INT J COMPUT COMMUN, ISSN 1841-9836
8(6):791-799, December, 2013.

Association Rule Mining using Path Systems in Directed Graphs

S. Arumugam, S. Sabeen

S. Arumugam*
1. National Centre for Advanced Research in Discrete Mathematics (n-CARDMATH)
Kalasalingam University
Anand Nagar, Krishnankoil-626126, INDIA.
2. School of Electrical Engineering and Computer Science
The University of Newcastle
NSW 2308, Australia.
*Corresponding author: s.arumugam.klu@gmail.com

S. Sabeen
Department of Computer Applications
Jaya Engineering College
Chennai-600054, INDIA.
sabeens@rediffmail.com

Abstract: A transaction database (TDB) consists of a set I of items and a multiset $\mathcal{D}$ of nonempty subsets of I, whose elements are called transactions. There are several algorithms for solving the popular and computationally expensive task of association rule mining from a TDB. In this paper we propose a data structure which consists of a directed graph D (loops and multiple arcs are permitted) and a system of directed paths in D to represent a TDB. We give efficient algorithms for generating the data structure, for extracting frequent patterns and for association rule mining. We also propose several graph theoretic parameters which lead to a better understanding of the system.
Keywords: Directed graphs, path system, in-degree, out-degree, association rule mining, frequent patterns, data mining.

1 Introduction

The task of association rule mining in a large database of transactions was proposed by Agarwal et al. [1]. Since then this problem has received a great deal of attention and association rule mining is one of the most popular pattern discovery methods in Knowledge Discovery from Database (KDD). A broad variety of efficient algorithms such as Apriori Algorithm [3], FP-growth [4], FP-tree [4], SETM [6], DIC [5] have been developed during the past few years. In this paper we propose a data structure consisting of a directed graph D and a multiset of directed paths in D to represent a database of transactions. We give an algorithm which generates the directed graph D and which also simultaneously computes several other measures such as in-degree, out-degree, total number of arcs, length of a largest transaction, frequency of occurrence of various nodes and the number of occurrences of each arc in D. The second algorithm generates all the patterns of the transaction database using the above data structure. This algorithm can be modified to extract frequent patterns. The third algorithm deals with association rule mining. In this process we scan the database exactly once and the digraph is constructed dynamically.

2 Directed Graphs and Path Systems

A *directed graph* $D = (V, A)$ consists of a finite nonempty set V and a multiset A of ordered pairs of elements of V. The elements of V are called *vertices* and the elements of A are called *arcs*. An ordered pair (v, v) is called a *loop* at v. We allow both loops and multiple arcs (that

is, an arc (u, v) appearing more than once) in D. If $a = (u, v)$ is an arc in D, then u is called the *tail* of a and v is called the *head* of a. For basic terminology in directed graphs we refer to the book by Chartrand and Lesniak [2]. A *directed path* in D is a sequence of distinct vertices $P = (v_1, v_2, \ldots, v_k)$ such that (v_i, v_{i+1}) is an arc in D for all $i, 1 \leq i \leq k-1$. A *directed cycle* in D is a sequence of vertices $C = (v_1, v_2, \ldots, v_k, v_1)$ such that (v_i, v_{i+1}) is an arc in D for all $i, 1 \leq i \leq k-1$, (v_k, v_1) is an arc in D and the vertices $v_1, v_2, \ldots, v_k$ are distinct. A directed graph is called *acyclic* if it contains no directed cycle. For any vertex v, the *in-degree* $id(v)$ is defined to be the number of arcs of the form (u, v) in D and the *out-degree* $od(v)$ is defined to the numbers of arcs of the form (v, u) in D. We observe that a loop at v contributes 1 to both $id(v)$ and $od(v)$ and an arc (u, v) contributes 1 to $od(u)$ and 1 to $id(v)$. A vertex v such that no arc is incident with v or all the arcs incident at v are loops is called an *isolated vertex* of D. We denote by D_1 the subdigraph of D obtained by removing all the loops in D. A *path system* or a *path cover* in a directed graph D is a multiset ψ of directed paths in D such that every arc of D is in exactly one path in ψ. We adopt the convention that all loops in D are members of ψ. If D contains multiple arcs, a directed path in D may occur more than once in ψ.

3 Digraph model for transaction database

Let $I = \{1, 2, \ldots, n\}$ be a set of objects whose elements are called items. We impose on I the natural ordering of the set of positive integers. Any nonempty subset of I is called a transaction. A transaction database $\mathcal{D}$ is a multiset of transactions in I. Thus a subset X of I may occur more than once in $\mathcal{D}$. We assume that for each $i \in I$, there is at least one transaction in $T \in \mathcal{D}$ with $i \in T$. In otherwords $\bigcup_{T \in \mathcal{D}} T = I$. We say that a transaction $T \in \mathcal{D}$ supports an item set $X \subseteq I$ if $X \subseteq T$. The support of X is defined by $supp(X) = \frac{|T \in \mathcal{D}: X \subseteq T|}{|\mathcal{D}|}$. Thus $supp(X)$ is the fraction of transactions supporting X. If we fix a threshold value s, then any subset X of I with $supp(X) \geq s$ is called a frequent pattern. An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The support of a rule $X \Rightarrow Y$ is defined to be $supp(X \cup Y)$. The confidence of the rule is defined as $conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$. The support and confidence values are usually normalized so that these values occur between 0% and 100% instead of 0 to 1.0. Normally we generate association rules for frequent patterns. For further terminology in TDB and association rule mining we refer to the book by Han and Kamber [7].

In this paper we propose a data structure which consists of a directed graph D and a system of directed paths in D for representing a transaction database. The vertex set of the directed graph is the item set I. We sort the elements of each transaction $T \in \mathcal{D}$ in the increasing order and represent T as a directed path in D. Thus the direction on each edge is given by the orientation from low-to-high. Any transaction T of the form $\{x\}$ is represented as a loop at x. We illustrate this database with a small example.

Example 1. Let $I = \{1, 2, 3, 4, 5\}$. Let $\mathcal{D} = \{T_1, T_2, T_3, T_4, T_5, T_6\}$ where $T_1 = \{1, 2, 5\}, T_2 = \{1, 4\}, T_3 = \{1, 2, 4\}, T_4 = \{3, 5\}, T_5 = \{2\}, T_6 = \{3\}$ and $T_7 = \{3, 4, 5\}$. The directed graph of $\mathcal{D}$ is given in Figure 1.

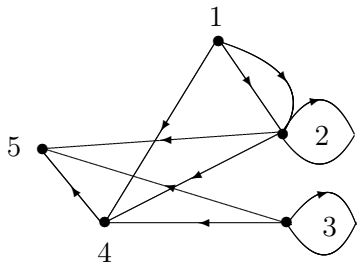

Figure 1

Since every transaction T in $\mathcal{D}$ gives a directed path in D, it follows that the multiset $\mathcal{D}$ of all transaction forms a path system in D. We observe that the directed graph D_1 which is obtained by removing all the loops in the directed graph D representing a TDB is an *acyclic directed graph.*

4 Directed graph of a transaction data base (DGTDB)

In this section we propose an algorithm to construct the directed graph representing a TDB. The algorithm scans the data exactly once, dynamically constructs the digraph D and simultaneously computes several parameters such as frequency of occurrence of each node, number of loops at each node, number of occurrence of each arc uv, total number of arcs in D, the maximum length of a transaction, in-degree and out-degree of each node.

The algorithm first creates all nodes of D, one node for each item, with support count 0. Then each transaction is scanned and the directed path in D representing the transaction is constructed. If $(i_1, i_2, \ldots, i_k)$ is a transaction, the arc (i_j, i_{j+1}) is represented as a linked list. The header of this list has two fields. One field is used to store the list of vertices $(i_1, i_2, \ldots, i_{j+1})$, which is called the label of the arc (i, j) and the other field is used to store the frequency of occurrence of the arc (i, j). For example in the digraph given in Figure 3, the arc (3, 5) occurs with values 2, (1,3,5) and also with values 1 (2,3,5), indicating that it occurs twice as part of (1,3,5) and occurs once as a part of (2,3,5). In general if an arc (i, j) has labels $k, (i_1, i_2, \ldots, i_r, i, j)$ it means that the arc (i, j) appears k items as part of $(i_1, i_2, \ldots, i_r, i, j)$. Dynamic memory allocation method is used for storing these values. The pseudo code for the construction of DGTDB is given in Figure 3. This algorithm also generates further informations which are given in the output.

Algorithm: Construction of DGTDB
Input: Transaction Database TDB, n: Number of distinct items in TDB, m: Number of Transactions.
Output: DGTDB: Directed Graph of TDB.

Method:

Create a node for each item and initialize the values $f(i)$, Out-Degree (i), In-Degree (i), In-Edge(i), Out-Edge (i) $L_c(i)$ to zero.

Initialize $predecessor = \emptyset$; and $X = \emptyset$;

1: **for** each transaction T_j and for each each item i in T_j, **do**
2: $X = \bigcup\{i\}$; $f(i) + +$;
3: **if** $(|X| > K)$ **then**

```
    K = |X|;
  end if
  if (|T_j| = 1) then
    L_c(i) + +;
    if (L_c(i) = 1) then
      // If an edge from i to i does not exist so far
      CreateEdge(i, i);
      Set Label (e_c, i) = ⟨X⟩;
      f(e_c, i) = 1;
      n(E) + +;
      Out-Degree(i) + +;
      In-Degree(i) + +;
      In-Edge(i) + +;
      Out-Edge(i) + +;
    end if
  end if
  if (L_c(i) > 1) then
    f(e_c, i) + +;
    Out-Degree(i) + +;
    In-Degree(i) + +;
  end if
  if (predecessor ≠ ∅) and (X ∉ any Label (e_c, i)) then
    CreateEdge (predecessor, i);
    n(E) + +;
    Set Label(e_c, i) = ⟨X⟩
    f(e_c, i) = 1;
    Out-Degree(predecessor) + +;
    In-Degree(i) + +;
    In-Edge(i) + +;
    Out-Edge(predecessor) + +;
  end if
  if (X ∈ Label (e_j, i)) then
    f(e_j) + +;
    Out-Degree(predecessor) + +;
    In-Degree(i) + +;
  end if
  predecessor = i;
end for
return DGTDB;
```

Figure 2. Pseudo code for DGTDB construction

We illustrate the algorithm DGTDB with a transaction database consisting of 12 items and 30 transactions, which is given in Table 1.

TID	PRODUCTS IN EACH TRANSACTION	TID	PRODUCTS IN EACH TRANSACTION
T001	1, 3, 7, 8	T016	2, 4, 6 , 8, 10, 12
T002	1, 2, 3, 8	T017	1, 3,5, 7
T003	2, 4, 5, 6	T018	9, 11
T004	2, 3, 5, 6	T019	10, 11,12
T005	1, 4, 5, 11	T020	6 , 7, 8,12
T006	3, 4, 5, 12	T021	3
T007	1, 2, 3, 4	T022	10, 11, 12
T008	4, 5, 11	T023	2, 4,6 , 8, 10,12
T009	1, 4, 5, 8	T024	1, 3, 5, 7, 9, 11
T010	3, 4, 10	T025	2, 4, 6, 8, 10, 12
T011	2, 3, 4	T026	3
T012	3	T027	2, 4, 6 , 8, 10, 12
T013	3, 6 , 9, 12	T028	3
T014	1, 10, 12	T029	12
T015	3, 6 ,9 ,12	T030	3

Table 1. Transaction Database, TDB

The DGTDB for the above transaction database is given in Figure 3.

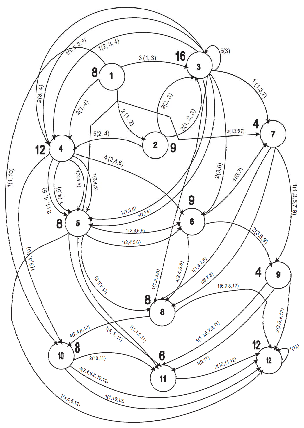

Figure 3. Directed Graph of TDB in Table 1.

The number given at each node represents the frequency of the corresponding item.

5 Algorithm for extracting frequent patterns (XoFP)

In this section we present an algorithm for extracting the set $\mathcal{L}$ of all frequent patterns from DGTDB constructed in Section 4. For each node i, the algorithm generates all frequent patterns L_1 with i as the last item of L_1. If in-edge $(i) = 0$, then $\{i\}$ is the only pattern with i as the last item. Otherwise for each edge e_c with i as head we consider all subsets X of label (e_c, i) such that $|X| \geq 2$ and $i \in X$. Then the frequency of X is the sum of the frequencies of all the edges e_c with i as head for which $X \subseteq label(e_c, i)$. If this frequency is greater than or equal to the minimum support threshold, then X is added to the set of frequent patterns. The algorithm XoFP is given in Figure 4.

Algorithm: XoFP, Extraction of Frequent Patterns from DGTDB.
Input: s: Minimum support threshold; DGTDB: Directed Graph of TDB.
Output: $\mathcal{L}$: The set of all frequent patterns mined from the DGTDB.
Method:

```
1: L = ∅; m = 0;
2: // Initialize set of frequent patterns
3: for each node i do
4:   if (f(i) ≥ s) then
5:     L = L ∪ {i}
6:     return {i}, f(i)
7:     m + +;
8:   end if
9:   if (In-Edge(i) > 0) then
10:    f = 0;
11:    for each Label (e_c, i), 1 ≤ c ≤ In-Edge(i) do
12:      f = f(e_c, i);
13:      W = Label(e_c, i);
14:      if (W ∉ L) then
15:        for each x ⊆ W, |x| ≥ 2 and i ∈ x do
16:          for each Label (e_y, i) do
17:            if (x ⊆ Label e_y(i) then
18:              f = f + f(e_y, i));
19:            end if
20:          end for
21:        end for
22:        if (f ≥ s) then
23:          L = L ∪ {x};
24:          return {x}, f
25:          m + +;
26:        end if
27:      end if
28:    end for
29:  end if
30: end for
31: return (L)
```

Figure 4. XoFP, Extraction of patterns from DGTDB

Example 2. By applying XoFP to the DGTDB given in Figure 3, the set of frequent patterns obtained using the items 1,2,3 and 4 along with the respective frequencies are given in Table 2.

Node id	Frequent Patterns	Frequency
1	$\{1\}$	8
2	$\{2\}$	9
	$\{1,2\}$	2
3	$\{3\}$	16
	$\{1,3\}$	5
	$\{2,3\}$	4
	$\{1,2,3\}$	2
4	$\{4\}$	12
	$\{1,4\}$	3
	$\{2,4\}$	7
	$\{3,4\}$	4
	$\{2,3,4\}$	2

Table 2. Extracted frequent patterns from the nodes 1, 2, 3 and 4 of DGTDB where $s = 2$.

6 Algorithm for generating association rules (GEAR)

In this section we present an algorithm for generating association rules and strong association rules from the set of frequent patterns mined from the given TDB. An association rule which satisfies both minimum support threshold and minimum confidence threshold is called a strong association rule. For each frequent pattern X and for each nonempty proper subset Y of X the algorithm computes the support and confidence of the association rule $Y \Rightarrow X - Y$.

Algorithm: GEAR, Generating association rules from the frequent patterns
Input: $\mathcal{L}$: set of all frequent patterns; c- Minimum confidence threshold of rule; s- Minimum support threshold.
Output: R: set of all strong association rules; $R\prime$: set of all association rules not in R.

Method:

1: $R = \emptyset; R\prime = \emptyset; N(R) = 0; N(R\prime) = 0;$
2: **for** each X, Y where $X \in \mathcal{L}$ with $|X| > 1$ and $Y \subseteq X, Y \neq \emptyset, Y \neq X$ **do**
3: Generate Rule $(Y \Rightarrow (X - Y))$
4: Compute $conf = \frac{f(X)}{f(Y)}$
5: // $f(X)$ is the frequency of pattern X. $f(Y)$ is the frequency of pattern Y.
6: **if** $(conf \geq c)$ **then**
7: $R = R \cup \{Y \Rightarrow (X - Y)$: support=s, confidence = $conf\}$;
8: $n(R) + +$;
9: **else**
10: $R\prime = R\prime \cup \{Y \Rightarrow (X - Y)$: support=s, confidence = $conf\}$;
11: $n(R\prime) + +$;
12: **end if**
13: **end for**
14: **return** (R) and **return** $(R\prime)$;

Figure 5. GEAR, Algorithm for Generating Association Rules

Example 3. From Table 2 we have $X = \{1, 2, 3\}$ is a frequent pattern with frequency 2. The set of all association rules generated from this pattern by using GEAR along with the confidence and support for each rule is given in Table 3. We have taken the minimum confidence threshold c and the minimum support threshold s as 50 and 6 respectively.

S.No	**Association Rules**	**Confidence of the Rule**	**Support of** the Rule	R **or** $R\prime$
1	$\{1\} \Rightarrow \{2, 3\}$	25	6	$R\prime$
2	$\{2\} \Rightarrow \{1, 3\}$	22.2	6	$R\prime$
3	$\{3\} \Rightarrow \{1, 2\}$	12.5	6	$R\prime$
4	$\{1, 2\} \Rightarrow \{3\}$	100	6	R
5	$\{1, 3\} \Rightarrow \{2\}$	66.67	6	R
6	$\{2, 3\} \Rightarrow \{1\}$	50	6	R

Table 3. Association Rules mined from the frequent pattern $\{1, 2, 3\}$.

For the TDB given in Table 1 we have generated 188 frequent patterns. The number of frequent patterns generated with various support counts is given in Table 4. The total number

of associate rule generated is 1588. The number of association rule with various support counts is given in Table 5. The breakup of the number of association rules generated with various levels of confidence is given in Table 6.

S.No	**Support %**	**No. of Patterns Generated**	**S.No**	**Support %**	**No. of Patterns Generated**
1	3	188	10	30	3
2	6	113	11	33	3
3	9	84	12	36	3
4	12	72	13	39	1
5	15	25	14	42	1
6	18	14	15	45	1
7	21	12	16	48	1
8	24	9	17	51	0
9	27	5	18	54	0

Table 4. Number of patterns with various support counts

S.No	**Support % ≥**	**No. of Association Rules Generated**	**S.No**	Support % ≥	**No. of Association Rules Generated**
1	3	1588	10	30	0
2	6	746	11	33	0
3	9	484	12	36	0
4	12	292	13	39	0
5	15	38	14	42	0
6	18	16	15	45	0
7	21	10	16	48	0
8	24	2	17	51	0
9	27	0	18	54	0

Table 5. Number of association rules with various support counts

S.No	Confidence %	No. of Association Rules Generated
1	< 50	465
2	50-59	162
3	60-69	81
4	70-79	36
5	80-89	95
6	≥ 90	749

Table 6. Number of strong association rules with various confidence levels

7 Conclusion

In this paper we have proposed a new data structure consisting of a directed graph D and a path system in D for representing a TDB. We have presented algorithms for constructing D, for generating frequent patterns using D and for generating association rules. During the entire process the data is scanned exactly once. Further it is possible to get several information about the TDB by using graph theoretic parameters. For example if in-degree$(i) = 0$ in the directed

graph D_1 obtained from D by removing all the loops, then the item i always appears as the first item in every transaction T with $i \in T$. Similarly if out-degree$(i) = 0$, then the item i always appears as the last item in every transaction T with $i \in T$. Our algorithm can be used to identify all such items. Use of other graph theoretic parameters to extract new knowledge about the TDB and the comparison of the performance of this algorithm with other existing algorithms in the literature using real data set will be reported in a subsequent paper.

Acknowledgment

The first author is thankful to the Department of Science and Technology, New Delhi for its support through the n-CARDMATH Project No. SR/S4/MS:427/07.

Bibliography

[1] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large database, In *Proc. of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 93)*, Washington, USA, 22(2)207-2016, May 1993.

[2] G. Chartrand and L. Lesniak, *Graphs and Digraphs*, Chapman and Hall, CRC, 4^{th} edition, 2005.

[3] R. Agrawal and R. Srikant, Fast Algorithms for mining association rules, In *Proc. of the 20^{th} International Conference on Very Large Database (VLDB' 94)*, Santiago, Chile, 487-499, June 1994.

[4] J. Han, J. Pei and Y. Yin, Mining Frequent Patterns without Candidate Generation, In *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, 29(2):1-12, May 2000.

[5] S. Brin, R. Motwani, J.D. Ullman and S. Tsur, Dynamic itemset counting and implications rules for masket basket data, In *Proc. of the ACM SIGMOD International Conference on Management Data*, 26(2):255-264, 1997.

[6] M. Hontsma and A. Swami, *Set oriented mining for association rules in relatrend database*, The technical report RJ9567, IBM Almaden Research Centre, San Jose, California, October 1993.

[7] J. Han and M. Kamber, Data minining, Concepts and Applications, Elsevier Inc., (2006).