Probability-based Incremental Association Rules Discovery Algorithm with Hashing Technique

Ratchadaporn Amornchewin

Abstract— **Discovery of association rule is one of the most interesting areas of research in data mining, which extracts together occurrence of itemset. In a dynamic database where the new transaction are inserted into the database, keeping patterns up-to-date and discovering new pattern are challenging problems of great practical importance. This may introduce new association rules and some existing association rules would become invalid. It is important to study efficient algorithms for incremental update of association rules in large databases. In this paper, we modify an existing incremental algorithm, Probability-based incremental association rule discovery. The previous algorithm, probability-based incremental association rule discovery algorithm uses principle of Bernoulli trials to find frequent and expected frequent k-itemsets. The set of frequent and expected frequent k-itemsets are determined from a candidate k-itemsets. Generating and testing the set of candidate is a time-consuming step in the algorithm. To reduce the number of candidates 2-itemset that need to repeatedly scan the database and check a large set of candidate, our paper is utilizing a hash technique for the generation of the candidate 2-itemset, especially for the frequent and expected frequent 2-itemsets, to improve the performance of probability-based algorithm. Thus, the algorithm can reduce not only a number of times to scan an original database but also the number of candidate itemsets to generate frequent and expected frequent 2 itemsets. As a result, the algorithm has execution time faster than the 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.**

Index Terms—**Incremental Association Rule Discovery; Association Rule Discovery; Data mining**

I. INTRODUCTION

Data mining has attracted great attention in database communities because of its wide applicability in many areas, including decision support, market strategy and financial forecast. 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.

The rules discovered from a database only reflect the current state of the database. However, in a dynamic database where the new transaction are inserted into the database, keeping patterns up-to-date and discovering new pattern are challenging problems of great practical importance. The updating of the database may cause not only the generation of new rules but also the invalidation of some existing rules. As a brute force approach, Apriori algorithm may be applied to mining a whole dynamic database when the database has been changed. To re-mining the frequent itemsets of the whole updated database is clearly inefficient, because all computations done in the previous mining are wasted. The main idea of the association rule mining in dynamic database refers to optimizations that can be done across mining computations over updated dataset based on previously stored knowledge. Then several research works [5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17] have proposed several incremental algorithms to deal with this problem. Review of previous works will be introduced in section 2.

In this paper, we modify an existing incremental algorithm, Probability-based incremental association rule discovery [15,17]. A previously proposed algorithm Probability-based incremental association rule discovery algorithm uses the principle of Bernoulli trials to find all frequent itemsets and expected frequent itemsets of an updated database efficiently. The expected frequent itemset is the itemsets that have capable of being frequent itemsets after a number of new records have been added to a database.

In addition, the produce candidate item sets are too many when the pattern is too long. Therefore when the database is too large, the running time of the algorithm would be too long. A hash-based technique can be used to reduce the size of the candidate k-itemset (especially when k=2). The key issue of this work is utilizing a hash technique for the generation of the candidate 2-itemset, especially for the frequent 2-itemsets, to improve the performance of probability-based algorithm. Our algorithm can reduce not only a number of times to scan an original database but also the number of candidate itemsets to generate frequent 2 itemsets. As a result, the algorithm has execution time faster than that of previous methods.

II. RELATED 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, it generates a number of

Manuscript received March 17, 2011.

Ratchadaporn Amornchewin is with the Faculty of Information Technology, Thepsatri Rajabhat University, Lopburi, 15000, Thailand. (e-mail: ramornchewin@yahoo.com).

candidate frequent itemsets and then verify them by scanning the database. For a frequent itemset, its support must be higher than a user-specified minimum support threshold. The association rule can be discovered based on frequent itemsets. DHP [3] algorithm is an extension of Apriori by further reducing the number of candidate itemsets using a hashing technique. DHP is very efficient for the generation of candidate large itemsets, in particular for the large 2-itemsets.In addition, DHP proposed the effective pruning techniques to progressively reduce the number of transaction database and the number of items in each transaction.

While both Apriori and DHP efficiently discover association rules from a database, the rules maintenance problem is not addressed. 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 [4]. FUP algorithm is the first incremental updating technique for maintaining association rules when new data are inserted into database. Based on the concepts of Apriori algorithm, FUP computes frequent itemsets using large itemsets found at previous iteration. The major idea of FUP is re-using frequent itemsets of previous mining to update with frequent itemsets of an incremental database. At each iteration, the supports of the size-k frequent item sets of an original database are updated by scanning an incremental database. As well as any k-frequent itemset of the incremental database are updated by scanning the original database to find the new frequent itemsets. As a result, FUP algorithm requires to scan passes over an original database several times when new frequent itemsets are found. This can degrade the performance of FUP algorithm.

To deal with the rescanning problem, negative border approach is presented by Toivonen [6], Thomas et al [7] and Feldman et al [10].This approach maintains both frequent itemsets and border itemsets. 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. Basically, the border-based algorithms start by scanning a new database. Then, the border-based algorithms update support counts of all frequent sets and border sets. Most updated frequent itemsets can be found not only from frequent itemsets but also from border itemsets. This can reduce scanning times of an original database. However, when new frequent itemsets are introduced as updated frequent itemsets, several database scanning is required to obtain support counts of the new frequent itemsets and their subsets. Adnan et al [11] shows that the execute time of the border-based algorithms can severely slower than that of Apriori when new frequent itemsets are introduced as updated frequent itemsets.

Although a large number of itemsets in the border itemsets is not become frequent items when a new database is added to an original database, the border-based algorithms still need to keep them in order to guarantee that all frequent itemsets can be found. Thus, the border-based algorithms need large memory space to keep the border itemsets.

To reduce memory space, Hong el al. [13] and Amornchewin and Kreesuradej [14] 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.

To deal with the problem that the previous approach can only allow very small size of an increment database to insert into an original database, Probability-based Incremental Association Rule Discovery Algorithm, is introduced by Amornchewin and Kreesuradej [15, 17]. Similar to the previous approach, the new algorithm also keeps both frequent itemsets and expected frequent itemsets. The Probability-based algorithm uses the principle of Bernoulli trials to estimate expected frequent itemsets. This technique can allow larger size of an increment database to insert into an original database than that of the previous approach.

To reduce the number of candidates 2-itemset that need to be checked for a given transaction, our algorithm is a improvement of Probability-based algorithm with inclusion of the direct hashing technique. Our approach shall apply the hashing technique for efficient the 2 frequent itemset generation. The preliminary experiments show that the new technique has execution time faster that that of previous methods.

III. USING HASH-BASED FOR PROBABILITY-BASED INCREMENTAL ASSOCIATION RULES 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.
In this section, we describe our algorithm into 2 subsections. Firstly, probability-based expected frequent itemsets is presented. Secondly, updating frequent and expected frequent k-itemsets is introduced.

A. Probability-Based Expected Frequent Itemsets

The key idea of previous incremental mining, Probability-based algorithm, is solve the updating problem of association rules after a number of new records have been added to a database. In this 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.

According to the Probability-based algorithm assumption is that the statistics of new transactions slightly change from original transactions. Then, the statistics of old transaction 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:

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

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 itemset in an updated database can be obtained as the following equations:

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

Here, an expected frequent itemset is an itemset that is not a frequent itemset but has its probability to be a frequent itemset greater than $Prob_{pl}$. $Prob_{pl}$ is a threshold constant specified by users. $Prob_{pl}$ indicates the minimum confidence level that a promising frequent itemset will be a frequent itemset after inserting new transaction into an original database. The higher $Prob_{pl}$ is set, the lesser expected frequent itemsets are kept. As results, the algorithm may need more number of rescanning times in the original database when the algorithm performs the discovering new frequent item task.

B. Updating frequent and expected frequent k- itemsets algorithm

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.

The notation for Updating frequent and expected frequent itemsets algorithm

DB	Original database
db	Incremental Database
UP	Updated database
k	Number of itemset
σ	Minimum support
ρ	Minimum Expected Frequent
C_k	Candidate k-itemset
F_k	Frequent k-itemset
EF_k	Expected Frequent k-itemset

Algorithm1: Main 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. *end if*
6. *if* $k = 2$
7. *Generate Hash _ Candidate*
8. *Update* $2-itemset$
9. $k = k + 1$
10. *else*
11. *for* $(k \geq 3; F_k^{UP} \neq \phi; k++)$ *do*
12. *Generate Candidate Itemset*
13. *Update* $k-itemset$ (*return m, Temp _ scanDB*)
14. *// m is the* max*imum itemset of Temp _ scanDB*
15. $k = k + 1$
16. *end do*
17. *end if*
18. $k = 2$
19. *while* $(Temp\_scanDBl_k \neq \phi$ *and* $(k \leq m)$ *do*
20. *Scan Original Database*$(Temp\_scanDB_k)$
21. $k = k + 1$
22. *end do*
23. *clear Temp _ scanDB*

Figure 1. Main Algorithm

Here, a new updating algorithm shown in Figure 1 is proposed in this paper. The algorithm consists of four phases. The first phase is updating 1- frequent and expected frequent itemsets, i.e. line1-3. The second phase is applying the hash technique to discover the candidate frequent 2-itemsest and update the frequent and expected frequent itemset, i.e. line 6-8. The third phase is repeatedly updating the other frequent and expected frequent itemsets by using only an incremental database, i.e. line 11-16. The last phase is scanning an original database, i.e. line 18-22.

Algorithm 2: Updating 1-itemsets

Input : $DB, db, \sigma^{UP}, \rho^{UP}, C_1^{DB}, F_1^{DB}, EF_1^{DB}, C_1^{db}$ *and their count*

Output : $F_1^{UP}, EF_1^{UP}, C_1^{UP}$ *and their count*

1. *for all* $x \in C_1^{DB} \cup C_1^{db}$ *do*
2. $c(x, UP) = c(x, DB) + c(x, db)$
3. *end do*
4. $F_1^{UP} = \{x \in C_1^{UP} \mid c(x, UP) \geq \sigma^{UP}\}$
5. $EF_1^{UP} = \{x \in C_1^{UP} \mid \rho^{UP} \leq c(x, UP) < \sigma^{UP}\}$

Figure 2. Update 1-itemsest Algorithm

Figure 2 shows the algorithm for updating 1- frequent and expected frequent itemsets. According to the algorithm, 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, i.e. line 1-4. Then, the 1-frequent and expected itemsets of an updated database can be found as shown in line 5 and 6 respectively.

The second phase has 2 major steps which are a discovering the frequent 2-itemset step in an increment database and an updating support count of frequent and expected frequent itemset in updated database step. To discover the set of frequent 2-itemsets, in view of the fact that any subset of a frequent itemset could also have minimum support, the Apriori algorithm uses L_1*L_1 to generate the candidates. The operation * is represented for concatenation in this case. Therefore, C_2 consists of 2-itemset generated by $\binom{|L_1|}{2}$ as candidates in the second iteration.

Since the set of candidate itemsets includes all the possible permutations of the elements, Probability-based algorithm may suffer from a very large set of candidate itemsets, especially from candidate 2-itemsets.

Algorithm 3: Generate Hash Candidate

Input: db, k

Output: C_2^{db} *and their count*

1. $k = 2$
2. *set all the bucket of* H_2 *to zero*
3. *for all transaction* $t \in db$ *do*
4. *for all* $2-$*subsets* y *of* t *do*
5. *// get the subsets of 2 that are candidates*
6. $H_2[h_2(y)]++$
7. *count* $c(y, db)$ *for all* $y \in C_2^{db}$
8. *end do*
9. *end do*
10. $C_2^{db} = \{y \in C_2^{db} \mid c(y, db) > 0\}$

Figure 3. Generate Hash_Candidate

Algorithm 4 : Updating 2-itemsets

Input: $\sigma^{db}, \sigma^{UP}, \rho^{UP}, F_2^{DB}, EF_2^{DB}, C_2^{db}$ *and their count*

Output: $F_2^{UP}, EF_2^{UP}, Temp\_scanDB_2$ *and their count*

1. *for all* $y \in (F_2^{DB} \cup EF_2^{DB} \cup C_2^{db})$ *do*
2. *if* $y \in (F_2^{DB} \cup EF_2^{DB})$ *and* $Y \in C_2^{db}$ *then*
3. $c(y, UP) = c(y, DB) + c(y, db)$
4. *else if* $y \in (F_2^{DB} \cup EF_2^{DB})$ *and* $y \notin C_2^{db}$ *then*
5. $c(y, UP) = c(y, DB)$
6. *else if* $y \notin (F_2^{DB} \cup EF_2^{DB})$ *and* $y \in C_2^{db}$ *then*
7. *if* $c(y, db) \geq \sigma^{db}$ *then*
8. $Temp\_scanDB_2 = \{y \mid (c(y, db) + (\rho^{DB} - 1)) \geq \sigma^{UP}\}$
9. *end if*
8. *end if*
9. *end do*
10. $F_2^{UP} = \{y \mid c(y, UP) \geq \sigma^{UP}\}$
11. $EF_2^{UP} = \{y \mid \rho^{UP} \leq c(y, UP) < \sigma^{UP}\}$

Figure 4. Update 2-itemsest Algorithm

The problem of a large set of candidate itemsets will hinder an effective use of the hashing technique [3]. In this phase, we use a hash technique for the generation of the candidate 2-itemset, especially for the frequent 2-itemsets, to improve the performance of probability-based algorithm.

After reading each new transaction, itemsets of 2-subset can be mapped to hash buckets and stored there. The algorithm for generating 2- incremental candidate itemsets based on the hash table is shown in Figure 3. For each itemset in a hash bucket, the following information is stored: 1) the length of the itemset, 2) hash address, which is used for identifying this itemset, , i.e. line 1-6 and 3) a count of the number of occurrences of the itemset as show in line 7.

Figure 4 shows an algorithm for updating support count of 2- updated frequent itemsets and 2- updated expected frequent 2-itemsets, as shown in line 1-11. To reduce the number of itemset for scanning original database, if sum of any new 2 frequent's support count and support of $prob_{pl}$ minus 1 is greater than minimum support of updated database, then it will be moved to Temp_scanDB, as show in line 7.

Algorithm 5: Generating Candidate k-itemsets

Input $F_l^{UP}, F_{k-l}^{UP}, F_{k-l}^{db}, k$

Output C_k^{new}

1. *if* $k > 2$ *then*
2. $C_k^{db} = F_{k-l}^{db} * F_{k-l}^{db}$
3. *forall* $X \in C_k^{db}$ *do*
4. $C_k^{new} = \{X \in C_k^{db} \mid X \in F_{k-l}^{UP} \text{ and } X \notin (F_k^{DB} \cup EF_k^{DB})\}$
5. *enddo*
6. *endif*

Figure 5. Generating Candidate k- itemsets Algorithm

Algorithm 6: Update (k ≥ 3) itemset

Input: $DB, db, \sigma^{UP}, \rho^{UP}, \rho^{DB}, F_k^{DB}, EF_k^{DB}$ *and their count*

Ouiput: F_k^{UP} *and* $EF_k^{UP}, F_k^{db}, Temp\_scanDB$ *and their count*, m

1. *Scan db and find count* $c(X, db)$ *and* $c(Y, db)$
2. $\backslash\backslash X \in (F_k^{DB} \cup EF_k^{DB})$ *and* $Y \in C_k^{new}$
3. *for all* $X \in (F_k^{DB} \cup EF_k^{DB} \cup C_k^{new})$ *do*
4. *if* $X \in (F_k^{DB} \cup EF_k^{DB})$ *and* $X \in C_k^{new}$ *then*
5. $c(X, UP) = c(X, DB) + c(X, db)$
6. *else if* $X \in (F_k^{DB} \cup EF_k^{DB})$ *and* $X \notin C_k^{new}$ *then*
7. $c(X, UP) = c(X, DB)$
8. *else if* $X \notin (F_k^{DB} \cup EF_k^{DB})$ *and* $X \in C_k^{new}$ *then*
9. $Temp\_scanDB_k = \{X \mid (c(X, db) + (\rho^{DB} - 1)) \geq \sigma^{UP}\}$
10. *end if*
11. *end do*

Figure 6. Update (k ≥ 2) itemset Algrithmo

The third phase has 2 major steps as the second phase except that it does not employ a hash table. The algorithm for generating k- incremental candidate itemsets for k greater than or equal to 3 is shown in Figure 5. In this phase, the algorithm is firstly found k- candidate itemsets of an incremental database, i.e. C_k^{db}, by joining F_{k-1}^{db} with F_{k-1}^{db}, i.e. line 2. 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, i.e. C_k^{new}, 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 itemsets, i.e. line 3-5. This can prune the k- candidate itemsets of an incremental database that can't be the k- updated frequent itemsets.

Algorithm 7 : Scanning an original database

Input : $Temp\_scanDB_k, \sigma^{UP}, \rho^{UP}, F_k^{UP}, EF_k^{UP}$ *and their count*
Output : F_k^{UP}, EF_k^{UP} *and their count*
1. *Scan DB and obtain count* $c(X, DB)$ *for all* $Temp\_scanDB_k$
2. *for all* $X \in Temp\_scanDB_k$ *do*
3. $c(X,UP) = c(X,DB) + c(X,db)$
4. *end do*
5. $F_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } c(X,UP) \geq \sigma^{UP}\}$
6. $EF_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } \rho^{UP} \leq c(X,UP) < \sigma^{UP}\}$
7. $F_k^{UP} = F_k^{UP} \cup F_k^{new}$
8. $EF_k^{UP} = EF_k^{UP} \cup EF_k^{new}$

Figure 7. Algorithm for scanning an original database

Figure 6 shows an algorithm for updating support count of *k*- updated frequent itemsets and *k*- updated expected frequent itemsets for *k* greater than or equal to 3. As shown in line 1, the algorithm scans an incremental database to find and update support count of the *k*- updated candidate itemsets, i.e. C_k^{UP}. When any *k*-itemsets are not in the union set of the original *k*-frequent and the original *k*- expected frequent itemsets, i.e. k-itemsets $\notin F_k^{DB} \cup EF_k^{DB}$, but is in the k-incremental candidate itemsets, i.e. *k*- itemsets $\in C_k^{new}$, their support counts need to be specially updated. This is the case because their support counts obtained from an original database are not available. Since these *k*- itemsets are not in $F_k^{DB} \cup EF_k^{DB}$, their support counts are at best equal to ρ^{DB} - 1. Here, their support counts are assumed to be equal to the sum of ρ^{DB} - 1 and their support counts obtained from an incremental database, i.e. $c(X, db) + (\rho^{DB} - 1)$. If any k- itemsets have support counts below updated min support count, i.e. σ^{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, i.e. line 9.

At the last 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 algorithm is shown in Figure 7. The support counts of the likely *k*-updated frequent itemsets are found and updated by scanning an original database as shown in line 1-6. Then, all *k*-updated frequent itemsets and *k*-updated expected frequent itemsets are found as shown in line 7-8.

IV. EXPERIMENT

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 T10I4D20K. The technique for generating the dataset is proposed by Agrawal and etc. [1]. The synthetic dataset comprises 250,000 transactions over 70 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 20,000 transactions. Then, the same sizes of incremental databases, i.e. 10% of the original database, are added to the original database. For comparison purpose, FUP, Border, Pre-large and Probability-based algorithm are also used to find association rules from the same original database and the same incremental databases. Figure 8 and Table 2 show the average of execution time for FUP, Border, Pre-large, Probability-based and our approach. The results also show that the proposed algorithm has much better running time than that of FUP, Border, Pre-large and Probability-based.

TABLE I. EXECUTION TIME OF ALGORITHM

Algorithm	Execution time (sec)	
	min_sup 3%	min_sup 4%
FUP	8288.03	3512.08
Border	10540	5214
Pre-large	9569.17	3235.39
Probability-based	2992.11	1647.7
Hash-Probability-based	2399.05	1072.09

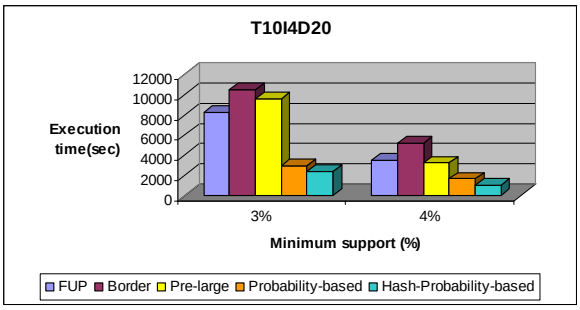

Figure 8. Execution time of FUP, Borders,Pre-large, Probability-based and the proposed algorithm

V. CONCLUSION

We have improved the probability-based incremental association rule discovery algorithm by using hashing technique. We can generate all of the 2-itemsets for each transaction and hash them into the different buckets of a hash table structure with increase the corresponding bucket counts. If any bucket count of 2-itemset in the hash table is less than the minimum support threshold, it cannot be frequent and then should be remove from the candidate set. Thus, the hash technique might substantially reduce especially the number of the candidate 2-itemsets.

In our study, we assume that the two thresholds, minimum support and confidence, do not change. So, the algorithm can guarantee to discover all frequent itemsets. From the

experiment, our algorithm has less running time than FUP, Borders, Pre-Large and Probability-based 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 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] 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, pp. 175-186, May 1995.

[4] 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.

[5] 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.

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

[7] 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.

[8] N.F. Ayan, 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, 287-291

[9] 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.

[10] 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.

[11] M. Adnan, R. Alhajj, and K. Barker, "Performance Analysis of Incremental Update of Association Rules Mining Approaches"; In Proceeding of 9th IEEE International Conference on Intellgent Engineering System 2005, Sept. 16-19, 2005. 129-134

[12] C. H. Lee, C. R. Lin, and M. S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining" , Proceeding of the ACM 10th International Conference on Information and Knowledge Management , November 2001.

[13] 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.

[14] 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.

[15] R. Amornchewin and W. Kreesuradej, "Probability-based incremental association rule discovery algorithm"; The 2008 International Symposium on Computer Science and its Applications (CSA-08), Australia (2008).

[16] R. Amornchewin and W. Kreesuradej, "False Positive Itemset Algorithm for Incremental Association Rule Discovery" ,International Journal of Multimedia and Ubiquitous Engineering. Vol. 4, No. 2, April,2009.

[17] R. Amornchewin and W. Kreesuradej, "Mining Dynamic Databases using Probability-Based Incremental Association Rule Discovery Algorithm", Journal of Universal Computer Science, vol. 15, no. 12 , 2009, pp. 2409-2428 .