An Efficient Algorithm for Mining Association Rules for Large Itemsets in Large Centralized Databases

Allan K.Y. Wong, S.L. Wu and L. Feng
Department of Computing, Hong Kong Polytechnic University,
Hunghom, Kowloon, Hong Kong, P.R. China, Email: csalwong@comp.polyu.edu.hk

ABSTRACT

The proposed algorithm is derived from the conventional Apriori approach with features added to improve data mining performance. These features are embedded in the encoding and decoding mechanisms. It has been confirmed by the preliminary test results that these features can indeed support effective and efficient mining of association rules in large centralized databases. The goal of the encoding mechanism is to reduce the I/O time for finding large itemsets, and to economize memory usage in a predictable manner. The decoding mechanism contributes to speed up the process of identifying different items in a transaction. The performance of three different decoding methods will be compared to demonstrate the potential gain delivered by any ingeniously devised decoding approach.

Keywords: Apriori algorithm, performance bottleneck, encoding, decoding, association rules

1. INTRODUCTION

In this paper an efficient algorithm for mining association rules [1,2] in large centralized databases is proposed. The new algorithm is derived from the conventional Apriori approach [3,4] by adding the encoding and decoding mechanisms. The encoding mechanism reduces the I/O time in tallying the large itemsets, and naturally economizes memory usage in a predictable manner. The decoding mechanism helps speed up the process of identifying specific items in a transaction. In order to demonstrate the impact by any ingeniously devised decoding approach on data mining performance, three different decoding methods will be empirically compared. The data used in the comparison were test results collected from different experiments performed on the same platform for sequential computation. This platform consists of a Sun Ultra machine, and a centralized database generated by the public IBM package described in [5]. The new algorithm can be represented conceptually by the eight steps as follows:

```
1. L_1=[frequent-1 itemsets] &   /* Read the whole database to identify Level 1's large itemsets $L_1$
     encode (whole_database);       and encode every transaction by the rule $V_t=\Sigma 2^X$ ; X=TRUE*/
2. For(k=2; L_{k-1}!={};K++){                              /* Loop until no more large itemset */
3    C_k=apriori_gen(L_{k-1});                    /*Generate itemsets $C_K$ from $L_{k-1}$ for level k*/
4.   For(m=0; m<no_transactions_in_encoded_database; m++){
5.      decode_transaction_m_for_level_k (encoded_database);          /*Work on $V_t$ value*/
6.      count_itemsets_for_C_k (transaction_m);}               /*For finding $L_k$ 's large itemsets*/
7.   L_k=[itemsets_for_C_k ≥Minsup]; }  /*Identify large itemsets (≥Minsup) for Level k→$L_k$)*/
8. Find_association_rules_for_large_itemsets (L_k);        /* From large itemsets at all levels */
```

The highlighted portions in step 1 and step 5 are the encoding and decoding mechanisms respectively. The decoding approach adopted in the latter mechanism is normally dictated by the philosophy implemented in the former. The proposed new algorithm can have many variants, which are characterized by the particular decoding methods incorporated. If the highlighted portions in step 1 and step 5 are deleted, then the new algorithm is reverted back to the conventional Apriori. The operational difference between the two algorithms is that in the new one the database is read only once for encoding purpose, as shown in Figure 1. No reading of the database is required again for the rest of the data mining life cycle. In the traditional Apriori, however, the whole database must be read once again when a deeper level denoted by L_{k+1} is mined for large itemsets. The encoding process (***encode (whole_database)***) transforms a large centralized database into a miniaturized and manageable form. It the transformation process, it produces the encoded value $V_t=\Sigma 2^X$ for every transaction in the database, where X marks the physical position of the item in the transaction. The basic logic for the decoding mechanism (the highlighted code segment) in step 5 is: *" if $(V_t \geq 2^X)\{V_t = V_t - 2^X$; transaction's_X_item = TRUE} "*.

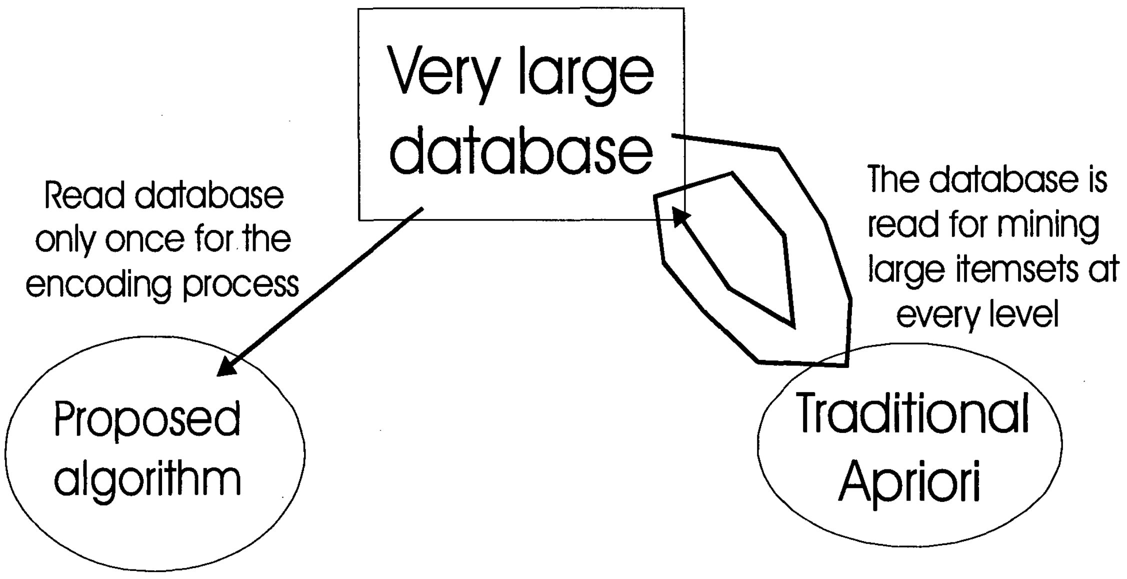
Very large database
Read database only once for the encoding process
The database is read for mining large itemsets at every level
Proposed algorithm
Traditional Apriori

Figure 1. The main difference between the two algorithms is in the I/O

An itemset is large because its chance of occurrence is greater than or equal to S (*minimum support count* or *Minsup*). An association rule X⇒Y holds provided that: (a) *X, Y* and *X∪Y* are large itemsets, and (b) *(support_count_of_X∪Y/support_count_of_X)* $\geq$ *minimum confidence C.* The symbol ⇒ abstracts the relationship *R* between *X* and *Y*.

2. THE NEW ALGORITHM

The development of the proposed algorithm is divided into three stages:

a) evaluate the I/O performance of the conventional Apriori approach,
b) devise the encoding method,
c) devise efficient decoding methods for the encoding measure above, and
d) test the overall data mining efficacy of the algorithm when the encoding and decoding methods are combined.

2.1 I/O Performance Evaluation

The aim is prove that an appropriate encoding method is necessary for improving the I/O performance of the traditional Apriori approach. The evaluation was performed with experiments in which the conventional Apriori algorithm was executed without the code segment for computation. This deletion of the code segment is illustrated by the absence of program statements after step 4 in the following pseudo-program:

```
1. $L_1$=[frequent-1 itemsets] &  /*Read whole database to identify Level 1's large itemsets $L_1$
       encode (whole_database);   and encode every transaction by the rule $V_t=\Sigma 2^x$ ; x=TRUE*/
2. For(k=2; $L_{k-1}$!={};K++){
3     $C_k$=apriori_gen($L_{k-1}$);          /*Generate itemsets $C_K$ from $L_{k-1}$ for level k*/
4 For(m=0; m<no_transactions_in_encoded_database; m++){ };
7.                 } /* The computation code segment is deleted from the traditional Apriori */
```

The results from many evaluation experiments with different database sizes indicate that the conventional Apriori approach consistently spends more than 70 % of its runtime on I/O accesses. Obviously, the performance bottleneck of the traditional approach is in the I/O operation. Therefore, the bit-encoding method is proposed for inclusion in the new algorithm to reduce the number of I/O accesses.

2.2 The Bit-Encoding Method

Encoding here means reorganizing and transforming a large database into a miniaturized and manageable structure to fulfill two objectives: (a) to reduce the number of I/O accesses in data mining, and (b) to speedup the decoding process. Generally speaking, such a data encoding process is an essential part of any typical knowledge discovery approach [6].

	2^0	2^1	2^2	2^3	
X →	1	2	4	8	
	CAT	DOG	RABBIT	CHICKEN	
1	0	1	1	1	14
2	0	0	1	0	4
3	1	0	1	0	5
4	0	1	0	1	10
5	1	0	0	1	9
...	...	...	...	...	

Encoded variable ($V_t=\Sigma 2^X$)

Table 1. Every item in a transaction is represented by its 2^X value (X is a bit's physical position)

The are two mandatory requirements for the bit-encoding method: (a) the database should be read only once within the whole life cycle of data mining, and (b) memory utilization should be maximized. In this encoding method every item in a transaction is represented by a 2^X value, where X marks the item's physical position in the transaction. The whole transaction is then represented by its unique encoded value. The bit-encoding concept that transforms a large database into its miniaturized and manageable form is exemplified by Table 1, where the encoded value for the first transaction is equal to $2^1 + 2^2 + 2^3 = 14$.

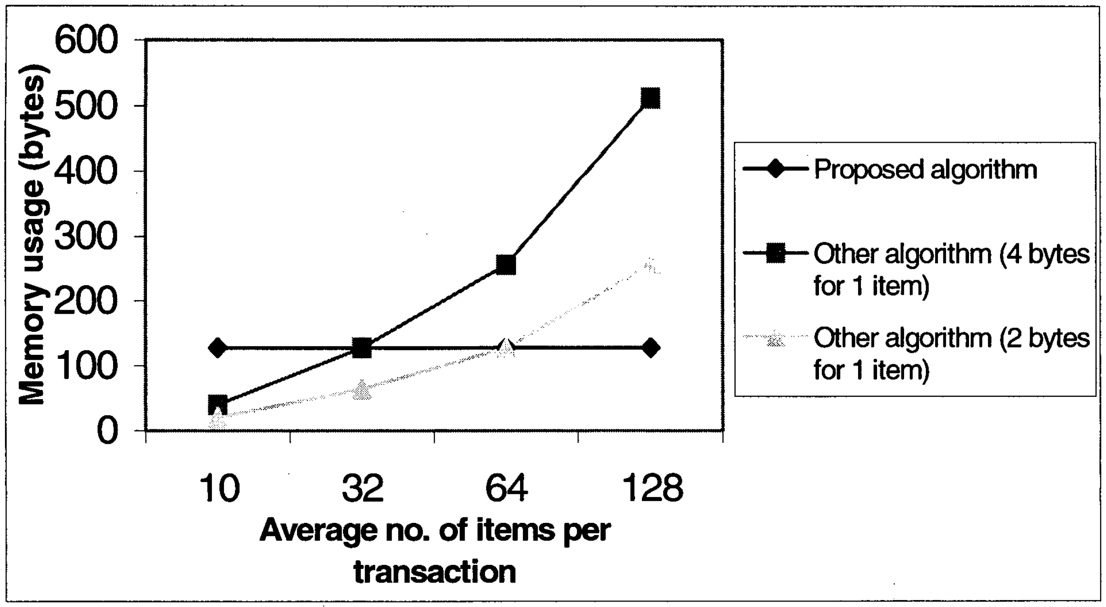

Figure 2. Efficient predictable memory utilization by the proposed encoding method

The bit-encoding method also maximizes memory utilization in a predictable manner. This is achieved for the following reasons: (a) memory usage is economized by encoding items in bits instead of bytes, and (b) bit representation is linear. On the contrary, it is necessary in the conventional Apriori to read and decode items that are encoded in a predefined number of bytes. For verifying the claim that memory usage is linear and economical, many experiments were performed with different database sizes generated by the public IBM package [4]. Figure 2 demonstrates the trend pinpointed by the results from these experiments. For the presented case the database had 100,000 transactions (D100K) constructed out of 1000 (N1000) possible items.

3. THREE DECODING METHODS

The decoding method can seriously affect the performance of the data mining process. For demonstration of this point, the test results from three different methods will be compared. The three methods are as follows:

a) *Basic decoding*: This algorithm of the following logic is executed repeatedly until i, which initialized to the encoded value V_t, is reduced 0:

```
if (i ≥ 2^X) then {i=i–2^X;
    X^th_item_in_transaction = TRUE;
      X=X–1} else X=X-1
```

In this approach the encoded value of 14 in Table 1 would represent the "*TRUE states*" for those items in the $2^{X=1}$, $2^{X=2}$, and $2^{X=3}$ positions of the first transaction. During the decoding operation the encoded transaction is scanned bit by bit starting from the most significant bit.

b) *Binary decoding*: This approach is based on the *basic decoding* approach except that the first step is to find "*the most significant TRUE position for X*" by binary search. The aim is to slash the scanning time of the *basic decoding* approach by half.

c) *Logarithmic decoding*: In this approach, which saves decoding time by eliminating the entire scanning process, the following logic is executed repeatedly until i (initialized to the encoded value V_t) is reduced to 0:

```
i = i – 2^{integer(log2(i))};
    [integer(log_2(i))]^th_item = TRUE
```

The *integer(log₂(i))* operation converts $log_2(i)$ into an integer by truncating whatever after the decimal point. For example, it yields the integer values of 3, 2, and 1 from the encoded value of $V_t = 14$ successively.

3.1 Test Results

The three aforementioned decoding methods were evaluated in different experiments performed on a Sun Ultra machine running on Unix. It was found that the *logarithmic approach* is consistently the most efficient among the three. The databases used in these experiments were generated by the same IBM package [4]. Each test case involved a specific combination of database size and number of items.

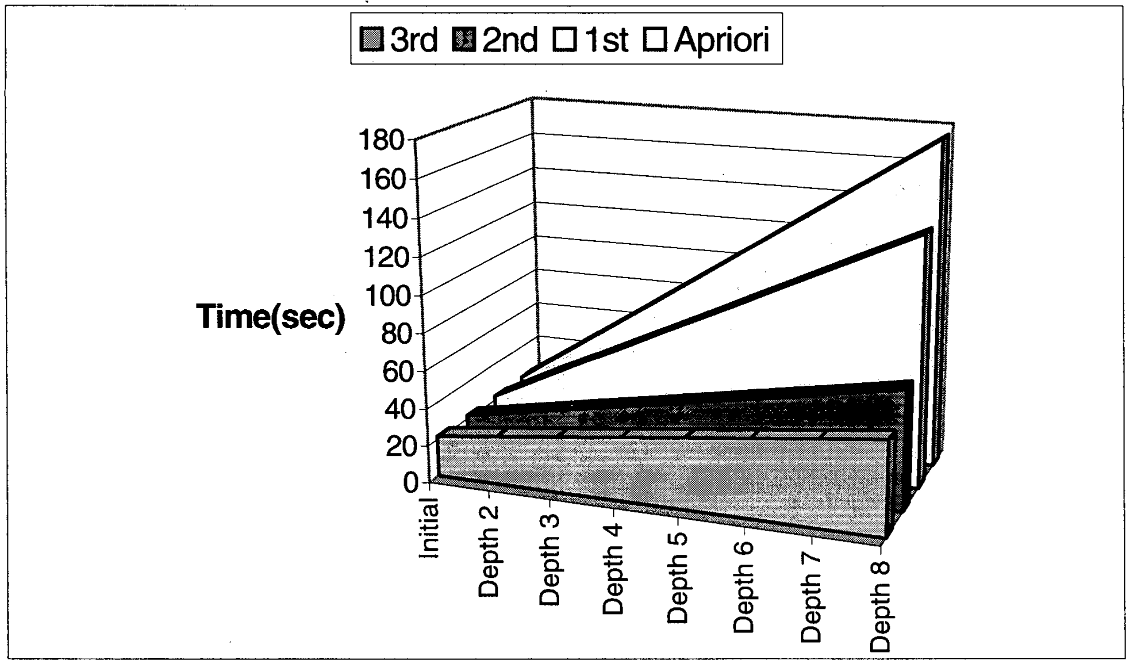

Figure 3. Comparison of four algorithms (T10.I4.D100K.N200)

Figure 3 demonstrates the difference in performance among the four algorithms, namely, the conventional Apriori (4 bytes is used in this case for encoding a data item), and three variants of the new algorithm. Each variant is backed either by the basic, the binary, or the logarithmic decoding method. The database for producing the test data for the comparison was generated by the public IBM package with the following statistics:

a) Total number of transactions in the database (D) is 1 00,000 (100K).
b) Average number of items per transaction (T) is 10.
c) Total number of items in the database (N) for forming transactions is 200.
d) For large itemsets: the number of transaction patterns is 1000; the average number items in a transaction (I) is 4.

The test programs for collecting the data for Figure 3 were written in C. In the data mining experiments, large itemsets were tallied up to the 8th level ($L_{k=8}$).

4. CONCLUSION

The aim of in this project is to improve the performance of the conventional Apriori algorithm that mines association rules. The approach to attain the desired improvement is to create a more efficient new algorithm out of the conventional one by adding the encoding and decoding mechanisms to the latter. In order to demonstrate the importance of efficient decoding to high data mining performance, three methods, namely, basic, binary, and logarithmic were evaluated. These three decoding methods were devised with respect to the bit-encoding approach that maximizes memory utilization in a predictable manner. The findings from different experiments have confirmed that the logarithmic decoding method is the most efficient among the three. It can speed up the data mining process significantly as demonstrated in the performance comparison. The imminent future work is to investigate how the logarithmic approach can be applied effectively and efficiently to large-scale distributed data mining, particularly in the Internet environment.

5. REFERENCES

[1] P. Adriaans and D. Zantinge, *Data Mining*, Addison Wesley, 1996.

[2] S.P. Jong, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 5, September/October 1997, pp. 813-825.

[3] R. Agrawal and J.C. Shafer, "Parallel Mining of Association Rules", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, December 1996, pp. 962-969.

[4] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", *Proceedings of International Conference on Management of Data (SIGMOD 93)*, May 1993, pp. 207-216

[5] IBM Almaden Research Center, "Synthetic Data Generation Code for Association and Sequential Patterns", *http://www.almaden.ibm .com/almaden/projects.html*, 1998

[6] "From Data Mining to Knowledge Discovery, An Overview", in *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, G. Piatetsky-Shapiro, P.Smyth, and R. Uthurusamy Ed., AAAI/MIT Press, 1996