Algorithms for Mining Share Frequent Itemsets Containing Infrequent Subsets

Brock Barber and Howard J. Hamilton

Department of Computer Science, University of Regina, Regina, SK., Canada S4S 0A2

Abstract. The share measure for itemsets provides useful information about numerical values associated with transaction items, that the support measure cannot. Finding share frequent itemsets is difficult because share frequency is not downward closed when it is defined in terms of the itemset as a whole. The Item Add-back and Combine All Counted algorithms do not rely on downward closure and thus, are able to find share frequent itemsets that have infrequent subsets. These heuristic algorithms predict which itemsets should be counted in the current pass using information available at no additional processing cost.

1 Introduction

Association rules identify items that occur together and those that are likely to occur, given that particular items have been selected (called itemsets). The discovery of association rules is a two-step process [1]: (1) discover all frequent itemsets meeting user-specified frequency criteria, and (2) generate association rules from the frequent itemsets. The second task is easier than the first [11]. Here, we study the first step in the context of itemset share, a measure of itemset importance [4].

Itemset share is the fraction of some numerical value, such as total quantity of items sold or total profit, contributed by items when they occur in an itemset. Unlike support [1], share can be applied to the non-binary numerical data associated with items in a transaction, allowing for a more insightful analysis of the impact of itemsets in terms of stock, cost or profit. In practice, itemset ranking by support and share can be significantly different [4].

Support frequency ($frequency_{sup}$) is downward closed, since all subsets of a frequent$_{sup}$ itemset are also frequent$_{sup}$ [3]. This property allows efficient algorithms to find all frequent$_{sup}$ itemsets while traversing only a part of the itemset lattice, e.g. [3, 6, 11]. Share frequency is also downward closed if we require each item in a frequent itemset to be frequent when it occurs in the itemset [4]. However, since share considers non-binary values, the share of an itemset can be greater than the share of its subsets. If the frequency requirement is based on the total share of the itemset, frequent itemsets might contain infrequent subsets. Thus, some frequent itemsets cannot be found using the downward closed share frequency definition. We describe heuristic algorithms to discover share frequent itemsets that do not rely on downward closure.

2 Review of the Support Measure

Itemset methodology is summarized as follows. [2]. Let $I = \{I_1, I_2, ..., I_m\}$ be a set of literals, called *items*. Let $D = \{T_1, T_2, \ldots, T_n\}$ be a set of n transactions, where for each transaction $T \in D$, $T \subseteq I$. A set of items $X \subseteq I$ is called an *itemset*. Transaction T *contains* X if $X \subseteq T$. Each itemset X is associated with a set of transactions $T_X = \{T \in D \mid T \supseteq X\}$, the transactions containing X. The *support s* of itemset X equals $|T_X|/|D|$.

Support is illustrated using the transaction database shown in Table 1. The TID column gives the transaction identifier values. Values under each item name are quantity of item sold. To calculate support, a non-zero quantity is treated as a 1. Table 2 shows the support for each possible itemset.

Table 1. Example Transaction Database

TID	Item A	Item B	Item C	Item D
T1	1	0	1	14
T2	0	0	6	0
T3	1	0	2	4
T4	0	0	4	0
T5	0	0	3	1
T6	0	0	1	13
T7	0	0	8	0
T8	4	0	0	7
T9	0	1	1	10
T10	0	0	0	18

Table 2. Itemset Support

Item-set	s	Item-set	s
A	0.30	BC	0.10
B	0.10	BD	0.10
C	0.80	CD	0.50
D	0.70	ABC	0.00
AB	0.00	ABD	0.00
AC	0.20	ACD	0.20
AD	0.30	BCD	0.10
		ABCD	0.00

Transaction data often contains information such as quantity sold or unit profit, that support cannot consider. For example, support for items C and D is 0.8 and 0.7 respectively. However, total quantity sold for C and D is 26 and 67, respectively, so D is sold more frequently than C. Itemsets BC and BD have support of 0.10, indicating equal frequency. However, the quantity of items sold in BC and BD is 2 and 11, respectively. If items B, C and D return a net profit of \$1.00, \$100.00 and \$0.10, then itemsets BC and BD return a net profit of \$101.00 and \$1.10. Yet support does not consider itemset BC to be more important than itemset BD. Support fails as a measure of relative importance of the itemsets in these instances. For target marketing, measures should consider both the frequency of an item contributing to a predictive rule and the value of the items in the prediction [7]. Support allows for neither, so measures based on specific numbers of items, such as percentage of gross sales, costs or net profit, cannot be calculated, and business payoff cannot be maximized.

3 Review of the Share Measure

A *measure attribute* (MA) is a numerical attribute associated with each item in each transaction. The *transaction measure value* of item I_p in transaction T_q, $tmv(I_p,T_q)$, is the value of a measure attribute associated with I_p in T_q. The *global measure value* of item I_p, $MV(I_p)$, is the sum of the *tmv*'s of I_p in all transactions in which I_p appears:

$$MV(I_p) = \sum_{T_q \in T_{Ip}} tmv(I_p, T_q) . \tag{1}$$

The *total measure value* (*MV*) is the sum of the global measure values for all items in *I* in every transaction in *D*, given as

$$MV = \sum_{p=1}^{m} MV(I_p) . \tag{2}$$

If x_i is the i^{th} item of itemset *X*, the *item local measure value* of x_i in *X*, $lmv(x_i,X)$, is the sum of the transaction measure values of x_i in all transactions containing *X*, given by

$$lmv(x_i, X) = \sum_{T_q \in T_X} tmv(x_i, T_q) . \tag{3}$$

The *itemset local measure value* of *X*, *lmv*(*X*), is the sum of the local measure values of each of the *k* items in *X* in all transactions containing *X*, given by

$$lmv(X) = \sum_{1}^{k} lmv(x_i, X) . \tag{4}$$

The *item share* of x_i in *X*, $SH(x_i,X)$, is the ratio of the local measure value of x_i in *X* to the total measure value, as given by

$$SH(x_i, X) = lmv(x_i, X) / MV . \tag{5}$$

The *itemset share* of *X*, *SH*(*X*), is the ratio of the local measure value of *X* to the total measure value, as calculated by

$$SH(X) = lmv(X) / MV . \tag{6}$$

Table 3 gives *lmv*(*X*) and *SH*(*X*) for itemsets in the sample database and $lmv(x_i,X)$ and $SH(x_i,X)$ for items in these itemsets. A dash means an item is not in an itemset.

Consider again itemsets C, D, BC and BD. Itemset C, ranked higher than itemset D by support despite having a lower quantity sold, is ranked lower by share, with *SH*(C) = 0.26 and *SH*(D) = 0.67. Itemsets BD and BC are ranked the same by support, although the quantity of items sold in BC is less. Using share, itemset BD is ranked higher than itemset BC, with *SH*(BD) = 0.11 and *SH*(BC) = 0.02.

Share can be incorporated into many algorithms developed for support [5]. Approaches have been proposed for extending support to quantitative measures, e.g. [10]. We feel share is simpler and more flexible. The problem of finding $\text{frequent}_{\text{sup}}$ itemsets with $lmv(X) \geq minvalue$ has been described [8]. No solution was presented.

Table 3. Itemset Share Summary for Sample Database

	Item A		**Item B**		**Item C**		**Item D**		*Itemset* X	
Itemset	**Lmv**	**SH**	**lmv**	**SH**	**lmv**	**SH**	**lmv**	**SH**	**lmv**	**SH**
A	6	0.06	-	-	-	-	-	-	6	0.06
B	-	-	1	0.01	-	-	-	-	1	0.01
C	-	-	-	-	26	0.26	-	-	26	0.26
D	-	-	-	-	-	-	67	0.67	67	0.67
AB	0	0.00	0	0.00	-	-	-	-	0	0.00
AC	2	0.02	-	-	3	0.03	-	-	5	0.05
AD	6	0.06	-	-	-	-	25	0.25	31	0.31
BC	-	-	1	0.01	1	0.01	-	-	2	0.02
BD	-	-	1	0.01	-	-	10	0.10	11	0.11
CD	-	-	-	-	8	0.08	42	0.42	50	0.50
ABC	0	0.00	0	0.00	0	0.00	-	-	0	0.00
ABD	0	0.00	0	0.00	-	-	0	0.00	0	0.00
ACD	2	0.02	-	-	3	0.03	18	0.18	23	0.23
BCD	-	-	1	0.01	1	0.01	10	0.10	12	0.12
ABCD	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00

4 Share Frequent Itemsets

Property *P* is *downward closed* with respect to the lattice of all itemsets if, for each itemset with property *P*, all of its subsets have property *P* [9]. Share frequency as originally defined is downward closed. Itemset *X* is *downward closed share frequent* (*DC-frequent*) if $\forall x_i \in X$, $SH(x_i,X) \geq minshare$, a user defined minimum share [4].

Theorem 1: DC-frequency is downward closed with respect to the lattice of all itemsets. **Proof**: To show DC-frequency is downward closed, we must show that if *X* is DC-frequent, then for all $X_j \subseteq X$, X_j must be DC-frequent. Suppose *X* is DC-frequent. By definition, for all $x_i \in X$, $SH(x_i,X) \geq minshare$. Since $X_j \subseteq X$, $lmv(x_i,X_j) \geq lmv(x_i,X)$ and $SH(x_i,X_j) = \text{lmv}(x_i,X_j)/MV \geq SH(x_i,X) = lmv(x_i,X)/MV$. Since for all $x_i \in X$, $SH(x_i,X) \geq minshare$, then for all $x_i \in X_j$, $SH(x_i,X_j) \geq minshare$. Therefore, by definition, X_j is DC-frequent. •

To find frequent itemsets with infrequent subsets, itemset *X* is defined to be share frequent, or simply frequent, if $SH(X) \geq minshare$. This definition removes the property of downward closure. Adding an item x_i to itemset *X* to create itemset *Y*, adds a restriction to the measure values of the items in *X*. Values associated with the items in *X* contribute to *lmv*(*Y*), only when they occur with x_i. Their contribution towards *lmv*(*Y*) must be less than or equal to their contribution to *lmv*(*X*). However, $lmv(x_i,Y)$ is added to *lmv*(*Y*), counteracting the effect of the additional restriction. Thus, *lmv*(*Y*) may be less than, equal to, or greater than *lmv*(*X*), depending on the relative effect of the restriction and the addition of the measure value for another item and, it is possible to have an itemset with share ≥ *minshare*, whose subsets have share < *minshare*.

Theorem 2: Share frequency is not downward closed with respect to the lattice of all itemsets. **Proof**: Proof by counterexample is sufficient. Consider itemset ACD in

Table 3. Assume *minshare* = 0.20. *SH*(ACD) = 0.23 and ACD is frequent. *SH*(A) = 0.06 and A is not frequent. A is a subset of ACD so share frequency based on the share of the itemset as a whole is not downward closed. •

5 Description of Algorithms

Figure 1 shows an algorithm space consisting of six algorithms. We use an exhaustive algorithm as a starting point and specialize it using different pruning and candidate itemset generation techniques to create the algorithms. The type of pruning added is shown on the edges between nodes.

The first pass through the data collects information about all 1-itemsets. Summary information is compiled, including *MV* and TC_T, the total number of transactions. C_k is the set of candidate itemsets for the k^{th} pass. C_2 is generated using information about the 1-itemsets and information about the candidate 2-itemsets is collected in pass 2. The process of building C_k using itemsets in C_{k-1} stops when no candidate itemsets are added to C_k. After the k^{th} pass, the local measure value and transaction count is available for each counted *k*-itemset.

Candidate itemset generation and itemset pruning are done in procedure **GenerateCandidateItemsets**. A discussion of this procedure suffices to describe algorithm differences. We use two early methods for generating candidates [3], [6]. *Combination generation* is generation of *k*-itemsets by combining itemsets in C_{k-1}, differing only in their last item. Unless otherwise noted, our algorithms use this type of

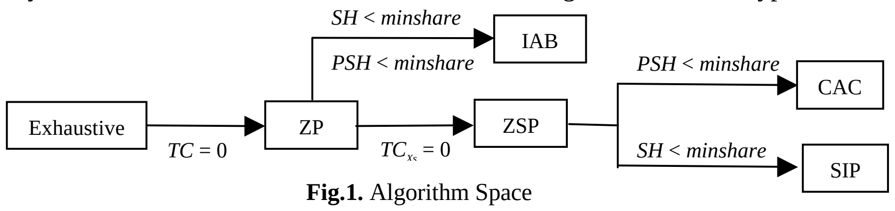

Fig.1. Algorithm Space

generation. *Item add-back generation* is generation of *k*-itemsets by adding to each $X_i \in C_{k-1}$, any item found in the first pass not contained in X_i. Generation of the next potential candidate itemset, X_{pc}, is represented by an iterator procedure **GenerateNextItemset**. The first call to the procedure returns the first generated itemset, and repeated calls cycle through all possible generated itemsets. When no more itemsets can be generated, the procedure returns false. We investigate two types of pruning. *Pre-generation pruning* prunes itemsets from C_{k-1} using information obtained during the *k*-1 pass, before any *k*-itemsets are generated. In *generation pruning*, potential candidate *k*-itemsets are generated and then pruned as required before they are added to C_k. Pre-generation (generation) pruning is done in procedure **PreGenPrune** (**PruneGeneratedItemset**). Procedure **GenerateCandidateItemsets** is written as:

```
1    PreGenPrune($C_{k-1}$)
2    while $X_{pc}$ := GenerateNextItemset() do
3        if PruneGeneratedItemset($X_{pc}$) = false then Add $X_{pc}$ to $C_k$
```

In the exhaustive algorithm, all possible k-itemsets are added to C_k. If m items are found in pass 1, 2^m itemsets are counted. All frequent itemsets are found.

The *Zero Pruning Algorithm* (ZP) is created from the exhaustive algorithm by adding *zero pruning*, pre-generation pruning of any itemset $X_i \in C_{k-1}$ for which TC_{Xi} = 0. This prevents the generation of k-itemsets from (k-1)-itemsets not in the data. The number of itemsets counted cannot exceed 2^m and all frequent itemsets will be found.

Even with zero pruning, X_{pc} can contain a (k-1)-subset, X_s, with $TC_{Xs} = 0$. The *Zero Subset Pruning Algorithm* (ZSP) adds *subset pruning*, generation pruning of any X_{pc} with $X_s \notin C_{k-1}$. The procedure **PruneGeneratedItemset** is written as:

```
3.1 foreach $x_i \in X_{pc}$
3.2     foreach $x_j \in X_{pc}$ where $i \neq j$
3.3         add $x_j$ to $X_s$
3.4     if $X_s \notin C_{k-1}$ then return true
3.5 return false
```

The number of itemsets counted by ZSP cannot exceed the number of itemsets counted by ZP and all frequent itemsets will be found.

The *Share Infrequency Pruning Algorithm* (SIP) is created from ZSP by adding s*hare infrequency pruning*, pre-generation pruning of any itemset $X_i \in C_{k-1}$ whose actual share $SH(X_i) < minshare$. SIP behaves like Apriori [3], building candidate k-itemsets using only frequent itemsets from the previous pass.

The *Combine All Counted Algorithm* (CAC) is created from ZSP by adding heuristic methods to calculate the *predicted share* of X_{pc}, $PSH(X_{pc})$, and generation pruning any X_{pc} whose $PSH < minshare$. For each subset X_s, there is a corresponding item x_i that is a member of X_{pc} but not a member of X_s. We use information about X_s and x_i to calculate the predicted share, since no additional work is required to determine their values (we store first pass information about all 1-itemsets). For $k > 1$, information about infrequent itemsets is discarded after construction of C_k. We calculate $P(X)$, the probability that any single transaction contains an itemset X, using $P(X) = TC_X/TC_T$. Assuming a uniform distribution of actual share over all $T \in T_X$, the share value in each of these transactions is $SH(X)/TC_X$. The predicted share of X in any single transaction, $PSH_1(X)$, is given by:

$$PSH_1(X) = P(X)*(SH(X)/TC_X) + (1-P(X))*0 = SH(X)/TC_T \ . \tag{7}$$

To calculate the predicted share when an itemset, item pair occurs together, equation 8 is used if $TC_{xi} < TC_{Xs}$, equation 9 is used if $TC_{Xs} < TC_{xi}$ and the average of equations 8 and 9 is used if $TC_{xi} = TC_{Xs}$.

$$PSH = SH(x_i) + PSH_1(X_s)*TC_{xi} \ . \tag{8}$$

$$PSH = SH(X_s) + PSH_1(x_i)*TC_{Xs} \ . \tag{9}$$

The average PSH of all X_s, x_i pairs is compared to *minshare*. This value is returned by the function **GetPredictedShare** and **PruneGeneratedItemset** becomes:

```
3.1 $PSH(X_{pc})$ := 0, $SubsetCount$ := 0
3.2 foreach $x_i \in X_{pc}$
3.3     foreach $x_j \in X_{pc}$ where $i \neq j$
3.4         add $x_j$ to $X_s$
3.5     if $X_s \notin C_{k-1}$ then return true
```

```
3.6         $PSH(X_{pc})$ := $PSH(X_{pc})$ + **GetPredictedShare**($x_i$,$X_s$)
3.7         $SubsetCount$ := $SubsetCount$ +1
3.8     **if** $PSH(X_{pc})/SubsetCount < minshare$ **then return true**
3.9     **return false**
```

In the *Item Add-back Algorithm* (IAB), no item with a non-zero measure value is completely discarded. Starting from the ZP algorithm, infrequency pruning is added. New itemsets are generated using item add-back generation. In the k^{th} pass, each single item found in the first pass is added to each frequent itemset from the (k-1) pass. We again use predictive pruning as described for CAC, except the predicted share value is the average *PSH* of the (k-1)-subset, x_i pairs available. Subset pruning is not used. The algorithm for **PruneGeneratedItemsets** differs from that for CAC only in Line 3.5, where the return true becomes a continue statement.

We now provide an example. Figure 2 gives the itemset lattice for the data in Table 1. Each node is labeled with the itemset name. Below the name are $lmv(X)$ and TC_X values, separated by a forward slash. *MV* is equal to 100 and *minshare* is assumed to be equal to 0.20. Frequent itemsets are shaded in Figure 2. For all algorithms, the first pass identifies 1-itemsets C and D as frequent itemsets.

In ZP and ZSP, 2-itemset AB is zero pruned since TC_{AB} = 0. Supersets of AB, (ABC, ABD and ABCD), are not generated or counted. ZSP does not subset prune any itemsets since C_{k-1} contains all subsets of the generated itemsets. All frequent itemsets are found.

In SIP, items A and B are infrequency pruned. Itemset CD is generated from the frequent items in C_1, counted in pass 2 and found to be frequent. SIP terminates because no 3-itemsets can be generated from a single 2-itemset. Frequent itemsets AD and ACD are missed, since item A was infrequency pruned after the first pass and cannot exist in any larger itemsets.

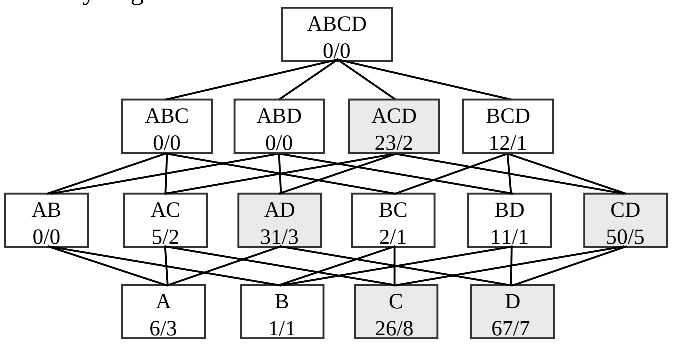

Fig. 2. Itemset Lattice

In CAC, all counted 1-itemsets are used to generate candidate 2-itemsets. Generated 2-itemsets are pruned based on predicted share value. Consider itemset AD. SH(A) = 0.06, TC_A = 3 and SH(D) = 0.67, TC_D = 7. Since $TC_A < TC_D$, we determine PSH_1(D) = SH(D)/TC_T = 0.67/10 = 0.067. Now PSH(AD) = $SH(A)$ + $PSH_1(D)*TC_A$ = 0.06 + 0.067(3)=0.26. PSH(AD) > *minshare*, so AD is added to C_2. Only AD and CD meet the condition $PSH \geq minshare$, so only they are counted in pass 2. CAC terminates after pass 2 because AD and CD cannot be used to generate a 3-itemset with all subsets in C_{k-1}. CAC counted one more 2-itemset than the SIP and it was frequent. CAC missed frequent 3-itemset ACD because one of its subsets, itemset AC, was not counted in the second pass.

IAB generates all possible 2-itemsets except AB. AB is not generated since both A and B are infrequent. As for CAC, itemsets AD and CD are predicted to be frequent and counted in pass 2. After pass 2, itemsets ABD, ACD and BCD are generated by adding single items to frequent itemsets AD and CD. To determine *PSH*(ACD), all available 2-itemset, item pairs are examined. Only 2-itemsets AD and CD exist in C_{k-1} so C plus AD and A plus CD are examined. *PSH*(ACD) = 0.30 > *minshare*, so ACD is added to C_k. The predicted share of ABD and BCD do not meet the minimum share requirement. In pass 3, ACD is counted and is found to be frequent. Itemset ABCD is generated after pass three but *PSH*(ABCD) < *minshare*, so IAB terminates. IAB counts one 3-itemset not counted by either SIP or CAC, and it was frequent.

Table 4 summarizes the performance of the algorithms for the sample data set. A “1” in a column labeled Gen (Cnt) indicates an itemset that was generated (counted). A value of NP in a *PSH* column indicates that no prediction was made because the itemset was not generated. Rows containing frequent itemsets are shaded. The trade off between the work an algorithm does and its effectiveness is evident. ZP and ZSP found all frequent itemsets, but counted most itemsets in the lattice. SIP performed very little work but missed two frequent itemsets.

6 Conclusions

Share can provide useful information about numerical values typically associated with transaction items, that support cannot. We defined the problem of finding share frequent itemsets, showing share frequency is not downward closured when it is defined in terms of the itemset as a whole. We presented algorithms that do not rely downward closure and thus, are able to find share frequent itemsets with infrequent

Table 4. Example Task Summary

	ZP	ZSP	SIP	CAC				IAB			
Itemset	Cnt	Cnt	Cnt	Gen	Cnt	PSH	SH	Gen	Cnt	PSH	SH
AB	1	1	0	1	0	0.02	0.00	0	0	NP	0.00
AC	1	1	0	1	0	0.14	0.05	1	0	0.14	0.05
AD	1	1	0	1	1	0.26	0.31	1	1	0.26	0.31
BC	1	1	0	1	0	0.04	0.02	1	0	0.04	0.02
BD	1	1	0	1	0	0.08	0.11	1	0	0.08	0.11
CD	1	1	1	1	1	0.85	0.50	1	1	0.85	0.50
ABC	0	0	0	0	0	NP	0.00	0	0	NP	0.00
ABD	0	0	0	0	0	NP	0.00	1	0	0.03	0.00
ACD	1	1	0	0	0	NP	0.23	1	1	0.30	0.23
BCD	1	1	0	0	0	NP	0.12	1	0	0.05	0.12
ABCD	0	0	0	0	0	NP	0.00	1	0	0.03	0.00
Sum	8	8	1	6	2			9	3		

subsets. Using heuristic methods, we generate candidate itemsets by supplementing the information contained in the set of frequent itemsets from a previous pass, with other information that is available at no additional processing cost. These algorithms count only those generated itemsets predicted be frequent.

References

1. Agrawal A., Imielinksi T., Swami A. 1993. Mining Association Rules between Sets of Items in Large Databases. *Proceedings of the ACM SIGMOD International Conference on the Management of Data*. Washington D.C., pp.207-216.
2. Agrawal A., Mannila H., Srikant R., Toivonen H. and Verkamo, A.I. 1996. Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining*, Menlo Park, California, pp.307-328.
3. Agrawal A., Srikant R. 1994. Fast Algorithms for Mining Association Rules. *Proceedings of the 20th International Conference on Very Large Databases*. Santiago, Chile, pp.487-499.
4. Carter C.L., Hamilton H.J., Cercone N. 1997. Share Based Measures for Itemsets. *Proceedings of the First European Conference on the Principles of Data Mining and Knowledge Discovery*. Trondheim, Norway, pp.14-24.
5. Hilderman R.J., Carter C., Hamilton H.J., Cercone N. 1998. Mining Association Rules from Market Basket Data using Share Measures and Characterized Itemsets. *International Journal of Artificial Intelligence Tools*. 7(2):189-220.
6. Mannila H., Toivonen H., Verkamo A.I. 1994. Efficient Algorithms for Discovering Association Rules. *Proceedings of the 1994 AAAI Workshop on Knowledge Discovery in Databases*. Seattle, Washington, pp.144-155.
7. Masland B., Piatetsky-Shapiro G. 1995. A Comparison of Approaches for Maximizing Business Payoff of Predictive Models. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon, pp.195-201.
8. Ng R.T., Lakshmanan L.V.S., Han J., Pang A. 1998. Exploratory Mining and Pruning Optimizations of Constrained Association Rules. *Proceedings ACM SIGMOD International Conference on Management of Data*. Seattle, Washington, pp.13-24.
9. Silverstein C., Brin S., Motwani R. 1998. Beyond Market Baskets: Generalizing Association Rules to Dependence Rules. *Data Mining and Knowledge Discovery*, 2(1):39-68.
10. Srikant R., Agrawal R. 1996. Mining Quantitative Association Rules in Large Relational Tables. *Proceedings of the ACM SIGMOD Conference on the Management of Data*. Montreal, Canada, pp.1-12.
11. Zaki M.J., Parthasarathy, M., Ogihara M., Li W. 1997. New Algorithms for Fast Discovery of Association Rules. *Proceedings of the Third International Conference on Knowledge Discovery & Data Mining*. Newport Beach, California, pp.283-286.