# Efficiently Mining Association Rules from Time Series

Liang-Xi Qin [1, 2, 3], Zhong-Zhi Shi [1]

[1] (Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)
[2] (Graduate School of Chinese Academy of Sciences, Beijing 100049)
[3] (College of Computer, Electronics and Information, Guangxi University, Nanning 530004)

{qinlx, shizz}@ics.ict.ac.cn

**Abstract**

Traditional association rules are mainly concerned about intra-transactional rules. In time series analysis, intra-transactional association rules can only reveal the correlations of multiple time series at same time. It is difficult to forecast the trend of time series. In this paper, it is studied the mining problem of inter-transactional association rules in time series with which the trend can be forecast by the time difference between the prerequisite and the consequent in a rule. A new algorithm, ITARM, for inter-transactional association rules mining is presented. It uses a compact FP-tree based and divide-and-conquer approach. After the frequent 1-itemsets is produced, the algorithm separately uses them as constraint conditions to construct compact FP-tree and mine inter-transactional association rules. It is presented that the main idea and the pseudo-code of ITARM. A performance test is done for the algorithm, the results show that ITARM is an algorithm with high temporal and spatial performances.

**Keywords**: data mining; time series; inter-transactional association rules; compact FP-tree.

## I. Introduction

Association rules mining is an important subject in the study of data mining. This problem is introduced by R. Agrawal, et al.[1] in 1993. Since then, it has been widely used in business, scientific research and management of enterprise, etc. Many researchers have studied the mining of one dimensional boolean association rules, multi-level association rules, multi-dimensional association rules and sequential patterns, but the traditional association rules are mainly concerned about the rules within same transaction or same sequence, the intra-transactional association rules.

A time series data set consists of sequences of values or events that change with time. Time series data is popular in many applications, such as the daily closing prices of a share in a stock market, the daily temperature value recorded at equal time intervals, and so on. According to the number of involved time series, the association rules mining problem can be divided into two categories: association rules mining from single series and that from multiple series. The association rules mining problem from single series cab be view as the mining problem of sequential patterns, there have been a lot of studied about it. If we take the values of different series at same time as a transaction, the association rules mining problem from multiple series can be divided into two types: (1) intra-transactional association rules mining, and (2) inter-transactional association rules mining.

We can use the mining algorithms for traditional association rules to mine intra-transactional association rules from time series. However, the intra-transactional association rules can only reveal the co-relations of multiple time series at same time. It has little contribution to prediction, the ultimate destination of time series analysis. Taking stock data as an example, it can be found the rules like "If stock A goes up and stock B goes up then stock C will goes up on the same day (5%, 80%)" with intra-transactional association rules, but it can not be found the rules like "If stock A goes up on the first day and stock B goes up on the second day then stock C will goes up on the third day (5%, 80%)". There is no time difference between the items in the intra-transactional association rules, so it can not be used to predict the trend of time series.

The inter-transactional association rules describe the relations between different transactions. If we use it in multiple time series analysis, it can be analyzed the relations of different series at different time. So, it can be used to find the rules like "If stock A goes up on the first day and stock B goes up on the second day then stock C will goes up on the third day (5%, 80%)".

It is evident that in the example above the time dimension is added, the events in the rule occur in different time. They have sequential order in time dimension. If in an association rule "$X(0) \Rightarrow Y(2)$", event Y occurs two days after event X, then the rule will have the ability of prediction. According to this rule, while event X occurs some day, we can infer that event Y will most likely occur two days after.

The inter-transactional association rules can be viewed as an extension of traditional intra-transactional association rules, and traditional intra-transactional association rules is a special case of inter-transactional association rules. The mining of inter-transactional association rules is much complex than traditional association rules, it challenges the efficiency of mining algorithms.

It is simply introduced the concept of time series, and compared the differences between inter-transactional association rules and traditional ones. The remainder of this paper is arranged as follows. Related works are introduced in section 2. Section 3 gives a description of the problem for mining inter-transactional association rules of time series. Section 4 introduces the main idea of the ITARM algorithm, and presents the pseudo-code of the algorithm. Section 5 presents the experimental results of performance test. And the last is the conclusion of this paper.

## II. Related works

The Apriori algorithm[2] proposed by Agrawal et al. is a classical algorithm for association rules mining. The name of the algorithm comes after a prior knowledge about frequent itemsets was used. The prior knowledge is that any non-empty subset of a frequent itemset is also frequent. Apriori algorithm uses a level-wised and iterative approach, it first generates the candidates then test them to delete the non-frequent itemsets. Most of previous studies adopted an Apriori-like candidates generation-and-test approach.

Han et al. developed the FP-growth algorithm[3] that is based on frequent pattern tree. Comparing with Apriori algorithm, this algorithm has following features. (1) It uses FP-tree to store the main information of the database. The algorithm scans the database only twice, avoids multiple database scans and reduces I/O time. (2) It does not need to generate candidates, reduces the large amount of time that is consumed in candidates generation and test. (3) It uses a divide-and-conquer approach in the mining process, so the searching space is significantly decreased. The efficiency of the FP-growth algorithm is about an order of magnitude faster than the Apriori algorithm.

However, there still exist some aspects in FP-growth algorithm that can be improved. For example, it needs to recursively generate huge number of conditional FP-trees that consumes much more memory and more time. It has appeared several improved FP-growth algorithms based on original one. Fan and Li [4] presented a constrained subtree based approach to avoid the generation of huge number of conditional FP-trees recursively in the mining process. They also reduced the fields in each FP-tree node. Their approach has better time and space scalability than FP-growth algorithm. Grahne and Zhu [5] proposed an array-based technique to reduce the time cost in FP-tree traverse.

They also implemented their own memory management for allocating and deallocating tree nodes. Qin et al. combined several advanced techniques above, presented a compact FP-tree based frequent patterns mining algorithm, CFPmine [6]. It uses constrained subtrees of a compact FP-tree to mine frequent pattern, so that it is doesn't need to construct conditional FP-trees in the mining process, so the memory cost is reduecd. It also uses an array-based technique to reduce the time on traverse of the CFP-tree, the efficiency is improved.

Agrawal et al. [7] studied the sequential patterns mining problem, and proposed the well known GSP algorithm[8]. Lu et al.[9][10] introduced multidimensional inter-transactional association rules mining problem, and presented two algorithms: E-Apriori and EH-Apriori. They are both Apriori-like algorithms. It is used the hashing technique for counting 2-itemsets in EH-Apriori, so the efficiency of EH-Apriori is higher than E-Apriori. Tung et al. [11] presented an algorithm, FITI (Fist Intra Then Inter), in which first mine the intra-transactional association rules, and then mine the inter-transactional association rules. Dong et al.[12] presented the ES-Apriori algorithm in which a divide-and-conquer approach is adopted for mining inter-transactional association rules in multiple time series step by step.

## III. Problem description

**Definition 1.** let $S=\{s_1, s_2,..., s_u\}$ be a set of time series, $T_i$ be the values of set S at time $i$, $T_i=\{s_1(i),s_2(i),...s_u(i)\}(1 \leqslant i \leqslant n)$, the union set of multiple time series D is defined as follows: $D=\{T_1,T_2,...,T_n\}$ . Each group of values in $D$ is a transaction, it has an identifier *TID*.

**Definition 2.** let $\Sigma_1=\{e_1, e_2 , ...,e_u \}$ be a set of events. They are attributes of time series $S$. $\Sigma =\{e_1(0),..., e_1(w-1), e_2(0),..., e_2(w- 1) , ..., e_u(0),..., e_u(w-1) \}$ be the set of possible extension of $\Sigma_1$. $w$ be the sliding window in $D$. Taking time spot $s$ $(1 \leqslant s \leqslant n-w+1)$ as a reference time spot, if $e_i$ occurs at time $s+x$ $(0 \leqslant x \leqslant w-1)$, then we mark $e_i(x)$ belongs to $T_s$. Each $e_i(x)$ is given an identifier, *IID*.

**Definition 3.** An inter-transactional association rule in multiple time series is an implication of the form "$X \Rightarrow Y$", and it satisfies the following conditions:

(1) $X \subset \Sigma, Y \subset \Sigma,\ X \cap Y = \Phi$

(2) $\exists e_i(0) \in X,\ 1 \leqslant i \leqslant u,$

(3) $\exists e_j(q) \in X,\ 1 \leqslant j \leqslant u, ((i=j) \wedge (1 \leqslant q < w-1)) \vee ((i \neq j) \wedge (0 \leqslant q < w-1)),$

(4) $\exists e_i(p) \in Y,\ 1 \leqslant i \leqslant u,\ max(q) < p \leqslant w-1.$

Let n be the number of transactions, $C_{XY}$ be the times that $X \cup Y$ appears in database, $C_X$ be the times that X appears in database, then the support and confidence of inter-transactional association rule can be defined as follows:

support = $C_{XY} / n$ , confidence = $C_{XY} / C_X$ .

**Definition 4.** An intra-transactional itemset is a set in which all items are from same transaction $T_i$. And an inter-transactional itemset is a set in which the items are from multiple transactions.

For inter-transactional itemsets, it has following lemmas about them.

**Lemmas 1.** Let F be an inter-transactional itemset, $A_i = \{ e_j \mid 1 < j < u, e_j(i) \in F \}$, there into $0 \leqslant i \leqslant (w-1)$. For all i, $0 \leqslant i \leqslant (w-1)$, $A_i$ must be an intra-transactional itemset.

Lemmas 1 is quite important, it means that we can regard an inter-transactional itemset as a combination of several intra-transactional itemsets, so that we can use a series of intra-transactional itemsets mining steps to mine inter-transactional itemsets.

Inter-transactional association rule is different from traditional association rule. It is also different from sequential pattern. Comparing to that for traditional association rules, the mining algorithm for inter-transactional association rules is more complex.

While the amount of data increases gradually, the frequent itemsets of inter-transactional association rule will become larger and larger, and hard to handle. In the inter-transactional analysis, because the sliding window is added, each itemset $\beta$ will extend to $\beta(0), \beta(1), ..., \beta(w)$, there into $w$ is the size of sliding window. Let there are 1000 frequent 1-itemsets, it will generate at most $\binom{1000}{2} = \frac{1000*999}{2} = 499500$ candidates of frequent 2-itemsets in traditional association rules, but it will generate at most $\binom{1000*w}{2} = \frac{1000*w*(1000*w-1)}{2}$ candidates in inter-transactional association rules. It grows with the times of $w^2$. If $w$=3, there will be 4498500 candidates, it is 9 times of traditional ones. And it will even worse while construct frequent 3-itemsets.

FP-growth algorithm has high performance to intra-transactional association rules, can it be used to mine inter-transactional association rules? From the analysis above, we can see that the frequent itemsets of inter-transactional association rules is more complex than those of intra-transactional association rules. If we use FP-growth directly in the mining of inter-transactional association rules, the FP-tree will become too large to be saved in the main memory. So, we must use the divide-and-conquer method to decompose the mining task to small ones. Based on the CFPmine algorithm introduced above, we proposed an algorithm ITARM (Inter-Transaction Association Rule Mining) in which compact FP-tree (CFP-tree) based and divide-and-conquer mining method is adopted.

## IV. Implementation of inter-transactional association rules mining

### 4.1 ITARM algorithm

The main idea of ITARM is based on CFPmine algorithm, and a decomposing approach to the mining task is adopted. While the frequent 1-itemsets are generated, they are used as constrained conditions separately to acquire constrained frequent 1-itemsets. Then using the constrained frequent 1-itemsets to construct a CFP-tree, mining the tree with CFPmine algorithm, thus we can get all frequent itemsets and association rules constrained by some itemset. The union of all constrained association rules composes the complete set of association rules. The process of ITARM is as follows.

1. First of all, reading all data from database and store them in an array. One dimension of the array represents different series, and another dimension represents time. Each element of the array stores the value of some series at some time spot. Because the data is stored in the main memory, thereafter the support counting of frequent itemsets does not need to scan the database, thus the I/O cost can be reduced. If the amount of data is much large, it can be also stored in a disk file, but the I/O cost will be increased.

2. The mining process is divided into two steps. First step is to find all intra-transactional frequent 1-itemsets that satisfied the minimum support threshold. Second step is, on the basis of first step, to mine inter-transactional frequent itemsets and association rules.

3. The second step adopts divide-and-conquer approach. For all $e_i(0)$ in frequent 1-itemsets ($1 \le i \le$ size of 1-itemsets), execute the following operations.

(3.1) Find out while $e_i(0)$ is occuring, all $e_i(0)$ constrained frequent 1-itemsets $F1_i$ (if $e_i(0)$ is added, they can be view as 2-itemsets ) in the sliding window.

(3.2) Sort $F1_i$ according to the order of ($e_{i+1}(0)$, ..., $e_u(0)$, $e_1(1)$, ..., $e_u(1)$, ..., $e_1(w\text{-}1)$, ..., $e_u(w\text{-}1)$), let it be $SF1_i$.

(3.3) Scan the data set D, take each sliding window as a transaction, find out all items in $SF1_i$, and construct a CFP-tree. The main differences between CFP-tree and FP-tree are as follows. 1) There are 6 fields in each node of FP-tree, while there are only 4 fields in each node of CFP-tree. So CFP-tree need only 2/3 memory space of FP-tree. 2) The nodes in FP-tree are unordered, while CFP-tree

is sorted according to item-no ascending order. 3) FP-tree is bi-directional, but CFP-tree is single directional. After the construction, CFP-tree only has paths from leaves to the root.

(3.4) Call CFPmine algorithm to mine CFP-tree. After the mining is completed, then output all frequent itemsets or association rules begin with $e_i(0)$ at once. Delete the CFP-tree, i+1, and goes to step 3 to mine the next item $e_{i+1}(0)$.

**Algorithm 1** ITARM algorithm
**Input:** the union set of multiple time series D, minimum support threshold min_sup, minimum confidence threshold min_conf, sliding windows size w
**Output:** the inter-transactional association rules in D
**Method**: Do the following program.

**Phase 1**

(1) $C_1=\{\{ e_i(x)\} | (e_i(x)\in\sum)\wedge(0\le x\le w-1)\}$
(2) for each inter-time series transaction $T_s$ in $D$
(3) for each candidate $c$: $e_i(x)\in C_1$ $(e_i(x)\in T_{s+x})$
(4) $c$.count ++;
(5) $L_1=\{c:\{ e_i(x) \}|(c\in C_1)\wedge(c.\text{count}\ge support)\}$

**Phase 2**

(6) for each item: $e_i(0)\in L_1$ {
(7) $C_2'=\{\{e_i(0),e_k(x)\}|e_k(x)\in L_1\ ((x\neq 0)\vee(x=0\wedge i<k))\}$
(8) for each candidate $c\in C_2'$: $\{e_i(0),e_k(x)\}$ {
(9) c.count++; }
(10) $L'_2=\{c:\{e_i(0),e_k(x)\}|(c\in C_2')\wedge(c.count\ge min\_sup)\}$ // $L'_2$ is $F1_i$
(11) Sort $L'_2$
(12) Scan data set D and construct CFP-tree
(13) Call CFP-tree mining algorithm CFPmine to generate all frequent itemsets begin with $e_i(0)$
(14) Generate association rules from frequent itemsets and output them.
(15) Delete CFP-tree.
(16) }

### 4.2 CFPmine algorithm

In ITARM algorithm, the mining of CFP-tree is completed by CFPmine. In CFPmine, a constrained subtrees based mining approach is adopted, and an array technique is used to reduce the traverse times of CFP-tree. For reduces the memory allocating and deallocating time, a unified memory management technique is also adopted.

#### 4.2.1 Constrained subtree

In the mining process, FP-growth algorithm must recursively generate huge number of conditional FP-trees that requires much more memory and costs more time. In CFPmine algorithm, we use a constrained subtrees approach to mine frequent pattern directly, it does not need to generate conditional FP-trees. The follows is the related definition of constrained subtrees.

**Definition 5.** Let $i_1\prec i_2\prec\ldots\prec i_k$ be item orders. N is a node in the CFP-tree. P is a sub-path from root to N. We say P is constrained by the itemset $\{i_1,i_2,\ldots,i_k\}$, if exist a N's descendant node M, such that $i_1,i_2,\ldots,i_k$ appear in the sub-path from N to M, and $i_1$ is the item-no of N's child, $i_k=$

M.item-no. N is called end node of sub-path P. Node M's count c is called the base count of constrained sub-path P.

**Definition 6.** In a CFP-tree, all of the sub-path constrained by itemset $\{i_1, i_2, ..., i_k\}$ make up of a subtree, it is called a subtree constrained by $\{i_1, i_2, ..., i_k\}$, nominated as ST($i_k, ..., i_2, i_1$).

ST($i_k, ..., i_2, i_1$) can be represented by an array, let it be EndArray, that every element of it has two fields: end-ptr (point to the end node) and base-count(store the base count of the constrained sub-path). The frequent items and count of ST($i_k, ..., i_2, i_1$) can be represented by ST($i_k, ..., i_2, i_1$).fitem[] and ST($i_k, ..., i_2, i_1$).count[] respectively.

### 4.2.2 The array technique

In the mining process of constrained subtree-based approach, the main works are recursively traverse a constrained subtree and generate a new subtree. For each frequent item k in constrained subtree ST(X), two traverses of ST(X) are needed for generate frequent items and counts of ST(X, k). For reducing the traverse time, we adopt an array technique. By means of this technique, while the constrained subtree ST(X) is constructed, the count of any two items in frequent itemset is calculated, so one traverse to the constrained subtree can be reduced.

### 4.2.3 Unified memory management

Because the CFP-tree could have millions of nodes, thus, it takes plenty of time for allocating and deallocating the nodes. Just like in [5], we also implement unified memory management for CFPmine algorithm. In the recursive mining process, the memory used in it is not frequently allocated and freed. It allocating a large chunk before the first recursion, and when a recursion ends, it doesn't really free the memory, only changes the available size of chunk. If the chunk is used up, it allocates more memory. And it frees the memory only when all the recursions have been finished. By the use of unified memory management, the time for allocating and deallocating memory is significantly reduced.

The following is the pseudocode of CFPmine. CFPmine is a main procedure, it output frequent 1-itemset and generate constrained subtree that has only one constraint item, then call mine procedure to generate frequent itemsets that have more than one item. The most work of mining is done by mine procedure.

**Algorithm 2.** CFPmine algorithm
**Input:** constructed CFP-tree and min_sup
**Output:** all frequent itemsets and the support
**Method:** call CFPmine (Tree)

```
Procedure CFPmine(T) {
        (1)  patlen=1;
        (2)  for(k=flen-1;k>=0;k--){// flen is the size of
                                                the frequent itemset
        (3)     pat[0]=fitem[k];
        (4)     output { pat[0] } with support count[k];
        (5)     generate ST(k).EndArray[];
        (6)     mine(ST(k)); }}

Procedure mine(ST(i_k,...,i_2,i_1)) {
(1)       generate ST(i_k,...,i_2,i_1).fitem[] and count[],
                let the size of fitem[] be listlen;
```

(2) if (listlen==0 ) then return;
(3) if (listlen==1) then {
pat[patlen]= ST($i_k$,...,$i_2$,$i_1$ ).fitem[0];
output pat with support ST($i_k$,...,$i_2$,$i_1$ ).count[0]; return; }
(4) if ST($i_k$,...,$i_2$,$i_1$ ) has only single path then
{ output pat ∪ all the combination of
ST($i_k$,...,$i_2$,$i_1$ ).fitem[]; return; }
(5) patlen++;
(6) for (k=listlen-1; k>=0; k--) {
(7) generate array;
(8) generate ST($i_k$,...,$i_2$,$i_1$,$k$ ).EndArray[];
(9) if ST($i_k$,...,$i_2$,$i_1$,$k$ ).EndArray[] is not NULL then
mine(ST($i_k$,...,$i_2$,$i_1$,$k$ )); }
(10) patlen--; }

In mine procedure, line 1 generate frequent itemset in the constrained subtree ST($i_k$,...,$i_2$,$i_1$ ). Line 2~3 process the condition while listlen is 0 or 1. Line 4 process the condition while constrained subtree has only single path. Line 6~10 generate new array and constrained subtree, then mine the new subtree.

## V. Performance test

For testing the performance of ITARM algorithm, we use the data of 500 shares from Chinese Stock market as test data. The test data is between 1997 and 2001, totally 1125 days. In the test we compare ITARM algorithm with E-Apriori algorithm. The test environment is Intel P4 CPU 1.7G, 512M main memory, Windows XP operating system. The variation range is divided into two parts, the maximal length of frequent itemset is set to 6, sliding window size is 3, and minimum support threshold is 1%. The data set with 5k to 30k items is used separately to test the two algorithms. The results are shown in figure 1.

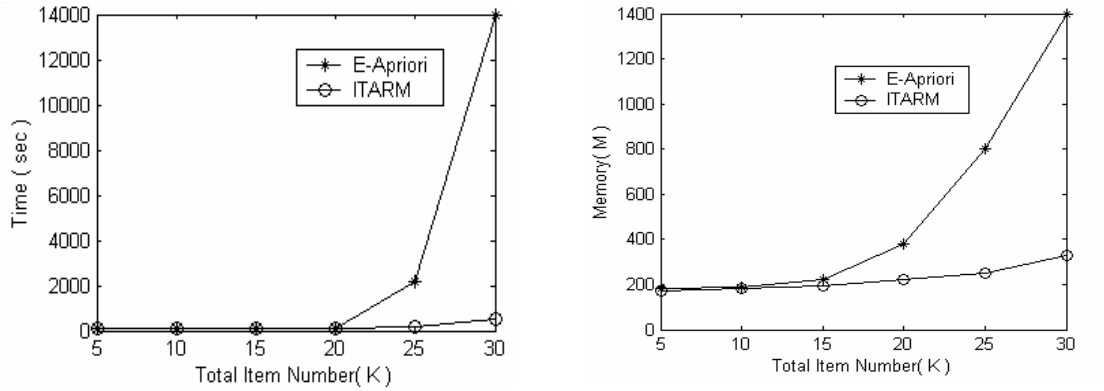


**Fig. 1.** The compare of ITARM and E-Apriori on temporal/spatial performances

From the figure we can see that while the data amount is increasing, the time and memory cost in E-Apriori is increasing rapidly, but that in ITARM is increasing is much slower. When the data amount is larger than 20K bytes, the memory required in E-Apriori is exceeding 512M main memory gradually and must use the virtual memory in the disks. It costs a great deal of additional I/O time, and the efficiency of itself is not as high as ITARM, so that the time consuming in E-

Apriori is increasing rapidly. The memory cost in ITARM is much less and not exceeds the capability of main memory, in addition to that the efficiency of the algorithm is much higher, so that the time consuming in ITARM is increasing slowly. From the analysis above, we can see that the ITARM has much higher performances in time and space than E-Apriori.

## VI. Conclusion

Traditional association rules are mainly concerned about intra-transactional rules. In time series analysis, intra-transactional association rules can only reveal the correlations of multiple time series at same time. It is difficult to forecast the trend of time series. In this paper, it is studied the inter-transactional association rules of multiple time series. We analyzed the problems existed in current algorithms. And a new algorithm for inter-transactional association rules mining, ITARM, is presented. It uses a compact FP-tree based and divide-and-conquer approach. After the frequent 1-itemsets is produced, the algorithm separately uses them as constraint conditions to construct compact FP-tree and mine inter-transactional association rules. It is introduced that the main idea and the pseudo-code of ITARM algorithm, and a performance test is done for the algorithm. The experimental results show that ITARM has higher temporal and spatial performances than E-Apriori.

## References


[1] R. Agrawal, T. Imielinski, A.Swami, "Mining association rules between sets of items in large database". In: P. Buneman, S. Jajodia eds. *Proc. of 1993 ACM SIGMOD Conf on Management of Data.* Washington DC: ACM Press, 1993. pp. 207-216

[2] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules". In: J.Bocca, M.Jarke, C.Zaniolo eds. *Proc. of the 20th Int'l Conf on Very Large DataBases* (VLDB'94). Santiago: Morgan Kaufmann, 1994. pp. 487-499

[3] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation". In: M. Dunham, J. Naughton, W. Chen eds. *Proc. of 2000 ACM-SIGMOD Int'l Conf on Management of Data* (SIGMOD'00). Dallas, TX, New York: ACM Press, 2000. pp. 1-12.

[4] M. Fan. and C.Li, "Mining frequent patterns in an FP-tree without conditional FP-tree generation". *Journal of computer research and development*, 2003, 40(8). pp. 1216-1222.

[5] G. Grahne, and J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets". In: *First Workshop on Frequent Itemset Mining Implementation* (FIMI'03). Melbourne, FL.

[6] L. Qin, P. Luo, Z. Shi, "Efficiently mining frequent itemsets with compact FP-tree". In: Z.Shi and Q.He eds. *Proc. of Int'l Conf. on Intelligent Information Processing 2004*(IIP2004), Beijing, China. Springer Press, 2004. pp. 397-406.

[7] R. Agrawal, R. Srikant, "Mining sequential patterns". In: *Proc. of ICDE'95*, Taipei, Taiwan. pp. 3-14.

[8] R. Srikant, R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements". In *Proc. of the 5th Int'l Conf on Extending Database Technology* (EDBT'96). Mar. 1996.

[9] H. Lu, J. Han, and L. Feng, "Stock movement and n-dimensional inter-transaction association rules". In *Proc. of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1998.

[10] H. Lu, L. Feng, and J. Han, "Beyond Intra-Transaction Association Analysis: Mining Multi-Dimensional Inter-Transaction association rules". *ACM Transactions on Information Systems*, Vol. 18, No. 4, October 2000, pp. 423–454.

[11] A. Tung, H. Lu, J. Han, and L. Feng, "Breaking the Barrier of Transactions: Mining Inter-Transaction association rules". In *Proc. of the Knowledge Discovery and Data Mining*, 1999

[12] Z. Dong, H. Li, Z. Shi, "An Efficient Algorithm for Mining lnter-transaction Association Rules in Multiple Time Series". *Journal of Computer Science*. 2004, 31(3). pp. 108-111.

Liang-xi Qin received his B.Sc degree from Guangxi University in 1983. He also finished M.Sc courses in Guangxi University in 1992, and now he is a Ph D. candidate in the Graduate school of Chinese Academy of Sciences. He has become an Associate Professor in Guangxi University since 1997. His research interests include data mining, evolutionary computation and software engineering.

Prof. Zhong-Zhi Shi received his B.Sc and M.Sc degree from the University of Science and Technology of China in 1964 and 1968. At the present, he is a supervisor of Ph.D candidates. Prof. Shi's research and teaching interests are in the areas on Artificial Intelligence, Neural Computing, Cognitive Science, Advanced Database Technology, New Generation Computer. He authored more than 10 books and published more than 200 technical papers on the above areas. Prof. Shi is also active in professional activities. He is the Chairman of Working Group 12.3 of IFIP, member of Standing Steering Committee of PRICAI, and member of Scientific Committee of PRICAI, vice president of Chinese Artificial Intelligence Society, and member of council of Chinese Computer Federation. He serves as an Academic Committee member of Institute of Computing Technology, Academia Sinica, and also a member of the Technical Committee of National Artificial Intelligence Laboratory. He is vice president of Chinese Society of Machine Learning and vice president of Chinese Society of Knowledge Engineering.