A Novel Boolean Algebraic Framework for Association and Pattern Mining

Hatim A. Aboalsamh
Department of Computer Sciences
King Saud University
P.O. Box 2454 Riyadh 11451
Saudi Arabia
Hatim@ccis.ksu.edu.sa
http://faculty.ksu.edu.sa/Aboalsamh/

Abstract:- Data mining has been defined as the non- trivial extraction of implicit, previously unknown and potentially useful information from data. Association mining and sequential mining analysis are considered as crucial components of strategic control over a broad variety of disciplines in business, science and engineering. Association mining is one of the important sub-fields in data mining, where rules that imply certain association relationships among a set of items in a transaction database are discovered. In Sequence mining, data are represented as sequences of events, where order of those events is important. Finding patterns in sequences is valuable for predicting future events. In many applications such as the WEB applications, stock market, and genetic analysis, finding patterns in a sequence of elements or events, helps in predicting what could be the next event or element. At the conceptual level, association mining and sequence mining are two similar processes but using different representations of data. In association mining, items are distinct and the order of items in a transaction is not important. While in sequential pattern mining, the order of elements (events) in transactions (sequences) is important, and the same event may occur more than once. In this paper, we propose a new mapping function that maps event sequences into itemsets. Based on the unified representation of the association mining and the sequential pattern, a new approach that uses the Boolean representation of input database D to build a Boolean matrix M. Boolean algebra operations are applied on M to generate all frequent itemsets. Finally, frequent items or frequent sequential patterns are represented by logical expressions that could be minimized by using a suitable logical function minimization technique.

Keywords:- Sequence mining, data mining, association mining, Boolean association expressions, Boolean matrix, Association matrix.

1 Introduction

Data mining is the process of discovering potentially valuable patterns, associations, trends, sequences and dependencies in data [2, 3, 4, 5, 11, 17, 25, 16, 14]. Key business examples include web site access analysis for improvements in e-commerce advertising, fraud detection, screening and investigation, retail site or product analysis, and customer segmentation. Data mining techniques can discover information that many traditional business analysis and statistical techniques fail to deliver. Additionally, the application of data mining techniques further exploits the value of data warehouse by converting expensive volumes of data into valuable assets for future tactical and strategic business development. Management information systems should provide advanced capabilities that give the user the power to ask more sophisticated and pertinent questions. It empowers the right people by providing the specific information they need.

Association mining and sequence mining are two of the central tasks in data mining. Association mining is the process of producing association rules to express positive connections between items 2, 3, 4, 5, 6, 18, 21, 29, 9, 20], while sequence mining is the task of discovering frequent patterns among a large sequence database [1, 7, 8, 10, 11,12, 15, 23, 26, 27, 14, 33]. There is a connection between the two tasks in the way of extracting knowledge. The two tasks are related in the way they are handled. The Apriori algorithm [2] has been conceptually used in handling the two tasks.

The framework we develop derives from the observation that association mining and sequence mining are two similar processes on different representations of transactions. In association mining, the order of items in a transaction is not important, and all elements are distinct. While, in sequential pattern mining, the order of elements (events) in the transaction (sequence) record is important, and the same event may occur more than once. So, first we need to define a mapping function that maps a sequence of elements, which are ordered and could be repeated, into a set of elements, where ordering is not important, and elements are distinct.

In this paper, we propose a new mapping function that maps event sequence into events set. Based on the unified representation of the association mining and the sequential pattern, a new approach that uses the Boolean representation of the input database D is introduced. Database D is scanned only once to build a Boolean matrix M. M has n columns and k rows, where N is the number of items in i, and k is the number of transactions in D. A position (i,j) in M is 1 iff in transaction I, item j exists, and 0 otherwise. Boolean algebra operations are applied on M to generate all frequent itemsets. Finally, frequent items or frequent sequential patterns are represented by a logical expression that could be minimized by using a suitable logical function minimization technique. The Boolean approach does not depend on a specific association mining technique. In this paper, we use the apriori like algorithm for demonstrating the Boolean mining approach.

The rest of this paper is organized as follows: In section 2, we give the problem definition. The sequential pattern mapping is given in section 3. In section 4, the Boolean approach is presented. The frequent logical expressions are described in section 5. The performance study is given in section 6. Finally, the paper is concluded in section 7.

2 Problem Definition

2.1 Notation

I	Set of all items $\{i_1, i_2, \ldots, i_n\}$
2^I	Set of all possible transactions.
T	A transaction
t	An element in T, $t \in T$
D_T	Set of transactions $\{T_1, T_2, \ldots, T_k\}$
S	A sequence
s	an element in S
D_S	Set of sequences $\{S_1, S_2, \ldots, S_k\}$
$P(T)$	A mapping function that maps T into a sequence S
$Q(S)$	A mapping function that maps S into a transaction T
$\|x\|$	Number of 1's in vector x

2.2 Association Mining

Association mining that discovers dependencies among values of an attribute was first introduced by Agrawal et al.[2] and has emerged as a prominent research area. The association mining problem also referred to as the *market basket* problem can be formally defined as follows. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items as $S = \{s_1, s_2, \ldots, s_m\}$ be a set of transactions, where each transaction $s_i \in S$ is a set of items that is $s_i \subseteq I$. An *association rule* denoted by $X \Rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = \Phi$, describes the existence of a relationship between the two itemsets X and Y.

2.3 Sequence Mining

Sequential patterns mining [1, 8, 10, 11, 17, 22, 25, 27, 28, 16, 9] is the process of finding frequent sequential patterns in large transaction databases. It can be defined as followed. Let I = $\{i_1, i_2, \ldots, i_n\}$ be a set of distinct items, and S = $e_1, e_2, \ldots, e_m$ be an ordered sequence of events, where $\forall e_i \epsilon S,\ 1 \le i \le m, e_i \epsilon I$. In a sequence S, an event e may occur more than once, and the ordering of events in S is important, i.e., sequence AB in not the same as sequence BA. The number, l, of events in S is called the length of sequence S, and S is called an l-sequence. As an example, S= BBBCCAD is an 8-sequence of events. An event sequence $S_1 = i_{a_1}, i_{a_2}, \ldots, i_{a_m}$, is contained in an event sequence $S_2 = i_{b_1}, i_{b_2}, \ldots, i_{b_k}$, $m \le k$, if $i_{a_1} \le i_{b_1}, i_{a_2} \le i_{b_2}, \ldots, i_{a_m} \le i_{b_m}$. If an event sequence S_1 is contained in event sequence S_2,

S_1 is called an event subsequence of S_2, and S_2 is called an event super-sequence of S_1. S_1 is an event subsequence of S_1, and S_1 is an event super-sequence of S_1.

3 Sequential Pattern Mapping

Let I = {i_1, i_2, … , i_n } be a set of distinct events [30], and S = e_1, e_2,…, e_m be an ordered sequence of events, where $\forall e_i \epsilon S,\ 1 \le i \le m, e_i \epsilon I$. In a sequence S, the order of events is important and an event e could occur more than once.

Example 3.1: Let I={A,B,C,D} be the set of all possible events in database D, where D is given in Table 1.

ID	*Sequence*
1	ADCCABB
2	DCACC
3	BABCCA
4	DCAA
5	DACCC

Table 1

In the above database D, the first transaction T_1 contains each of the events A, B and C twice, and sequence 'ADC' in transaction 1 is not the same as sequences 'DCA' and 'DAC' in transactions 4 and 5, respectively. As a sequence representation, the two transactions 2 and 5 are not the same, while if we consider D as a database of sets of items (ignore the repeated events), transactions 2 and 5 are identical transactions.

To unify the two representations of sets and sequences, we map the sequences into sets by identifying sequence elements by their positions. For a sequence S = e_1 e_2,…, e_m, each element $e_j, 1 \le j \le m,$ could be one of the possible n elements in I. If all possible elements at position j are labeled by j, then instead of writing S= i_{a_1}, i_{a_2},,…, i_{a_m}, S could be written as S = $i_{a_1,1}$, $i_{a_2,2}$,,…, $i_{a_m,m}$. In this case, event $i_{a_j}, 1 \le j \le m,$ that has order j in sequence S, would have a label j describing its position.

Example 3.2: For the database D given in example 3.1, the mapping function applied on sequence transactions S produces the database D_T given in Table 2.

In example 3.2, sequence transaction S_i is mapped into item transaction T_i by attaching element position j of elements in S_i to be mapped into T_i.

ID	*S*	*T*
1	ADCCABB	A_1 D_2 C_3 C_4 A_5 B_6 B_7
2	DCACC	D_1 C_2 A_3 C_4 C_5
3	BABCCA	B_1 A_2 B_3 C_4 C_5 A_6
4	DCAA	D_1 C_2 A_3 A_4
5	DACCC	D_1 A_2 C_3 C_4 C_5

Table 2

Definition 3.1: Let I = {i_1, i_2, … , i_n } be a set of distinct events, then for a sequence S = e_1, e_2,…, e_m , the transaction mapping function P applied on S, P: S→T is defined as

$$P(S)=\{t_{i,j} \mid \exists e_j \epsilon\ S \wedge t_i = e_j \wedge\ t_{i,j} = t_i\ \}$$

According to definition 3.1, $t_{i,j} = t_i$ for all $1 \le j \le m$, and $t_i = t_{i,j}$ for all $1 \le j \le m$.

Lemma 3.1: The order of elements in T=P(S), is not important.

Proof: Since each element $e_j \epsilon\ S$ is mapped to element $t_{i,j}\ \ \epsilon\ T$, where position j is preserved, if we change the order of elements in T, they are still identified by their positions.

Definition 3.2: Let I = {i_1, i_2, … , i_n } be a set of distinct events, then for a transaction T = $\{t_{i,j} \mid\ t_{i,j} = t_i\ 1 \le i \le n, t_i \epsilon I\}$ of cardinality m and ordered according to j, the sequence mapping function Q applied on T, Q: T→S is defined as

$$Q(T)= e_1, e_2,…, e_m, e_j = t_{i,j} \text{ and } t_i \epsilon\ I$$

Definition 3.3: Let I = {i_1, i_2, … , i_n } be a set of distinct events, then for a transaction T = $\{t_{i,j} \mid\ t_{i,j} = t_i\ 1 \le i \le n, t_i \epsilon I\}$ of cardinality m and ordered according to j, the mapping factor *mf* of event t_i; $mf(t_i)$ is defined as the mapping space for t_i; i.e., $\| t_{i,j} \|$

Example 3.3: For the database D_T given in table 2, the mapping function applied on item transactions T's produces the database D_s shown in table 3.

ID	T	*Orderd T*	S
1	A_3 D_4 C_1 C_5 A_2 B_7 B_6	$C_1A_2A_3D_4C_5B_6B_7$	CAADCBB
2	D_1 C_5 A_4 C_3 C_2	$D_1C_2C_3A_4C_5$	DCCAC
3	B_6 A_1 B_5 C_2 C_3 A_4	$A_1C_2C_3A_4B_5B_6$	ACCABB
4	D_1 C_3 A_2 A_4	$D_1A_2C_3A_4$	DACA
5	D_4 A_5 C_3 C_2 C_1	$C_1C_2C_3D_4A_5$	CCCDA

Table 3

4 The Boolean Approach

Given a set of items $I = \{i_1, i_2, \ldots, i_n\}$, a transaction t is defined as a set of items such that $t \in 2^I$, where $2^I = \{\varnothing, \{i_1\}, \{i_2\}, \ldots, \{i_n\}, \{i_1, i_2\}, \ldots, \{i_1, i_2, \ldots, i_n\}\}$. *Let* $T \subseteq 2^I$ be a given set of transactions $\{T_1, T_2, \ldots, T_k\}$.

Definition 4.1: Let $D_T = \{T_1, T_2, \ldots., T_k\}$ be the set of all transactions in database D, with k is the number of such transactions. The *Association Matrix* $M(D_T)$ is a $K \times N$ Boolean matrix where each element $a_{i,j}$, $1 \le i \le K$, $1 \le j \le N$, $N \le n$, in $M(D_T)$ is defined as

$$a_{i,j} = \begin{cases} 1 & if\ t_j \in T_i \\ 0 & other\ \ wise \end{cases}$$

Definition 4.2: Let $D_T = \{T_1, T_2, \ldots., T_k\}$ be the set of all transaction in database D, with K is the number of such transaction. For element e_j, $1 \le j \le N$, Vector

$$f_j = \begin{bmatrix} a_{1,j} \\ a_{2,j} \\ \\ \\ a_{K,j} \end{bmatrix}$$

is called the projection vector of e_j, and

$$\text{support}(e_j) = \left\| f_j \right\|_1 .$$

Lemma 4.1: For elements e_j and e_k with projection vectors f_j and f_k,

$$\text{support}(e_j,e_k) = \|f_j \wedge f_k\|.$$

Proof: Following the definition of Boolean operators, the proof of lemma 4.1 is straightforward.
By generalizing lemma 4.1, we get the following lemma.

Lemma 4.2: For elements $e_1, e_2, \ldots, e_m$, with projection vectors $f_1, f_2, \ldots, f_m$,

$$\text{support } (e_1, e_2, \ldots, e_m) = \|f_1 \wedge f_2 \wedge \ldots \wedge f_m\| .$$

Proof: By induction,
for m=2 , support $(e_1, e_2) = \|f_1 \wedge f_2\|$.
suppose it is true for m=n,

$$\text{support } (e_1, e_2, \ldots, e_n) = \|f_1 \wedge f_2 \wedge \ldots \wedge f_n\|,$$

then

$$\text{support } ((e_1, e_2, \ldots, e_n), e_{n+1}) = \|((f_1 \wedge f_2 \wedge \ldots \wedge f_n) \wedge f_{n+1}\|. \text{ Q.E.D}$$

Example 4.1: Let I={A,B,C,D, E, F} be the set of all possible events in database D, where D is given in table 4.

ID	*Transaction*
1	ADCF
2	ABCD
3	ACEF
4	BCDE
5	CDEF

Table 4

The corresponding Boolean matrix M is given in table 5.

ID	A	B	C	D	E	F
1	1	0	1	1	0	1
2	1	1	1	1	0	0
3	1	0	1	0	1	1
4	0	1	1	1	0	0
5	0	0	1	1	1	1

Table 5

The projection vectors of A, B, C, D, E, and F are given in table 6.

ID	f_A	f_B	f_C	f_D	f_E	f_F
1	1	0	1	1	0	1
2	1	1	1	1	0	0
3	1	0	1	0	1	1
4	0	1	1	1	0	0
5	0	0	1	1	1	1
$\|f\|$	3	2	5	4	2	3

Table 6

The projection vectors of AB, BCD, and BDEF are given in table 7.

ID	f_{AB}	f_{BCD}	f_{BDEF}
1	0	0	0
2	1	1	0
3	0	0	0
4	0	1	0
5	0	0	0
$\|f\|$	1	2	0

Table 7

The above Boolean expressions for calculating support using projection vectors could handle itemsets calculations where itemsets AB and BA are treated in the same way. For sequence database D_S, with maximum record length N, and by using the transaction mapping function discussed in section 3, event A will be represented by items A_i, $1 \le i \le N$. Projection vector

$$f_A = \vee^{1 \le i \le N}(f_{A_i}) \quad .$$

Also, sequence AB will be represented by sequences A_iB_j , $1 \le i \le N$, $i < j \le N$. Projection vector

$$f_{AB} = \vee^{1 \le i \le N, i < j \le N}(f_{A_i} \wedge f_{B_j}) \quad .$$

Example 4.2: Let I={A,B,C,D} be the set of all possible events in database D, where D is given in table 8. The association mapping of D is given in table 9, and the corresponding Boolean matrix M is given in table 10.

The projection vector of A, given in table 11, is the result of applying the Boolean operation $\vee$ on f_{A_1} and f_{A_3}.

ID	*Transaction*
1	CDA
2	ADCD
3	ADC
4	BCAAD
5	CBAD

Table 8

ID	*Transaction*
1	C_1 D_2 A_3
2	A_1 D_2 B_3C_4 D_5
3	A_1 D_2 C_3
4	B_1 C_2 A_3 A_4 D_5
5	C_1 B_2 A_3 D_4

Table 9

ID	A_1	A_3	A_4	B_1	B_2	B_3	C_1	C_2	C_3	C_4	D_2	D_4	D_5
1	0	1	0	0	0	0	1	0	0	0	1	0	0
2	1	0	0	0	0	1	0	0	0	1	1	0	1
3	1	0	0	0	0	0	0	0	1	0	1	0	0
4	0	1	1	1	0	0	0	1	0	0	0	0	1
5	0	1	0	0	1	1	1	0	0	0	0	1	0

Table 10

In table 10, the mapping factors of events A, B, C, and D are 3, 3, 4, and 3, respectively.

ID	f_A		f_{A_1}		f_{A_3}		f_{A_4}
1	1		0		1		0
2	1		1		0		0
3	1	=	1	$\vee$	0	$\vee$	0
4	1		0		1		1
5	1		0		1		0
$\|f\|$	5		--		--		--

Table 11

The projection vectors of A, B, C, and D are given in table 12.

ID	f_A	f_B	f_C	f_D
1	1	0	0	1
2	1	0	1	1
3	1	0	1	1
4	1	1	1	1
5	1	1	1	1
F	5	2	4	5

Table 12

The projection vectors of AB are given in table 13.

ID	f_{AB}		f_{A_1}		f_{B_2}		f_{A_1}		f_{B_3}	
1	0		0		0		1		0	
2	0		1		0		0		1	
3	0	= (	1	$\wedge$	0	) $\vee$ (	0	$\wedge$	0	)
4	0		0		0		1		0	
5	1		0		1		1		1	
$\|f\|$	1		--		--		--		--	

Table 13

Lemma 4.3: For elements e_l and e_k in a sequence S, with projection vectors

$$\{f_{l,1}, f_{l,2}, \dots, f_{l,N}\} \text{ and } \{f_{k,1}, f_{k,2}, \dots, f_{k,N}\},$$

respectively,

$$\text{support}(e_l, e_k) = \left\| \vee^{1 \le i \le N, i < j \le N} (f_{e_{l,i}} \wedge f_{e_{k,j}}) \right\|$$

Proof: Following the definition of Boolean operators, the proof of lemma 4.3 is straightforward.

Lemma 4.4: For elements $e_1, e_2, \dots, e_n$ in a sequence database D_S, with projection vectors

$$\{f_{1,1}, f_{1,2}, \dots, f_{1,N}\}, \{f_{2,1}, f_{2,2}, \dots, f_{2,N}\}, \text{and } \{f_{n,1}, f_{n,2}, \dots, f_{n,N}\}$$

, respectively,

$$\text{Support}(e_1, e_2, \dots, e_n) = \left\| \vee^{1 \le i_1 \le N, i_1 < i_2 \le N, \dots, i_{n-1} < i_n \le N} (f_{e_{1,i_1}} \wedge f_{e_{2,i_2}} \wedge \dots \wedge f_{e_{n,i_n}}) \right\|$$

Proof: Following the definition of Boolean operators, the proof of lemma 4.4 is straightforward.

In this paper, we have modified the Apriori like algorithm to demonstrate the Boolean approach used for data mining. The approach is assuming that the mining problem is already in the association mining domain. For sequence mining, we should include two modules to the algorithm. The first module, maps the sequence database into an itemset database, while the second module maps back the association mining results into the sequence mining domain. The Boolean approach is given in figure 1.

Generate large itemsets using Boolean approach
Input: Transaction database D_T
Output: large itemsets L

$\boldsymbol{begin}$
  $\boldsymbol{for\ (k = 2; L_{k-1} \neq \emptyset; K++)}$
    $\boldsymbol{begin}$
      $\boldsymbol{L_k = \emptyset}$
      $\boldsymbol{C_k = apriori - gen(L_{k-1});}$
      $\boldsymbol{for\ all\ c \in C_k\ do}$
        $\boldsymbol{begin}$
          $\boldsymbol{calculate\ f_c\ as\ given\ in\ lemma\ 4.4\ ;}$
          $\boldsymbol{L_k = L_k \cup \{c \mid f_c \ge minsup\};}$
        $\boldsymbol{end;}$
      $\boldsymbol{end;}$
      $\boldsymbol{L = \cup_K L_K}$
$\boldsymbol{end.}$

Fig. 1 Association Mining Using Boolean Approach

5 Frequent Logical Expressions

In the previous section, we have shown that frequent items or frequent sequential patterns are represented by a logical expression. In order to maximize the performance of our technique, we need to minimize the generated Boolean expressions by using a suitable minimization technique.

Example 5.1: Using the Boolean matrix M in Table 10, with minimum support 2, the resulted frequent sequences are

1-frequent sequences:

$(A_1 \vee A_3 \vee A_4) \vee (B_1 \vee B_2) \vee (C_1 \vee C_2 \vee C_3) \vee (D_2 \vee D_4 \vee D_5)$

2- frequent sequences:

$(A_1 \wedge C_2) \vee (A_1 \wedge C_3) \vee (A_1 \wedge C_4) \vee (A_3 \wedge C_4) \vee (A_1 \wedge D_2) \vee (A_1 \wedge D_4) \vee (A_1 \wedge D_5) \vee (A_3 \wedge D_4) \vee (A_3 \wedge D_5) \vee (A_4 \wedge D_5) \vee (B_1 \wedge A_3) \vee (B_1 \wedge A_4) \vee (B_2 \wedge A_3) \vee (B_2 \wedge A_4) \vee (B_3 \wedge A_4) \vee$

$(B_1 \wedge D_2) \vee (B_1 \wedge D_4) \vee (B_1 \wedge D_5) \vee (B_2 \wedge D_4) \vee (B_2 \wedge D_5) \vee (B_3 \wedge D_4) \vee (B_3 \wedge D_5) \vee (C_1 \wedge A_3) \vee (C_1 \wedge A_4) \vee (C_2 \wedge A_3) \vee (C_2 \wedge A_4) \vee (C_1 \wedge D_2) \vee (C_1 \wedge D_4) \vee (C_1 \wedge D_5) \vee (C_2 \wedge D_4) \vee (C_2 \wedge D_5) \vee (C_3 \wedge D_4) \vee (C_3 \wedge D_5) \vee (D_2 \wedge C_3)$

3- frequent sequences:

$(A_1 \wedge D_2 \wedge C_3) \vee (A_1 \wedge D_2 \wedge C_4) \vee (B_1 \wedge A_3 \wedge D_4) \vee (B_1 \wedge A_3 \wedge D_5) \vee (B_2 \wedge A_3 \wedge D4) \vee (B_2 \wedge A_3 \wedge D_5) \vee (C_1 \wedge A_3 \wedge D_4) \vee (C_1 \wedge A_3 \wedge D_5) \vee (C_2 \wedge A_3 \wedge D_4) \vee (C_2 \wedge A_3 \wedge D_5) \vee C_3 \wedge A_4 \wedge D_5)$

After doing hand Boolean minimization, the Boolean expression of frequent sequences in D is

$(A_1 \wedge C_2) \vee (A_3 \wedge C_4) \vee (A_1 \wedge D_4) \vee (A_1 \wedge D_5) \vee (B_1 \wedge A_4) \vee (B_2 \wedge A_4) \vee (B_3 \wedge A_4) \vee (B_1 \wedge D_2) \vee (B_3 \wedge D_4) \vee (B_3 \wedge D_5) \vee (C_1 \wedge D_2) \vee (C_3 \wedge D_4) \vee (C_3 \wedge D_5) \vee (D_2 \wedge C_3) \vee (A_1 \wedge D_2 \wedge C_3) \vee (A_1 \wedge D_2 \wedge C_4) \vee (B_1 \wedge A_3 \wedge D_4) \vee (B_1 \wedge A_3 \wedge D_5) \vee (B_2 \wedge A_3 \wedge D_4) \vee (B_2 \wedge A_3 \wedge D_5) \vee (C_1 \wedge A_3 \wedge D_4) \vee (C_1 \wedge A_3 \wedge D_5) \vee (C_1 \wedge A_4 \wedge D_5) \vee (C_2 \wedge A_3 \wedge D_4) \vee (C_2 \wedge A_3 \wedge D_5) \vee (C_2 \wedge A_4 \wedge D_5) \vee (C_3 \wedge A_4 \wedge D_5)$

Which can be written as

$(A \wedge C) \vee (A \wedge D) \vee (B \wedge A) \vee (B \wedge D) \vee (C \wedge D) \vee (D \wedge C) \vee (A \wedge D \wedge C) \vee (B \wedge A \wedge D) \vee (C \wedge A \wedge D)$

Or

AC, AD, BA, BD, CD, DC, ADC, BAD, CAD

The classical Karnaugh [15] maps approach could be used in minimizing Boolean expressions. The only problem of the Karnaugh maps approach is its limited task that is suited for at most 6 input variables and practical only for up to 4 variables. It is even harder to be carried out in product term sharing for multiple output functions. Also, the Karnaugh maps technique can not be automated in a computer program. In Boolean data mining solution, we are usually concerned with the large number of variables.

Quine and McCluskey [24] introduced the tabular method as the first alternative method to Karnaugh maps. It has two phases. It starts with the truth table for a set of logic expressions, a set of prime implicants is composed. In the second phase, A systematic procedure is followed to find the smallest set of prime implicants of output functions [19].

Although this Quine-McCluskey algorithm is very well suited to be implemented in a computer program.It produces the optimal logical expressions. The only problem we have with Quine-McCluskey algorithm is that it is not efficient in terms of processing time and memory usage. Increasing the number of variables by even one more variable to the function will double processing time and memory usage. The truth table length increases exponentially with the number of variables. So, the Quine-McCluskey method is practical only for functions with a limited number of input variables and output functions.

A different approach to this issue is the Espresso approach [13]. Instead of expanding a logic function into minterms, the approach processes cubes, representing the product terms. The output from this approach is not the optimum one, but it is very closely approximated, while the solution is always free from redundancy. This approach is more efficient than many of the other methods. It reduces memory computation time by several orders of magnitude. The important part of this approach is it has no restriction on the number of variables, output functions and product terms of a combinational function block. In our experimental study, we have used the Espresso approach for minimization.

6 Performance Study

In this section we present the performance results that have been collected. As a mining algorithm, we have used an Apriori-like algorithm as a local procedure to generate frequent itemsets (sequences). We would like to emphasize on the fact that our approach does not depend on the approach used to generate frequent itemsets.

We ran our experiments on a 2.4 GHz machine, with 4 GB of RAM and running windows Vista. The databases used were generated synthetically, to evaluate the performance of the algorithms over a range of data parameters. For 1000 distinct events (items), we assume that average value of the mapping factor for each event is set to 20.

The range of database sizes is varied between 10,000 to 100,000 transactions. The results were evaluated against the BIDE algorithm [26], which is one of the latest algorithms designed for mining closed frequent sequences. Our study is based on the total mining time consumed for the whole data mining process. We have measured the mining time for the different database sizes on various minimum support values ranges between 0.2% and 5%.

Figures 2, 3 and 4 show the execution times for the three synthetic databases of sizes 100,000, 50000, and 10000 transactions, respectively, for minimum support values varies from 0.5% to 5%. As the minimum support decreases, the execution times of the two algorithms increase because of increases in the total number of candidate and large itemsets. In figures 2, 3, 4, the performance results are shown.

Figure 5 shows the execution times for minimum support value equals 1%, and database sizes between 10000 and 100000 transactions. . As the database size increases, the execution times of all the algorithms increase because of increases in the total number of candidate and large itemsets. In figure 5, the performance results are shown.

It is clear that from the preliminary results depicted in figures 2-5, our implementation that is based on the Boolean approach has a better performance than the BIDE algorithm.

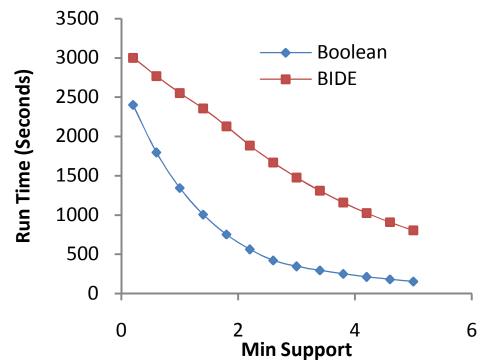

Fig. 2 Database Size is 100000 transactions

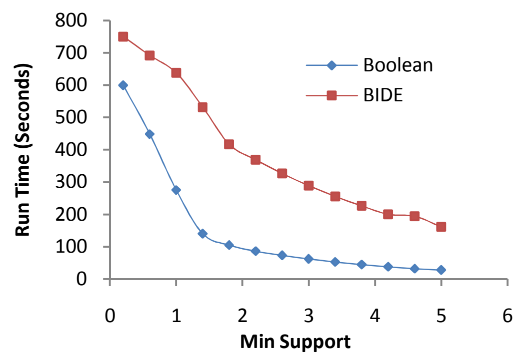

Fig. 3 Database Size is 50000 transactions

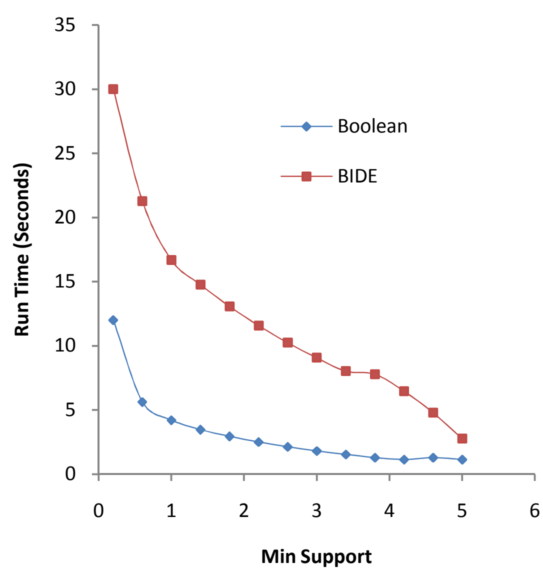

Fig. 4 Database Size is 10000 transactions

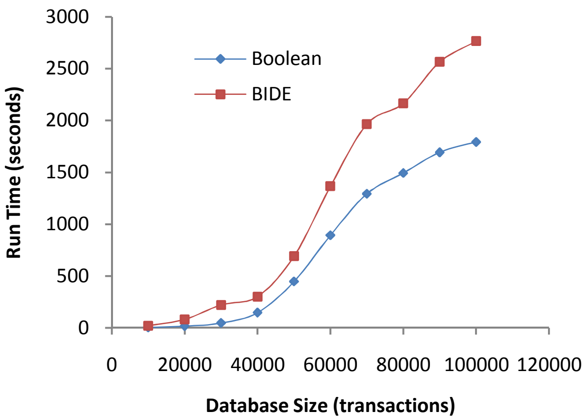

Fig. 5 Minimum Support equals 1%

7 Conclusions

In this paper, we have proposed a new mapping function that maps event sequence into events set. Based on the unified representation of the association mining and the sequential pattern, a new approach that uses the Boolean representation of the input database D has been introduced, where database D is scanned only once to build a Boolean matrix M. M has N columns and K rows, where N is the number of items in I, and K is the number of transactions in D. A position (i,j) in M is 1 iff in transaction I, item j exists, and 0 otherwise. Boolean algebra operations are applied on M to generate all frequent itemsets. Finally, frequent items or frequent sequential patterns has been represented by a logical expression that could be minimized by using a suitable logic function minimization technique. The Boolean approach does not depend on the association mining methodology. In this paper, we have used the apriori like algorithm for demonstrating the Boolean mining approach.

In the performance results, we have shown that our implementation that is based on the Boolean approach has a better performance than the BIDE algorithm.

References:

[1] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, Sequential Pattern Mining Using a Bitmap presentation, Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '02), pp. 429-435, July 2002.

[2] R. Agrawal, T. Imilienski, and A. Swami, Mining Association Rules between Sets of Items in Large Databases, Proc. of the ACM SIGMOD Int'l Conf. On Management of data, May 1993.

[3] R. Agrawal and R. Srikant, Mining sequential patterns, In 11th Int'l Conference on Data Engineering, 1995.

[4] R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules, Proc. Of the 20th VLDB Conference, Santiago, Chile, 1994.

[5] R. Agrawal, J. Shafer, Parallel Mining of Association Rules, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, Dec. 1996.

[6] C. Agrawal, and P. Yu, Mining Large Itemsets for Association Rules, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1997.

[7] Y. L. Chen, H. P. Kao, and M.-T. Ko, Mining dag patterns from dag databases, In Web Age Information Management Conference, 2004.

[8] Y. Chi, Y. Yang, and R.R. Muntz. Hybridtreeminer: An efficient algorithm for mining frequent rooted trees and free trees using canonical forms. In 16th International Conference on Scientific and Statistical Database Management, 2004.

[9] Renata Ivancsy , Istvan Vajk, Efficient sequential pattern mining algorithms, Proceedings of the 4th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering Data Bases, p.1-6, February 13-15, 2005, Salzburg, Austria

[10] X. Ji, J. Bailey, and G. Dong, Mining Minimal Distinguishing Subsequence Patterns with Gap Constraints, Proc. IEEE Int'l Conf. Data Mining (ICDM '05), pp. 194-201, Nov. 2005.

[11] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation, In ACM SIGMOD Conference on Management of Data, 2000.

[12] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining, Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '00), pp. 355-359, Aug. 2000.

[13] J.P. Hayes, Digital Logic Design, Addison Wesley, 1993.

[14] Ming-Chuan Hung, Shi-Qin Weng, Jungpin Wu and Don-Lin Yang. Efficient Mining of Association Rules Using Merged Transactions Approach, WSEAS TRANSACTIONS on COMPUTERS, Issue 5, Volume 5, 916-923, May 2006.

[15] Karnaugh, Maurice, The Map Method for Synthesis of Combinational Logic Circuits, Transactions of American Institute of Electrical Engineers part I **72** (9): 593–599**.**, November 1953.

[16] Ioannis N. Kouris, Christos H. Makris, Athanasios and K. Tsakalidis. A Spatiotemporal View of Transactional Data for Data Mining, WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 8, Volume 2, 1179-1183, August 2005.

[17] Jiangbo Liu and Yufen Huang. Healthcare Data Analysis using Data Mining Algorithms, WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 5, 1389-1397, , June 2006.

[18] H. Mannila, H. Toivonen, and A. Verkamo, Efficient Algorithms for Discovering Association Rules, AAAI Workshop on Knowledge Discovery in databases (KDD-94) , July 1994.

[19] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill Science Engineering, 1994.

[20] Stergios Papadimitriou , Seferina Mavroudi , Spiridon Likothanassis, Mutual information clustering for efficient mining of fuzzy association rules with application to gene expression data analysis, Proceedings of the 9th WSEAS International Conference on Computers, p.1-9, July 14-16, 2005, Athens, Greece.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, Discovering Frequent Closed Itemsets for Association Rules, Proc. Int'l Conf. Database Theory (ICDT '99), pp. 398-416, Jan. 1999.

[22] J. Pei, J. Han, and R. Mao, CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets, Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD '00), pp. 21-30, May 2000.

[23] J. Pei, J. Liu, H. Wang, K. Wang, P.S. Yu, and J. Wang, Efficiently Mining Frequent Closed Partial Orders, Proc. IEEE Int'l Conf. Data Mining (ICDM '05), pp. 753-756, Nov. 2005.

[24] W.V. Quine, A Way to Simplify Truth Functions, American Mathematics Monthly, 62: 627-631, 1955.

[25] R. Srikant and R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements, Proc. Int'l Conf. Extending Database Technology (EDBT '96), pp. 3-17, Mar. 1996.

[26] J. Wang, J. Han, C. Li, Frequent Closed Sequence Mining without Candidate Maintenance, IEEE Transactions on Knowledge and Data Engineering, 19(8):1042-1056, August 2007.

[27] J. Wang, J. Han, and J. Pei, CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets, Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD'03), pp. 236-245, Aug. 2003.

[28] X. Yan, J. Han, and R. Afshar, CloSpan: Mining Closed Sequential Patterns in Large Databases, Proc. SIAM Int'l Conf. Data Mining (SDM '03), pp. 166-177, May 2003.

[29] M.J. Zaki. Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering, 12(3):372–390, 2000.

[30] M. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, Machine Learning, vol. 42, pp. 31-60, 2001.

[31] M. Zaki and C. Hsiao, CHARM: An Efficient Algorithm for Closed Itemset Mining, Proc. SIAM Int'l Conf. Data Mining (SDM '02), pp. 457-473, Apr. 2002.

[32] M.J. Zaki, N. Parimi, N. De, F. Gao, B. Phoophakdee, J. Urban, V. Chaoji, M.A. Hasan, and S. Salem. Towards generic pattern mining, In International Conference on Formal Concept Analysis, 2005

[33] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, New Algorithms for Fast Discovery of Association Rules, Proc. of the 3rd Int'l Conf. On Knowledge Discovery and data Mining (KDD-97), AAAI Press, 1997.