AIP Conference Proceedings

# The improved Apriori algorithm based on matrix pruning and weight analysis

Cite as: AIP Conference Proceedings **1955**, 040113 (2018); https://doi.org/10.1063/1.5033777
Published Online: 18 April 2018

Zhenhong Lang

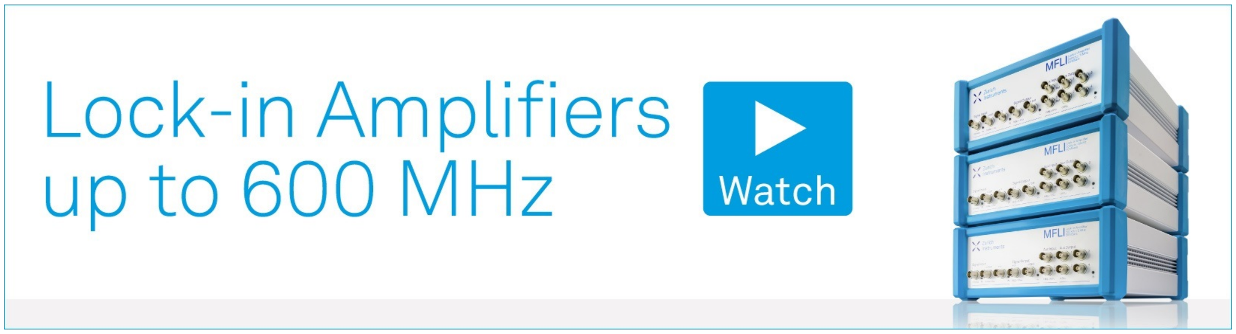
View Online Export Citation

ARTICLES YOU MAY BE INTERESTED IN

An improved Apriori algorithm for mining association rules
AIP Conference Proceedings **1820**, 080005 (2017); https://doi.org/10.1063/1.4977361

Mining algorithm for association rules in big data based on Hadoop
AIP Conference Proceedings **1955**, 040035 (2018); https://doi.org/10.1063/1.5033699

Rare itemsets mining algorithm based on RP-Tree and spark framework
AIP Conference Proceedings **1967**, 040070 (2018); https://doi.org/10.1063/1.5039144



AIP Publishing

© 2018 Author(s).

# The Improved Apriori Algorithm based on Matrix Pruning and Weight Analysis

Zhenhong Lang [a)]

*Tianjin Electronic Information College Tianjin, 300350 China*

[a)] zhenhonglang2000@126.com

**Abstract.** This paper uses the matrix compression algorithm and weight analysis algorithm for reference and proposes an improved matrix pruning and weight analysis Apriori algorithm. After the transactional database is scanned for only once, the algorithm will construct the boolean transaction matrix. Through the calculation of one figure in the rows and columns of the matrix, the infrequent item set is pruned, and a new candidate item set is formed. Then, the item's weight and the transaction's weight as well as the weight support for items are calculated, thus the frequent item sets are gained. The experimental result shows that the improved Apriori algorithm not only reduces the number of repeated scans of the database, but also improves the efficiency of data correlation mining.

**Key words:** association rules; matrix compression; weight; Apriori algorithm; frequent item sets.

## INTRODUCTION

The core idea of data mining is that association rules are excavated in virtue of generated frequent item sets in massive data, and in turn, the valuable information and interesting relationships are searched among data sets. [1] The most classic algorithm was the Apriori algorithm put forward by Agrawal et al in 1993 and the association rule mining algorithm of frequent item set which was issued in the following year. [2] The algorithm's mining process was the iterative process of multiple scanning the transactional database. Based on the two thresholds of complete dependency support [3] and confidence [4], the candidate itemsets were filtrated and the frequent item sets were formed, thereby the association rules were extracted.

From this we see that, there are two flaws in the Apriori Algorithm: for one thing, scanning the transactional database repeatedly leads to excessive consumption of space and time, so that vast candidate sets are generated and the efficiency of data mining is reduced; for another thing, because the importance of item and transaction are not completely considered and the minimum support is excessively depended on, ultimately, the rules of association that might be found are not ideal. In view of this, the improved Apriori algorithm based on matrix pruning and weight analysis is put forward in this paper in order to enhance the algorithm's property.

## THE APRIORI ALGORITHM AND THE ASSOCIATION RULES

Apriori is a classical algorithm to mine frequent item set. The process of Apriori is that firstly, with the help of a priori knowledge of frequent item sets and by means of layer by layer search the candidate sets are found; secondly, frequent item sets are generated by iteration operation; [5] finally, the loop operation is not executed on this basis until new frequent item sets are generated. Then the algorithm is over. The strong association rules are structured on the basis of minimum support and minimum confidence. Although the frequent item sets generated by the algorithm are efficient, in the search for frequent item sets, you need to perform multiple scans of the transactional database, so a lot of time is wasted. In terms of the efficiency generated by the frequent item sets, the improved algorithm uses the transcendental judgment in order to compress search space [6].

One of the main purposes of constructing frequent item sets is to explore the association rules. The item set that meets the minimum support is called the frequent item set. There are two important properties in the mining of association rules.[7] The first one is that all non-empty subsets of frequent item sets must be frequent item sets, any non-frequent item set cannot be a subset of frequent item sets; the other is that any superset of non-frequent item sets is a set of non-frequent items.

## THE IMPROVED APRIORI ALGORITHM

### The Design Idea of the Improved Apriori Algorithm

In the face of the ever increasing data volume, the Apriori algorithm has been improved in this paper in order to avoid repeated scanning in which the database leads to higher system overhead and lower efficiency of Apriori algorithm mining association rules in the search of frequent item sets. On the one hand, after the transacted data items are stored on the basis of boolean matrix and the support counting is analyzed, [8] the compression of frequent item sets are realized and the purpose of pruning is achieved. On the other hand, on the basis of boolean matrix for transaction information and in virtue of weight calculation of item and transaction,[9] the frequent finding item sets is quickly realized and the consumption of space and time is dramatically reduced so that more valuable association rules are excavated.

*1)* Hypothesis transactional databases D is scanned, each transaction item $I_j$ is defined as $D_j$= ($a_{1j}$ $a_{2j}$……$a_{mj}$), including n transactions and m transaction items, thus the boolean transaction matrix of n row and m columns is structured, it is named S, the matrix S as follows:

$$S = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{n1} & a_{n2} & a_{nn} \end{pmatrix}, a_{ij} = \begin{cases} 0, & a_{ij} \notin T_i \\ 1, & a_{ij} \in T_i \end{cases}$$

value range of i is [1, n],value range of j is [1, m]. The total number of transaction items contained in the transaction $T_i$ forms matrix A, which supports the counting of $I_{ij} = \sum_{i=1}^{n} a_{ij}$, and that forms matrix B [10]. By comparing the support counting of item set $I_j$ with the minimum support counting, the smaller than minimum support counting is automatically deleted, then frequent 1-item sets $L'_1$ is generated.

*2)* According to the formula P ($I_j$)=1/ (B [i]/N), the weight of item is calculated, in which N represents the total number of transactions, B [i] expresses the support counting of transaction $I_j$, namely, occurrence frequency of $I_j$ in the transaction matrix.

According to the formula $\mathrm{PT}(T_i) = \sum_{j=1}^{n} P\left(I_j\right)/A[i]$, transaction weight is calculated, $T_i$ expresses number i transaction, $I_j \in T_i$, the weight of transaction is the average value of included transaction item.

*3)* According to the formula $\mathrm{PS}(I_i) = \sum_{i=1}^{m} PT\,(T_i)/\sum_{i=1}^{m} PT\,(T)$, the *s*upport counting of item weight is calculated, $I_i \in T_i$, obviously, the data shows that a transaction weight containing a certain item holds the proportion of all transaction weights, so the item information contained in each transaction is gained. Hence, the weighted value of candidate k-item sets $C_k$ is gained, this value is compared to minimal support, frequent k-item sets $L_k$ is generated, the intersection of $L'_k$ and $L_k$ is calculated, and thus the pruning operation is finished, the frequent item sets are formed, entering the operation of candidate (k+1)-item set.

*4)* The candidate (k+1)-item sets $C_{(k+1)}$ is formed by self-join of $L_k$, defining (k+1)-item sets {$I_i$,$I_j$} as:

$D_{ij} = D_i \wedge D_j = \begin{bmatrix} d_{1i} \wedge d_{1j} \\ d_{2i} \wedge d_{2j} \\ \cdots\cdots \\ d_{ni} \wedge d_{nj} \end{bmatrix}$ support counting of $\{I_i, I_j\} = \sum_{k=1}^{n}\left(d_{ki} \wedge d_{kj}\right)$ the icolumn and the jcolumn execute "AND operation" by bit in The boolean matrix, an item sets which is smaller than the minimum support counting is deleted, and frequent (k+1)-item sets $L'_{(k+1)}$ is generated. According to the above formula, the weight of candidate (k+1)-item sets $C_{(k+1)}$ is unceasingly calculated, frequent (k+1)-item sets $L_{(k+1)}$ is gained.

*5)* The intersection of $L'_{(k+1)}$ and $L_{(k+1)}$ is unceasingly calculated, the pruning operation is finished, the frequent item sets are formed, and the loop operation is carried out in this way, until the candidate item sets are null, and the algorithm is over.

## The application example of the improved Apriori algorithm

The transaction data is shown in table 1, including four transactions, D={T1, T2, T3, T4}, corresponding to five item sets, I={$I_1$, $I_2$, $I_3$, $I_4$, $I_5$}, the hypothesis minimum support being 30%, and the minimum support counting $30\%\times4\approx1$.

*6)* Transactional database is scanned and boolean matrix is structured, the 1 element of the row and column in the matrix is summed, matrix A is generated by the sum of 1 element in all rows, matrix B is generated by the sum of 1 element in all columns.

**TABLE 1.** transaction data

| TID | List of transaction items |
|---|---|
| T1 | $I_2$, $I_4$, $I_5$ |
| T2 | $I_1$, $I_3$, $I_4$ |
| T3 | $I_1$, $I_2$, $I_3$, $I_4$ |
| T4 | $I_1$, $I_3$ |

$$D=(D_1, D_2, D_3, D_4)=\begin{bmatrix}0&1&0&1&1\\1&0&1&1&0\\1&1&1&1&0\\1&0&1&0&0\end{bmatrix} \quad A=\begin{bmatrix}3\\3\\4\\2\end{bmatrix}$$
$$B=\begin{bmatrix}3&2&3&3&1\end{bmatrix}$$

Every element value B[i] is greater than or equal to the minimum support counting 1 in the matrix B, so frequent 1-item sets $L'_1$={$I_1$:3, $I_2$:2, $I_3$:3, $I_4$:3, $I_5$:1} is generated.

*7)* According to the formula $P(I_j)=1/(B[i]/N)$, the weight of every item is calculated, for example: $P(I_1)=1/(3/4)=1.33$; similarly, the weight of the rest of the items are calculated, as shown in Table 2. According to the formula $PT(T_i)=\sum P(I_j)/A[i]$, the weight of every transaction is calculated, for example: $I_2I_4I_5$ are included in transaction T1, so $PT(T_1)=(2+1.33+4)/3=2.44$; similarly, the weight of the rest of the transactions are calculated, as shown in Table 3.

**TABLE 2.** weight value of $I_J$ item

| $I_j$ | $P(I_j)$ |
|---|---|
| $I_1$ | 1.33 |
| $I_2$ | 2 |
| $I_3$ | 1.33 |
| $I_4$ | 1.33 |
| $I_5$ | 4 |

**TABLE 3.** weight value of $T_I$ transaction

| TID | $PT(T_i)$ |
|---|---|
| $T_1$ | 2.44 |
| $T_2$ | 1.33 |
| $T_3$ | 1.50 |
| $T_4$ | 1.33 |

*8)* According to the formula $PS(I_i)=\sum PT(T_i)/\sum PT(T)$, the weight support of the item is calculated, such as transaction item $I_1$, by referring to the transaction matrix it can be known that transaction $T_2T_3T_4$ have included $I_1$, $PS(I_1)=(1.33+1.50+1.33)/(2.44+1.33+1.50+1.33)=0.630$; similarly, weight support of the rest of the items are calculated in the candidate 1-item sets $C_1$, as shown in Table 4. Compared with the minimum support of 30%, the value of all $PS(I_i)$ is more than 30%, so the frequent 1-item sets $L_1$= {$I_1$, $I_2$, $I_3$, $I_4$, $I_5$} is gained. By now, there is no need to prune $L'_1=L_1$. The candidate 2-item sets $C_2$ = {$I_1$ $I_2$, $I_1$ $I_3$, $I_1$ $I_4$, $I_1I_5$, $I_2$ $I_3$, $I_2$ $I_4$, $I_2$ $I_5$, $I_3$ $I_4$, $I_3$ $I_5$, $I_4$ $I_5$} is formed by self-join of $L_1$, the matrix $D_{ij}$ as shown below is generated, for instance, i=1, j=2.

It is known by the above $D_{12}$ matrix, the minimum support counting for $I_1$ and $I_2$ $=\sum(d_{k1}\wedge d_{k2})=1$, the value of k is 1~4.Similarly, the minimum support counting for $I_1$ and $I_3=\sum(d_{k1}\wedge d_{k3})=(0\ 1\ 1\ 1)^T=3$;the minimum support counting for $I_1$ and $I_4=\sum(d_{k1}\wedge d_{k4})=(0\ 1\ 1\ 0)^T=2$; the minimum support counting for $I_1$ and $I_5=\sum(d_{k1}\wedge d_{k5})=(0\ 0\ 0\ 0)^T=0$; the minimum support counting for $I_2$ and $I_3=\sum(d_{k2}\wedge d_{k3})=(0\ 0\ 1\ 0)^T=1$; the minimum support counting for $I_2$ and $I_4=\sum(d_{k2}\wedge d_{k4})=(1\ 0\ 1\ 0)^T=2$; the minimum support counting for $I_2$ and $I_5=\sum(d_{k2}\wedge d_{k5})=(1\ 0\ 0\ 0)^T=1$; the minimum support counting for $I_3$ and $I_4=\sum(d_{k3}\wedge d_{k4})=(0\ 1\ 1\ 0)^T=2$; the minimum support counting for $I_3$ and $I_5=\sum(d_{k3}\wedge d_{k5})=(0\ 0\ 0\ 0)^T=0$; the minimum support counting for $I_4$ and $I_5$ $=\sum(d_{k4}\wedge d_{k5})=(1\ 0\ 0\ 0)^T=1$.The minimum support count is greater than 1, including: $I_1$and$I_3$, $I_1$and$I_4$, $I_2$and$I_4$, $I_3$and$I_4$, so frequent 2-item sets $L'_2$={$I_1$ $I_3$:3, $I_1$ $I_4$:2, $I_2$ $I_4$:2, $I_3$ $I_4$:2}.

**TABLE 4.** weight value of candidate 1-item sets $C_1$

| $C_1$ | PS ($I_i$) |
|---|---|
| $I_1$ | 0.630 |
| $I_2$ | 0.597 |
| $I_3$ | 0.630 |
| $I_4$ | 0.798 |
| $I_5$ | 0.370 |

$$D_{12}=D_1\wedge D_2=\begin{pmatrix} d_{11}\wedge d_{12} \\ d_{21}\wedge d_{22} \\ d_{31}\wedge d_{32} \\ d_{41}\wedge d_{42} \end{pmatrix}=(0\ 0\ 1\ 0)^T$$

*9)* On the basis of transaction data matrix and the formula of the item's weighted support, the weight values of the candidate 2-item sets $C_2$ is sequentially calculated, as shown in table 5. Compared with the minimum support 30%, the frequent 2-item sets $L_2$={$I_1$ $I_3$, $I_1$ $I_4$, $I_2$ $I_4$, $I_2$ $I_5$, $I_3$ $I_4$, $I_4$ $I_5$}, by now, $L'_2\neq L_2$, there arises the pruning problem, the intersection between $L'_2$ and $L_2$ is calculated, so the eventual $L_2$={$I_1$ $I_3$, $I_1$ $I_4$, $I_2$ $I_4$, $I_3$ $I_4$}. The candidate 3-item sets $C_3$ ={$I_1$ $I_3$ $I_4$, $I_1$ $I_2$ $I_4$, $I_2$ $I_3$ $I_4$, $I_1$ $I_2$ $I_3$ } is formed by self-join of $L_2$, because of { $I_1$ $I_2$, $I_2$ $I_3$} the nonfrequent 2-item sets, so { $I_1$ $I_2$ $I_4$, $I_2$ $I_3$ $I_4$, $I_1$ $I_2$ $I_3$} is deleted from $C_3$, $C'_3$={$I_1$ $I_3$ $I_4$}, matrix $D_{ijk}$ is generated, namely: $D_{134}=(D_1\wedge D_3)\wedge D_4=(0\ 1\ 1\ 1)^T\wedge(1\ 1\ 1\ 0)=(0\ 1\ 1\ 0)^T$, thereinto: i=1, j=3, k=4. So the minimum support count of $I_1$, $I_3$ and $I_4$ $=\sum(0\ 1\ 1\ 0)^T=2$,thus frequent 3-item sets $L'_3$={ $I_1$ $I_3$ $I_4$:2} is formed.

*10)* The weight values of the candidate 3-item sets $C_3$ is sequentially calculated, as shown in Table 6. Compared with the minimum support 30%, the frequent 3-item sets $L_3$={ $I_1$ $I_3$ $I_4$}, by now, $L'_3=L_3$, there does not exist the pruning problem. So the cycle is calculated, a new item sets cannot be found, thus $C_4=\Phi$, the algorithm is terminated. The minimum support count of $D_{134}=2>1$; PS($I_1$ $I_3$ $I_4$)=0.429>30%, so a strong association rule is produced from {$I_1$ $I_3$ $I_4$} is: $I_1\wedge I_3\rightarrow I_4$, $I_1\wedge I_4\rightarrow I_3$, $I_1\rightarrow I_4\wedge I_3$.

## ALGORITHM EXPERIMENT AND PERFORMANCE ANALYSIS

In this paper, we have selected college students' employment information and academic performance as the basic data for 3 consecutive years, and the association rules mining operation is carried out. Based on the same experimental environment, the improved Apriori algorithm proposed in the paper, the original Apriori algorithm, the Apriori algorithm based on matrix in the literature review, the Apriori algorithm based on matrix and weight were respectively tested on the time consumed in the process of the mining of the association rules. The advantage of the improved Apriori algorithm was highlighted in the operating time. The comparison information of time consumption is shown in figure 1.

**TABLE 5.** weight value of candidate 2-item sets $C_2$

| $C_2$ | PS ($I_i$ $I_j$) |
|---|---|
| $I_1$ $I_2$ | 0.227 |
| $I_1$ $I_3$ | 0.630 |
| $I_1$ $I_4$ | 0.429 |
| $I_1$ $I_5$ | 0 |
| $I_2$ $I_3$ | 0.227 |
| $I_2$ $I_4$ | 0.597 |
| $I_2$ $I_5$ | 0.370 |
| $I_3$ $I_4$ | 0.429 |
| $I_3$ $I_5$ | 0 |
| $I_4$ $I_5$ | 0.370 |

**TABLE 6.** weight value of candidate 3-item sets $C_3$

| $C_3$ | PS ($I_i$ $I_j$ $I_k$) |
|---|---|
| $I_1$ $I_3$ $I_4$ | 0.429 |
| $I_1$ $I_2$ $I_4$ | 0.227 |
| $I_2$ $I_3$ $I_4$ | 0.227 |
| $I_1$ $I_2$ $I_3$ | 0.227 |

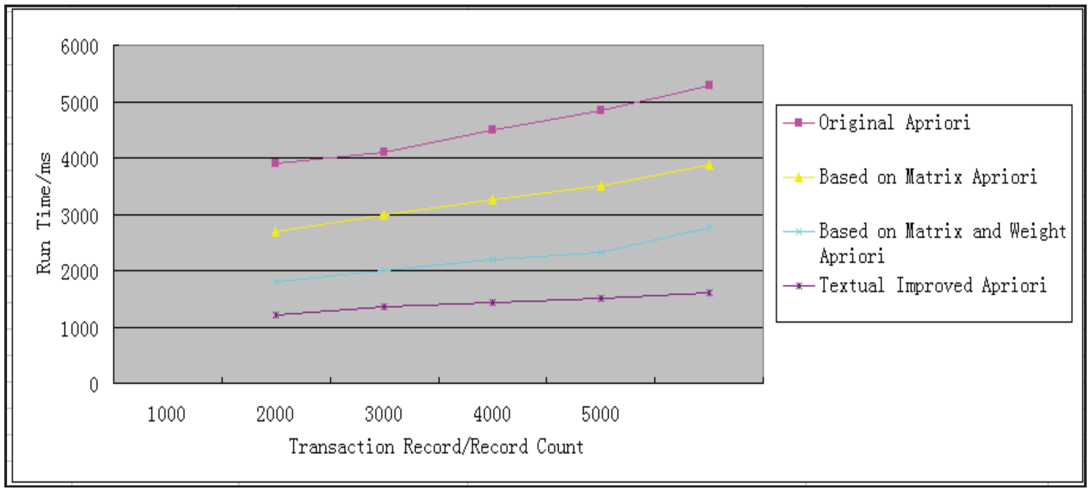


**TABLE 1.** Comparison of the Four Algorithms Runtime Performance

The experimental results show that the improved Apriori algorithm in this paper takes the least time when the same number of transactions is conducted. With the increase of the pending transaction data, the operation time of the algorithm is linear. This indicates that in the big data environment, the higher the amount of transaction of information is, the more obvious advantages of the algorithm time-consuming there are. The main reason is that only one scan of the transaction database is required when a frequent item set is generated. If the database is larger, the original Apriori algorithm will significantly increase the time consumption for multiple scans of the database. The other two algorithms also scan the database for only once, but in the process of pruning to compare with the minimum support, a large number of operations are needed that increases the overhead of system space and time. The improved Apriori algorithm is the clever use of the matrix operations and the knowledge of probability theory. In virtue of calculation of weights of transactions and items and adding weight support and minimum support count, the operation time of pruning and generating frequent item sets is greatly reduced. Furthermore, the performance of mining association rules is obviously improved. In addition, the algorithm can generate strong association rules. Through this analysis, the neglected valuable information is excavated in order to provide useful data for decision making.

## CONCLUSION

With the advent of the era of big data, the status of data mining technology has become increasingly high. The mining of association rules has been hailed as the main research subject in this field. On the basis of inheriting the classical algorithms, the research and improvement of Apriori algorithm is of great theoretical significance and practical value. In this paper, the improved Apriori algorithm based on matrix pruning and weights has been analyzed and shown effective to reduce the number of scanning the transaction database, to prune operation to compress search space, and to improve the efficiency of generating frequent item-sets. Through the calculation of weight and weight support of the transaction, the runtime of the algorithm is greatly saved and the impact of association-rule mining is avoided, a problem that is caused by support which is set either too high or too low. Hence strong association rules among transaction items are effectively excavated and valuable information is obtained.

Capital projects:

Tianjin science and technology commission fund project " Research on intrinsic driving forces by in-depth integration of university enterprise cooperation to stimulate the innovation and entrepreneurship of teachers in higher vocational colleges "

Tianjin the education seminar on higher vocational technology subject" Research on the implementation of 'hybrid' teaching on the online and offline courses in the core curriculum of vocational software technology "

## REFERENCES

1. Baralis E, Cagliero L, Cerquitelli T, et al. NEMICO: Mining Network Data through Cloud-Based Data Mining Techniques [C] // Utility and Cloud Computing, 2014 IEEE/ACM 7th International Conference on. IEEE, 2014:503-504.
2. Zheng Lin. A Divide-and-Conquer Apriori Algorithm Directly Generating Frequent Itemsets [J]. Computer Applications and Software, 2014, 31 (4):297-301.
3. Fu Sha, Zhou Hong-jun. The Research and Improvement of Apriori Algorithm for Mining Association Rules [J]. Microelectronics & Computer, 2013, 30 (9):110-114.
4. Luo Dan, Li Tao-shen. The Improved Research on Apriori Algorithm Based on Compression Matrix [J]. Computer Science, 2013, 40 (12):75-80.
5. Zhang S, Du Z, Wang J T L. New techniques for mining frequent patterns in unordered trees. [J]. IEEE Transactions on Cybernetics, 2015, 45 (6):1113-1125.
6. Zhou Fa-chao. Research and Improvement of Apriori Algorithm for Mining Association Rules [J]. Journal of Frontiers of Computer Science and Technology, 2015, 9 (9):1075-1083.
7. Zhu Yi-xia. Algorithm for Association Rule Minning Based on Ordinal Matrix and Tree [J]. Computer Science, 2006, 33 (7):196-198.
8. Fu Sha. Study on the improved algorithm of Apriori based on matrix [J]. Microelectronics & Computer, 2012, 29 (5):156-160.
9. Bian Gen-qing, Wang Yue. An Improved Apriori Algorithm Based Matrix and Weight [J]. Microelectronics & Computer, 2017, 34 (1):136-140.
10. Qu Rui, Zhang Tian-jiao. Improved Apriori Algorithm Based on Matrix Compression [J]. Computer Engineering and Design, 2017, 38 (8):2127-2131.