*Journal of Intelligent Learning Systems and Applications*, 2011, 3, 181-189
doi:10.4236/jilsa.2011.33019 Published Online August 2011 (http://www.SciRP.org/journal/jilsa)

# DARM: Decremental Association Rules Mining

**Mohamed Taha[1], Tarek F. Gharib[2,3], Hamed Nassar[4]**

[1]Faculty of Computers & Informatics, Benha University, Benha, Egypt; [2]Faculty of Computer & Information Sciences, Ain Shams University, Cairo, Egypt; [3]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia; [4]Faculty of Computers & Informatics, Suez Canal University, Ismailia, Egypt.
Email: mohamed.taha@fci.bu.edu.eg, tfgharib@cis.asu.edu.eg, nassar@ci.suez.edu.eg

Received December 29th, 2010; revised April 1st, 2011; accepted April 8th, 2011.

## ABSTRACT

*Frequent item sets mining plays an important role in association rules mining. A variety of algorithms for finding frequent item sets in very large transaction databases have been developed. Although many techniques were proposed for maintenance of the discovered rules when new transactions are added, little work is done for maintaining the discovered rules when some transactions are deleted from the database. Updates are fundamental aspect of data management. In this paper, a decremental association rules mining algorithm is present for updating the discovered association rules when some transactions are removed from the original data set. Extensive experiments were conducted to evaluate the performance of the proposed algorithm. The results show that the proposed algorithm is efficient and outperforms other well-known algorithms.*

***Keywords*: *Decremental Mining*, *Association Rules Maintenance*, *Updating Association Rules***

## 1. Introduction

Association rules mining is a data mining technique which discovers strong associations or correlation relationships among data [1]. Recently, it has attracted much attention in data mining research because of its wide applicability in many areas, including decision support, market strategy and financial forecast. An association rule is a relation between items in a set of transactions. This rule must have two measures to express its statistical significance: the support and the confidence [2].

There are many mining algorithms to find out association rules in large databases: the Apriori and its variations, Tree-Projection algorithm, FP-growth and others [3-5]. The Apriori algorithm is the first successful algorithm for mining association rules. The main idea of Apriori-based algorithms is to run number of iterations. In each iteration, a set of candidate itemsets is generated then the database is scanned to count the number of transactions that contain each candidate set. The candidates with support counts greater than or equal to the minimum support count are the frequent itemsets. These mining algorithms differ in the techniques used to create the candidate sets. The smaller the number of candidate sets is, the faster the algorithm would be.

The Apriori-based algorithms belong to a generate-and-test paradigm. They compute frequent itemsets by generating candidates and checking their frequencies against the transaction database. Another paradigm is based on a test-only approach. It does not generate candidates and only tests for frequency. The FP-growth algorithm belongs to this paradigm. It uses a tree structure to represent all the transactions of the database. The frequent itemsets are generated with only two passes over the database and without any candidate generation process [6].

Due to advances in information technology, a large amount of data could be collected easily. Databases may be frequently or occasionally updated. Such updates may change the characteristic of the database and hence invalidate the existing discovered rules (some of the rules may still be valid and new rules may appear). A brute force approach to solve the update problem is to re- mine the entire updated database to find the new association rules. However, this approach is not efficient because it ignores the previous computation that has been performed. The present approach to the update problem is to use incremental and decremental mining algorithms. Incremental mining refers to mining the database when new transactions are added to it while decremental mining refers to mining the database when some obsolete transactions are deleted from it [7,8]. The main purpose of these mining algorithms is to update the previously discovered rules by benefiting from the previously discovered information without repeating all the work done

 

previously. Much effort has been devoted to the problem of incremental mining and several algorithms have been already proposed to update the association rules after adding new transactions but decremental mining is not as fortunate.

In this paper, a decremental updating algorithm called Decremental Association Rules Mining algorithm (DARM) has been introduced which aims to store and maintain the itemsets that are not frequent at present but may be frequent after updating the database. Therefore the processing cost of the updated database can be reduced.

The rest of the paper is organized as follows. In the next section, some preliminaries in association rules mining are illustrated. Section 3 surveys the related work. In Section 4, the DARM algorithm is presented. Section 5 shows the experimental results. Finally, conclusions are presented in Section 6.

## 2. Preliminaries

Let $I = \{i_1, i_2, \cdots, i_m\}$ be a set of items and *DB* be a transaction database, where each transaction *T* is a set of items such that $T \subseteq I$. For an itemset $A \subseteq I$, a transaction *T* contains *A* if and only if $A \subseteq T$. An association rule is defined as an implication of the form $A \Rightarrow B$, where $A \subseteq I$, $B \subseteq I$ and $A \cap B = \varnothing$. The association rule $A \Rightarrow B$ has support *s* in DB if *s* is the percentage of transactions in *DB* contains $A \cup B$. The association rule $A \Rightarrow B$ holds in *DB* with confidence *c* if *c* is the percentage of transactions in *DB* that contain *A* also contain *B*. The problem of mining association rules is to extract all the association rules whose support and confidence are not less than given threshold values called *min_sup* and *min_conf*. The association rules that satisfy both the minimum support threshold and minimum confidence threshold are called strong rules to distinguish them from the weak ones. The support of an itemset *A* is the percentage of transactions in *DB* that contain *A*. An itemset *A* is frequent if its support is not less than a minimum support threshold *min_sup* [9-12].

Let *L* be the set of frequent itemsets in the database *DB*, *D* be the number of transactions in *DB*, *s* be the minimum support and *X.support* be the support count for an itemset *X* which is the number of transactions in *DB* containing *X*. Assume that for each $X \in L$, *X.support* is available. After some updates, a set of old transactions *db* (decrement database) is deleted from *DB*, and *d* is the number of transactions in *db*. Using the same minimum support *s*, an itemset *X* is frequent in the updated database ($DB - db$) if the support of *X* in ($DB - db$) is not less than or equal to *s*, *i.e.*, $X.support \geq s \times (D - d)$ [9,11]. Therefore, the problem of updating association rules can be defined as finding the set *L`* of frequent itemsets in $DB - db$.

## 3. Related Work

In the literature, the majority of the research in updating the discovered association rules is focused on the incremental mining. This is due to updating databases by adding new transactions has many applications such as web log records, stock market data, grocery sales data, transactions in electronic commerce, and daily weather/ traffic records, to name a few [9]. Several incremental mining algorithms are proposed such as Fast UPdate algorithm (FUP) [10], The Update Large Itemsets algorithm (ULI) [13], Negative Border with Partitioning (NBP) [2], Update with Early Pruning algorithm (UWEP) [11], New Fast UPdate algorithm (NFUP) [14] and Granular Computing Based Incremental Frequent Itemset Mining in distorted databases (GrC-IFIM) [15]. Unfortunately, little work has addressed the decremental mining although deletion is one of the most frequently operations in many database systems. In many applications, the user may want to remove the out-of-date data from the database.

Cheung *et al.* proposed the first decremental mining algorithm called FUP2 [12]. In fact, it is an extension of the incremental algorithm FUP to update the discovered association rules when transactions are deleted from the database. Mainly, the framework of FUP2 is similar to that of Apriori. It contains number of iterations. At each iteration, the frequent itemsets of the same size are found. However, FUP2 has two main problems; it generates large number of candidate itemsets and it requires multiple scans of the database.

Lee *et al.* proposed a Sliding-Window Filtering technique (SWF) for updating the discovered association rules [9]. The SWF technique partitions the transaction database into a set of partitions and employs the minimum support threshold in each partition to filter unnecessary candidate itemsets. The algorithm uses a scan reduction technique in which it generates all the candidate itemsets of all sizes from the candidate 2-itemsets to reduce the number of database scans. However, this leads to generating larger number of candidates. In fact, SWF does not utilize previously discovered frequent itemsets which are used in the other algorithms to improve incremental or decremental mining. Since the SWF is based on partitioning the database, it requires the data to be uniformly distributed to work well. Chang *et al.* extended the SWF algorithm by incorporating previously discovered information and proposed two algorithms to enhance the performance for incremental mining [16]. However, they are only suitable for cases in which a whole partition of data is deleted from a database at a time. In practice, situations requiring the deletion of continuous blocks of data from databases are not very common [17].

 

Lian *et al.*, [18] addressed the relationships between old and new maximal frequent itemsets and proposed an algorithm called IMFI, which is based on these relationships to reuse previously discovered knowledge. The algorithm follows a top-down mechanism rather than traditional bottom-up methods to produce fewer candidates. Moreover, they integrated SG-tree into IMFI to improve the counting efficiency.

Zhang *et al.* proposed the Decrement Updating Algorithm (DUA) [7,8]. The algorithm tries to find those itemsets that are most likely to change their frequentness in the remaining database by: 1) finding the frequent itemsets of the decrement database using a threshold value less than the minimum support. 2) finding the frequent itemsets of a random set of transactions from the original database. One important drawback of this algorithm is that its accuracy drops when a lower minimum support threshold is used because it can not find all the frequent itemsets in the updated database. Another drawback is that the DUA algorithm is efficient as long as the size of the decrement database is less than 30% of the total data in the original database [7].

Also, Zhang *et al*. proposed another algorithm called Efficient Dynamic Database Updating Algorithm (EDUA) [17]. It is similar to the SWF in adopting the scan reduction technique. The EDUA consists of two procedures: the pre-mining procedure which uses the scan reduction technique to calculate all the frequent itemsets of the original database and store all the candidate 2-itemsets for the following procedure; the dynamic maintaining procedure in which all the candidate 2-itemsets in the deleted database are calculated with their support counts and subtracted from their corresponding 2-itemsets in the original database then the scan reduction technique is used again to generate all the candidates of all sizes. Finally, the new candidate itemsets are checked against the updated database in order to discover the new frequent itemsets. The main advantage of the EDUA over the SWF is that it can deal with the case in which transactions are deleted from any part of the database, while the SWF can only function when the whole first partition is deleted from the database. Like SWF, EDUA does not benefit from the previously discovered frequent itemsets. A pruning technique was also proposed in [17] to enhance the performance of the EDUA that tries to use heuristic knowledge of the prior mining results.

Mengling Feng *et al.*, [19] addressed the maintenance of the frequent patterns space of both incremental and decremental updates. The author's conducted an in-depth investigation on how the frequent pattern space evolves under both incremental and decremental updates. Based on the evolution analysis, a new data structure, Generator-Enumeration Tree (GE-tree), is developed to facilitate the maintenance of the frequent pattern space. With the concept of GE-tree, Mengling Feng *et al.*, proposed two novel algorithms, Pattern Space Maintainer+ (PSM+) and Pattern Space Maintainer− (PSM−), for the incremental and decremental maintenance of frequent patterns.

## 4. The DARM Algorithm

The main idea of the proposed algorithm (DARM) is to store and maintain the itemsets that are not frequent at present but may be frequent after deleting a set of transactions. We will call these itemsets the *semi-frequent itemsets* and denote them by *SF*. In practice, the decremental algorithm is not invoked every time a transaction is deleted from the database. However, it is invoked after a non-trivial number of transactions are deleted. Assume that the algorithm will be invoked every time $d$ transactions are deleted from the database, so an itemset $X$ is semi-frequent if it satisfies the following condition:

$$s \times D > X.support \geq s \times (D - d)$$

This means that the semi-frequent itemsets are those itemsets that may be frequent after $d$ transactions are deleted from the database where their support counts are less than the minimum support count and greater than or equal to $s \times (D - d)$.

When deleting a set of transactions from a database, all the itemsets in the updated database can be categorized into four classes as illustrated in **Table 1**. The itemsets in classes A and D need some processing because they may be frequent or infrequent in the updated database. However, the itemsets in classes B and C are straight forward. If an itemset is frequent in the original database and infrequent in the decrement database, it will be frequent in the updated database. Also, if an itemset is infrequent in the original database and frequent in the decrement database, it will be infrequent in the updated database.

In order to find the new frequent itemsets of the updated database, the DARM algorithm must handle all the itemsets in the four classes. The itemsets of class A, B and C can be obtained easily by using the frequent itemsets of the original and the decrement databases. How-

**Table 1. The four classes for an itemset in the updated database.**

| *Class* | *Original Database* ($DB$) | *Decrement Database* ($db$) | *Updated Database* ($DB - db$) |
|---|---|---|---|
| A | Frequent | Frequent | Frequent or Infrequent |
| B | Frequent | Infrequent | Frequent |
| C | Infrequent | Frequent | Infrequent |
| D | Infrequent | Infrequent | Frequent or Infrequent |

 

ever, the itemsets of class D need further consideration that is why the semi-frequent itemsets are stored. **Table 2** shows the definitions of symbols used in the pseudo-code of the DARM algorithm. The actual steps of the DARM algorithm are shown in **Figure 1**. The first step of the algorithm is to find the frequent itemsets of the decrement database with their support counts. Then, it subtracts the support counts of the frequent itemsets of the decrement database from their corresponding frequent itemsets of the original database. The new support counts of these itemsets are then compared to the minimum support to determine which itemsets will be still frequent and which will not. By this way, the algorithm handles those itemsets falling in class A.

To determine the itemsets of class D, the algorithm eliminates each frequent itemset in the decrement database from *SF*. After this elimination, the remaining itemsets in *SF* are the infrequent itemsets in both the original and the decrement database (class D). These itemsets require a scan of the decrement database to update their support counts then the algorithm verifies them against the minimum support. The rest of the itemsets that are frequent in the original database and they do not occur in the frequent itemsets of the decrement database will be frequent in the updated database and do not need to be verified (class B).

However, the DARM algorithm needs a scan to the decrement database for these itemsets to update their support counts for further runs. Hence, the algorithm scans the decrement database only once to handle all the itemsets in class B and class D. Also, the rest of frequent itemsets of decrement database that do not have a corresponding frequent itemsets in the original database will be infrequent so that they do not need any processing at all (class C).

For the subsequent runs of the algorithm, the semi-frequent itemsets need to be updated. **Figure 2** shows the Update_SF procedure that maintains the itemsets in SF. First, the procedure computes the negative border of new frequent itemsets of the updated database. Then, it computes the support count of each itemset in the negative border in the updated database and checks if the itemset satisfies the condition of semi-frequent itemsets.

**Example 4.1** Consider the database DB presented in **Figure 3(a)** with ten transactions indexed from T1 to T10. All the frequent and semi-frequent itemsets of DB are known in advance and shown in **Figure 3(b)** with a

**Table 2. The list of symbols used in the DARM Algorithm.**

| *Symbol* | *Meaning* |
|---|---|
| $DB$ | The original database |
| $D$ | The number of transactions in $DB$ |
| $db$ | The decrement database |
| $d$ | The number of transactions in $db$ |
| $DB - db$ | The updated database |
| $D'$ | The number of transactions in the updated database |
| $S$ | The minimum support |
| $SF$ | The set of semi-frequent itemsets |
| $L$ | The set of frequent itemsets in $DB$ |
| $L_{dec}$ | The set of frequent itemsets in $db$ |
| $L'$ | The set of frequent itemsets in $DB - db$ |
| $X$ | An itemset |
| $X.support$ | The number of transactions containing $X$ in the database |
| $X.support_{DB}$ | The number of transactions containing $X$ in $DB$ |
| $X.support_{db}$ | The number of transactions containing $X$ in $db$ |
| $X.support_{DB-db}$ | The number of transactions containing $X$ in $DB - db$ |

**Inputs:**
$DB$, $db$, $L$, $SF$ and $s$

**Output:**
$L'$ ($L'$ is initially set to ∅) and $SF$

**Algorithm:**

1- Find the set $L_{dec}$ of frequent itemsets in the decrement database db.

2- For all $X \in L$
  If $X \in L_{dec}$ then
    $X.support_{DB-db} = X.support_{DB} - X.support_{db}$
    If $X.support_{DB-db} \geq s \times (D - d)$ then
      $L' = L' \cup \{X\}$
    Remove $X$ from $L$ and $L_{dec}$

3- For all $X \in SF$
  If $X \in L_{dec}$ then
    Remove $X$ from $SF$

4- Scan $db$ and compute $X.support_{db}$ for all $X \in L \cup SF$

5- For all $X \in L \cup SF$
  $X.support_{DB-db} = X.support_{DB} - X.support_{db}$
  If $X \in L$ then
    $L' = L' \cup \{X\}$
  Else
    If $X.support_{DB-db} \geq s \times (D - d)$ then
      $L' = L' \cup \{X\}$

6- Update_$SF$ ($L'$) //a procedure to maintain the itemsets in $SF$

7- Return $L'$

**Figure 1. The DARM algorithm.**

Function Update_$SF$ ($L'$)

1- Compute Negative border of $L'$ $NBd$ ($L'$)
2- Scan $DB - db$ and compute $X.support_{DB-db}$ for all $X \in NBd$ ($L'$)
3- For $X \in NBd$ ($L'$)
  If $X.support_{DB-db} < s \times D'$ && $X.support_{DB-db} \geq s \times D'$
    $SF = SF \cup \{X\}$
End Function

**Figure 2. The update procedure of *SF*.**

 

minimum support 30%. After a period of time, four transactions are deleted from *DB*: T1, T7, T9 and T10. To find the new frequent itemsets of the updated database, the algorithm computes the frequent itemsets of the decrement database and the result is shown in $L_{dec}$ in **Figure 3(b)**.

After that, it begins to handle the itemsets of class A which are the common itemsets in $L$ and $L_{dec}$. All the support counts of the frequent itemsets in $L_{dec}$ are subtracted from their corresponding frequent itemsets in $L$ (the highlighted itemsets in **Figure 3(b)**) and these itemsets are removed from the two lists. The results of the subtraction are shown in class A in **Figure 3(c)**. The itemsets in class A for which the support count is not less than ($6 \times 30\% = 1.8 \approx 2$) are moved to $L'$ as shown in **Figure 3(c)**. Note that the itemsets EF and CEF have not enough support count so they are not moved to $L'$. The remaining itemsets in $L$ and $L_{dec}$ are shown in class B and class C in **Figure 3(d)** respectively. By default, all the itemsets in class B will be frequent in the updated database and there is no need to be verified but their support counts need to be updated. Only their support counts in *DB* are known so they need a *db* scan to determine their support counts. The DARM algorithm postpones this scan until it determines the itemsets of class D to handle both of them in one scan. Using the itemsets in *SF* and in $L_{dec}$, the algorithm can determine the itemsets of class D by subtracting the itemsets of $L_{dec}$ from *SF*. Here, the itemset BEF marked by an oval in **Figure 3(b)** is eliminated from *SF* because it appears in $L_{dec}$ and all the remaining itemsets in *SF* will belong to class D as shown in **Figure 3(d)**. Then, the algorithm scans *db* to update the support counts of the itemsets of class B and D. **Figure 3(e)** shows the itemsets of class B and D with their new support counts in the updated database. All the itemsets of class B are then moved to $L'$ without any verification while the itemsets of class D are verified against the minimum support count of the updated database. The itemsets D and AC are the only itemsets from class D that are moved to $L'$.

**Original database (*DB*)**

| TID | Transaction |
|---|---|
| T1 | A B E F |
| T2 | B C F |
| T3 | A C E F |
| T4 | B D E |
| T5 | A B C E |
| T6 | A B D E |
| T7 | C E F |
| T8 | A B F |
| T9 | B C E F |
| T10 | B F |

**Decrement database (*db*)**

| TID | Transaction |
|---|---|
| T1 | A B E F |
| T7 | C E F |
| T9 | B C E F |
| T10 | B F |

(a)

| ***L*** | | ***$L_{dec}$*** | | ***SF*** | |
|---|---|---|---|---|---|
| Itemset | Count | Itemset | Count | Itemset | Count |
| A | 5 | B | 3 | D | 2 |
| B | 8 | C | 2 | AC | 2 |
| C | 5 | E | 3 | ABF | 2 |
| E | 7 | F | 4 | AEF | 2 |
| F | 7 | BE | 2 | BCE | 2 |
| AB | 4 | BF | 3 | BCF | 2 |
| AE | 4 | CE | 2 | BEF | 2 |
| AF | 3 | CF | 2 | | |
| BC | 3 | EF | 3 | | |
| BE | 5 | BEF | 2 | | |
| BF | 5 | CEF | 2 | | |
| CE | 4 | | | | |
| CF | 4 | | | | |
| EF | 4 | | | | |
| ABE | 3 | | | | |
| CEF | 3 | | | | |

(b)

| **Class A** | | ***L′*** | |
|---|---|---|---|
| Itemset | Count | Itemset | Count |
| B | 5 | B | 5 |
| C | 3 | C | 3 |
| E | 4 | E | 4 |
| F | 3 | F | 3 |
| BE | 3 | BE | 3 |
| BF | 2 | BF | 2 |
| CE | 2 | CE | 2 |
| CF | 2 | CF | 2 |
| EF | 1 | | |
| CEF | 1 | | |

(c)

| **Class B** | | **Class C** | | **Class D** | |
|---|---|---|---|---|---|
| Itemset | Count | Itemset | Count | Itemset | Count |
| A | 5 | BEF | 2 | D | 2 |
| AB | 4 | | | AC | 2 |
| AE | 4 | | | ABF | 2 |
| AF | 3 | | | AEF | 2 |
| BC | 3 | | | BCE | 2 |
| ABE | 3 | | | BCF | 2 |

(d)

| **Class B** | | **Class D** | | ***L′*** | |
|---|---|---|---|---|---|
| Itemset | Count | Itemset | Count | Itemset | Count |
| A | 4 | D | 2 | A | 4 |
| AB | 3 | AC | 2 | B | 5 |
| AE | 3 | ABF | 1 | C | 3 |
| AF | 2 | AEF | 1 | D | 2 |
| BC | 2 | BCE | 1 | E | 4 |
| ABE | 2 | BCF | 1 | F | 3 |
| | | | | AB | 3 |
| | | | | AC | 2 |
| | | | | AE | 3 |
| | | | | AF | 2 |
| | | | | BC | 2 |
| | | | | BE | 3 |
| | | | | BF | 2 |
| | | | | CE | 2 |
| | | | | CF | 2 |
| | | | | ABE | 2 |

(e)

**Figure 3. An illustrative example.**

## 5. Experimental Results

Several experiments are designed to assess the perform-

 

ance of the proposed algorithms and compare it with the performance of two other decremental algorithms: DUA and EDUA. These two algorithms are chosen because they are the most recent algorithms for updating association rules when some transactions are deleted from the database. The comparisons are based on the following metrics: run time, minimum support, original database size and decrement database size.

In this section, the experiments of the DARM algorithm are reported. All the programs are implemented using C# and run on a PC with 1.8 GHz Intel Core 2 Duo processor and 1GB memory running on Windows XP Professional. In the experiments of data mining, it is common to use the notation [Tx-Iy-Dm-dn] to represent a dataset in which x is the mean size of a transaction, y is the mean size of maximal frequent itemsets, m is the number of transactions in the original database (in thousands) and n is the number of transactions in the decrement database (in thousands).

The experiments carried out performed using the synthetic data used in the experimental results of the previously mining algorithms introduced in [7-9,15,16]. Readers are referred to these papers for a detailed description. In general, several different transaction databases are generated from a set of potentially frequent itemsets using the parameters mentioned in the notion [Tx-Iy-Dm-dn]. These transactions mimic the transactions in the retailing environment.

The first experiment measures the relative run time of the DARM, DUA and EDUA algorithms. The three algorithms are tested against different datasets with minimum support thresholds varying from 0.1% to 1%. The results are shown in **Figures 4-6** with original database sizes 50 k, 100 k and 150 k respectively. The size of the decrement database is equal to 20% of the transactions in the original database. It is clear from these figures that the DARM algorithm reduces the run time required to

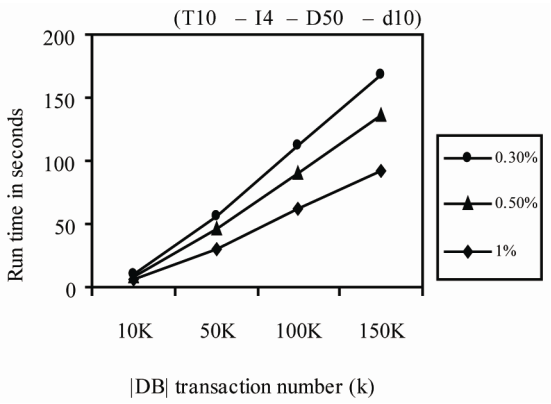


**Figure 4. The relative performance of the decremental algorithms and the DARM Algorithm.**

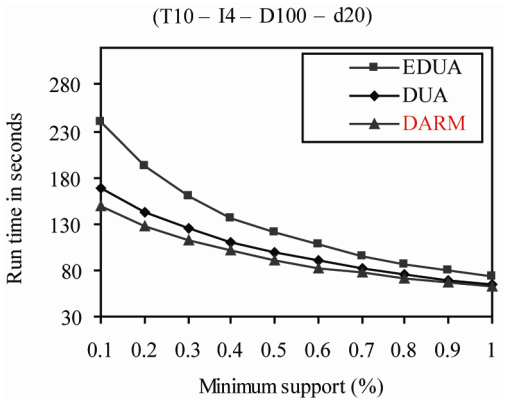


**Figure 5. The relative performance of the decremental algorithms and the DARM Algorithm.**

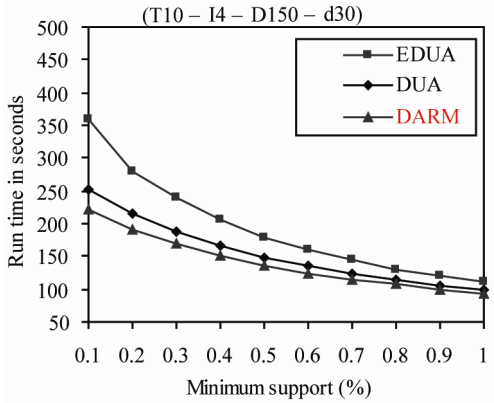


**Figure 6. The relative performance of the decremental algorithms and the DARM Algorithm.**

perform the update operation than other two decremental algorithms.

By examining these figures more carefully, it can be noticed that at low support thresholds, the performance difference among the three algorithms is significant while it decreases as the support thresholds increase. This is because at low support thresholds, large numbers of frequent itemsets are produced hence the performance differences among different algorithms are prominent. However at high support thresholds, there is a restricted number of frequent itemsets so the run time curves of different algorithms approach each other in this region. Also, if we examine these figures more carefully, we can notice sharper differences in the run times of EDUA at high support thresholds than other two algorithms. To understand this phenomenon, recall that EDUA generates the entire candidate itemsets of all sizes from the candidate 2-itemsets to reduce the number of database scans. However, the DUA and the DARM algorithm are based on handling the frequent itemsets.

**Figure 7** shows the average speedup ratio achieved by

 

the DARM algorithm with respect to DUA and EDUA algorithms. It can be seen that the average speed is 1.14 with the DUA and 1.62 with the EDUA.

In the second experiment, the frequent itemsets generated by each algorithm are verified to determine its accuracy. The accuracy is defined as the ratio of the number of frequent itemsets found by the algorithm against the number of frequent itemsets found by running Apriori on the updated database [7,8]. **Table 3** shows the number of frequent itemsets generated by each algorithm. The accuracy of the DUA algorithm is 89.77%, 99.69% and 95.95% for the datasets 50 k, 100 k and 150 k respectively while the accuracy of the DARM algorithm and EDUA is equal to 100% for all datasets. The reason for the accuracy difference among these algorithms is in the way they handle the itemsets of class D. The DUA tries to find the class D itemsets by reducing the minimum support by which the mining is performed on the decrement database and by finding the frequent itemsets of a random set of transactions from the original database. However, this does not guarantee to find all itemsets of class D. For the EDUA, it depends on storing all candidate 2-itemsets and generating all the candidates of all sizes from them so no frequent itemset will be missed. For the DARM algorithm, only the semi-frequent itemsets are stored and this guarantees finding all frequent itemsets without any miss.

The next experiment is performed to find out how the size of decrement database affects the performance of the DARM algorithm. In this experiment, the size of original database is fixed while the size of the decrement database is changing. As shown in **Figure 8**, the results are very encouraging. It shows that the DARM algorithm not only can work with small decrement, but also can work very well in case of large decrement. In general, the larger the decrement is, the longer it would take to do the update.

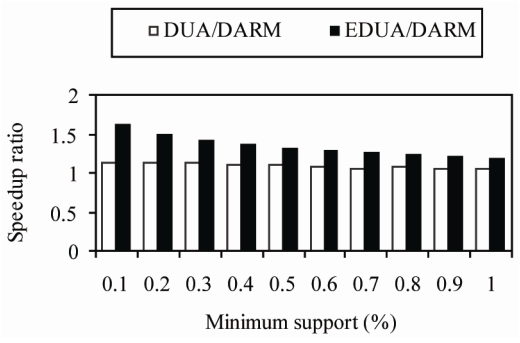


**Figure 7. The speedup ratio.**

**Table 3. Accuracy comparsion of the decremental algorithms.**

| Datasets | Number of frequent itemsets generated | | | |
|---|---|---|---|---|
| | DUA | EDUA | DARM | Apriori |
| **T10-I4-D50-d10** | 13,404 | 14,931 | 14,931 | 14,931 |
| **T10-I4-D100-d20** | 27,377 | 27,461 | 27,461 | 27,461 |
| **T10-I4-D150-d30** | 22,470 | 23,419 | 23,419 | 23,419 |
| *Accuracy* | 89.77% to 99.69% | 100% | 100% | |

Another similar experiment is done to know the influence of changing the size of the original database. **Figure 9** shows the scalability of the DARM algorithm with respect to changing the size of the original database. It shows that the proposed algorithm is adaptive to size increase and can be applied to very large databases.

The last experiment evaluates the reduction in the total number of candidates generated by the DARM algorithm. In this experiment, the number of candidates generated by the DARM algorithm is counted and compared with the number of candidates generated by the other algorithms. The results are shown in **Table 4**. Note the significant reduction in total number of candidate itemsets compared to the other algorithms. The DARM algorithm achieves a reduction rate equal to 34.98% relative to the DUA and 47.73% relative to the EDUA. **Figure 10** shows the candidate reduction rates of the DARM algorithm with respect to the DUA and EDUA algorithms.

## 6. Conclusions

The maintenance of the frequent pattern is a challenging task in data mining and machine learning. In this paper, a

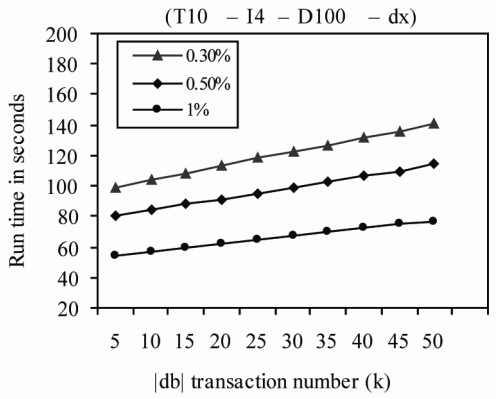


**Figure 8. Scalability with the number of transactions in db.**

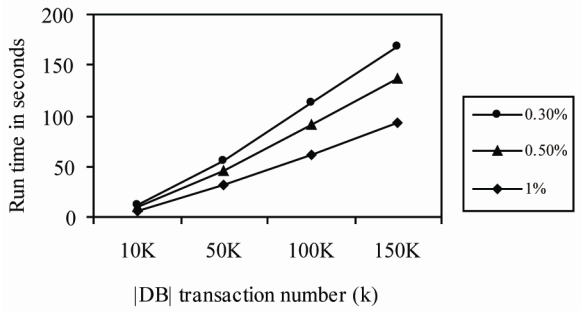


**Figure 9. Scalability with the number of transactions in db.**

 

**Table 4. Comparing the number of candidates.**

| Itemsets | Number of candidates generated | | |
|---|---|---|---|
| | DUA | EDUA | DARM |
| 1 | 2,427 | 2,000 | 1,611 |
| 2 | 33,459 | 37,801 | 20,530 |
| 3 | 24,433 | 30,852 | 15,877 |
| 4 | 19,147 | 23,341 | 12,667 |
| 5 | 11,431 | 15,203 | 7,890 |
| 6 | 5,335 | 9,612 | 3,838 |
| 7 | 1,965 | 2,757 | 1,409 |
| 8 | 543 | 1,002 | 377 |
| 9 | 100 | 359 | 67 |
| 10 | 9 | 36 | 6 |
| **Total** | 98,849 | 122,963 | 64,272 |

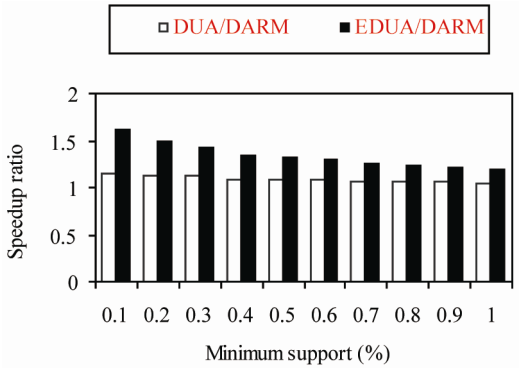


**Figure 10. Candidate reduction.**

decremental association rules mining algorithm (DARM) is presented. The proposed algorithm studied the problems of pattern maintenance when some transactions are removed from a large database.

The main idea of the DARM algorithm is to store and maintain the itemsets that are not frequent at present but may be frequent after deleting a set of transactions. Therefore the processing cost of the original database can be reduced. The DARM algorithm does not require rescanning the original database and can determine the new frequent itemsets by scanning the decrement database only. Experiments have shown that the DARM algorithm not only attain highly accurate mining results, but also run significant faster than existing algorithms for updating the frequent itemsets. The DARM algorithm outperforms the DUA in terms of both run time and in the accuracy.

## REFERENCES

[1] Z. Q. Zhou and C. I. Ezeife, "A Low-Scan Incremental Association Rule Maintenance Method," *Proceedings of the* 14*th Canadian Conference on Artificial Intelligence* (*AI* 2001), Ottawa, 2001, pp. 26-35.

[2] R. Kashef and Y. El-Sonbaty, "NBP: Negative Border with Partitioning Algorithm for Incremental Mining of Association Rules," *Proceedings of the International Conference on Intelligent Systems Design and Applications* (*ISDA*'04), Budapest, Hungary, August 2004.

[3] G. Grahne and J. Zhu, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 10, 2005, pp. 1347-1362. doi:10.1109/TKDE.2005.166

[4] R. C. Agarwal, C. C. Aggarwal and V. V. V. Prasad, "A Tree Projection Algorithm for Generation of Frequent Item Sets," *Parallel and Distributed Computing Journal*, Vol. 61, No. 3, 2001, pp. 350-371. doi:10.1006/jpdc.2000.1693

[5] J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *Proceedings of ACM-SIGMOD International Conference on Management of Data*, Dallas, 2000, pp. 1-12. doi:10.1145/342009.335372

[6] A. Ceglar and J. F. Roddick, "Association Mining," *ACM Computing Surveys Journal*, Vol. 38, No. 2, 2006, Article 5.

[7] S. C. Zhang, J. L. Zhang and Z. Jin, "A Decremental Algorithm of Frequent Itemset Maintenance for Mining Updated Databases," *The Journal of Expert Systems with Applications*, Vol. 36, No. 8, 2009, pp. 10890-10895.

[8] S. C. Zhang, X. D. Wu, J. L. Zhang and C. Q. Zhang, "A Decremental Algorithm for Mining Dynamic Databases," *Proceedings of* 7*th International Conference on Data Warehousing and Knowledge Discovery* (*DaWak* 2005), Copenhagen, 2005, pp. 305-314.

[9] C. H. Lee, C. R. Lin and M. S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining," *Proceedings of* 10*th International Conference on Information and Knowledge Management*, Atlanta, 2001, pp. 263-270.

[10] D. W. Cheung, J. Han, V. T. Ng and C. Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," *Proceedings of* 12*th International Conference on Data Engineering*, New Orleans, 1996, pp. 106-114.

[11] N. F. Ayan, A. U. Tansel and E. Arkun, "An Efficient Algorithm to Update Large Itemsets with Early Pruning," *Proceedings of* 5*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, 1999, pp. 287-291.

[12] D. W. Cheung, S. D. Lee and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," *Proceedings of* 5th *International Conference on Database Systems for Advanced Applications*, Melbourne, 1997, pp. 185-194. doi:10.1142/9789812819536_0020

[13] S. Thomas, S. Bodagala, K. Alsabti and S. Ranka, "An Efficient Algorithm for the Incremental Updating of Association Rules in Large Database," *Proceedings of* 3*rd International Conference on Data Mining and Knowledge Discovery*, Newport Beach, 1997, pp. 263-266.

[14] C.-C. Chang, Y.-C. Li and J.-S. Lee, "An Efficient Algorithm for Incremental Mining of Association Rules," *Proceedings of* 15*th International IEEE Workshop on Research Issues in Data Engineering*: *Stream Data Min-*

 

*ing and Applications* (*RIDE-SDMA*'05), Tokyo, 2005.

[15] C. Xu and J. Wang, "An Efficient Incremental Algorithm for Frequent Itemsets Mining in Distorted Databases with Granular Computing," *Proceedings of* 2006 *IEEE*/*WIC*/*ACM International Conference on Web Intelligence*, Hong Kong, 2006, pp. 913-918.

[16] C.-H. Chang and S.-H. Yang, "Enhancing SWF for Incremental Association Mining by Itemset Maintenance," *Proceedings of 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (*PAKDD* 2003), Seoul, 2003, pp. 301-312.

[17] S. C. Zhang, J. L. Zhang and C. Q. Zhang, "EDUA: An Efficient Algorithm for Dynamic Database Mining," *Information Sciences Journal*, Vol. 177, No. 13, 2007, pp. 2756-2767. doi:10.1016/j.ins.2007.01.034

[18] W. Lian, D. W. Cheung and S. M. Yiu, "Maintenance of Maximal Frequent Itemsets in Large Databases," *Proceedings of* 2007 *ACM Symposium on Applied Computing* (SAC'07), Seoul, 2007, pp. 388-392.

[19] M. L. Feng, G. Z. Dong, J. Y. Li, Y.-P. Tan and L. Wong, "Pattern Space Maintenance for Data Updates and Interactive Mining," *Computational Intelligence*, Vol. 26, No. 3, 2010, pp. 282-316. doi:10.1111/j.1467-8640.2010.00360.x

 