Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules

Anthony K. H. Tung[1] Hongjun Lu[2] Jiawei Han[1] Ling Feng[3]

[1]Simon Fraser University, British Columbia, Canada. {khtung,han}@cs.sfu.ca

[2]The Hong Kong University of Science and Technology, Hong Kong, China. luhj@cs.ust.hk

[3]The Hong Kong Polytechnic University, Hong Kong, China. cslfeng@comp.polyu.edu.hk

Abstract

Most of the previous studies on mining association rules are on mining *intra-transaction associations*, i.e., the associations among items within the *same transaction*, where the notion of the transaction could be the items bought by the *same customer*, the events happened on the *same day*, etc. In this study, we break the barrier of transactions and extend the scope of mining association rules from traditional intra-transaction associations to inter-transaction associations.

Mining inter-transaction associations poses more challenges on efficient processing than mining intra-transaction associations because the number of potential association rules becomes extremely large after the boundary of transactions is broken. In this study, we introduce the notion of inter-transaction association rule, define its measurements: *support* and *confidence*, and develop an efficient algorithm, FITI (an acronym for "*First Intra Then Inter*"), for mining inter-transaction associations. We compare FITI with EH-Apriori, the best algorithm in our previous proposal, and demonstrate a substantial performance gain of FITI over EH-Apriori.

1 Introduction

As an important theme on data mining, association rule mining research [AIS93] has progressed in various directions, including efficient, Apriori-like mining methods [AS94, KMR+94, Toi96], mining generalized, multi-level, or quantitative association rules [SA95, SA96, KHC97], mining sequential patterns and episodes [SA95, MTV97], association rule mining query languages [MPC96] and constraint-based rule mining [SVA97, NLHP98, LNHP99] etc.. However, there is an important form of association rules which are useful but could not be discovered within the existing association rule mining framework.

Taking stock market database as an example, association rule mining can be used to analyze the share price movements. Suppose a database registers the price of every stock at the end of each trading day. Association mining may find rules like

R_1 : *When the prices of IBM and SUN go up, at 80% of probability the price of Microsoft goes up on the same day.*

While R_1 reflects some relationship among the prices, its role in price prediction is limited; and traders may be more interested in the following type of rules:

R_2 : *If the prices of IBM and SUN go up, Microsoft's will most likely (80% of probability) go up* **the next day** *and then drop* **four days later**.

Unfortunately, current association rule miners cannot discover this type of rules as there is a fundamental difference between the rules like R_2 and those like R_1. A classical association rule like R_1 expresses the associations among items within the *same transaction*. On the other hand, rule R_2 represents some association relationship *among the field values from different transaction records*. To distinguish these two types of association, we name the classical association rules as *intra-transaction associations* and the latter as *inter-transaction associations*. Inter-transaction associations are different from the mining sequential patterns in transaction data [AS95], and the mining of episodes by Mannila, *et al.* [MTV97], although all of them involve a temporal component. This is because both the mining of sequential patterns and episodes involve forming transactions from given data before mining take place.

Based on the above example, a new type of rules call *inter-transaction association rules* is proposed and an efficient algorithm for mining them is described in this paper. An interesting property, *"A frequent inter-transaction itemsets must be made up of frequent intra-transaction itemsets"* is observed and is used to mine inter-transaction association rules from large databases efficiently.

The remaining of the paper is organized as follows. In Section 2, inter-transaction association rules will be defined formally. In Section 3, an efficient algorithm call FITI ("First Intra, Then Inter") is introduced for mining inter-transaction association rules. A prelimary performance study is presented in Section 4 and the study will conclude with Section 5.

2 Problem Description

Let's first introduce the notion of inter-transaction association rules.

Definition 2.1 *Let* $\sum = \{e_1, e_2, \ldots, e_u\}$ *be a set of literals, called items. Let* D *be an attribute and* $Dom(D)$ *be the domain of* D*. A* **transaction database** *is a database containing transactions in the form of* (d, E) *where* $d \in Dom(D)$ *and* $E \subseteq \sum$*.* □

The attribute D in the transaction database is called a *dimensional attribute*. It describes the properties associated with the items, such as time and location. It is assumed that the domain of the dimensional attribute is ordinal and can be divided into equal length intervals. For example, time can be divided into day, week, month etc. These intervals can be represented by integers 0, 1, 2, etc., without loss of generality.

When an association rule involves items which are m intervals apart, the association rule is said to have spanned across m intervals. A mining parameter called **maxspan** (or *sliding_window_size*) denoted by w is introduced and only rules which span less than or equal to w intervals will be mined. Using w, a sliding window in the transaction database is defined as follows.

Definition 2.2 *A* **sliding window** W *in a transaction database* T *is a block of* w *continuous intervals along domain* D*, starting from interval* d_0 *such that* T *contains a transaction at interval* d_0*. Each interval* d_j *in* W *is called a* **subwindow** *of* W *denoted as* $W[j]$ *where* $j = d_j - d_0$*. We call* j *the subwindow number of* d_j *within* W*.* □

Every sliding window in the transaction database forms a *mega-transaction* defined below.

Definition 2.3 *Let* W *be a sliding window with* w *intervals and* u *be the number of literals in* $\sum$*. We define a* **mega-transaction** M *contained within* W *to be* $M = \{e_i(j) | e_i \in W[j], 1 \leq i \leq u, 0 \leq j \leq w-1\}$*.* □

As can be seen from the definition, a mega-transaction in a sliding window W is just the set of items in W, each appended with the corresponding subwindow number of the interval that contains the item.

To distinguish the items in a mega-transaction from the items in a traditional transaction, the items in a mega-transaction are called **extended-items**. We denote the set of all possible extended-items as $\sum'$. Given $\sum$ and w, we will have

$$\sum' = \{e_1(0), \ldots, e_1(w-1), e_2(0), \ldots, e_2(w-1), \ldots, e_u(0), \ldots, e_u(w-1)\}$$

Next we will introduce two terms: *intra-transaction itemset* and *inter-transaction itemset*.

Definition 2.4 *An* **intra-transaction itemset** *is a set of items* $A \subseteq \sum$*. An* **inter-transaction itemset** *is a set of extended items* $B \subseteq \sum'$ *such that* $\exists e_i(0) \in B, 1 \leq i \leq u$*.* □

Now we are ready to define the concept of *inter-transaction association rule*.

Definition 2.5 *An* **inter-transaction association rule** *is an implication of the form* $X \Rightarrow Y$*, where*

1. $X \subseteq \sum', Y \subseteq \sum'$.
2. $\exists e_i(0) \in X, 1 \leq i \leq u$.
3. $\exists e_i(j) \in Y, 1 \leq i \leq u, j \neq 0$.
4. $X \cap Y = \emptyset$. □

Similar to the studies in mining intra-transaction association rules, we introduce two measures of inter-transaction association rules: *support* and *confidence*.

Definition 2.6 *Let* S *be the number of transactions in the transaction database. Let* T_{xy} *be the set of mega-transactions that contain a set of extended-items* $X \cup Y$ *and* T_x *be the set of mega-transactions that contain* X*. Then the* **support** *and* **confidence** *of an inter-transaction association rule* $X \Rightarrow Y$ *are defined as* [1]

$$support = \frac{|T_{xy}|}{S}, \qquad confidence = \frac{|T_{xy}|}{|T_x|}$$

□

Given a minimum support level *minsup* and a minimum confidence level *minconf*, our task is to mine the *complete* set of inter-transaction association rules from the transaction database with $support \geq minsup$ and $confidence \geq minconf$.

3 The FITI Algorithm

The problem of mining inter-transaction association rules can be decomposed into two subproblems:

1. Find all inter-transaction itemsets with support higher than *minsup*. We call these itemsets **frequent inter-transaction itemsets**.

[1] Notice that in our definition, we also count the last several transactions in sequence which do not cover the full sliding window, i.e., those not containing the full maxspan of the sliding window.

2. For every frequent inter-transaction itemset F and for all possible combination of $X \subset F$, output a rule $X \Rightarrow (F - X)$ if its confidence is higher than $minconf$.

We will only focus on subproblem 1 in this paper as solving subproblem 2 is trivial. Two algorithms are derived for solving subproblem 1. The first algorithm, *EH-Apriori* was described in [LHF98]. In this section, the second algorithm, FITI will be introduced.

Unlike EH-Apriori which is modified from the Apriori algorithm, FITI is an algorithm designed specifically for discovering frequent inter-transaction itemsets. FITI makes use of the following property to enhance its efficiency in discovering frequent inter-transaction itemsets.

Property 3.1 *Let F be a frequent* **inter**-*transaction itemset. Let $A_i = \{e_j | 1 \leq j \leq u, e_j(i) \in F\}$ where $0 \leq i \leq (w-1)$. For all i, $0 \leq i \leq (w-1)$, A_i must be a frequent* **intra**-*transaction itemset.* □

Property 3.1 is an important property as it provides a different view of mining frequent inter-transaction itemsets. Instead of viewing mining as an attempt to identify frequently occurring patterns formed from the extended-items, we can view it as an attempt to discover frequently occurring patterns formed from frequent intra-transaction itemsets. As such in FITI, we first try mine frequent intra-transaction itemsets and then mine frequent inter-transaction itemsets from them. This gives rise to the name of FITI which stands for *First Intra Then Inter*. Generally, FITI consists of the following phases.

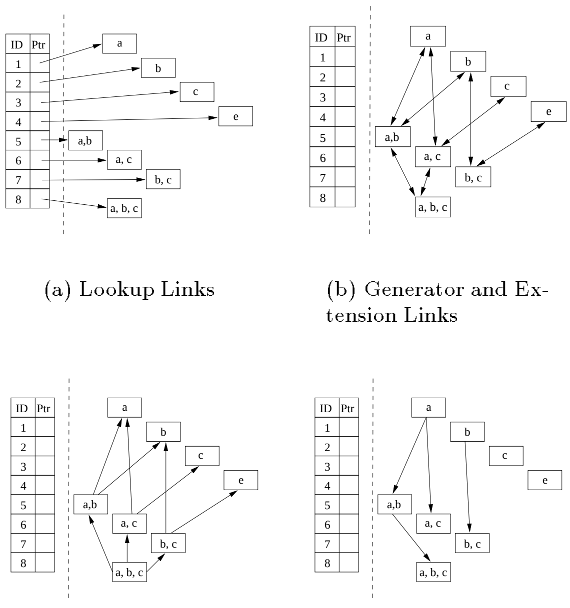
1. Phase I : Discovering Frequent Itemsets and Database Transformation
2. Phase II: Mining Frequent Inter-Transaction Itemsets

3.1 Phase I: Discovering Frequent Itemsets and Database Transformation

In this phase, frequent intra-transaction itemsets are first mined using the Apriori algorithm and then stored in a data structure, called **Frequent-Itemsets Linked Table**, or simply **FILT** which is depicted in Figure 1.

The data structure consists of an *ItemSet Hash Table*, with nodes linked by several kinds of links. The following four kinds of links are built in FILT.

1. **Lookup Links:** Each frequent intra-transaction itemset is assigned a unique **ID** number that corresponds to a row number in the *ItemSet Hash Table*. Each itemset is stored in a node pointed to by a **lookup link** from the corresponding row in the table.



(a) Lookup Links

(b) Generator and Extension Links

(c) Subset Links

(d) Descendant Link

Figure 1: Data Structure for Storing Intra-Transaction Itemsets

2. **Generator and Extension Links:** Given a node N_F that contains an intra-transaction k-itemset F, the generator links of N_F point to the two $(k-1)$-itemsets that are combined to form F in the Apriori algorithm.
3. **Subset Links:** Given a node N_F that contains an intra-transaction k-itemset F, the subset links of N_F point to all subsets of F with size $k-1$.
4. **Descendant Links:** As mentioned earlier, FILT is composed of an array and a hash-tree. Given a node N_F that contains an intra-transaction k-itemset F, the descendant links of N_F point to all of its descendants in the hash-tree. If $F = \{e_1, \ldots, e_k\}$, then its descendants will be a set of $(k+1)$-itemsets of the form $\{e_1, \ldots, e_k, e_{k+1}\}$. □

After FILT is formed, the database is transformed into a set of *encoded Frequent-Itemset Tables*, (called **FIT** tables). We have in total $\mathbf{max_k}$ FIT tables, $\{F_1, \ldots, F_{max_k}\}$, where max_k is the maximum size of the intra-transaction itemsets discovered. Each table F_k will be of the form $\{d_i, IDset_i\}$ where d_i is the value of the dimensional attribute and $IDset_i$ is the IDs of frequent k-itemsets that are found in the transaction.

3.2 Phase II: Mining Frequent Inter-Transaction Itemsets

In this phase, inter-transactions itemsets are represented by their *ID encoding* defined below.

Definition 3.1 *Let F be an inter-transaction itemset. Let $A_i = \{e_j | 1 \leq j \leq u, e_j(i) \in F\}$, where $0 \leq i \leq (w-1)$. The* **ID encoding** *of F is of the form $I = \{I_0, \ldots, I_{w-1}\}$, where I_i is the ID of A_i if $|A_i| > 0$, and 0 otherwise.*

Following the Apriori principle [AS94], level-wise mining is performed by using inter-transaction frequent k-itemsets (for $k \geq 2$) to form candidate frequent $(k+1)$-itemsets.

The generation of frequent 2-itemsets is similar to EH-Apriori in which a hashing approach introduced in [PCY95] is used to prune off candidate inter-transaction itemsets.

To generate candidate frequent inter-transaction k-itemsets for $k > 2$, frequent $(k-1)$-itemsets are first joined to form a set of k-itemsets. Their counts are then generated by scanning the relevant FIT tables once. Two kinds of joins can be performed to create new candidate itemsets: *intra-transaction join* and *cross-transaction join*. These two joins are defined as follow.

Definition 3.2 *Let $I = \{I_0, I_1, \ldots, I_{w-1}\}$ and $J = \{J_0, J_1, \ldots, I_{w-1}\}$ be two frequent intra-transaction itemsets. An* **intra-transactions join** *can be performed on I and J iff both of the following conditions are true.*

1. *There exists a p such that I_p and J_p are the generators of a frequent intra-transaction itemset K_p.*
2. *For all q, $q \neq p$, $I_q = J_q$.*

If I and J can be intra-transaction joined, the resulting candidate itemset will be $K = \{K_0, \ldots, K_p, \ldots, K_{w-1}\}$, where $K_q = I_q = J_q$ for $p \neq q$.

Definition 3.3 *Let $I = \{I_0, I_1, \ldots, I_{w-1}\}$ and $J = \{J_0, J_1, \ldots, J_{w-1}\}$ be two frequent intra-transaction itemsets. A* **cross-transaction join** *can be performed for I and J iff all the following conditions are true:*

1. *There exists a p such that $I_p \neq 0$ and $J_p = 0$.*
2. *There exists a q, $q \neq p$ such that $I_q = 0$ and $J_q \neq 0$.*
3. *For all r, $r \neq p$, $r \neq q$, $I_r = J_r$.*
4. *I_p and J_q are the last non-zero subwindows in I and J respectively.*
5. *All subwindows of I and J either contain no itemset or only frequent intra-transaction 1-itemsets.*

If the above conditions are satisfied, the resulting candidate itemsets will be $K = \{K_0, \ldots, K_p, \ldots, K_q, \ldots, K_{w-1}\}$, where $K_p = I_p$, $K_q = J_q$ and for all r, $r \neq p$, $r \neq q$, $K_r = I_r = J_r$.

To implement the defined joins, we first observe the following property:

Property 3.2 *If two itemsets I and J can be intra-transaction or cross-transaction joined, then they will become an equivalent itemset if each of them has a relevant subwindow set to 0.*

With this observation, we came up with the hash-table method for generating candidate $(k+1)$-itemsets from frequent inter-transaction k-itemsets. Each hash bucket in the table points to a linked list of frequent inter-transaction k-itemset that is hashed to the hash bucket. A k-itemset will be hashed multiple times, each time with one of its subwindows being set to 0. When a k-itemset, I, is hashed to a particular linked list, FITI will go through the linked list to perform intra-transaction join or cross-transaction join between I and itemsets that satisfied the joining condition. I will then be added to the end of the list.

While the candidate itemsets are being generated, they are inserted into a hash-tree to facilitate efficient counting of support [AS94].

4 Prelimary Performance Study

To assess the performance of the proposed algorithms, experiments were conducted using synthetic data. The method used by this study to generate synthetic transactions is similar to the one used in [AS94] with some modifications noted below.

To generate the synthetic data, We first generate a set L of potentially frequent inter-transaction itemsets chosen from $\sum$ items. The size of each itemset in $|L|$ follows a Poisson distribution with mean $|I|$. When generating the transactions, the size of each transaction is determined from a Poisson distribution with mean $|T|$. Itemsets are then assigned to the transaction. These assigned itemsets are either newly picked from the set $|L|$ or are picked when generating earlier transactions. Since itemsets may span multiple transactions, they may be stored in a list for inclusion in subsequent transaction.

Using the method above, we generated a set of synthetic data with 10000 transactions using the following setting: $|T|$=5, $|L|$=1000, $|I|$=5, $|\sum|$=500 and R=4.

To compare the performance of EH-Apriori and FITI, we vary the support level and run the two algorithms on the data set with maxspan equal to 4 intervals. The running time of EH-Apriori and FITI against the support level are shown in Figure 2. The experiment shows that FITI outperforms EH-Apriori by a large margin for the data set. This is not surprising because FITI avoids a large amount of recomputation by making use of FILT and the FIT tables. Due to the fact that FITI is a clear winner, our next experiment on scalability is performed only on FITI.

We test the scalability of FITI by increasing the number of transactions in the dataset from 10k to 1000k with support=0.1% and maxspan=4. The result in Figure 3 shows that the running time of FITI

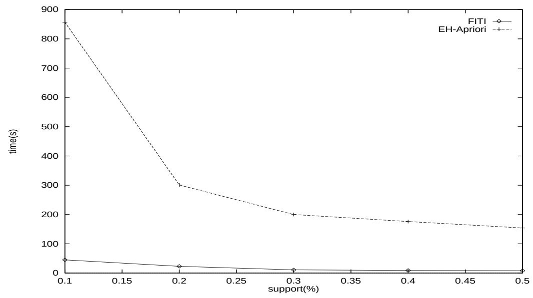


Figure 2: Comparison of FITI and EH-Apriori

increases linearly with the number of transactions in the database.

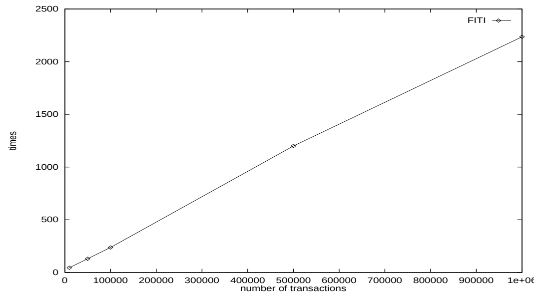


Figure 3: Effect of increasing the no. of transactions

5 Conclusion

In this paper, the concept and definition of inter-transaction association rules are introduced. Two algorithms, EH-Apriori and FITI were implemented for mining inter-transaction association rules. FITI proves to be much faster than EH-Apriori and its performance was found to be acceptable for real life applications. While we believe that the present form of inter-transaction association rules will prove to be useful in providing prediction capability along a single dimension, we feel that this usefulness can be further enhanced if prediction along multiple dimensions is possible. The success of FITI provides us with the necessary ground work for mining this new form of association rules.

Acknowledgmemt: The first and third author were partially supported by Natural Science and Engineering Research (NSERC) of Canada and Networks of Centres of Excellence (NCE) of Canada.

References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., May 1993.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering*, pages 3–14, Taipei, Taiwan, March 1995.

[KHC97] M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, pages 207–210, Newport Beach, California, August 1997.

[KMR+94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proc. 3rd Int. Conf. Information and Knowledge Management*, pages 401–408, Gaithersburg, Maryland, Nov. 1994.

[LHF98] H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. In *Proc. 1998 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98)*, pages 12:1–12:7, Seattle, Washington, June 1998.

[LNHP99] L. V. S. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data*, pages 157–168, Philadelphia, PA, June 1999.

[MPC96] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Proc. 1996 Int. Conf. Very Large Data Bases*, pages 122–133, Bombay, India, Sept. 1996.

[MTV97] H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[NLHP98] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, pages 13–24, Seattle, Washington, June 1998.

[PCY95] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data*, pages 175–186, San Jose, CA, May 1995.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. 1995 Int. Conf. Very Large Data Bases*, pages 407–419, Zurich, Switzerland, Sept. 1995.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 1–12, Montreal, Canada, June 1996.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, pages 67–73, Newport Beach, California, August 1997.

[Toi96] H. Toivonen. Sampling large databases for association rules. In *Proc. 1996 Int. Conf. Very Large Data Bases*, pages 134–145, Bombay, India, Sept. 1996.