Proceedings of the ICEECE
December 22-24, Dhaka, Bangladesh

Association Rule Mining in Dynamic Database Using the Concept of Border sets

Ferdous Ahmed Sohel*[1] and Chowdhury Mofizur Rahman**

*Dept. of CSE, International Islamic University Chittagong – Dhaka Campus
** Dept. of CSE United International University, Dhaka.

ABSTRACT

Mining of association rules is one of the important tasks in Data Mining. Association Rules find the influence of one set of items on another set of items. There are many influential algorithms to determine association rules [1], [2], [3], [4]. Most of the algorithms assume a static database. There are a few algorithms, which find association rules for dynamic database. Border algorithms [5], [6] are such algorithms, which use the concept of Border set for incremental database. But there are such situations where data are to be upgraded or deleted from the database. This paper presents an algorithm, which uses the concept of border algorithm for diminishing database.

Keywords: Rule mining, frequent set, support, border set, promoted border set, downgraded border set, incremental mining.

1. INTRODUCTION

Association rules are of the form "80% of the people who buy breads also buy butter". Association rules have got numerous applications in the real world. Most research of data mining has focused on the problems of mining association rules [7], [2], [8], [9], [10]. Some researchers study the constraint data mining [4], [11], [12], [13]: sequence mining [8], [14], [15]. In this case, the data being mined has a time stamp; the data will increase with the time. If we re-run the algorithm of data mining to analyze the whole database including the incremental data and the original data, it is obviously inefficient and time consuming. There are many data mining algorithms for incremental data mining [17], [18], [16], [19], [20]. The border algorithm [5] is one of them. In the border algorithm, the whole database is to be scanned every time whenever a border set becomes a large set. Detailed discussion about the border algorithm is given in section 2. In this paper we have discussed an enhanced version of the border algorithm [6].

In most cases, the data that occurred a long time ago must be deleted from the database, so the result of data mining can correctly reflect the current context. Thus, we propose an algorithm to re-compute frequent sets, and rules in the updated database based on the results of previously mining in the original database, and thus a technique to form a system that will work in ever-changing (for both incremental and diminishing) database.

The terms most frequently used in relation to association rules are itemset, support, confidence, frequent itemsets and large itemsets. Itemset means a nonempty set of items. The support of an itemset X, denoted by $\sigma(X)$, is defined as the number of transaction/records in a database that contains X. An itemset with the support greater than the predefined minimum support (min_sup) is called frequent (large) itemset. An association rule between two disjoint and frequent itemset X and Y, denoted by $X \rightarrow Y$, exists if the support of $X \cup Y$ is at least min_sup and confidence is minimum confidence. Confidence is denoted by $\rho(X \rightarrow Y)$, which is defined as the percentages of transactions/records that contain X also contain Y. The rest of the paper is organized as follows: Section 2 presents the Border algorithm and discusses the merits and demerits of it; in section 3 an enhanced version of Border algorithm for incremental database is discussed. In the section 4 another modified version of the border algorithm for decreasing database has been proposed. In section 5 experimental results have been described.

2. BORDER ALGORITHM

Most of the association rule mining algorithms assume that the database is static. However, in reality, every database is dynamic. They are being updated constantly. As a result, the existing frequent itemset may become infrequent and infrequent may become frequent in the context of updated database. So special algorithms are required to handle the frequent itemset computation for the changing database. Border algorithm [5] is one such algorithm, which uses the concept of *border set* and *promoted border set* to find the frequent itemsets when new records are added to the database. An itemset X is called a border set if X is not frequent but all of its subsets are frequent. An itemset that was a border set before update and has become frequent set after update is called promoted border set. The border algorithm maintains support count for all the frequent sets

[1]Corresponding author E-mail: ferasohel27@yahoo.com

as well as for all the border sets. The following symbols are used in the algorithm in figure 1.

T_{old} = The old database
T_{new} = The new records/ transactions
T_{whole}= The whole database, i. e., $T_{old} \cup T_{new}$
L_{old}=Set of frequent itemsets in T_{old}
B_{old}= Set of border itemsets in T_{old}
L_{whole}= Set of frequent itemsets in T_{whole}
B_{whole}= Set of border itemsets in T_{whole}
$S(X)_y$= Support count of the itemset X in the database y.

Given L_{old} and $B_{old,}$ the task of the border algorithm is to find L_{whole} and $B_{whole.}$ The algorithm works as follows:

Initially, L_{old} and B_{old} are generated with their respective supports. The algorithm starts by making one pass over the new database T_{new} and counts the support of the itemsets of $L_{old} \cup B_{old}$ for T_{new}. During this pass the algorithm generates two categories of itemsets – F and B, where F contains the itemsets of L_{old}, which are frequent in Twhole, and B contains the itemsets of B_{old}, which are frequent in T_{whole}. The set B here is the promoted border set. If B is null then F contains all frequent itemsets of T_{whole}. But, if there is at least one promoted border set, the algorithm generates new candidate sets and makes one pass over the whole database to count the support of them. So the algorithm makes one pass over the new database and at most one pass over the whole database.

1. Read T_{new} and increment the support count of $X \in L_{old} \cup B_{old}$
2. $F:= \{X | X \in L_{old}$ and $S(X) \geq \alpha\}$
3. $B:= \{X | X \in B_{old}$ and $S(X) \geq \alpha\}$
4. Candidate Generation:
a) $C_1 = \phi$, k:=2;
b) For all itemsets $l_1 \in B_{k-1} \cup C_{k-1}$ do begin
i) For all itemsets $l_2 \in B_{k-1} \cup C_{k-1} \cup F_{k-1}$do begin
a) if $l_1[i] = l_2[I]$, i= 1,2,3 … .. . , k-1 and $l_1[k-1] < l_2[k-1]$ then
b) $c = l_1[1], l_1[2], ..., l_1[k-1], l_2[k-1]$
ii) $C_k = C_k \cup \{c\}$
iii) End do
c) End do
5. Prune C_k:
a) All the subsets of k-1 size should be present in $B_{k-1} \cup C_{k-1} \cup F_{k-1}$
b) k = k+1
c) Candidate $C = \cup C_k$
d) Read T_{whole} and count the support for each itemset in C;
e) new_frequent_sets = $\{X \in C | S(X)_{Twhole} \geq \alpha\}$
f) $L_{whole} = F \cup$ new_frequent_sets $\cup B$
g) $B_{whole} = (B_{old} \backslash B) \cup (F_{old} \backslash F) \cup \{X \in C$ and $S(X) < \alpha$ and all its subsets are in $L_{whole}\}$
6. Prune $B_{whole:}$
a) All the subsets of any element in B_{whole} must be in $L_{whole.}$

Figure 1: Border Algorithm

The algorithm is robust enough to handle the incremental database. However, an important drawback of the algorithm is that if any of the itemsets of the border set B_{old} becomes large on adding new transactions the whole database has to be scanned because the candidate sets are computed based on $L_{old,}$ which ultimately makes the algorithm expensive. In the following work an attempt has been made to address this problem by modifying the existing Border algorithm.

3. AN ENHANCED BORDER ALGORITHM FOR THE INCREMENTAL DATABASE

The purpose of the following algorithm [6] is the same as that of the border algorithm, i. e., to generate frequent itemsets for an incremental database. However, unlike the border algorithm it uses two border sets, where the first border set is $B'_{old} = \{X | S(X) \geq \beta$ and $S(X) < \alpha\}$, where α is the minimum support and β is a positive constant such that $(\alpha - \beta \geq 0)$. The second Border set is $B''_{old} = \{X | S(X) < \alpha$ and $\notin B'_{old}\}$. Initially B''_{old} is calculated as $\{X | S(X) < \beta\}$. The other symbols are same as those of Border algorithm. When new transactions are added the first border set is used in candidate generation, whereas, the elements of the second border ser are not used in candidate generation. Another requirement of the algorithm is that all the subsets of an itemset in $B'_{old} \cup B''_{old}$ must be frequent or belong to $B'_{old.}$ The algorithm generates these two border sets using a modified version [6] of the Apriori algorithm [2] as stated in the figure 2. Obviously, the first border set contains the itemsets with high probability of being frequent when new transactions are added. The important significance of introducing two border sets is that if some of the elements of the first border set become frequent on addition of new transactions, it's not required to generate new candidate sets unlike the border algorithm, because they have already been used to generate candidate sets. New candidate sets will be generated only when any of the elements of the second border set become large on addition of new transactions. If new candidate itemsets are generated, one scan over the whole database is required to count the support of the new candidate itemsets.

Let us consider the following synthetic market basket database T_{old} in table 1 of dimensionality 5 and assume α= 40% and β=30%; now applying modified Apriori algorithm L_{old}, B'_{old} and B''_{old} are assumed to be known with their respective support counts. Here $L_{old} = \{(I_1, I_2, I_3), (I_1I_2)\}$, $B'_{old} = \{(I_3)\}$, $B''_{old} = \{(I_5), (I_1I_4, I_2I_4, I_1I_3, I_2I_3, I_3I_4,)\}$. The algorithm starts by making one pass over the new database T_{new} and counts the supports of the elements of $L_{old} \cup B'_{old} \cup B''_{old}$ in T_{new}. During the pass, the algorithm generates 5 categories of itemsets- F, B′, B″, B‴ and B⁗ – as defined in the algorithm in figure 3. If B″ is null then F and B′ contain all the frequent sets in T_{whole}. If B″ contains at least one itemset then new candidate sets are required to be generated. If B″ is null then no new candidate set is generated, i.e., all the generated candidate sets will be infrequent.

Item→ Transaction↓	I_1	I_2	I_3	I_4	I_5
T1	1	1	0	0	1
T2	0	0	1	0	0
T3	0	0	0	1	0
T4	0	0	0	1	0
T5	0	1	0	1	0
T6	1	1	1	0	0
T7	1	0	0	0	0
T8	0	0	0	0	0
T9	1	1	0	1	0
T10	1	1	1	1	1

Table 1: An example database.

If new candidate sets are generated, the algorithm makes one pass over the entire database to count the support of the new candidate sets. So the algorithm makes one pass over the new database and at most one pass over the whole database. At the end, the algorithm generates L_{whole}, B'_{whole} and B''_{whole}, which are counterparts of the L_{old}, B'_{old}, B''_{old} respectively for the whole database. For better understanding let's take the above example again. Let the T_{new} be as shown in table 2.

Item→ Transaction ↓	I_1	I_2	I_3	I_4	I_5
T11	0	0	1	1	1
T12	1	1	0	0	1
T13	1	1	1	0	1
T14	0	0	1	0	1

Table 2: New Transactions are to be added

When T_{new} is added to T_{old}, B', B'', $B'''$and B'''' are calculated as follows:
$F= \{(I_1, I_2, I_4), (I_1I_2)\}$; $B' = \{(I_3)\}$, $B''= \{(I_5)\}$, $B'''= \phi$ and $B''''= \phi$.
Since B'' is not null, new candidate itemsets will be generated. The new candidate itemsets after pruning will be $\{(I_1I_5, I_2I_5, I_3I_5, I_4I_5), (I_1I_2I_5)\}$. The enhanced algorithm is as follows:

1. $F= \{X| X \in L_{old}$ and $S(X)_{Twhole} \geq \alpha\}$
2. $B'= \{X| X \in B'_{old}$ and $S(X)_{Twhole} \geq \alpha\}$
3. $B''=\{X| X \in B''_{old}$ and $S(X)_{Twhole} \geq \alpha\}$
4. $B'''= \{X| X \in B''_{old}$ and $S(X)_{Twhole} \geq \beta$ and $S(X)_{Twhole} < \alpha\}$
5. $B''''= \{X| X \in B'_{old} \cup L_{old}$ and $S(X)_{Twhole} \geq \beta$ and $S(X)_{Twhole} < \alpha\}$
6. Prune B''''; all the subsets of $X \in B''''$ should be in $F \cup B'$.
7. Prune B'''; all the subsets of $X \in B'''$ should be in $F \cup B' \cup B''''$.
8. If $B'' \neq \phi$ then $k=2$ and $C_1 = \phi$;
9. do
a. $C_k = \phi$
b. $L= B''_{k-1} \cup C_{k-1} \cup B'''_{k-1;}$
c. $M= B''_{k-1} \cup C_{k-1} \cup B'''_{k-1} \cup F_{k-1} \cup B'_{k-1} \cup B''''$;
d. For all itemsets in $l_1 \in L$ do
e. For all itemsets in $l_2 \in M$ do
i. If $l_1[i]= l_2[i]$ for $1 \leq i \leq k-2$ and $l_1[k-1] < l_2[k-1]$ then
$C= \{ l_1[1], l_1[1], \ldots \ldots \ldots, l_1[k-2], l_1k-1], l_2[k-1]\}$;
ii. $C_k= C_k \cup C$;
f. Prune C_k: All the subsets of C_k of size k-1 must be present in M;
g. K=k+1;
10. until (no new candidate can be generated)
11. Candidate set $C= \cup C_k$
12. Read the whole database and count the support of the candidate sets $X \in C_k$ and generate L_{whole}, B'_{whole}, B''_{whole} as follows:
a. new_large_sets= $\{X \in C| S(X)_{whole} \geq \alpha\}$
b. $L_{whole}= F \cup B' \cup B'' \cup$ new_large_sets
c. $B'_{whole}= B'''' \cup B''' \cup \{X \in C| S(X) < \alpha$ and $S(X) \geq \beta\}$
d. $B''_{whole}=\{X \in B''/ B'' \cup B'''|$ all subsets of X are also in $L_{whole} \cup B'_{whole}\} \cup \{X \in C| S(X) < \beta$ and all the subsets of X are in L_{whole} and $B'_{whole}\}$;
13. Else:
- $L_{whole}= F \cup B'$;
- $B'_{whole}= \{X \in B'''' |$ all the subsets of X are also in $L_{whole}\}$
- $B''_{whole}= \{X \in B''_{old} |$ all the subsets of X should also be in $L_{whole} \cup B'_{whole}\}$;
14. Endif

Figure 2: Enhanced Border Algorithm for incremental database.

Here, whole database will be scanned only when some of the itemsets of the second border set become large, whereas the border algorithm scans the whole database when any element of the border set (first or second) becomes large. When this enhanced algorithm requires whole database scanning, performance will be the same as the border algorithm.

The value of β plays a vital role. When α - β=0, the algorithm becomes the same as the border algorithm. With smaller β value performance of this enhanced algorithm is better than the border algorithm in terms of execution time. However, with β memory requirement increases due to the additional candidate sets. However, the cost of additional memory requirement is negligible in comparison to the total memory requirement of whole database scanning. The trade off in choosing β should be considered when there is a scarcity of memory. But the above algorithm does not consider the situation of diminishing database.

4. PROPOSED BORDER ALGORITHM FOR DIMINISHING DATABASE

In most situations, the data that occurred a long time ago will have side effects on the results of data mining. In order to guarantee that the result of data mining can really reflect the current context, some old data are needed to be deleted from the original database. So we propose another modified border algorithm and also ideas of modified enhanced border algorithm to deal with the diminishing database.

Here we have considered two types of frequent itemsets – *most frequent itemset* and *down graded border set.* The members in the downgraded border set are likely to be no more frequent if some of the transactions are deleted from the original database. We are again considering 2 threshold values α and β, where $\alpha-\beta \geq 0$ and also β is now the min_sup. Let's have the following table:

M_{old}= Most frequent itemsets in $T_{old,}$ i.e., support count $\geq \alpha$
D_{old}= Down graded border set in $T_{old,}$ i.e., support count is between α and β.
T_{del}= Transactions to be deleted
T_{cur}= The new database after deletion
M_{cur}= Most frequent itemsets in T_{cur}
D_{cur}= Down graded border set in T_{cur}

We can use another modified version of the Apriori algorithm as stated in figure 3. The algorithm works as follows: initially M_{old} and D_{old} are calculated with their respective support count from the initial database. The algorithm starts by making one pass over the database to be deleted T_{del} and updates the support counts of the itemsets in $M_{old} \cup D_{old.}$ During this pass five categories of itemsets are generated – M: the itemsets, which are still most frequent; MD: The itemsets which move graded from most frequent set to down graded border set; D: which are holding their positions in the down graded border set; DLF: itemsets which move from down graded border set to less frequent itemsets; MLF: itemsets which move from most frequent itemsets to less frequent itemsets. $D_{cur}= D \cup MD.$

1.Initialize k:=1, C_1:= all the 1-itemset
2. Read the database T to count the support of C_1 and determine L_1.
3. L_1 := {frequent 1-itemset}
4. k=2
5. while ($L_{k-1} \neq \phi$)do
a) C_k:=gen_candidate_itemsets from L_{k-1}
b) Prune C_k
c) for all the transactions $t \in T$
1. increment the count of all the candidates in C_k which are contained in T
d) M_k:={$c \in C$| $S(c) \geq \alpha$}
e) D_k:= { $c \in C$| $S(c) \geq \beta$ and $S(c) < \alpha$}
f) k:=k+1
6. M_{old}:=$\cup M_k$
7. D_{old}:= $\cup D_k$

Figure 3: Modified Apriori for diminishing database.

If either M or D∪MD set is empty then the algorithm scans over the current database to find out M_{cur} and $D_{cur,}$ i.e., the entire database is required to be scanned only when either M or D_{cur} is empty. As a result we need not scan over the entire database every time it's updated by deletion of transactions. Let's explain the algorithm with an example: Let us assume the initial database is as in table 1. So for α=40% and β=30% we have, M_{old}= {(I_1, I_2, I_3), (I_1I_2)}, D_{old} = {(I_3)}. Now if we delete the transaction T1 then M ={(I_1, I_2, I_3)}, MD={(I_1I_2)}, D={(I_3)}, MLF=ϕ, DLF=ϕ. Neither M nor D∪MD is empty. So the elements in these sets will be used to generate candidate itemsets to form large itemsets. Now if we delete T9 then M becomes empty. So it is now required to scan over the latest (entire) database to find out the itemsets for M_{old} and D_{old}. We can also maintain 3 sets of frequent itemsets rather than 2 sets. In that case if any two of the three sets become empty the entire database is needed to be scanned for support count with the new values of the thresholds. Again, as in the enhanced border algorithm for incremental database there is a tradeoff between memory requirement and execution time in case of choosing 2 or 3 sets for frequent itemsets.

1. Read T_{del} and find the support count of $X \in M_{old} \cup D_{old}$ by subtracting.
2. M:= {$X|X \in M_{old}$ and $S(X) \geq \alpha$}
3. D:= {$X|X \in D_{old}$ and $S(X) \geq \beta$}
4. MD:= {$X|X \in M_{old}$ and $\alpha > S(X) \geq \beta$}
5. DLF:={$X|X \in D_{old}$ and $S(X) < \beta$}
6. MLF:= {$X|X \in M_{old}$ and $S(X) < \beta$}
7. If $M \neq \phi$ and $D \cup MD \neq \phi$ then
8. Candidate Generation:
a) $C_1 = \phi$, k:=2;
b) For all itemsets $l_1 \in D_{k-1} \cup MD_{k-1} \cup C_{k-1}$ do begin
c) For all itemsets $l_2 \in D_{k-1} \cup MD_{k-1} \cup C_{k-1} \cup M_{k-1}$do begin
i) if $l_1[i] = l_2[I]$, i= 1,2,3 … .. . , k-1 and $l_1[k-1] < l_2[k-1]$ then
ii) c = $l_1[1]$, $l_1[2]$, .. … …, $l_1[k-1]$, $l_2[k-1]$
iii) $C_k = C_k \cup \{c\}$
d) End do
e) End do
9. Prune C_k:
a. All the subsets of k-1 size should be present in $D_{k-1} \cup MD_{k-1} \cup C_{k-1} \cup M_{k-1}$
b. k = k+1
c. Candidate C= $\cup C_k$
d. Read T_{cur} and count the support for each itemset in C;
e. new_frequent_sets = {$X \in C|$ $S(X)_{Twhole} \geq \alpha$}
f. $M_{cur} = M \cup$new_frequent_sets.
g. $D_{cur} = (D_{old} \backslash D) \cup (M_{old} \backslash M) \cup \{X \in C$ and $\beta \leq S(X) < \alpha\}$
10. Else
a) M_{cur} and D_{cur} from the database T_{cur}.

Figure 4: Modified Border Algorithm for Diminishing Database

The MLF and DLF sets can be used to find out border and promoted border sets in incremental database (when new transactions are added after deletion of some records). Because these are more probable to become frequent on occasion of addition of new transactions.

5. EXPERIMENTAL RESULTS

The proposed border algorithm for diminishing database was tested on an Intel Processor (P IV 1GHz, 256 MB RAM) using various sized synthetic data generated using the techniques given in [2]. The data sets used in the

experimentation are shown in Table 3. The original size of the synthetic database was 60,000 records with 255 dimensionality and the size of decrements were 2000 records at each spell. Initially the modified Apriori algorithm was used on the original database to find out the M_{old} and $D_{old.}$ Table 3 shows the parameters used in the synthetic data generation. For all the data sets the number of items and number of potential large itemsets were taken as 250 and 500 respectively:

Data Set	|T|	|I|	|D|
T5. I2. D100K	5	2	100K
T10. I2. D100K	10	2	100K
T20. I4. D100K	20	4	100K
T20. I6. D100K	20	6	100K

Table 3: Parameters used for synthetic data

Where |D|= Number of transactions, |I|= Average size of potentially large itemsets and |T|= Average size of transactions. Also we have used N=1000 and |L|= 2000, where N is number of items and |L| is number of maximal potentially large itemsets. We have tried to keep observation on the performances of the algorithm for various values of α, β. Our initial values of α, β pairs are 5%, 3% and 5%, 2% and 4%, 2%. The results are summarized in table 4.

Decr-ement Spell	α= 5%, β=3%		α= 5%,β=2%		α= 4%,β=2%	
	Time (sec)	Scan Req.	Time (sec)	Scan Req.	Time (sec)	Scan Req.
1	330	N	339	N	350	N
2	665	Y	335	N	351	N
3	315	N	672	Y	348	N
4	660	Y	340	N	675	Y

Table 4: Results for the modified border algorithm for decreasing database

From the above table we have the following observations:

Large values of α, β lead faster process due to less itemsets are to be considered in the most frequent itemset and downgraded itemsets. But in this case the probability of scanning over the whole database increases because these two itemsets are more probable to become empty at large values of α, β. In this case, time requirements ultimately increase.

Smaller values of α, β lead steady process and less requirement for scanning over the whole database. We found that at lower values of α, β and (α - β) the modified border algorithm for decreasing database works well.

We can also use this modified algorithm along with border algorithm and enhanced algorithm for an increasing environment. In this case, the datasets in MLF and DLF will make support to the system to form promoted border sets. So we can easily use the proposed border algorithm and any one of the incremental border algorithms in order to form a complete rule mining system in an ever-changing environment.

6. CONCLUSION

In this paper we discuss border algorithm and an enhanced version of border algorithm for incremental database and also present a modified form of border algorithm for diminishing database. Choosing the values for the parameters α and β is very much important for these algorithms. Again, the variants of Apriori [2] algorithms can be used to improve the performances.

REFERENCES

[1]: R. Agarwal, H. Mannila, R. Shrikant, H. Toivonan, and A. I. Verkamo, "Fast discovery of association rules," in U. Fayyad et el., editors *Advances in Knowledge Discovery and Data Mining,* MIT Press, 1996.

[2]: R. Agarwal and R. Srikant, "Fast algorithms for mining association rules," in *20th VLDB Conference*, Santiago, Chile, September 1994.

[3]: H. Mannila, R. Shrikant, H. Toivonan, and A. I. Verkamo, "Efficient algorithms for discovering association rules," in *AAAI Workshop, Knowledge Discovery in Databases,* July 1994.

[4]: A. Savasere, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association Rules for Large Databases," In *Proceedings of 21st Conference of VLDB,* Zurich, Switzerland, September 1995.

[5]: R. Feldman, Y. Aumann, A. Amir, and H. Mannila, "Efficient algorithms for discovering frequent sets in incremental databases," in *Proceedings of ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD),* 1997.

[6]: A. K. Das, D. K. Bhattacharyya, "Efficient Rule mining: Dynamic Database," in *proceedings of conference ITPC 2003,* vol. 1, Nepal, May 23 – 26, 2003.

[7]: R. Agarwal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," In *ACM SIGMOD International Conference on Management of Data*, May 1993.

[8]: H. Mannila, H. Toivonan, and A. I. Verkamo, "Discovering frequent episodes in sequences," in *KDD-95,* pages 210-215, August 20-21, 1995.

[9]: U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, "Advances in Knowledge Discovery and Data Mining," *AAAI/MIT Press*, 1995.

[10]: R. Agarwal, and J. Shafer, "Parallel mining of association rules," in *IEEE trans., on Knowledge and data Engg.,* 1996.

[11]: R. Srikant and R. Agarwal, "Mining Quantitative Association Rules in Large Relational Tables," in *ACM SIGMOD Conference on Management of Data*, June 1996.

[12]: R. Srikant and R. Agarwal, "Mining Sequential Patterns: Generalization and Performance Improvements," in *5th International Conference on Extending Database Technology*, May 1996.

[13]: H. Toivonen, "Sampling Large Databases for Association Rules," in *22nd VLDB Conference, 1996.*

[14]: H. Mannila, and H. Toivonen, "On an Algorithm for Finding All Interesting Sequences," in cybernetics and systems, Volume ii, *the 13th European Meeting on Cybernetics and Systems Research*, Vienna, Austria, April 1996.

[15]: R. Agarwal and R. Shrikant, "Mining Sequential Patterns," in *Proceedings of the 11th International Conference on Data Engineering,* Taipei, Taiwan, March 1995.

[16]: M. Klemettinen, H. Mannila, P. Ponkainen, H. Toivonen, and A. I. Verkamo, "Finding Interesting Rules from Large Sets from Discovered Association Rules," in *3rd International Conference on Information and Knowledge Management,* November 1994.

[17]: D. W. Cheung, J. Han, V. T. Nag, and C. Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Update Technique," In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96),* pages 106-114, New Orleans, Louisiana, USA, February 1996.

[18]: D. Cheuang, S. D. Lee, and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," In *Proceedings of the 5th International Conference on Database Systems for Advance Applications (DASFAA '97)*, pages 185-194, Melbrone, Australia, April 1997.

[19]: S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An Efficient algorithm for the incremental updating of association rules in large database," In *the Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97),* pages 263-266, Newport Beach, California, USA, August 1997.

[20]: Ahmed Ayad, Nagwa El- Makky, and Yoursy Taha, "Incremental Mining of Constructed Association Rules," *First SIAM International Conference on Data Mining,* April 5-7, 2001, Chicago, USA.