A NOVEL APPROACH FOR MINING INTER-TRANSACTION ITEMSETS

S. Nandagopal

Associate Professor, Nanda College of Technology, Erode, Tamilnadu, India

V.P. Arunachalam

Principal, SNS College of Technology, Coimbatore, Tamilnadu, India

S. Karthik

Dean, Department of CSE, SNS College of Technology, Coimbatore, Tamilnadu, India

Abstract

Aim: To address the problem of inter-transaction association rule mining, where the frequent occurrence of a large number of items results in a combinatorial explosion that limits the practical application of the existing mining algorithms. **Methodology**: We propose an efficient algorithm called IAR Miner (Inter-transaction Association Rule Miner), for mining inter-transaction itemsets. Our proposed algorithm consists of two phases. First, we scan the database once to find the frequent items. For each frequent item found, the IAR Miner converts the original transaction database into a set of domain attributes, called a dataset. Then, it enumerates inter-transaction itemsets using an Itemset-Dataset tree, called an ID-tree. By using the ID-tree and datasets to mine inter-transaction itemsets, the IAR Miner can embed effective pruning strategies to avoid costly candidate generation and repeated support counting. **Results:** Our proposed algorithm can efficiently mine inter-transaction patterns. The performance study on the synthetic datasets shows that the IAR Miner algorithm is more efficient than the EH-Apriori, FITI, ClosedPROWL and ITP-Miner algorithms in most cases. **Conclusion**: The IAR Miner algorithm can efficiently mine the inter-transaction patterns. In the future work, we will address a number of research issues related to the IAR Miner algorithm.

Keywords: Data mining, association rules, inter-transaction itemsets, ID-tree, mining ntertransaction, mining algorithms, IAR miner

1.Introduction

Mining association rules, which is a fundamental problem in the area of data mining, has been extensively studied in recent years (Agrawal and Srikant, 1994; Berzal *et al*., 2001; Chen and Hsu, 2007; Han and Fu, 1995; Jea and Chang, 2008; Lee *et al*., 2007; Lee *et al*., 2006; Masseglia *et al*., 2003; Palshikar *et al*., 2007; Tseng and Lin, 2007; Yao and Hamilton, 2006). Traditional association rule mining algorithms focus on association rules among itemsets within a transaction. Taking stock market databases as an example, association rule mining can be used to analyze the share price movements. Suppose a database records the price of every stock at the end of each trading day, an association rule might be: ‘‘if the stock prices of Microsoft and IBM go up, the price of Apple is likely to go up on the same day.” This classical association rule expresses the associations among items within the same transaction, thus we call it intra-transactional association rule. However, the traditional approaches cannot capture a rule like: ‘‘if the stock prices of Microsoft and IBM go up, the price of Apple is likely to go up two days later.” This rule represents some association relationships among the itemsets from different transactions, thus we call it inter-transaction association rule.

A number of methods have been proposed for mining inter-transaction association rules (Chen *et al*., 2005; Feng *et al*., 2002; Lee and Wang, 2007; Lu *et al*., 1998; Lu *et al*., 2000; Lu *et al*., 2005; Tung *et al*., 2003; Lu *et al*., 2000, Park *et al*., 1995) first used inter-transaction association rules to predict stock market movements. Subsequently, two algorithms, E-Apriori and EH-Apriori, for finding frequent inter-transaction itemsets were proposed in Ref. (Lu *et al*., 2000;Bastide *et al* ., 2000), where EH-Apriori adopts an additional pruning technique used in Ref. (Feng *et al*., 2002)used several optimization techniques, namely, joining, converging and speeding, to enhance the EH-Apriori algorithm for mining inter-transactional association rules under rule templates. Then, (Tung *et al*., 2003) proposed the FITI algorithm, which is implemented in two phases. First, the frequent intra- transaction itemsets are discovered. Then, these itemsets are used to form the frequent inter-transaction item- sets. More recently, (Lee *et al*. 2007) proposed an algorithm, called ITP-Miner, which uses an ITP-tree to mine all frequent inter-transaction itemsets in a depth-first search manner. It has been shown that the ITP-Miner algorithm outperforms the previous inter-transaction mining algorithms.

Besides, (Li *et al*., 2005) extended the inter-transaction association rules to a more general form of association rules, called generalized multidimensional inter-transactional association rules, which expand rule contexts from point-wise (e.g., two days latter) to scope-wise (e.g. within three days).

In inter-transaction itemsets mining, there are a large number of frequent itemsets and the mining process could be extremely time-consuming. Thus, we incorporate the concept of closed itemsets into inter-transaction itemsets mining. That is, we only mine closed inter-transaction itemsets, instead of all frequent itemsets. A frequent itemset X is said to be closed if the database does not contain a superset of X with equal support (Han *et al*., 2002; Pei *et al*., 2000), where the support of an itemset is defined as the number of transactions containing the itemset in the database. Generally speaking, mining itemsets is more effcient than mining a complete set of frequent itemsets. Therefore, we will mine ones in our proposed method.

To resolve this problem, we propose an approach called IAR Miner that efficiently mines inter-transaction itemsets. Our proposed algorithm consists of two phases. First, we scan the database to find all frequent items. For each frequent item found, we scan the database to find a set of domain attributes (called a dataset) containing the frequent item. Next, the IAR Miner constructs an ID-tree to generate the inter-transaction patterns in a Breath-first Search (BFS) manner. By using the ID-tree and datasets, the effective pruning strategies can be embedded to avoid costly candidate generation and repeated support counting. Therefore, the IAR Miner algorithm can efficiently mine the inter-transaction patterns.

2. Problem definition

In this section, we define some terms used later.

Definition 1: Let I = {i1, i2,..., i_m} be a set of distinct items and A = {a1, a2, .. ., a_n} be a set of domain attributes. A transaction database D contains a set of transactions in the form of ‹dat, Tdat›, where dat ε A, Tdat is an itemset and dat is the domain attribute of Tdat that describes a property, such as the time stamp associated with Tdat. We can also say that the Tdat itemset occurs in a transaction with the domain attribute dat, or occurs at dat.

For example, Fig. 1 shows a transaction database D containing six transactions, namely T1,T2,T3,T4,T5 and T6. T1 is an itemset {a, b, d, e} and occurs at dat = 1. If this database registers the go-up stocks at the end of each trading day, the first transaction means the stock prices of a, b , d and e go up on day 1.

Dat	Itemset
1	{a,b,d,e}
2	{ b, e }
3	{a,b}
4	{a,b,c}
5	(a,b,c,d}
6	{a,c}

I = {*a*, *b*, *c*, *d*}

A = {1, 2, 3, 4, 5, 6}

D= {<1, *T1*>, <2, *T2*>, <3, *T3*>, <4, *T4*>, <5, *T5*>, <6, *T6*>}

T1 = {*a*, *b*, *d*, *e*} *T2* = {*b*, *e*}

T3 = {*a*, *b*} *T4* = {*a*, *b*, *c*}

T5 = {*a*, *b*, *c*, *d*} *T6* = {*a*, *c*}

Fig 1.A transaction database D

Definition 2: Let <u, Tu> and <v, Tv> be two transactions in a transaction database. The relative distance between u and v is defined as (u-v), where u > v and v is called the reference point. With respect to v, an item ik at u is called an extended item and denoted as ik (u - v), where (u-v) is called the span of the extended item. Similarly, with respect to v (or the transaction at v), a transaction Tu at u is called an extended transaction and denoted as Tu(u-v). Therefore, an extended transaction consists of a set of extended items, i.e., Tu(u- v)= {i1(u- v), .. ., is(u- v)}, where s is the number of items in Tu.

For example, in the database shown in Fig. 1, the extended transaction of the transaction at dat = 3 with respect to the transaction at dat = 1 is {a(2), b(2)}.

Definition 3: Let xi(di) and xj(dj) be two extended items. xi(di) < xj(dj) if (1) di < dj, or (2) di = dj and xi < xj. Moreover, xi(di)= xj(dj) if di = dj and xi = xj.

For example, a(0) < b(0), and b(0) < a(1).

3. Mining inter-transaction itemsets

We now introduce our proposed algorithm, IAR Miner. We first present the data structure ID-pair and the search tree ID-tree. Then, we discuss the pruning strategies and describe the algorithm.

3.1 ID-pair and ID-tree

Since the proposed algorithm uses an itemset-dataset pair (ID-pair) to record information about an inter- transaction itemset in an itemset-dataset tree (ID-tree), we begin by defining ID-pair and ID-tree.

Definition 4: Every node in an ID-tree consists of an itemset-dataset pair (ID-pair), X<t(X)>, where X is a pattern and t(X) is the dataset in which X occurs among the transactions. Let Y1<t(Y1)>, Y2<t(Y2)>, .. ., Ym<t(Ym)> be the m children of the X<t(X)> in an ID-tree..

Let us consider the database shown in Fig. 1 again and assume that minsup = 3 and maxspan = 1. All frequent inter-transaction itemsets can be enumerated in the ID-tree, where the sibling nodes are sorted alphabetically. In the ID-tree, the ID-pair {b(0)}<1, 2, 3, 4, 5> denotes that pattern {b(0)} occurs in the transactions with domain attributes 1, 2, 3, 4 and 5. The children of node {b(0)}<1, 2, 3, 4, 5> are {b(0), a(1)}<2, 3, 4, 5>, {b(0), b(1)}<1, 2, 3, 4> and {b(0), c(1)}<3, 4, 5>. Moreover, the pattern represented by a node is a sub-pattern of the patterns represented by its child nodes. Note that an ID-tree can be condensed to a smaller one in the proposed algorithm by using some pruning strategies.

3.2 Pruning strategies

The proposed algorithm applies two major pruning strategies to reduce the search space, namely, the down- ward closure property and the four properties for patterns. The downward closure property means that if a pattern is frequent, then all of its sub-patterns are frequent. In other words, if a pattern is not frequent, all of its super-patterns are not frequent. We observe that when joining two frequent patterns of length greater than one, the operation simply intersects their datasets.

We use the four properties for patterns to prune redundant patterns and reduce the search space. Assume there are two ID-pairs Xi<t(Xi)> and Xj<t(Xj)> in a joinable class [N], where Xi is a k-pattern and Xj is an m-pattern, k > 1 and m > 1, respectively. The four properties used to join them are described as follows:

Property 1 = If $t(X_i)= t(X_j)$, then replace X_i with $X_i \cup X_j$ and remove $X_j<(X_j)>$ from [N]
Property 2 = If $t(X_i)\ t(X_j)$, then replace X_i with $X_i \cup X_j$
Property 3 = If $t(X_i)\ t(X_j)$, then add $X_i \cup X_j <t(X_i \cup X_j)>$ to $[X_i]$ and remove $X_j<t(X_j)>$ from [N]
Property 4 = If $t(X_i) \neq t(X_j)$, then add Xi U $X_j<t(X_i \cup X_j)>$ to $[X_i]$

3.3 The IAR Miner algorithm

Our proposed algorithm, IAR Miner, consists of two phases. First, we scan the database to find all frequent items. For each frequent 1-pattern found, we convert the original transaction database into a set of domain attributes called a dataset. Then, we enumerate all patterns in a depth-first search manner. The IAR Miner algorithm is detailed in Fig. 2.
In step 1 of Fig. 2, the database is scanned to find all frequent 1-patterns, each of which is associated with a dataset. In step 2, a hash table HT is constructed for pruning unnecessary candidate 2-patterns. Then, all 1- patterns are added to the root class [{ }] in step 3. In step 4, starting from the root class [{ }], we recursively call the BFS procedure to generate all inter-transaction itemsets.
We adopt three techniques to speed up the mining process. First, in step 2 of Fig. 2, we use a hashing approach to check the corresponding bucket in the hash table HT before joining a pair of 1-patterns.

```
Algorithm: IAR Miner(D, minsup, maxspan)
Input: a transaction database D, minimum support minsup, and maximum span maxspan.
Output: all closed inter-transaction itemsets C.
(1) Scan the transaction database D to find all frequent 1-patterns, each of which is associated
with a datset.
(2) Construct a hash table HT;
(3) All 1-patterns found are added to the root class [{}];
(4) BFS([{}], C,minsup, maxspan);
(5) Return C;
```

Fig. 2.The IAR miner algorithm

The method is applied as follows. When we scan the transaction database to g*et all* frequent 1-patterns, we construct a hash table HT in which each bucket accumulates the supports of all 2-patterns hashing to it. Let {$x_i(0)$, $x_j(k)$} be a candidate 2-pattern read from the database.Then, the hash value of the 2-pattern, hv, is equal to ((x_i (maxspan+ 1) +k) $_j I_j$ + x_j) mod hashsize, where 0 ≤ k≤ maxspan, k is the span between xi and xj, jIj is the number of distinct items and hashsize is the size of the hash table. The support of HT[hv] is incremented by one. By scanning the database once, the HT can be constructed.

SHT, where its hash value is computed by shv (N i) = $\sum_{dat}$ ε t(Ni)(dat-dat1) is the first dat value in t(Ni). This hash value is the span sum of all dats in t(Ni). If the hash value and the support of Ni are equal to those of a frequent pattern Z in SHT, then (1) if Ni is a sub-pattern of Z, Ni is not and should not be added to SHT, or (2) if Ni is a super-pattern of Z, then Z is not . Thus, Z should be removed from SHT and Ni should be added to SHT. By doing so, we can avoid comparing Ni with all patterns in C and quickly retrieve related patterns.

4. Materials and method

To evaluate the performance of the proposed method, we compare it with the EH-Apriori (Palshikar *et al*., 2007) , FITI (Tung *et al.*, 2003),ClosedPROWL and ITP-Miner (Lu *et al*., 2000) algorithms.

Table 1:Description of parameters

D-Number of transactions
λt -Average length of the transactions
MaxT -Maximum length of the transactions
L - Number of potentially frequent inter-transaction itemsets
λf - Average length of potentially frequent inter-transaction itemsets
MaxI -Maximum length of potentially frequent inter-transaction itemsets
E -Number of distinct items
R -Maximum span

We introduces the experiment setup, synthetic data generations and presents the experimental results of synthetic datasets.

Table 2: Default value of two synthetic datasets

Parameters	Dataset 1	Dataset 2
D	100000	10000
λ_t	10	100
MaxT	10	200
L	1000	1000
λ_f	10	12
MaxI	10	10
E	500	500
R	6	10

4.1 Experiment setup and dataset description

To compare the proposed method with the EH-Apriori, FITI, ClosedPROWL and ITP-Miner algorithms, we conducted the experiments on synthetic datasets. All the experiments were performed on an Intel Core I5 Processor 3.4GHz, 4 GB memory, running on the Linux platform. All the algorithms were programmed using J2ME and compiled by gcc 4.1.1. The memory consumption was measured by using the GNU glibc function mallinfo, which can record the maximum heap memory used.

The algorithms are evaluated on synthetic datasets. First, the synthetic datasets, we use the method to generate transactions. The parameters are shown in Table 1. This process consists of two steps. We first generate potentially frequent inter-transaction itemsets and then generate transactions in the database from those itemsets. The length of each potentially frequent inter-trans- action itemset is derived from a Poisson distribution with mean = kf, while the length of each transaction is derived from a Poisson distribution with mean = kt.

Table 2 lists the parameters of two synthetic datasets used in the experiments. The first dataset has more transactions, but fewer items in each transaction. On the other hand, the second dataset has fewer transactions, but each transaction contains more items. We analyze the performance of the EH-Apriori, FITI, ClosedPROWL, ITP-Miner and IAR Miner algorithms by varying one parameter of the datasets and keeping the other parameters at the default values shown in Table 2.

5.Results and discussion

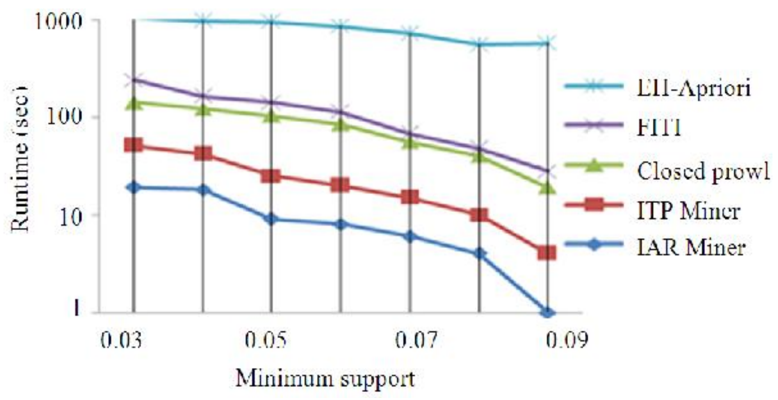
5.1 Basic experiments

Figure 3 shows the runtime versus the minimum support where the latter varies from 0.03% to 0.09% for Dataset1 and from 10-15% for Dataset2. Note that the minimum support is defined the fraction of mega-transactions containing the pattern in the database in the following experiments. As the minimum support increases, the runtime of the algorithms decreases because of the reduction in the total number of frequent itemsets. In Fig. 3, the IAR Miner runs 55-2500 times faster than the EH-Apriori, 3-231 times faster than the FITI, 2-120 times faster than the ClosedPROWL and 1-5 times faster than the ITP-Miner.

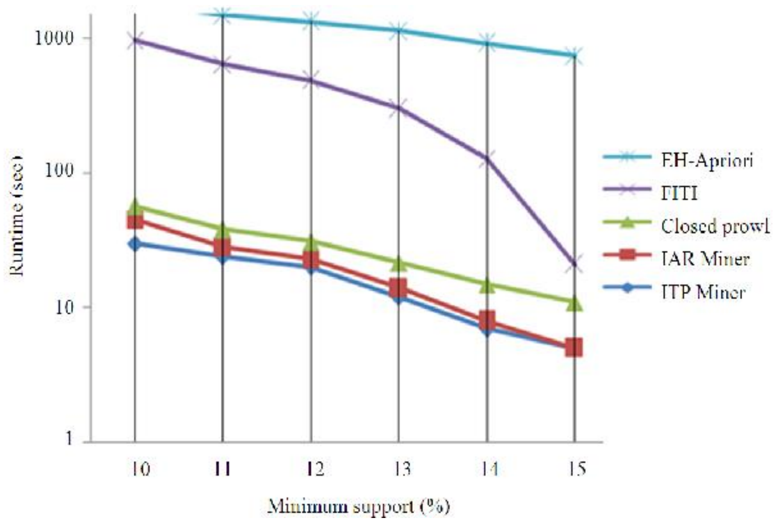
The IAR Miner algorithm outperforms the EH-Apriori, FITI, ClosedPROWL and ITP-Miner algorithms by order(s) of magnitude on both datasets. This is because the EH-Apriori generates a large number of candidates and needs to rescan the database and the FITI generates a large number of candidates and needs to rescan the FIT tables to count the support for all the candidates. Although the ClosedPROWL avoids generating a large number of candidates by scanning the transactions of the corresponding dataset within the maximum span for each frequent pattern to count the support, it cannot efficiently prune non-closed patterns. However, the IAR Miner does not need to rescan the database because it intersects and shifts the datasets for each newly generated pattern to speed up the mining process. Moreover, it can use the pruning strategies to remove many non-closed patterns, so it finds the inter-transaction itemsets more efficiently. Note that the IAR Miner outperforms the ITP-Miner is because when the minimum support decreases, the ratio of the number of closed patterns to the number of frequent patterns (called closed ratio) also decreases.

Figure 4 shows the memory usage as the minimum support increases from 0.03%-0.09% for Dataset1 and from 10%-15% for Dataset2. The IAR Miner requires 140-210 times less memory than the EH-Apriori and 50-55 times less memory than the FITI in both datasets. However, it requires more memory than the ITP-Miner in both datasets and slightly more memory than the ClosedPROWL in Dataset2.

The EH-Apriori consumes the largest amount of memory because it generates a huge number of candidates as the minimum support decreases. The FITI requires more memory than the IAR Miner and ClosedPROWL because it has to store the FILT structure, the FIT tables and a large number of candidates.

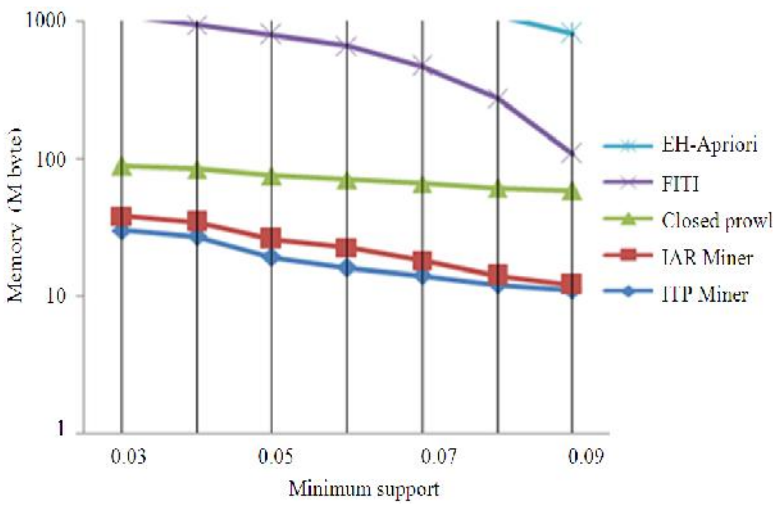


(a)

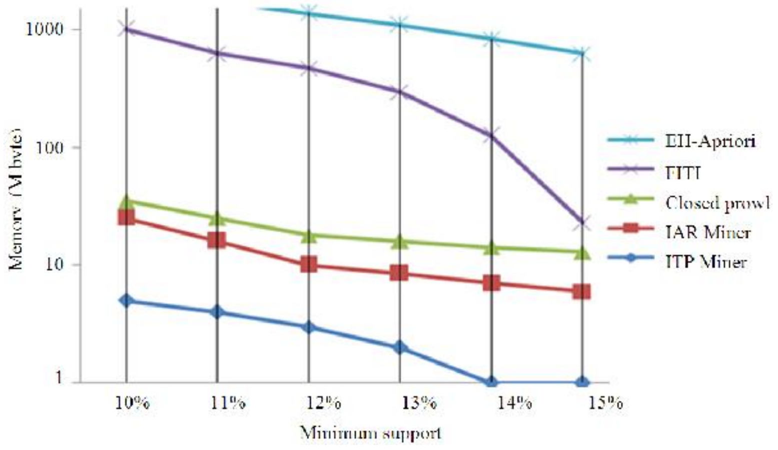


(b)

Fig. 3 Runtime versus minimum support (a) Dataset 1 with maxspan = 3 (b) Dataset 1 with maxspan = 5

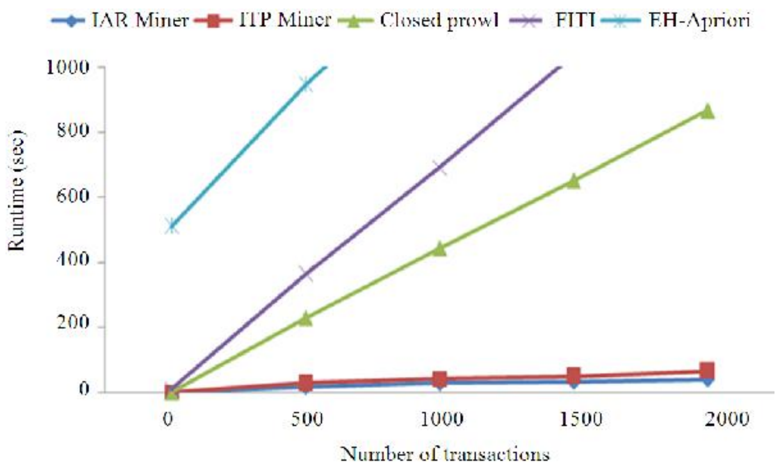


(a)

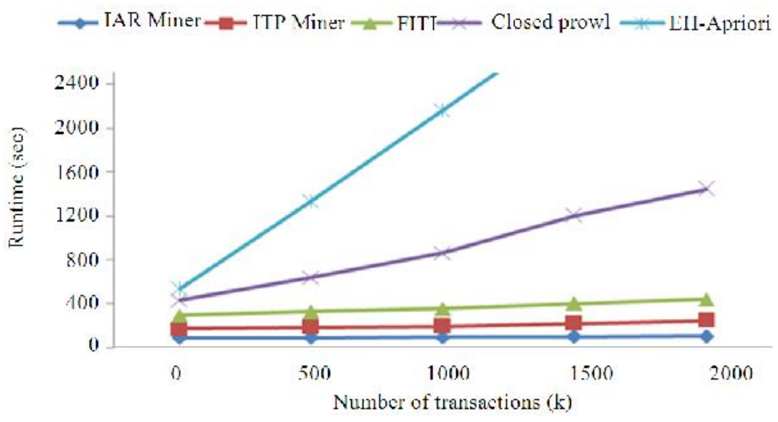


(b)

Fig 4: Memory usage versus minimum support (a) Dataset1 with maxspan = 3 (b) Dataset2 with maxspan = 5

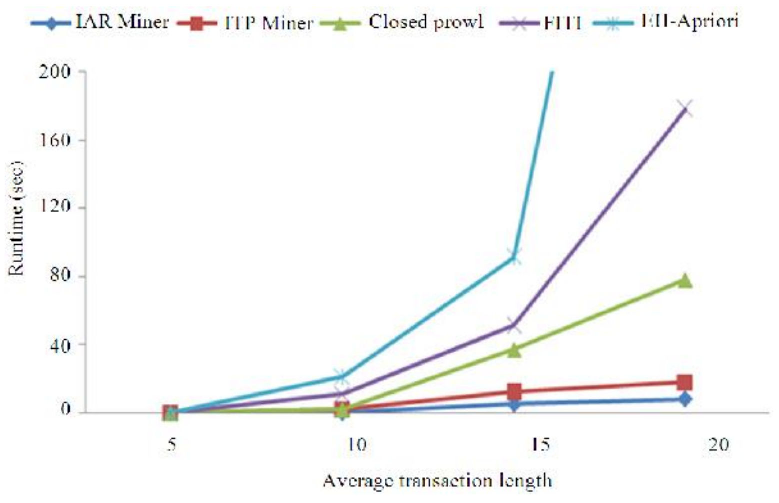


(a)

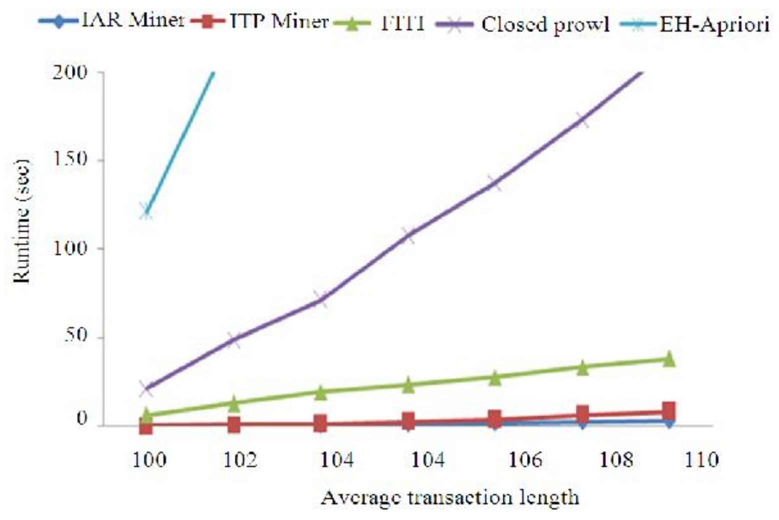


(b)

Fig 5: Scale up: Number of Transactions (a) Dataset1 with minsup = 0.05% and maxspan = 3 (b) Dataset2 with minsup =13% and maxspan = 5

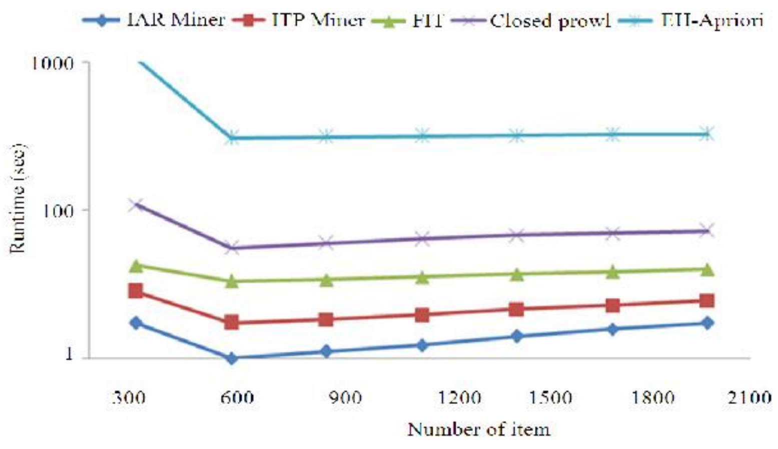


(a)

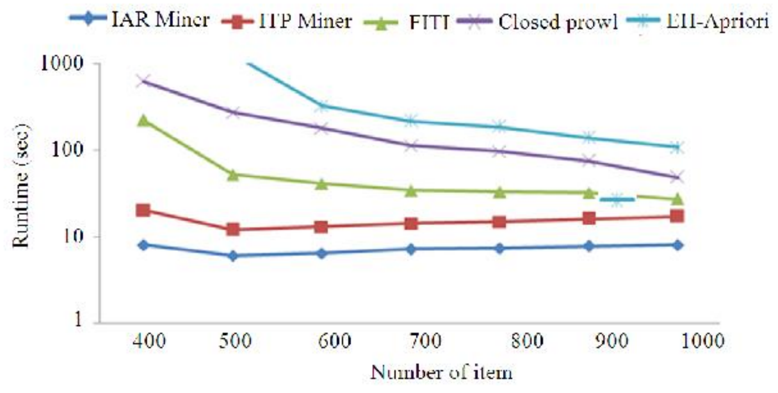


(b)

Fig 6: Scale up: Average length of transactions (a) Dataset1 with minsup = 0.05% and maxspan = 3 (b) Dataset2 with minsup = 13% and maxspan = 5



(a)



(b)

Fig 7: Scale up: Number of items (a) Dataset1 with minsup = 0.05% and maxspan = 3 (b) Dataset2 with minsup = 13% and maxspan = 5

Thus, as the minimum support decreases, the memory usage increases rapidly. However, the ClosedPROWL uses a data structure to store the intra-transaction itemsets and datasets and a compressed table to store the transformed database.

The IAR Miner, on the other hand, uses an ID-tree to store the inter-transaction itemsets and datasets. The memory usage of the IAR Miner and the ClosedPROWL increases slowly as the minimum support decreases. Besides, we have noted that the IAR Miner requires more memory than the ITP-Miner in all the cases. This is because the IAR Miner uses a hash table for the subsumption checking that requires extra memory space to store candidate inter-transaction patterns.

5.2 Scale-up experiments

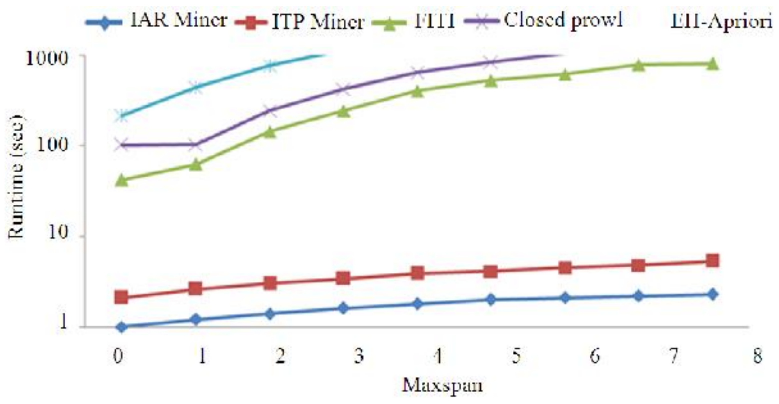
First, we investigate the scalability as the number of transactions increases. Figure 5 shows the runtime versus the number of transactions as the number of transactions increases from 10 K -2000 K [1] for Dataset1 and from 1 K-20 K [1] for Dataset2. Basically, the number of frequent patterns generated is not affected by the number of transactions. However, the EH-Apriori takes time to rescan the database, the FITI takes time to rescan the large transformed database, while the ClosedPROWL takes time to scan the transactions of the large dataset within the maximum span for each frequent itemset. Meanwhile, the IAR Miner and the ITP-Miner spend time in processing the large datasets. Thus, as the number of transactions increases, the runtime of each algorithm increases linearly.

Next, we investigate the scalability as the average length of transactions increases. Figure 6 shows the runtime versus the average length of transactions when the average length of transactions varies from 5-20 for Dataset1 and from 100-110 for Dataset 2. When the average length of transactions increases, extra frequent itemsets are generated. We observe that the IAR Miner algorithm is more efficient than the EH-Apriori, FITI, ClosedPROWL and ITP-Miner algorithms.

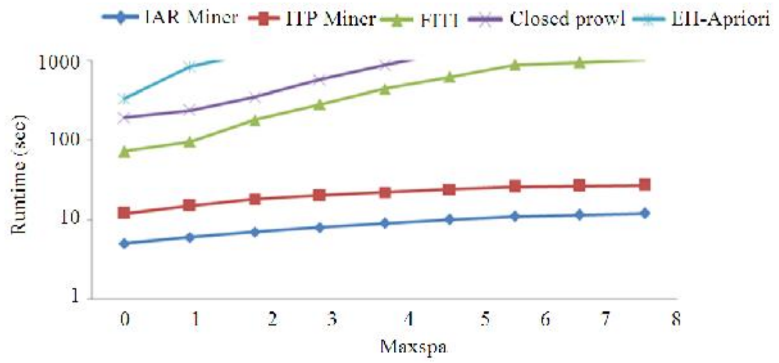
Finally, we examine the effect of increasing the number of items. Figure 7 illustrates the runtime versus the number of items when the number of items changes from 300-2100 for Dataset1and from 400-1000 for Dataset2. We can see that the runtime of the FITI and the ClosedPROWL drastically increases as far as the number of items decreases. The reason is that both the support of each candidate and the number of frequent patterns increase when the number of items decreases. Besides, we can also see that the gap of runtime between the IAR Miner and the ITP-Miner increases as the number of items decreases. This is because the closed ratio decreases when the number of items decreases.

5.3 Effect of the maximum span

Next, we examine the effect of the maximum span. Figure 8 illustrates the runtime versus the maximum span when the latter increases from 0-8. Clearly, the number of frequent itemsets grows as the maximum span increases. Figure 8 shows that the IAR Miner algorithm is more efficient than the EH-Apriori, FITI, Closed-PROWL and ITP-Miner algorithms.



(a)



(b)

Fig 8: Effect of the Maximum span (a) Dataset1 with minsup = 0.05% (b) Dataset1 with minsup = 11%

6.Conclusion

We have proposed an efficient algorithm, called IAR Miner, for mining inter-transaction itemsets. By using the ID-tree, we can efficiently mine the inter-transaction patterns. Moreover, we can avoid generating many frequent but non-closed patterns by using pruning strategies for patterns. Thus, our proposed algorithm can efficiently mine inter-transaction patterns. The performance study on the synthetic datasets shows that the IAR Miner algorithm is more efficient than the EH-Apriori, FITI, ClosedPROWL and ITP-Miner algorithms in most cases.

We will aim to address a number of research issues related to the IAR Miner algorithm. First, we can extend our algorithms to mine the generalized inter-transaction association rules (Li *et al*., 2005), which can expand rule contexts from point-wise (e.g., two days latter) to scope-wise (e.g., within three days). Second, in the future, we may develop efficient method to mine other concise representations of frequent patterns, such as non-derivable itemsets (Calders and Goethals, 2002), essential itemsets. Therefore, we may extend the IAR Miner algorithm to keep track of the set of patterns and their associated minimal generators during the mining process. Third, we will apply our proposed algorithm to other domains, such as marketing data and web access logs, to assess its usability. Finally, without generalization, too many patterns may be mined and they may be too detailed. However, by generalizing with a concept hierarchy, we may be able to obtain patterns or rules that are more abstract and meaningful.

References:

Agrawal, R. and R. Srikant, 1994. Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB '94), Santiago, Chile, pp. 487-499.

Bastide. Y., N. Pasquier, R. Taouil, L. Lakhal and G. Stumme, 2000. Mining minimal non-redundant association rules using frequent closed itemsets. Proceedings of the 1st International Conference Computational Logic, (CL' 00), Springer-Verlag London, UK., pp: 972-986.

Berzal. F., J.C. Cubero, N. Marın and J.M. Serrano, 2001. TBAR: an efficient method for association rule mining in relational databases. Data Knowl. Eng., 37: 47-64.

Calders. T. and B. Goethals, 2002. Mining all non-derivable frequent itemsets. Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, (PDMKD' 02), Springer-Verlag, London, UK, pp: 74-85.

Chen. J., L. Ou, J. Yin and J. Huang, 2005. Effcient mining of cross-transaction web usage patterns in large database. Proceedings of the 3rd International Conference on Networking and Mobile Computing, (NMC' 05) Springer-Verlag Berlin, Heidelberg, pp: 519-528.

Chen. T.S. and S.C. Hsu, 2007. Mining frequent tree-like patterns in large datasets. Data Knowl. Eng., 62: 65-83.

Feng. L., J.X. Yu, H. Lu and J.A. Han, 2002. A Template model for multidimensional inter-transactional association rules. VLDB J., 11: 153-175.

Han. J., J. Wang, Y. Lu and P. Tzvetkov, 2002. Mining top-k frequent closed patterns without minimum support. Proceedings of the 2002 IEEE International Conference on Data Mining, (ICDM' 02), IEEE Xplore Press, Maebashi, Japan, pp: 211-218.

Han. J. and Y. Fu, 1995. Discovery of multiple-level association rules from large databases. Proceedings of the 21th International Conference on Very Large Data Bases, (VLDB' 95), Francisco, CA, USA, pp: 420-431.

Jea. K.F. and M.Y. Chang, 2008. Discovering frequent itemsets by support approximation and itemset clustering. Data Knowl. Eng., 65: 90-107.

Lee, A.J.T., R.W. Hong, W.M. Ko, W.K. Tsao and H.H. Lin, 2007. Mining spatial association rules in image databases. Inform. Sci., 1593-1608.

Lee, A.J.T., W.C. Lin and C.S. Wang, 2006. Mining association rules with multi-dimensional constraints. J. Syst. Software 79: 79-92.

Lee, A.J.T and C.S. Wang, 2007. An efficient algorithm for mining frequent inter-transaction patterns. Inform. Sci., 3453-3476.

Li, Q., L. Feng, A. Wong, 2005. From intra-transaction to generalized inter-transaction: landscaping multidimensional contexts in association rule mining. Inform. Sci., 172: 361-395.

Lu, H., L. Feng and J. Han, 2000. Beyond intratransaction association analysis: mining multidimensional inter-transaction association rules. ACM Trans. Inform. Syst., 18: 423-454.

Lu, H.S., G. West and S. Venkatesh, 2005. An extended frequent pattern tree for intertransactionassociation rule mining. Curtin University of Technology, Perth, Western Australia.

Masseglia. F., P. Poncelet and M. Teisseire, 2003. Incremental mining of sequential patterns in large databases. Data Knowl. Eng., 46: 97-121.

Palshikar. G.K., M.S. Kale and M.M. Apte, 2007. Association rules mining using heavy itemsets. Data Knowl. Eng., 93-113.

Park, J.S., M.S. Chen, P.S. Yu, 1995. An effective hash-based algorithm for mining association rules. Proceedings of the ACM-SIGMOD International Conference on Management of Data, (ICMD’ 95), ACM Press New York, NY, USA, pp: 175-186.

Pei. J., J. Han and R. Mao, 2000. Closet: An efficient algorithm for mining frequent closed itemsets. Proceedings of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (RIDMKD’ 00), ACM, Dallas, TX, USA, , pp. 11-20.

Tseng, M.C. and W.Y. Lin, 2007. Efficient mining of generalized association rules with non-uniform minimum support. Data Knowl. Eng.,

Tung, A.K.H., H. Lu, J. Han and L. Feng, 2003. Efficient mining of intertransaction association rules. IEEE Trans. Knowl. Data Eng., 15: 43-56.

Yao, H. and H.J. Hamilton, 2006. Mining itemset utilities from transaction databases. Data Knowl. Eng.,