# Implementation Method and Comparative Experiment Based on Association Rule Mining Algorithm

**Yongfu Liu, Jian Jiao**
*Gansu Open University, Lanzhou, Gansu, China*

**Abstract: With the rapid development of the information age and the continuous improvement of data storage technology, association rule mining algorithms, as an important data mining technique, play a key role in discovering the relationships between items in a dataset. This article aims to explore the implementation methods of association rule mining algorithms and analyze the advantages, disadvantages, and applicability of different algorithms through comparative experiments. In order to compare and analyze the performance of these two algorithms, this paper designed a set of comparative experiments. The experimental dataset contains multiple transactions, each consisting of a set of items. The experiment evaluates the performance of different algorithms by calculating their efficiency, accuracy, and memory usage in discovering frequent itemsets and association rules. The experimental results show that the Apriori algorithm is relatively simple to implement, easy to understand and implement, and therefore still has certain application value when dealing with small-scale datasets.**

**Keywords**: **Association Rule Mining; Apriori Algorithm; Implementation Method; Comparative Experiment**

## 1. Introduction

With the rapid development of information technology, the education sector has also ushered in a new era driven by data. The university teaching management system has accumulated a large amount of student performance data, which hides rich teaching information and potential patterns behind it [1]. However, traditional data analysis methods often only perform simple statistics and queries on data, and cannot deeply explore the correlations and potential value between data. Therefore, introducing advanced data mining techniques such as Apriori algorithm for in-depth analysis of data in the grade management system isof great significance for improving teaching management efficiency, optimizing course settings, and enhancing teaching quality [2]. The Apriori algorithm is a classic association rule mining algorithm widely used in fields such as market basket analysis and recommendation systems. In the field of education, the Apriori algorithm can also be used for grade data analysis to help students and teachers discover correlations between courses, optimize learning methods, and improve teaching effectiveness [3]. In order to verify the performance and effectiveness of different association rule mining algorithms, researchers conducted a large number of comparative experiments. The experimental results show that the FP Growth algorithm is significantly more efficient than the Apriori algorithm when processing large-scale datasets [4]. This is because the FP Growth algorithm effectively compresses the dataset by constructing FP trees, reducing the generation and deletion of candidate itemsets, thereby improving mining efficiency. In addition, the FP Growth algorithm also performs well in terms of result quality, generating more accurate and useful association rules. However, the FP Growth algorithm also has some shortcomings [5]. For example, in the process of building FP tree, the algorithm requires a large amount of memory resources, which poses certain difficulties in implementation for massive databases. In addition, the performance of FP Growth algorithm may be affected when dealing with sparse datasets. Therefore, in practical applications, it is necessary to select appropriate association rule mining algorithms based on specific data characteristics and requirements. Association rule mining algorithms play an important role in the field of data mining [6]. With the continuous growth of data volume and the increasing diversification of application requirements, association rule mining algorithms are also constantly developing and improving. In the future, we can look forward to the

 

emergence of more efficient and accurate association rule mining algorithms, providing strong support for the development of various fields [7]. At the same time, we also need to pay attention to the performance and effectiveness of algorithms in practical applications, continuously optimize and improve algorithms to adapt to changes in different scenarios and requirements.

## 2. Principle of Apriori Algorithm

A itemset is a collection of one or more items in a dataset. In grade data analysis, an item can represent a course.

Support: The support of an itemset refers to the ratio of the number of transactions containing that itemset to the total number of transactions. Support is used to measure the prevalence of an item set in a dataset.

Frequent itemsets: itemsets with support exceeding a predefined threshold are called frequent itemsets.

Confidence: Confidence refers to the proportion of transactions that contain both the items in the rule and the items on the right side of the rule [8]. Confidence is used to measure the reliability of association rules.

Frequent itemsets: If an itemset appears more frequently in the dataset than the predetermined minimum support threshold, it is called a frequent itemset. Frequent itemsets are the foundation of association rule mining.

Association rules: Association rules describe the relationships between items in a dataset, usually expressed in the form of "if A, then B". Among them, A and B are itemsets, and the association rules need to meet the minimum support and minimum confidence thresholds.

## 3. Apriori Algorithm Steps

### 3.1 Apriori Algorithm Steps

The core idea of Apriori algorithm is to discover frequent itemsets through layer by layer search iteration, and then generate association rules [9]. The specific steps are as follows: Scan the database and generate a candidate itemset: First, scan the database, calculate the support of each item, and generate a candidate itemset C1. Pruning and generating frequent itemsets: Based on the minimum support threshold, select the frequent itemset L1 from the candidate itemset C1. Then, use L1 to generate candidate set C2, and perform pruning again to generate frequent itemset L2. Repeat this process until no new frequent itemsets can be generated. Generate association rules: For each frequent itemset Lk, generate all its non empty subsets and calculate confidence. If the confidence level meets the minimum confidence threshold, then the Apriori algorithm for generating association rules is a classic algorithm in association rule mining, as defined by R Agrawal and R Srikant proposed it in 1994 [10]. This algorithm uses a layer by layer search iterative method to search for k+1 itemsets from k itemsets, mainly used to find frequent patterns, correlations, or causal structures in information carriers such as transaction data and related data. However, although the Apriori algorithm plays an important role in the field of association rule mining, it also has some significant shortcomings and improve the speed of finding frequent itemsets.

### 3.2 Formulas and Calculations

Support calculation formula: Support (X)=P(X)=number of transactions containing itemset X/total number of transactions;Confidence calculation formula:Confidence(X→Y)=P(Y|X)=number of transactions containing itemsetsX and Y/number of transactions containing itemset X; The formula for calculating lift: Lift(X→Y)=P(X,Y)/(P(X)*P(Y))=Support (X,Y)/(Support(X)*Support (Y)); The degree of improvement is used to measure the degree of correlation between two itemsets. A degree of improvement greater than 1 indicates a positive correlation, while a degree of improvement less than 1 indicates a negative correlation.

### 3.3 Shortcomings of Apriori Algorithm

Multiple scans of the database: The Apriori algorithm requires multiple scans of the transaction database to generate candidate sets and support counts. Each iteration requires scanning the entire database, which can result in significant I/O load and time overhead on large-scale datasets [11]. Generate a large number of redundant candidate sets: In the process of generating frequent k-itemsets from frequent (k-1) itemsets, a large number of candidate itemsets will be generated, many of which are redundant. These redundant candidate sets not only increase computational burden, but may also lead to memory overflow issues. Inefficient: Due to the frequent itemset

 

generation and pruning operations requiring a large amount of pattern matching and comparison, the Apriori algorithm has a high time complexity. Especially when dealing with large-scale datasets, the execution time of algorithms will significantly increase. Difficulty in handling sparse data: The Apriori algorithm shows significant performance degradation when dealing with high-dimensional sparse data, as it requires scanning a large number of zero values. Unable to effectively handle multi-dimensional data: The Apriori algorithm is mainly targeted at single dimensional datasets, and requires additional processing steps for multi-dimensional spatiotemporal data.

## 4. Improvement of Apriori Algorithm

### 4.1 Improved Algorithm Based on Frequent Itemset Grouping Parallel

4.1.1. Improvement method:

This algorithm groups frequent k-1 itemsets according to a certain rule, and each group of frequent k-1 subsets directly generates frequent k subsets [12]. Then, the frequent k subsets generated by each group are collected together. In this way, each group reduces a lot of connection attempts during self connection, and can process connection and pruning behaviors in parallel, thereby reducing waiting time and improving the speed of searching for frequent itemsets.

4.1.2 Specific deduction steps:

Assuming the size of the Lk-1 itemset is N, it is necessary to determine the number of connections LN as follows: LN=∑n (where n is the length of the Lk-1 itemset). The calculation formula for the total number of self connections PLN after grouping is: PLN = ∑n+∑n+…+∑n, Where i is the frequent (k-1)-itemset grouping quantity, ni is the frequent (k-1) - sub itemset length of each group, n1+n2+…+ni=N. Obviously, the value of PLN is much smaller than LN. Assuming N=4, according to the original formula, LN=3+2+1=6 times. If divided into two groups, each group has a frequent (k-1) - sub item set length of n1=2, n2=2, According to the formula after grouping, PLN=1+1=2 (times) is obtained, which has many fewer useless connections compared to LN=6 (times) before grouping.

### 4.2 Improved Algorithm Based on User Interest Constraints

4.2.1 Improvement method:

This algorithm uses user interest items to remove irrelevant items from the transaction database item set, reducing the number of database scans and candidate item sets, thereby improving the efficiency of the algorithm.

4.2.2 Specific deduction steps

Assuming that the original transaction database needs to be scanned at least M times to find all candidate itemsets, and the new transaction database needs to be scanned N times, the percentage improvement in efficiency is:(M-N)/M. Assuming that the original transaction database generates approximately X candidate itemsets when generating k candidate itemsets, and the new transaction database generates approximately Y candidate itemsets when generating k candidate itemsets, the percentage improvement in efficiency in terms of the number of candidate itemsets is:(X-Y)/X. Assuming that the original transaction database needs to be scanned at least 7 times to find all candidate itemsets, and the new transaction database needs to be scanned 5 times, the percentage improvement in efficiency is: (7-5)/7 ≈ 28.6%.

## 5. Efficient Implementation Method of Apriori Algorithm 4.1 Main Ideas

The main idea of the efficient implementation method of Apriori algorithm is shown in Figure 1.

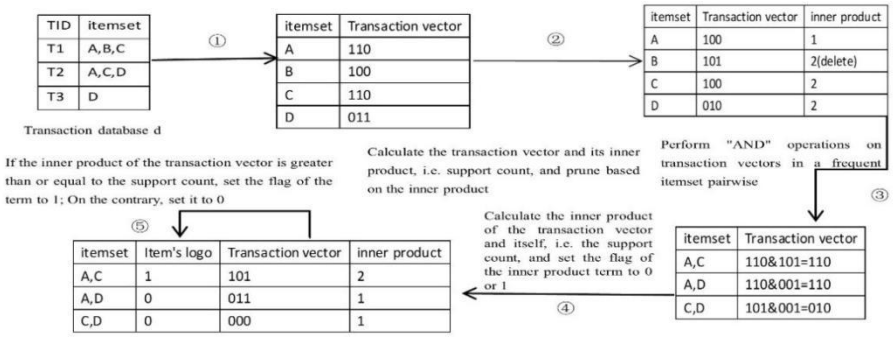


**Figure 1. The Main Idea of the Apriori Algorithm Implementation Method**

 

By scanning the dataset layer by layer and calculating support, frequent itemsets are discovered, and then association rules are constructed using these frequent itemsets. If an itemset appears frequently, then all its subsets are also frequent. Based on this idea, the algorithm continuously generates candidate frequent itemsets and uses a minimum support threshold to filter out itemsets that do not meet the criteria, ultimately obtaining frequent itemsets.

This method adds an identifier TID (Transaction ID) to each transaction in the transaction database d to mark each transaction record. This method changes the storage method of the itemset by storing its items in a linear table and adding a Boolean flag to each item, where "0" indicates that the item's support is less than smin, and "1" indicates that the item's support is greater than or equal to smin; In another linear table, the transaction vector corresponding to the item is stored in the form of a bit vector. [13] The order of each element in the transaction vector corresponds one-to-one with the order of transactions in the database. The state of the transaction vector is represented in the form of a Boolean variable, where "1" indicates that the item appears in the corresponding transaction, and "0" indicates that the item does not appear in the corresponding transaction. The main pseudocode for implementing this method is:

The main pseudocode for implementing this method is:

```
Input: transaction database d, Sc = smin.
Output: All frequent itemsets L.
C1={set of all items}
For all c∈ C1 do Vc=ϕ//Set the transaction vector to empty
//Generate transaction vectors for each item in C1 for all c∈C1 do
for all t∈D do
if c∈t then Vc .append(1)elseVc .append(0)
//Calculate the inner product of the transaction vector for each term in C1 for all c∈C1 do
S (c)=c.Vcc. Vc/Calculate vector inner product ifS (c)<S c then delete c from C1//Delete items in C1 with support less than the minimum support
L1=C; S1 = {S(c)|c∈L1}
Sort (L1)//Sort frequent itemsets in ascending support order
//Generate L2
if c1.
flag ==1&&c2.flag == 1 then
for all c1∈C1 do for all c2∈C1 do Vc = c1.Vc &c2.Vc
//Connect the transaction vectors pairwise with the terms in operation c=c1∞ c2//C1 pairwise. S (c)=Vc·Vc//Calculate the vector inner product if S (c)>S c then
C.lag=1, C2. append (c)//Generate candidate frequent 2-itemset else c. lag=0, C2.append(c)
L2= {c∈C2|c1.flag=1};
S2={S(c)|c∈L2}
//Generate Lk
Lk-1(k>2)={c∈Ck-1|c.flag=1} sort( if c1.flag ==1&&c2.flag ==1&&c1,c2∈Ck-1then
//Sort the frequent k-1_itemsets in ascending order of support for all c1∈Ck-1 do Lk -1)
for all c2∈Ck-1 do
//The transaction vectors perform pairwise AND operations with Vc=c1. Vc&c2.Vc
c= c1∞c2
//Pre pruning the newly generated term c using the non frequent term ck-1
for c subset∈subset(c) then
//Search through Hash in Ck-1
//Find the k-1-dimensional subset ck-1 of item c
ck-1=Hash(c subset ) in Ck-1
//If the newly generated k-dimensional term c is a k-1-dimensional subset
//If the flag of the subset in Ck-1 is set to "0", then set the flag of c to "0"
if ck-1.flag == 0 then
c flag=0
if c.flag! = 0 then S (c)=Vc·Vc
If S(c)>Sc then//Generate candidate frequent 2-itemset c. lag=1, Ck.append(c)
```

else c.flag=0 ,Ck.append(c )
Lk={c∈Ck|c1.flag=1}; Sk={S(c)|cLk}
printL={L1∪L2 ⋯∪ Lk-1∪Lk}L

Considering the large average width of transaction vectors in practical applications and their sparsity, the algorithm compresses and stores transaction vectors to reduce their memory usage during implementation. In experiments, transaction vector dynamic memory allocation is the best implementation method for association rule analysis in large-scale databases.

To provide a more detailed description of the implementation method mentioned above, an example is introduced to illustrate the steps of the algorithm in detail. Firstly, set the support count Sc=2 and smin=2/9, and the transaction database is shown in Table 1.

**Table 1. Transaction Database**

| TID | itemset | TID | itemset |
|---|---|---|---|
| T1 | A, B, E | T6 | B, C |
| T2 | B, D | T7 | A, C |
| T3 | B, C | T8 | A, B, C |
| T4 | A, B | T9 | A, B, C, E |
| T5 | A, C | | |

(1) Generate L1. Scan the transaction database to generate transaction vectors for each item in the candidate itemset (with the vector position consistent with the transaction order), and then calculate the inner product of the transaction vector for each item in the candidate frequent itemset and itself in sequence, denoted as S(i). The inner product of the transaction vector and itself is equal to the number of non-zero elements in the vector. If S (i)<Sc, delete it. The set of remaining items is the frequent itemset, and the generated frequent itemset is shown in Table 2. In Table 2, the transaction vector of itemset D is 010101100 ". SinceS(4)=1<2, i.e. itemset D does not meet the minimum support count, itemset D is deleted when generating frequent itemsets.

**Table 2. Frequent 1_itemsets Generated**

| itemset | Transaction vector | | | | | | | | | inner product |
|---|---|---|---|---|---|---|---|---|---|---|
| *A B C DE* | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 6 |
| | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 7 |
| | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1((Delete) |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 |

There are |Ck| items in Ck. There are N transactions in the transaction database, and the average width of the transactions is n. Calculate all in Ck.The support level of the item, the number of operations of the Apriori algorithm is $\sum_{i=1}^{|C_k|} N_n$. Calculating the inner product of the transaction vector and itself in Ck requires 2k-1 operations. The number of operations required to calculate the support of all terms in Ck using the above method is $\sum_{i=1}^{C_k|} N_n 2k-1$ . Due tok<<(n,N), $\sum_{i=1}^{C_k|} N_n 2k-1$ , The number of operations is much smaller than $\sum_{i=1}^{C_k|} N_n$ .The above method only requires scanning the database once and applies.The inner product calculation of vectors can avoid the operation of calculating support by counting the number of times each item appears in the transaction database, greatly reducing the I/O device overhead of the computer.

(2) Generate L2. Perform an AND operation on the transaction vectors in the frequent 1-point set to obtain the transaction vectors of the candidate frequent 2-point set, and connect the corresponding items to generate the items in the candidate frequent 2-point set. Check if the transaction vectors of the items in the candidate frequent itemset and their inner product meet the minimum support count requirement: if S (i)<Sc, set the flag of the item in the candidate itemset to "0"; On the contrary, set it to "1". The set of items marked as "1" in the candidate frequent 2 item set is called the frequent 2 item set. The frequent itemsets generated are shown in Table 3. In Table 3, itemset A and itemset B are connected, and the transaction vector of the two terms is obtained by taking the "AND" operation. The result is "10010011", and its inner product S(1)=4>2, indicating that it is a frequent itemset. Set the flags of itemsets A and B to "1".

**Table 3. Frequent 2-itemsets Generated**

| itemset | Item's logo | Transaction vector | inner product |
|---|---|---|---|
| *A, B* | 1 | 1 0 0 1 0 0 1 1 | 4 |
| *A, C* | 1 | 0 0 0 0 1 0 1 1 | 4 |
| *A, E* | 1 | 1 0 0 0 0 0 0 0 1 | 4 |
| *B, C* | 1 | 0 0 1 0 0 1 0 1 1 | 4 |
| *B, E* | 1 | 1 0 0 0 0 0 0 0 1 | 2 |
| *C, E* | 0 | 0 0 0 0 0 0 0 0 1 | 1(Delete) |

The above method avoids the operation of taking the intersection of identifier lists, and no longer allocates storage space separately for frequent itemsets, improving the efficiency of the algorithm and reducing its memory

 

consumption.

(3) Generate Lk. When Ck (k≥3) is generated by connecting items pairwise in Lk-1, in order to determine whether the generated items are invalid, The Apriori algorithm will scan Lk-1 multiple times. To reduce the number of scans of Lk-1, a non frequent itemset is used to pre prune the generated items: the items marked as "1" in Ck-1 are connected pairwise to generate candidate items c(c⊆Ck), Determine whether the flag of all k-1 dimensional subsets of c in Ck-1 is "0". If so, directly set the flag of c in Ck to "0"; If not, set the flag for c in Ck to "1". Calculate the inner product of the transaction vector and the item marked as "1" in Ck again, and determine whether the inner product of the transaction vector and the item meets the support count requirement. If it does not meet the requirement, set the item's flag to "0". Finally, the set of items marked as "1" in Ck is the frequent k-item set Lk. In addition, before generating Ck, sort the items in Ck-1 in ascending order of their transaction vectors and inner product, so that the generated Ck has the fewest items. The schematic diagram of pre pruning is shown in Figure 2. In the pre pruning operation, a Hash function is constructed for the terms in Ck-1 to reduce the time complexity of finding each term to 1.

If there are |Ck| terms in Ck, and i∈(1, Ck), then there is a subset of Ni terms in Ck. The number of operations required for this method is Ni×|Ck|, compared to $\sum_{i=0}^{C_k} k \ \sum_{i=0}^{Ni} k-1$ | Significantly reduced, thereby improving the efficiency of the algorithm.

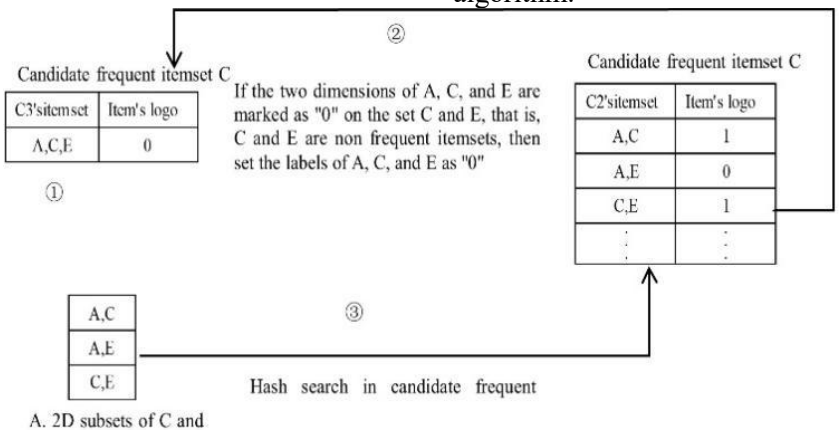


**Figure 2. Schematic Diagram of Pre Pruning**

In practical applications, transaction vectors stored in vector form have a large number of elements, and transaction vectors are often sparse in actual data, resulting in low vector storage efficiency. For this purpose, the sparse vector is compressed and stored.

Define two vectors (all of which are Boolean vectors):A= {x1,x2, … ,xn }B={y1,y2, … ,yn }, Before the transaction vector is compressed and stored, the expression for the inner product of two vectors is:A.B = x 1 × y1 + x2 × y2 + … + xn × yn.requires a total of 2n-1 operations to be performed. In the process of mining frequent itemsets, compressing and storing transaction vectors to remove the interference of "0" elements in the vectors can reduce the computational load of the computer.

Before the transaction vector is compressed and stored, the expression for the "AND" operation between two vectors is:A&B={x1&y1,x2&y2,⋯,xn&yn},The computer needs to perform a total of n operations. Due to the sparsity of the vector, the index Loci of the transaction vector is stored using compressed storage as shown in equationLoci=

$$\begin{cases} A: < Loc1 > , < Loc2 > , ..., < Loci > \\ B: < Loc1 > , < Loc2 > , ..., < Loci > \end{cases} ,$$

When performing an AND operation between vectors, first construct a Hash function to hash the elements of vector B into the hash table HT; Secondly, loop A in the vector and use the same hash function to search for the index of the element corresponding to A in B. If it can be found, set the result of the "AND" operation between the corresponding elements to "1". The total number of operations required is min (i,j). Due to the sparsity of transaction vectors, i.e. i<<n in min (i,j), j<<n,Therefore, min (i, j)<<n. This method accelerates the speed of algorithm mining association rules and saves storage space. However, the Apriori algorithm still has shortcomings in practical applications.To this end, the EI_Spriori algorithm was proposed and

applied to the correlation analysis of college student grades.

## 6. Application of EI_Spriori Algorithm

Using the grades of 4 courses from 7460 students in a certain university's student performance system as the database for algorithm correlation analysis, there are 7460 records in the student performance database. The performance data of some students are shown in Table 4.

**Table 4. Score Data of Some Students**

| Number | Score | | | |
|---|---|---|---|---|
| | Construction of pavement engineering | Fundamentals of Computer Science | Fundamentals of Data Mining | construction project management |
| 1 | 77 | 73 | 90 | 80 |
| 2 | 64 | 66 | 73 | 70 |
| 3 | 64 | 64 | 87 | 76 |
| 4 | 83 | 71 | 69 | 73 |
| 5 | 83 | 78 | 79 | 90 |

Firstly, the score data is subjected to unified grading processing: the scores are divided into five levels: I (≥90), II (80~<90), III (70~<80), IV(60~<70), and V(<60). Then, using string type data such as "Road Construction", "Computer Fundamentals", "Probability Theory", and "Construction Project Management" to represent the courses of Road Construction, Computer Fundamentals, Data Mining Fundamentals, and Construction Project Management, the association rule of the algorithm is: Course1:III ⇒Course2: II, that is,"Road Construction:Grade III ⇒Computer Fundamentals:GradeII". Set smin=10%, cmin=50%. Based on the data structure of actual student grades, make the following modifications to the EI_Spriori algorithm:

(1) Add constraints on the order of courses in association rule mining, and teaching management personnel arrange the teaching sequence of courses based on the inherent connections between courses [14]. However, algorithms generate a large number of association rules that have no practical significance, such as "computer application foundation ⇒ road engineering construction", "computer application foundation, data mining foundation"⇒"road engineering construction, data mining foundation", etc. In response to this, constraints on the order of courses were added to the original algorithm and the algorithm was adjusted. After completing the pruning operation based on the minimum confidence level (cmin), the EIApriori algorithm checks the number of reverse arranged courses in the association rules and deletes association rules with a reverse order greater than 0.

Firstly, the standard order of course arrangement is determined according to the order of teaching, which includes pavement engineering construction, computer engineering foundation, data mining foundation, and construction project management. According to equation, the association rules generated between frequent itemsets can be directly checked for their inverse ordinals [15]. For association rules generated from frequent k-itemsets (k>1) and other itemsets, treat the left and right ends of the symbol "⇒" in the association rule as two parts, and check the number of new permutations formed by the sequence of courses on the left end of each symbol "⇒" and courses on the right end of each symbol"⇒".For example, for the association rule "Pavement Engineering Construction,Line Data Mining Foundation⇒Construction Project Management, Probability Theory", check if the number of reverse orders for "Pavement Engineering Construction, Computer Application Foundation, Data Mining Foundation, Construction Project Management, Data Mining Foundation, and Construction Project Management" is greater than 0. If the number of reverse orders for "Construction Project Management, Computer Application Foundation, and Data Mining Foundation" is 2, then delete this association rule.

(2)To find association rules that meet practical needs, constraints are set in the association rules for student grade ranges. Teachers sometimes pay attention to situations where students with a "pavement construction grade" of II have a grade greater than or equal to II. To this end, when applying the EI_Spriori algorithm to student performance correlation analysis, search for association rules between the right end of the

 

symbol "⇒" and the left end of the symbol "⇒", where the course grade level is higher (lower) than the symbol "⇒", that is course1:II ⇒ course2:I, course1:III, course2:III⇒course 3:I.

Specific steps:The method for generating association rules and calculating the confidence level of association rules using EI_Spriori remains unchanged, and filtering association rules are added to the algorithm; This method selects the level of the highest (lowest) score in the set of course grades on the left side of the symbol "⇒" as the benchmark, and judges whether the course grades on the right side of the symbol "⇒" are in the range greater (less) than the benchmark level one by one; If not present, delete the association rule; if present, retain the association rule. Taking equation as an example, the grade level II of"Course1"on the left end of the symbol"⇒" is selected as the benchmark to determine whether the grade of "Course 2" on the right end of the symbol "⇒"is in[II,I]. SinceI ∈ [II,I], the association rule is retained. Taking equation as an example, select the level III of "Course1" with the highest grade level on the left end of the symbol "⇒" as the benchmark to determine whether the grade level of "Course3" on the right end of the symbol "⇒"is in[III,I]. SinceI ∈ [III,I], the association rule is retained.

The partial association rules generated by the EI_Spriori algorithm are shown in Figure (3) From Table 5, it can be seen that the confidence level of the association rule "Road Construction:II⇒Data Mining Foundation:I" is 0.7172, indicating that if the score of the data mining foundation can reach 80 points or above, the probability of the score of probability theory reaching 90 points or above is relatively high. This is because the knowledge of road engineering construction plays a significant auxiliary role in learning the foundation of data mining. The knowledge of advanced mathematics is the foundation of data mining, and the idea of limits in data mining is the basis for learning concepts such as random variable distribution, data mining density, and the law of large numbers in data mining. The preparation of knowledge in advanced mathematics has a great impact on the performance of data mining foundation. From Table 5, it can also be seen that the confidence level of "Road Construction:V⇒Data Mining FundamentalsV" has reached0.7857. This is because there is a strong correlation between the grades of Computer Engineering Fundamentals and Advanced Computer Application, and the two courses have strong continuity in content. The latter is an elevation and improvement based on the former, and the teaching order of the two courses cannot be reversed or interrupted; The foundation of data mining:II ⇒construction project management: I"also contains certain rules. After statistical analysis of all similar association rule data generated, it was found that among the student performance data that met such association rules, multiple mathematics scores were either excellent or poor, and the reasons for this could not exclude the influence of personal talent and teaching quality. These association rules can provide important references for teaching management personnel and teachers. Through these association rules, they can have a clearer understanding of the inherent laws or connections of student performance, adjust the teaching process in a timely manner, and determine the direction of teaching reform.

**Table 5. Partial Association Rules Generated by EI_Spriori Algorithm**

| number | Association Rules | Confidence level |
|---|---|---|
| 1 | Road construction:II⇒Data mining foundation:I | 0.7172 |
| 2 | Computer Fundamentals: III⇒ Data Mining Fundamentals:III | 0.5787 |
| 3 | Fundamentals of Data Mining:II⇒Construction Project Management:I | 0.7311 |
| 4 | Road construction:V⇒ Building project management:V | 0.5167 |
| 5 | Computer Fundamentals:III⇒ Construction Project Management:II | 0.5347 |
| 6 | Fundamentals of Data Mining:V⇒Construction Project Management:V | 0.6514 |
| 7 | Road construction:V⇒Computer foundation:V | 0.7857 |
| 8 | Road engineering construction:I. Computer foundation: II⇒Building project management:I | 0.5461 |
| 9 | Computer Fundamentals:IV⇒Construction Project Management:III | 0.5346 |
| 10 | Computer Fundamentals:IV⇒Data Mining Fundamentals:II | 0.7152 |

 

## 7. Comparative Experiment

To demonstrate the efficiency of the algorithm proposed in this paper, comparative experiments were conducted with Apriori algorithm, MCApriori algorithm, and the algorithm in reference [12]. A correlation analysis was conducted on 7460 records of 4 course grades from 7460 students in a certain university's student performance system as a sample. The experiment was conducted in the Windows operating system, CPU3.70GHz Intel(R)Core(TM)i36100; The memory is 6GB. Implement algorithms using Python language. The stable running time after running the program is taken as the final result.

(1) Read 7460 records from the transaction database into the program, and set the minimum support at different scales as variable conditions to compare the running time of our algorithm with the other three algorithms. When dealing with databases with the same number of transactions, the execution efficiency of this algorithm and the other three algorithms is mainly reflected in the running time. The running time of different algorithms at different scales of minimum support is shown in Figure 3.

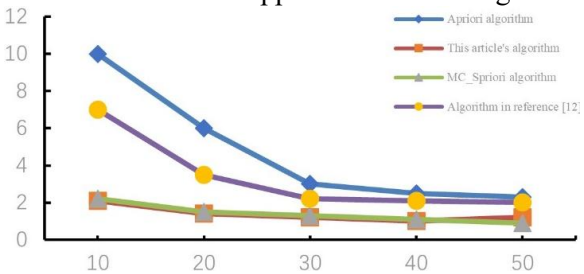


**Figure 3. The Running time of Different Algorithms at Different Scales of Minimum Support**

From Figure 3, it can be seen that the running time of our algorithm at different scales of minimum support is less than the other three algorithms; When the set minimum support is small, the difference in running time between our algorithm and Apriori algorithm is significant. This is because when the minimum support is small, due to the large number of frequent itemsets that need to be searched, the difference in the number of times the two algorithms scan the database and low dimensional frequent itemsets determines the length of the algorithm's running time. The algorithm in this article only needs to scan the transaction database once and no longer scans the low dimensional frequent itemsets multiple times, which consumes much less time than the Apriori algorithm. From Figure 3, it can be seen that the time consumption of the MC_Spriori algorithm is also lower than that of the Apriori algorithm. However, due to the higher time consumption of matrix operations compared to the vector inner product and "AND" operations of our algorithm, the overall performance of our algorithm is better.

(2) Several algorithms were set to run at smin=50%, and the running time of different algorithms for handling databases with different transaction numbers was compared. The results are shown in Figure 4.

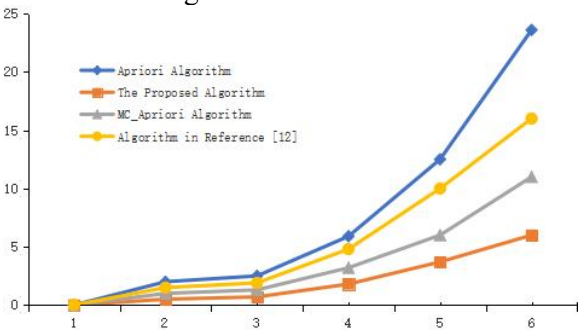


**Figure 4. Algorithm Running Time of Different Algorithms Processing Databases with Different Transaction Numbers**

From Figure 4, it can be seen that the other three algorithms take longer to process databases with different transaction numbers than the algorithm proposed in this paper; As the number of transactions to be processed increases, the difference in running time between the Apriori algorithm and the algorithm proposed in this paper becomes larger and larger. This is because the algorithm in this article calculates the support of items and discovers frequent itemsets through the inner product of transaction vectors and the "AND" operation, and removes a large number of invalid itemsets through pre pruning operations, greatly reducing the computational workload of the algorithm in processing itemsets.

The growth rate of the running time of the four algorithms with the number of transactions is shown in Table 6. From Table 6, it can be seen that as the number of transactions increases, the growth rate of the algorithm's running time in this paper gradually decreases and is lower than the other three algorithms; The growth rates of MC_Spriori algorithm and the algorithm in reference [12] are both on a downward trend, but compared to the algorithm in this paper, the degree of decrease is smaller, making it more

 

suitable for association rule analysis in larger databases.

**Table 6. The Growth Rate of Running Time of Four Algorithms with the Number of Transactions**

| Number of transactions/item | Growth rate of running time | | | |
|---|---|---|---|---|
| | Apriori algorithm | MCAprior algorithm | This article's algorithm | Algorithm in reference [12] |
| 1000~2000 | 0.514 | 0.527 | 0.501 | 0.583 |
| 2001~3000 | 0.510 | 0.524 | 0.467 | 0.566 |
| 3001~4000 | 0.517 | 0.476 | 0.374 | 0.552 |
| 4001~5000 | 0.547 | 0.445 | 0.372 | 0.543 |
| 5001~6000 | 0.579 | 0.435 | 0.322 | 0.531 |

## 8. Conclusion

The conclusion of the efficient implementation method of the Apriori algorithm in the performance comparison experiment shows that by using specific optimization techniques such as vector based storage structure and pre pruning, the efficiency of the Apriori algorithm can be significantly improved, and association rules that meet practical needs can be accurately found. In practical implementation, these methods reduce the number of times the database and low dimensional frequent itemsets are scanned, thereby reducing the computation time of the algorithm and the I/O operation time overhead of the database. For example, in the application scenario of student performance analysis, by adding constraints on the order of courses and the range of grades, the adjusted Apriori algorithm (such as EI_Spriori) can more effectively mine the association rules between courses, providing valuable auxiliary suggestions and decisions for teaching management and student course selection. Overall, the performance comparison experiment of Apriori algorithm shows that the efficient implementation method not only improves the execution efficiency of the algorithm, but also enhances its accuracy and practicality in practical applications.

Association rule mining algorithm is an important technique in the field of data mining, aimed at discovering the relationships between items in a dataset. The Apriori algorithm iteratively generates candidate itemsets and calculates their support to generate frequent itemsets and association rules. The core idea of this algorithm is to generate new candidate frequent itemsets using known frequent itemsets, and verify the frequency of the candidate frequent itemsets by scanning the database. The Apriori algorithm has poor performance in mining long frequent patterns because it requires multiple scans of the dataset and generates a large number of candidate sets. The Apriori algorithm may generate a large number of redundant rules, increasing the complexity of result analysis. Association rule mining algorithms have significant value in discovering interesting relationships between items in datasets. Apriori algorithm and FP Growth algorithm are two commonly used algorithms, each with its own advantages and disadvantages. Future research should focus on improving algorithm efficiency, adapting to different data characteristics, implementing multi-layer and multi-dimensional association rule mining, and providing interactive mining and result visualization functions. In addition, with the continuous development of machine learning technology, association rule mining algorithms will be applied and promoted in more fields.

## Acknowledgments

This project is a phased research outcome of the "Research on the Development and Application of an Adult Education Score Management System Based on PHP," which is funded by the 2026 Gansu Provincial University Teacher Innovation Fund (Project No. 2026A-406)

## References

[1] Wang, J, & Wang, Y. (2020). An Improved Apriori Algorithm for Mining Association Rules in Educational Data. Journal of Physics: Conference Series, 1646(1), 012016.

[2] Zhang, L, & Liu, Y. (2021). An Efficient Algorithm for Mining Association Rules with Constraints in Large Databases. International Journal of Database Theory and Application, 14(1), 1-12.

[3] Li, M, & Chen, L. (2022). A Novel Approach for Mining Frequent Itemsets with FP-Growth Algorithm in Healthcare Data. Journal of Healthcare Informatics Research,

 

6(1), 1-13.

[4] Wu, H, & Zhang, W. (2020). Data Association Rules Mining Method Based on Improved Apriori Algorithm. In Proceedings of the 4th International Conference on Big Data Research (pp. 1-6). DOI: 10.1145/3445945.3445948.

[5] Chen, X, & Liu, B. (2021). An Efficient Parallel Algorithm for Mining Association Rules in Big Data Environments. IEEE Transactions on Knowledge and Data Engineering, 33(1), 136-149.

[6] Liu, Y, & Wang, J. (2022). A Novel Hybrid Algorithm for Mining Association Rules in E-commerce Data. Journal of Electronic Commerce Research, 19(1), 56-70.

[7] Zhang, Q, & Li, H. (2020). An Improved Apriori Algorithm Based on Matrix Transposition for Mining Association Rules. Journal of Computer Science and Technology, 35(6), 1291-1302.

[8] Wang, L, & Sun, J. (2021). An Efficient Algorithm for Mining Association Rules in High-Dimensional Data. Information Sciences, 560, 19-33.

[9] Yang, M, & Zhang, Y. (2022). A Parallel FP-Growth Algorithm for Mining Association Rules in Large Databases. Journal of Parallel and Distributed Computing, 156, 1-12.

[10] Li, X, & Wang, Y. (2023). An Enhanced Apriori Algorithm for Mining Association Rules in Sparse Datasets. Knowledge-Based Systems, 253, 109406.

[11] Agrawal, R, Imieliński, T, & Swami, A. (1993). Mining association rules between sets of items in large databases. ACM SIGMOD Record, 22(2), 207-216.

[12] Han, J, Pei, J, & Yin, Y. (2000). Mining frequent patterns without candidate generation. ACM SIGMOD Record, 29(2), 1-12.

[13] Zaki, M. J. (1999). SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 35(1-2), 121-145.

[14] Srikant, R, & Agrawal, R. (1996). Mining generalized association rules. Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96), 407-419.

[15] Fournier-Viger, P, Wu, C.-W, Zida, S., & Nkambou, R. (2014). A survey of itemset mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(5), 351-377

 