed algorithm has a good performance.*

## 1. Introduction

Data mining is one of the processes of Knowledge Discovery in Database (KDD) that is used for extracting information or pattern from large database. One major area of data mining is association rule mining [1] that discovers hidden knowledge in database. The association rule mining problem is to find out all the rules in the form of $X \Rightarrow Y$, where X and $Y \subset I$ are sets of items, called itemsets. The association rule discovery algorithm is usually decomposed into 2 major steps. The first step is find out all large itemsets that have support value exceed a minimum support threshold and the second steps is find out all the association rules that have value exceed a minimum confidence threshold.

However, a database is dynamic when new transactions are inserted into the database. This may introduce new association rules and some existing association rules would become invalid. As a brute force approach, apriori algorithm may be applied to mining a whole dynamic database when the database has been changed. However, this approach is very costly even if small amount of new transactions is inserted into a database. Thus, the association rule mining for a dynamic database is an important problem. Several research works [6, 7, 8, 9] have proposed several incremental algorithms to deal with this problem. Review of related works will be introduced in section 2.

In this paper, a new incremental algorithm, called probability-based incremental association rule discovery, is introduced. The goal of this work is to solve the updating problem of association rules after a number of new records have been added to a database. Based on probabilistic approach, our incremental algorithm predicts infrequent itemsets that have capable of being frequent itemsets after a number of new records have been added to a database. That infrequent itemsets is called expected frequent itemsets. Our algorithm can reduce a number of times to scan an original database. As a result, the algorithm has execution time faster than that of previous methods.

## 2. Previous work

An influential algorithm for association rule mining is Apriori [2]. Apriori computes frequent itemsets in a large database through several iterations based on a prior knowledge. Each iteration has 2 steps which are a joining step and a pruning step. For a frequent itemset, its support must be higher than a user-specified minimum support threshold. The association rule can be discoverd based on frequent itemsets.

For dynamic databases, several incremental updating techniques have been developed for mining association rules. One of the previous works for incremental association rule mining is FUP algorithm that was presented by Cheung et al [3]. The major idea of FUP is re-using frequent itemsets of previous mining to update with frequent itemsets of an incremental database based on the concepts of Apriori.

Furthermore, negative border approach is presented by Toivonen [5], Thomas et al [6] and Feldman et al [8]. The negative border approach is an incremental mining algorithm based on FUP. The border itemset is not a frequent itemset but all its proper subsets are frequent itemsets. The approach need to keep a large number of border itemsets in order to reduce scanning times of an original database.

To reduce memory space, Hong et al [9] and Amornchewin et al[10] propose a new approach. The approach maintains both frequent itemsets and expected frequent itemsets. An expected frequent itemset is not a frequent itemset but is expected to become a frequent itemset when a new database is added to an original database. In order to guarantee that all frequent itemsets can be found when a new database is added to an original database, the approach can only allow very small size of an incremental database to insert into an original database.

Similarly, the proposed method in this paper also keeps not only frequent itemsets but also expected frequent itemsets from an original database. Unlike the previous works, this paper proposes a new technique for predicting expected frequent itemsets. Here, the principle of Bernoulli trials is used to predict the expected frequent itemsets. The expected frequent itemsets obtained from the proposed technique has lesser members than the border itemsets and the expected frequent itemsets obtained from the previous technique. This work also proposes a new updating algorithm that guarantee to find all frequent itemsets of a dynamic database efficiently.

## 3. Probability-based Incremental Association Rule Discovery Algorithm

When a dynamic database is inserted new transactions, not only some existing association rules may be invalidated but also some new association rules may be discovered. This is the case because frequent itemsets can be changed after inserting new transactions into a dynamic database. Therefore, an association rule discovery algorithm for a dynamic database has to maintain frequent itemsets when new transactions are inserted into the dynamic database.

The task of an association rule discovery algorithm for a dynamic database can be divided into three tasks. The first task is to update support count of existing frequent itemsets. The second task is to prune existing frequent itemsets that have support count below a minimum support threshold after updating the database. The third task is to discover new frequent itemsets that have support count equal or above a minimum support threshold after updating the database.

In this section, we describe our algorithm into 2 subsections. Firstly, probability-based expected frequent itemsets is presented. Secondly, updating frequent and expected frequent itemsets is introduced.

### 3.1 Probability-Based Expected Frequent Itemsets

For our algorithm, an original database, which is a database before being inserted new transactions, is firstly mined to find all frequent itemsets that satisfy a minimum support count, denoted $k_{original}$. Furthermore, the proposed algorithm also predicts and keeps expected frequent itemsets that may become frequent itemsets if new transactions are inserted into the original database.

Our assumption for the new algorithm is that the statistics of new transactions slightly change from original transactions and the maximum number of new transactions that be allowed to insert into an original database is available. According to the first assumption, the statistics of old transactions, obtained from previous mining, can be utilized for approximating that of new transactions. Therefore, the new algorithm uses support count of itemsets obtained from previous mining to approximate the probability of infrequent itemsets in an original database that may be capable of being frequent itemsets when new transactions are inserted into the original database.

Here, the process of inserting m transactions into an original database of n transaction can be considered as (m+n) Bernoulli trials, which are (m+n) sequence of identical trials. Each itemset has its probability of appearing in a transaction, denoted by p, i.e., the probability of success. According to the principle of Bernoulli trials, the probability of the number of an itemset to appearing in (n+m) transactions, denoted by P(x), can be found by the following equation:

$$P(x)=\binom{n+m}{x}\cdot p^{x}\cdot(1-p)^{n+m-x} ,$$

where p is the probability of an itemset appearing in a transaction, m is a number of new transactions, and n is a number of transactions of an original database.

Thus, if k is a minimum support count after inserting new transactions into an original database, the probability of an itemset to be a frequent intemset in an updated database can be obtained as the following equations:

$$P\left(x\geq k\right)_{item}=1-P(x<k)_{item} \qquad (1)$$

Here, an expected frequent itemset is an itemset that is not a frequent itemset but has its probability to oe a frequent intemset greater than $\text{Prob}_{pl}$. $\text{Prob}_{pl}$ is a threshold constant specified by users. $\text{Prob}_{pl}$ indicates the minimum confidence level that a promising frequent itemset will be a frequent itemset after inserting new transaction into an original database.

### 3.2. Updating frequent and expected frequent itemsets

When new transactions are added to an original database, an old frequent k-itemset could become an infrequent k-itemset and an old expected frequent k-itemset could become a frequent k-itemset. This introduces new association rules and some existing association rules would become invalid. To deal with this problem, all k-itemsets must be updated when new transactions are added to an original database. The notation used in this section is given in Table1.

**Table 1.** The notation for Updating frequent and expected frequent itemsets algorithm

| | |
|---|---|
| DB | Original database |
| db | Incremental Database |
| UP | Updated database |
| k | Number of itemset |
| $\sigma$ | Minimum support |
| $\rho$ | Minimum Expected Frequent |
| $C_k$ | Candidate k-itemset |
| $F_k$ | Frequent k-itemset |
| $EF_k$ | Expected Frequent k-itemset |

Here, a new updating frequent and expected frequent itemsets algorithm shown in Figure 1 is proposed in this paper. The algorithm consists of three phases. The first phase is updating 1- frequent and expected frequent itemsets, i.e. line1-3. The second phase is repeatedly updating the other frequent and expected frequent itemsets by using only an incremental database, i.e. line 6-11. The third phase is scanning an original database, i.e. line 13-22.

The First phase is updating 1- frequent and expected frequent itemsets. According to our propose, the 1-candidate itemsets of an updated database, i.e. $C_1^{UP}$, can be found by combining the 1-candidate itemsets of an original database, i.e. $C_1^{DB}$, with the 1-candidate itemsets of an incremental database, i.e. $C_1^{db}$. Then, the support count of $C_1^{UP}$ can be updated by scanning only an incremental database. Then, the result of this phase is consist of 1-frequent and expected 1-itemsets of an updated database.

The second phase has 2 major steps which are a generating $k$- incremental candidate itemsets step and an updating support count of k- incremental frequent and k- incremental expected frequent itemsets step for $k$ greater than or equal to 2. For $k$=2, the 2- incremental candidate itemsets are easily obtained by joining $F_1^{UP}$ with $F_1^{UP}$. For $k$>2, the algorithm is firstly find $k$-candidate itemsets of an incremental database, i.e. $C_k^{db}$, by joining $F_{k-1}^{db}$ with $F_{k-1}^{db}$. Similar to Apriori algorithm, the $k$- candidate itemsets of an incremental database can be the updated frequent itemsets, i.e. $F_k^{UP}$, only if the subsets of the $k$- candidate itemsets of an incremental database must be in the ($k$-1 )- updated frequent. Thus, the $k$- incremental candidate itemsets, will keep only the k- candidate itemsets of an incremental database whose subsets of the $k$- candidate itemsets are in the ($k$-1) - updated frequent intemsets. This can prune the $k$- candidate itemsets of an incremental database that can't be the $k$- updated frequent intemsets.

**Algorithm1 : Updating frequent and expected frequent itemsets Algorithm**

```
Input : $DB, db, k, \sigma^{UP}, \rho^{UP}, \rho^{DB}, C_1^{DB}, F_1^{DB}, EF_1^{DB}$ and their count
Output : $F_k^{UP}, EF_k^{UP}$
1.  $k = 1$
2.  if $k = 1$
3.      Update 1-itemset
4.      $k = k + 1$
5.  else
6.      for $(k = 2; F_k^{UP} \neq \phi; k++)$ do
7.          Generate Candidate Itemset
8.          Update k-itemset(return m, Temp_scanDB)
9.          // m is the maximum itemset of Temp_scanDB
10.         $k = k + 1$
11.     end do
12. end if
13. $k = 2$
14. while ($Temp\_scanDB_k \neq \phi$ and $(k \leq m)$ do
15.         Scan Original Database for all $X \in Temp\_scanDB_k$
16.         for all $X \in Temp\_scanDB_k$ do
17.             $c(X,UP) = c(X,DB) + c(X,db)$
18.         end do
19.         $F_k^{UP} = F_k^{UP} \cup \{X \mid X \in Temp\_scanDB_k$ and $c(X,UP) \geq \sigma^{UP}\}$
20.         $EF_k^{UP} = EF_k^{UP} \cup \{X \mid X \in Temp\_scanDB_k$ and $\rho^{UP} \leq c(X,UP) < \sigma^{UP}\}$
21.         $k = k + 1$
22. end do
23. clear Temp_scanDB
```

**Figure 1.** Updating frequent and expected frequent itmesets Algorithm

When any $k$-itemsets are not in the union set of the original $k$-frequent and the original $k$- expected frequent itemsets, but is in the k- incremental candidate itemsets, their support counts need to be specially updated. This is the case because their support counts obtained from an original database are not available. Here, their support counts in an original database are assumed to be equal to the sum of $\rho^{DB}$ - 1 and their support counts from an incremental database. If any k-itemsets have support counts below updated min support count, i.e. $\sigma^{UP}$, the $k$-itemsets can't be the $k$-updated frequent itemsets. On the other hand, if any $k$-itemsets have support counts above or equal to an updated min support count, the $k$-itemsets are likely to be the $k$-updated frequent itemsets. Thus, the $k$-itemsets, which have support counts above or equal to an updated min support count, are set aside for finding their true support counts from an original database.

At the third phase, an original database is scanned to find true support counts for the $k$-itemsets that are likely to be the $k$-updated frequent itemsets. The support counts of the likely $k$-updated frequent itemsets are found and updated by scanning an original database. Then, all $k$-updated frequent itemsets and $k$-updated expected frequent itemsets are found.

## 4. Experiments

To evaluate the performance of probability-based incremental association rules discovery algorithm, the algorithm is implemented and tested on a PC with a 2.8 GHz Pentium 4 processor, and 1 GB main memory. The experiments are conducted on a synthetic dataset, called T10I4D10K. The technique for generating the dataset is proposed by Agrawal and etc. [1]. The synthetic dataset comprises 110,000 transactions over 100 unique items, each transaction has 10 items on average and the maximal size itemset is 4.

Firstly, the proposed algorithm with $Prob_{pl} = 0.06$ used to find association rules from an original database of 10,000 transactions. Then, the same sizes of incremental databases, i.e. 10% of the original database, are added to the original database for 100 trials. For comparison purpose, FUP and Borders algorithm are also used to find association rules from the same original database and the same incremental databases. Figure 2 and Table 2 show the average of execution time for FUP, Borders and our approach. The results also show that the proposed algorithm has much better running time than that of FUP.

**Table 2.** Average of Execution time

| | Average of Execution time for 100 trials | | |
|---|---|---|---|
| Min_sup (%) | FUP | Borders | Probability-Based |
| 3% | 3195.6939 | 2660.9297 | 2354.24533 |
| 4% | 2274.4477 | 2087.8636 | 1931.19147 |

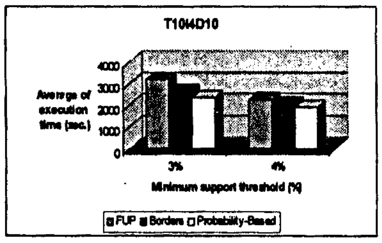

**Figure 2.** The execution time of FUP, Borders and the proposed algorithm

## 5. Conclusion

We have proposed probability-based incremental association rule discovery algorithm. Assuming that the two thresholds, minimum support and confidence, do not change, the algorithm can guarantee to discover all frequent itemsets. From the experiment, our algorithm has better running time than that of FUP and Borders algorithm. In the future, further researches and experiments on the proposed algorithm will be presented.

## 6. References

[1] R. Agrawal., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", In Proceeding of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In Proceedings of 20 th Intl Conf. on Very Large Databases (VLDB'94), pages487-499, Santiago, Chile, September 1994, pp. 478 -499.

[3] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique", In 12th IEEE International Conference on Data Engineering, February 1996, pp. 106-114.

[4] D. W. Cheung, S.D. Lee, B. Kao, "A General incremental technique for maintaining discovered association rules" , In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997, pp. 185-194.

[5] H. Toivonen. "Sampling Large Databases for Association Rules", Proceeding of the 22th International conference on Very Large Data Bases, September 1996,pp. 134-145.

[6] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases" , In Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97), New Port Beach, California, August 1997, pp. 263-266.

[7] C.C. Chang, Y.C. Li and J.S. Lee, "An efficient algorithm for incremental mining of association rules", Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA'05), IEEE, 2005.

[8] R. Feldman, Y. Aumann, and O. Lipshtat. "Borders: An efficient algorithm for association generation in dynamic databases",Journal,Intelligent Information System, 1990, pp. 61-73.

[9] T.P. Hong C.Y. Wang and Y.H. Tao, "A new incremental data mining algorithm using pre-large itemsets", Journal, Intelligent Data Analysis, Vol. 5, No.2, pp. 111-129, 2001.

[10] R. Amornchewin and W. Kreesuradej, "Incremental association rule mining using promising frequent itemset algorithm", In Proceeding 6th International Conference on Information, Communications and Signal Processing, Dec. 10-13 2007, pp.1-5.

# Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm

Ratchadaporn Amornchewin
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520 Thailand
ramornchewin@yahoo.com

Worapoj Kreesuradej
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520 Thailand
worapoj@it.kmitl.ac.th

***Abstract*— Association rule discovery is an important area of data mining. In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the maintenance of association rules for dynamic databases is an important problem. In this paper, promising frequent itemset algorithm, which is an incremental algorithm, is proposed to deal with this problem. The proposed algorithm uses maximum support count of 1-itemsets obtained from previous mining to estimate infrequent itemsets, called promising itemsets, of an original database that will capable of being frequent itemsets when new transactions are inserted into the original database. Thus, the algorithm can reduce a number of times to scan the original database. As a result, the algorithm has execution time faster than that of previous methods. This paper also conducts simulation experiments to show the performance of the proposed algorithm. The simulation results show that the proposed algorithm has a good performance.**

***Keywords*—association rule, maintain association rule, incremental associatin rule**

## I. Introduction

Data mining is one of the processes of Knowledge Discovery in Database (KDD) that is used for extracting information or pattern from large database. One major application area of data mining is association rule mining [1] that discovers hidden knowledge in database. The association rule mining problem is to find out all the rules in the form of X => Y, where X and Y $\subset$ I are sets of items, called itemsets. The association rule discovery algorithm is usually decomposed into 2 major steps. The first step is find out all large itemsets that have support value exceed a minimum support threshold and the second steps is find out all the association rules that have value exceed a minimum confidence threshold.

However, a database is dynamic when new transactions are inserted into the database. This may introduce new association rules and some existing association rules would become invalid. As a brute force approach, apriori may be reapplied to mining the whole dynamic database when the database has been changed. However, this approach is very costly even if small amount of new transactions is inserted into a database. Thus, the association rule mining for a dynamic database is an important problem. Several research works [7, 8, 9, 10, 11] have proposed several incremental algorithms to deal with this problem. Review of related works will be introduced in section 2.

In this paper, a new incremental algorithm, called promising frequent itemset algorithm, is introduced. The goal of this work is to solve the efficient updating problem of association rules after a nontrivial number of new records have been added to a database. Our approach introduces a promising frequent itemset for an infrequent itemset that has capable of being a frequent itemset after a number of new records have been added to a database. This can reduce a number of times to scan an original database. As a result, the algorithm has execution time faster than that of previous methods.

The remaining of this paper is organized as follows. We brief review of related works in Section 2. The Promising large itemset algorithm is described in Section 3. We evaluate the performance in Section 4. Finally, we conclude the work of this paper in section 5.

## II. Related Work

An influential algorithm for association rule mining is Apriori [2]. Apriori computes frequent itemsets in the large database through several iterations based on a prior knowledge. Each iteration has 2 steps. For each iteration with 2 steps, processes are join and prune step. For an frequent itemset, its support must be higher than a user-specified minimum support threshold. The association rule can be discoverd based on frequent itemsets that must be higher than user-specified minimum confidence.

For dynamic databases, several incremental updating techniques have been developed for mining association rules. One of the previous work for incremental association rule mining is FUP algorithm that was presented by Cheung et al [3]. FUP algorithm is the first incremental updating technique for maintenance association rules when new data are inserted to database. Based on the framework of Apriori algorithm, FUP computes frequent itemsets using large itemsets found at the previous iteration. The major idea of FUP is re-use frequent itemsets of previous mining to update with incremental database. The key performance of FUP is pruning technique to reduce the number of candidate set in update process. The extension algorithm of FUP is FUP2 [4] that is proposed to handle all update cases when database are added to, deleted from a database.

Ayan et al [5] present an algorithm called UWEP (Update With Early Pruning). UWEP follows the approaches of FUP and partition algorithm. It employs a dynamic look-ahead strategy in updating existing large itemsets by detecting and pruning superset of large itemsets in an original database that will no longer remain large in updated database. UWEP scans at most once in both original database and incremental database. UWEP generates smaller candidate set from the set of itemsets that are large both an original and incremental database.

Furthermore, negative border algorithm [6], an incremental mining algorithm based on FUP, reduces to scan original database and keeps track of large itemsets and negative border when transaction is added to or delete from database. Negative border consists of all itemsets that are candidates of the level-wise method. An itemset is in the negative border did not have enough support but all its subsets are frequent. If the negative border of large itemsets expands, this algorithm is required to full scan a whole database. This is the case because negative border algorithm does not cover all large itemsets in updated database.

## III. PROMISING FREQUENT ITEMSET ALGORITHM

In an observation the itemset will be frequent itemset in updated database if it is member of large itemset in original database or incremental database. The main problem of incremental update is changing of frequent itemset that cause to re-execute from original database again.

In this paper we present the new idea to avoid scanning the original database. Then we compute not only frequent itemset but also compute itemset that may be potentially large in an incremental database called "Promising frequent Itemset".

An algorithm find all possible k-itemset of promising frequent itemset in original database. If member of frequent for each iteration is more than or equal to k-itemset. This idea is guarantee that promising frequent itemset algorithm are cover all frequent itemset that occur in updated database.Thus, updating the new transactions are quickly because it can use the information from the existing original database.

In this section we describe our algorithm into 2 subsection. In our approach, an original database is firstly mined and all frequent itemsets and promising frequent itemset. Secondly each incremental dataset in mined and updated to frequent and promising frequent itemset. The result of updating, some infrequent itemsets or new itemsets may be changed into frequent itemset.

### A. *Original database Discovery*

A dynamic database may allow insert new transactions. This may not only invalidate existing association rules but also activate new association rules. Maintaining association rules for a dynamic database is an important issue. Thus, this paper proposes a new algorithm to deal with such updating situation. Our assumption for the new algorithm is that the statistics of new transactions slowly change from original transactions. According to the assumption, the statisti :s of old transactions, obtained from previous mining, can be utilized for approximating that of new transactions. Therefore, Support count of itemsets obtained from previous mining may slightly different from support count c f itemsets after inserting new transactions into an original database that contains old transactions. The new algorithm uses maximum support count of 1-itemsets obtained from previous mining to estimate infrequent itemsets of an original database that will capable of being frequent itemsets when new transactions are inserted into the original database. With maximum support count and maximum size of new transactions that allow insert into an original database, support count for infrequent itemsets that will be qualified for frequent itemsets, i.e. min_pl, is shown in equation 1:

$$\mathrm{min\_sup}_{DB} - \left( \frac{\mathrm{max}\mathit{supp}}{\mathit{total\ size}}) \times \mathit{inc\_size} \right) \leq \mathrm{min\_}PL < \mathrm{min\_sup}_{DB} \quad (1)$$

where $\mathrm{min\_sup}_{(DB)}$ is minimum support count for an original database, maxsupp is maximum support count of itemsets, current size is a number of transaction of an original database and inc_size is a maximum number of new transactions.

Here, a promising frequent itemsets is defined as following definition:

Definition A promising frequent itemset is an infrequent itemset that satisfies the equation 1.

As an example, an original database shown in figure 1. has 10 transactions, i.e. |DB|=10. Then, three new transactions is inserted into the original database, i.e. |db| =3. Here, minimum support count for mining association rules is set to 4 (40 percent). From figure 1, maximum support count of 1-itemsets of the original database is 7. min_PL is computed as the follows:

$$\mathrm{min\_}\ PL = 4 - \left( \frac{7}{10} \times 3 \right) = 2 \quad (2)$$

According to equation 2, if any itemset has support count at least 2 but less than 4, then it will be promising frequent itemsets. Thus, the frequent 1- itemset is {A, B, C} and the promising frequent 1- itemset is {D, E}.

In this paper, apriori algorithm is applied to find all possible frequent k- itemsets and promising frequent k-itemsets. Apriori scans all transactions of an original database for each iteration with 2 steps processes are join and prune step. Unlike typical apriori algorithm, items in both frequent k- itemsets and promising frequent k-itemsets can be joined together in the join step. For a frequent item, its support count must be higher than a user-specified minimum support count threshold and for a promising frequent item, its support count must be higher than min_PL but less than the user-specified minimum support count. As examples, figure 2. and 3. show the promising frequent and frequent 2- itemsets and the promising frequent and frequent 3- itemsets respectively.

| TID | List of item |
|---|---|
| 1 | A, B, E |
| 2 | B, D |
| 3 | B,C |
| 4 | A, B,D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, E |
| 10 | A,C |

| Itemset | support |
|---|---|
| A | 7 |
| B | 7 |
| C | 6 |
| D | 2 |
| E | 3 |

Figure 1. Transaction data and candidate 1-itemsets

| C2 | support |
|---|---|
| AB | 4 |
| AC | 4 |
| AD | 1 |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |
| CD | 0 |
| CE | 1 |
| DE | 0 |

| L2 | Support |
|---|---|
| AB | 4 |
| AC | 4 |
| PL2 | support |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |

Figure 2. candidate , frequent and promise frequent 2-itemset

| C3 | Support |
|---|---|
| ABC | 1 |
| ABE | 3 |
| ACE | 1 |
| BCD | 0 |
| BCE | 0 |
| BDE | 0 |

| PL3 | support |
|---|---|
| ABE | 3 |

Figure 3. candidate, frequent and promise frequent 3- itemset

## *B. Updating frequent and promising frequent itemsets*

When new transactions are added to an original database, an old frequent k-item could become an infrequent k-item and an old promising frequent k-item could become a frequent k-item. This introduces new association rules and some existing association rules would become invalid. To deal with this problem, all k-items must be updated when new transactions are added to an original database. In this section, we explain how to update all old items.

The size of an updated database increases when new transactions are inserted into an original database. Thus, min_PL must be recalculated in order to associate with the new size of an updated database. min_PL $_{(update)}$ is computed as the follows:

$$\min\_PL_{DB\cup db} = \min\_\sup_{DB\cup db} - \left(\frac{\max supp}{total\ size} \times inc\_size\right) \quad (3)$$

Then, If any k-item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this itemset is moved to a frequent k-item of an updated database. In the other case, if any k-item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$,, this k-item is moved to a promise frequent itemset of an upated database. The following algorithms are developed to update frequent and promising frequent k-tems of an updated database.

The first algorithm, shown in figure 4, updates a frequent and a promising frequent 1-itemset. After obtaining support count of candidate 1-itemsets of an incremental database, the support count of the candidate 1-items is summed to that of the 1-items of an original database. Then, If any item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this item is moved to a frequent 1-itemset of an updated database. In the other case, if any item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$ , this item is moved to a promise frequent 1-itemset of an updated database. In addition, if any item is a new frequent item or a new promising frequent item, this item will be added to Temp1 set. Then, Temp1 is joined and pruned with both a promise frequent k-itemset and a frequent k-itemset to obtain Temp_newCk. The algorithm for joining and pruning items is shown in figure 5.

The k-itemsets of the Temp_newCk are scanned in an incremental database. A k-itemset of Temp_newCk can become a frequent itemset in an updated database only if the k-itemset of the Temp_newCk is a frequent itemset in an incremental database. Thus, if a itemset of the Temp_newCk has support count greater than or equal to min_sup(db), the item is moved to an estimated frequent k-itemset. Similarly, a k-itemset of Temp_newCk can become a promising frequent itemset in an updated database only if the k-itemset of the Temp_newCk is a frequent itemset in an incremental database. Thus, if a k-itemset of Temp_newCk has support count less than min_sup(db) but greater or equal to min_PL(update) or min_PL(DB), the k-itemset is moved to an estimated promising frequent k-itemset. Then, both the estimated promise frequent k-itemsets and the estimated frequent k-itemsets are added to Temp_scanDB. Figure 6 shows updating k-itemset algorithm for k>=2.

As the last phase for the incremental algorithm, the k-items of Temp_scanDB is scanned in an original database to update their support count. Like previous cases, If any k-item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this k-item is moved to a frequent k-itemset and if any k-item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$ , this k-item is moved to a promise frequent k-itemset. The algorithm for finding support count of k>=2 itemsets shows in figure 7.

**Algorithm 1 Updating frequent and promising frequent 1-itemset**

**Input :**
(1) $L^1_{DB}$: the set of all frequent 1-itemset in original database,
(2) $PL^1_{DB}$ : the set of all promising frequent 1-itemset original database,
(3) $C^1_{DB}$ : candidate 1-itemset of original database,
(4) $C^1_{db}$: candidate 1-itemset of incremental database.

**Output :**
(1) $L^1_{(DB\cup db)}$: frequent itemset in updated database,
(2) $PL^1_{(DB\cup db)}$ : promising frequent itemset in updated database,
(3) new frequent itemsets : new frequent itemset in updated database ,
(4) new promising frequent itemsets : new promising frequent itemset in updated database
(5) Temp_newCk : new candidate 2-itemset in updated database

```
$C^1_{db}$ = all 1-itemsets in db with support >0
k=1
While $C^1_{db}$ > 0 do
For each X $\in C^1_{DB}$ do
    X.support$_{(DB\cup db)}$ = X.support$_{DB}$ + X.support$_{db}$
    If (X $\notin L^1_{DB}$ or X $\notin PL^1_{DB}$) and
        (X.support$_{(DB\cup db)}$) >= min_sup$_{(DB\cup db)}$ Then
        Add X to $L^1_{(DB\cup db)}$
        Add X to templ
    For each X $\in L^1_{DB}$ do
        If X.support$_{(DB\cup db)}$>= min_sup$_{(DB\cup db)}$ Then
            Add X to $L^1_{(DB\cup db)}$
        Else
            If X.support$_{(DB\cup db)}$>= min_PL$_{(DB\cup db)}$ Then
                Add X to $PL^1_{(DB\cup db)}$
    For each X $\in PL^1_{DB}$ do
        If X.support$_{(DB\cup db)}$>= min_sup$_{(DB\cup db)}$ Then
            Add X to $L^1_{(DB\cup db)}$
            Add X to templ
        Else
            If X.support$_{(DB\cup db)}$>= min_PL$_{(DB\cup db)}$ Then
                Add X to $PL^1_{(DB\cup db)}$
For each X $\notin C^1_{DB}$ do
    Add X to $C^1_{(DB\cup db)}$
    If X.support$_{db}$ >= minsup$_{(DB\cup db)}$ (new item in db) Then
        Add X to $L^1_{(DB\cup db)}$
        Add X to templ
    Else
        If X.support$_{(db)}$>= min_PL$_{(DB\cup db)}$ Then
            Add X to $PL^1_{(DB\cup db)}$
            Add X to templ
 If templ $\neq$ {} Then
    Y $\in$ templ
    $C^2_{(DB\cup db)}$ (new) = gen_newcandidate (Y)
    Clear templ
    Add $C^2_{(DB\cup db)}$(new) to Temp_newCk
k=k+1
```

Figure 4. Updating frequent and promising frequent 1-itemset algorithm

**Algorithm 2 Gen_newcandidate**

**Input :**
(1 )$L^k_{(DB\cup db)}$: frequent k-itemset in updated database,
(2) $PL^k_{DB\cup db)}$ : promising k-itemset in updated database.
(3) Templ : new frequent k-itemset in updated database

**Output :**
(1) new $C^{k+1}$: new candidate k+1-itemset in updated database.

```
If k <= (length(L) + length(PL))
    For each Y $\in$ Templ
        $C^{k+1}$ (new) = Y *($L^k_{(DB\cup db)}$)$\cup PL^k_{(DB\cup db)}$)
    For c $\in$ $C^{k+1}$(new)
        Delete c from $C^{k+1}_{DB}$(new) if all subset of c is in $L^k$ or $PL^k$
```

Figure 5. Generating new candidate itemset algorithm

**Algorithm 3 Update frequent and promising frequent itemsets for k>= 2 itemset**

**Input :**
(1) $L^k_{DB}$ : frequent k-itemset in original database,
(2) $PL^k_{DB,}$ : promising frequent k-itemset in original, database
(3) Temp_newCk : new candidate k-itemset in updated database.

**Output :**
(1) $L^k_{(DB\cup db)}$: frequent k-itemset in updated database,
(2) $PL^k_{DB\cup db)}$ : Promising frequent k-itemset in updated database,
(3) Temp_scanDB : estimated frequent k-itemset and estimated promising frequent k-itemset in updated database
(4) Templ : new estimated frequent k-itemset and new estimated promising frequent k-itemset in updated database
(5) Temp_newCk : new candidate k+1-itemset in updated database.

```
k=2
While k <= (length($L^k$) +length($PL^k$)) do
  Scan db for $\forall(L^k)$ , $\forall(PL^k)$ and $\forall$(items) $\in$ Temp_newCk
     X.support$_{(DB\cup db)}$ = X.support$_{DB}$ + X.support$_{db}$
  For each X $\in$ $L^k_{DB}$ do
        If X.support$_{(DB\cup db)}$>= min_sup$_{(DB\cup db)}$ Then
          Add X to $L^k_{(DB\cup db)}$
        Else
        If X.support$_{(DB\cup db)}$>= min_PL$_{(DB\cup db)}$ Then
          Add X to $PL^k_{(DB\cup db)}$
  For each X $\in$ $PL^k_{DB}$ do
        If X.support$_{(DB\cup db)}$>= min_sup$_{(DB\cup db)}$ Then
          Add X to $L^k_{(DB\cup db)}$
          Add X to templ
        Else
          If X.support$_{(DB\cup db)}$>= min_PL$_{(DB\cup db)}$ Then
            Add X to $PL^1_{(DB\cup db)}$
  For each Y $\in$ Temp_newCk do
   If Y.support$_{(db)}$>= min_sup$_{(db)}$ Then
        Add Y to Temp_scanDB($L^k_{(DB\cup db)}$)
        Add Y to Templ($L^k_{(DB\cup db)}$)
   Else
      If Y.support$_{(db)}$>= (min_PL$_{(DB\cup db)}$)
        or Y.support$_{(db)}$>= min_PL$_{(DB)}$ )Then
          Add Y to Temp_scanDB($PL^k_{(DB\cup db)}$)
          Add Y to Templ($PL^k_{(DB\cup db)}$)
  Clear Temp_newCk
  If Templ $\neq$ {} Then
      Y $\in$ Templ
      $C^{k+1}_{(DB\cup db)}$ (new) = gen_newcandidate (Y)
      Clear Templ
      Add $C^{k+1}_{(DB\cup db)}$(new) to Temp_newCk
k=k+1
If Temp_scanDB $\neq$ {} Then Find_SuppcountDB
```

Figure 6. Update k >= 2 itemset algorithm

```
Algorithm 4  Find_SuppcountDB
Input :
  (1) $L^k_{(DB\cup db)}$ ∈ Temp_scanDB : Estimated frequent k-itemset,
  (2) $PL^k_{(DB\cup db)}$ ∈ Temp_scanDB : Estimated promising frequent k-itemset
Output :
  (1) $L_{(DB\cup db)}$ : frequent k-itemset in updated database,
  (2) $PL_{(DB\cup db)}$ : promising frequent k-itemset in updated database

1    For each W ∈ Temp_scanDB
2      Scan DB for W
3      $W.support_{(DB\cup db)} = W.support_{DB} + W.support_{db}$
4        If $W.support_{(DB\cup db)} >= min\_sup_{(DB\cup db)}$ Then
5          Add W to $L^k_{(DB\cup db)}$
6        Else
7          If $W.support_{(DB\cup db)} >= min\_PL_{(DB\cup db)}$ Then
8            Add W to $PL^k_{(DB\cup db)}$
9      Clear Temp_scamDB
```

Figure 7. Finding support count in an original database algorithm

## IV. EXPERIMENT

To evaluate the performance of promising frequent algorithm, the algorithm is implemented and tested on a PC with a 2.8 GHz Pentium 4 processor, and 1 GB main memory. The experiments are conducted on a synthetic dataset, called T10I4D10K. The technique for generating the dataset is proposed by Agrawal and etc. [1]. The synthetic dataset comprises 20,000 transactions over 100 unique items, each transaction has 10 items on average and the maximal size itemset is 4

Firstly, the proposed algorithm is used to find association rules from an original database of 10,000 transactions. Then, several sizes of incremental databases, i.e. 10%, 20%, 30%, 40% and 50% of the original database, are added to the original database. For comparison purpose, FUP algorithm is also used to find association rules from the same original database and the same incremental databases. The experimental results with various minimum support thresholds are shown in Table1 and Figure 8. From the results, the proposed algorithm has better running time than that of FUP algorithm.

TABLE I. EXECUTION TIME WITH VARYING SIZE OF INCREMENTAL DATABASE

| Min_sup | Algorithm | Execution time (sec.) | | | | |
|---|---|---|---|---|---|---|
| | | Percent of Incremental database size | | | | |
| | | 10% | 20% | 30% | 40% | 50% |
| 4% | Promising Frequent Itemset | 185.2 | 546 | 1047 | 1563.4 | 1935.5 |
| | FUP | 2521.5 | 5012 | 7110.7 | 9996.3 | 11539 |
| 5% | Promising Frequent Itemset | 191.26 | 430.11 | 632.06 | 855.39 | 1048 |
| | FUP | 2023.6 | 4162.4 | 5970.4 | 8678.1 | 10681 |
| 6% | Promising Frequent Itemset | 93.562 | 180.41 | 304.56 | 366.78 | 452.7 |
| | FUP | 1680.5 | 3868 | 5446.8 | 6889.5 | 9516.3 |
| 7% | Promising Frequent Itemset | 52.985 | 98.296 | 463.22 | 1808.4 | 2147.1 |
| | FUP | 1350.6 | 2723.7 | 4873.3 | 5972.3 | 8856.3 |

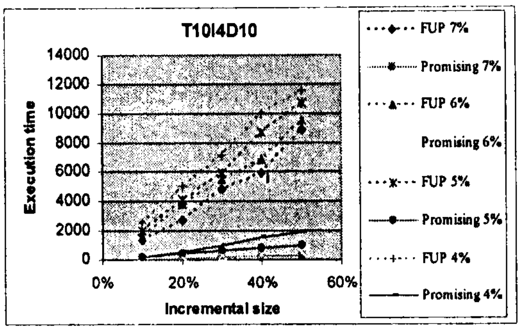

Figure 8. Execution Time comparison

## V. CONCLUSIONS

We have proposed promising frequent algorithm for incremental association rule mining. Assuming that the two thresholds, minimum support and confidence, do not change, the promising frequent algorithm can guarantee to discover frequent itemsets. From the experiment, our algorithm has better running time than that of FUP algorithm. In the future, further researches and experiments on the proposed algorithm will be presented.

## REFERENCES

[1] R. Agrawal., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules." In Proceedings of 20 th Intl Conf. on Very Large Databases (VLDB'94), pages487-499,Santiago, Chile, 1994.

[3] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique,"In 12th IEEE International Conference on Data Engineering,1996.

[4] D. W. Cheung, S.D. Lee, B. Kao, "A General incremental technique for maintaining discovered association rules," In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997.

[5] N.F. Ayn, A.U. Tansel, and E. Arun, "An efficient algorithm to update large itemsets with early pruning." Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999.

[6] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases," In Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97), New Port Beach, California,1997.

[7] C. H. Lee, C. R. Lin , and M. S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining ," ACM , 2000

[8] C.C. Chang, Y.C. Li and J.S. Lee, "An efficient algorithm for incremental mining of association rules," Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA'05) ,IEEE, 2005

[9] A. A. Veloso et al., "Mining frequent itemsets in evolving databases," In Proc. 2nd SIAM Intl. Conf. on Data Mining, Arlington, VA, Apr. 2002

[10] K.L. Lee, G. Lee and A. L.P. Chen, "Efficient Graph-based algorithm for discovering and maintaining knowledge in large database," Proceedings of the third pacific-asia conference on methodologies for knowledge discovert and data mining, April 1999.

[11] N. L. Sarda and N. V. Srinivas, "An adaptive algorithm for incremental. mining of association rules," In Proc. 9th Intl.Workshop on Database and Expert System Applications,Vienna, Austria, pp. 240-245, Aug 1998.

-4