Mining of Sequential Patterns using Directed Graphs

Sabeen S, R. Arunadevi, Kanisha, R.Kesavan

Abstract: Sequential pattern mining is one of the important functionalities of data mining. It is used for analyzing sequential database and discovers sequential patterns. It is focused for extracting interesting subsequences from a set of sequences. Various factors such as rate of occurrence, length, and profit are used to define the interestingness of subsequence derived from the sequence database. Sequential pattern mining has abundant real-life applications since sequential data is logically programmed as sequences of cipher in many fields such as bioinformatics, e-learning, market basket analysis, texts, and webpage click-stream analysis. A large diversity of competent algorithms such as PrefixSpan, GSP and Freespan have been proposed during the past few years. In this paper we propose a data model for organizing the sequential database, which consists of a directed graph DGS (cycles and several edges are allowed) and an organization of directed paths in DGS to represent a sequential data for discovering sequential patterns from a sequence database. Competent algorithms for constructing the digraph model (DGS) for extracting all sequential patterns and mining association rules are proposed. A number of theoretical parameters of digraph model are also introduced, which lead to more understanding of the problem.

Keywords: sequential pattern, data mining, directed graph, directed path, frequent item, association rule, minimum support.

I. DIGRAPH MODEL FOR SEQUENCE DATABASE

Consider a set of items, \( I = \{1, 2, \ldots, n\} \). "A sequence, \( s \) is any nonempty subset of items in \( I \). A sequence database \( S \) is a set of sequences in \( I \). A subsequence \( x \subseteq s \subseteq S \) may take place more than once in \( DGS \). We assume that for each \( i \in I \), there is at least one sequence \( s \in S \) with \( i \in S \). In other words \( \bigcup_{s \in S} s = I \). A sequence \( s \in S \) is said to support a subsequence \( x, x \subseteq S \) if \( s \subseteq S \). The support of \( x \) is defined by \( \text{sup}(x) = \frac{|\{s \in S : x \subseteq s\}|}{|S|} \). Thus \( \text{sup}(x) \) is the percentage of sequences contains \( x \). If a threshold \( \text{min\_sup} \) is fix, then any subsequence \( x \) of \( s \) with \( \text{sup}(x) \geq \text{min\_sup} \) is called a sequential pattern”. "An association rule is an implication of the form \( x \Rightarrow y \) where \( x, y \subseteq S \) and \( x \cap y = \emptyset \). The support of rule \( x \Rightarrow y \) is defined to be \( \text{sup}(x \cup y) \). The confidence of the rule is defined as \( \text{conf}(x \Rightarrow y) = \frac{\text{sup}(x \cup y)}{\text{sup}(x)} \).

The support and confidence values are usually normalized so that these values occur between 0 and 100 instead of 0 and 1.0”. Association rules can be generated from the extracted sequential patterns[1][4][8][13]. The sequential pattern mining problem was first proposed by Agrawal[1]. Consider a set of sequences, where every sequence contains a set of subsequence and each subsequence consists of a list of items. For a given user-defined minimum support, the problem of sequential pattern can be defined as the extraction of all the frequent subsequences.

II. DEVELOPMENT OF DIGRAPH MODEL FOR THE SEQUENCE DATABASE, DGS

A method to construct the data model, \( DGS \) for a sequence database, \( S \) is proposed in this section. This method dynamically constructs the data model \( DGS \) by scanning the database only once and at the same time extract a number of factors such as incidence frequency of each node, how many loops are there in each node, occurrence frequency of each edge, how many edges in \( DGS \), the highest length of a sequence, and each node’s in-degree and out-degree.

The algorithm for constructing data model, \( DGS \) of a sequence database, initially constructs separate nodes for each item with occurrence frequency 0. Then scanning the sequence database one time and read each sequence \( S_j \) and construct the directed route or path representing the sequence in \( DGS \). Let \( \langle e_1, e_2, \ldots, e_k \rangle \) be a sequence, with \( k \) elements \( e_1, e_2, \ldots, e_k \). Each element \( e_i \) is an itemset denoted as \( (i_1, i_2, \ldots, i_j) \) where \( i_j \) is an item, \( i_j \in I \). An edge \( (i_1, i_{j+1}) \) is implemented by linked list. Each node in the linked list has two fields for storing prefix patterns \( (i_1, i_2, \ldots, i_j) \), which is name of the edge(i, j) and occurrence frequency of the edge(i, j). For instance the data model given in Figure 2, the edge (a, b) occurs with label \( \langle ab \rangle \) and also with values \( \langle ef(ab) \rangle \) indicating that it occurs once as part of the sequences \( S_{10} \) and \( S_{30} \).
These values are stored in memory dynamically. The algorithm for developing the data model DGS is given in Figure 1.

A. Algorithm for Directed graph of Sequence Database, DGS

The pseudo code for constructing a directed graph for a sequence database is follows:

Algorithm: DGS, Directed graph of Sequence database

Input: S, Sequence database,
I, set of items
n, items count
m, total sequences

Output:
\( f(i) \): frequency of each item
K: Highest Length of a sequence
N(E): Total edges in DGS
id(i): in-degree of a node i.
\( od(i) \): out-degree of a node i.
\( L_r(i) \): Number of loops at node i.
\( InEdge(i) \): Number of distinct edges to node i.
\( OutEdge(i) \): Number of distinct edges from node i.
Label \( (e_c, i) \): Label assigned to the edge \( e_c \) with head i.
\( f(e_c, i) \): frequency of edge \( e_c \) and Label \( (e_c, i) \)
\( l(S_j) \): length of sequence \( S_j \) (i.e., count of items in \( S_j \))
\( E(S_j) \): set of all subsequence in \( S_j \)
\( N(S_j) \): count of subsequences in \( S_j \)
K: Highest Length of sequence
\( N(S_j^c) \): Number of instances in \( S_j \)

DGS: The directed graph of the sequence database.

Method:

for each item \( i \in I \), \( 1 \leq i \leq n \), do
CreateNode(i);
\( f(i) = 0; \)
\( od(i) = 0; \)
\( id(i) = 0; \)
\( InEdge(i) = 0; \)
\( OutEdge(i) = 0; \)
\( L_r(i) = 0; \) // \( 1 \leq i \leq n \)
\( K = 0 \)
end for

for each sequence \( S_j \in S \), \( 1 \leq j \leq m \), do
\( E(S_j) = GetElements(S_j); // assigning all elements of \( S_j \) to \( E(S_j) \) \)
\( N(S_j) = |E(S_j)|; \)
\( item = \phi; \)
\( prefix = \phi; \)
\( N(S_j^c) = 0; \)
for each element in \( S_j \), \( E_k(S_j) \in E(S_j), 1 \leq k \leq N(S_j) \), do
\( predecessor = r = \phi; \)
\( X = \phi; \)
for each item \( i \in E_k(S_j) \)
\( N(S_j^c) ++; \)
end for
end for

if \( (i \notin item) \) then
\( f(i)++; \)
item = \( \bigcup \{i\} \);
\( x = \bigcup \{i\} \);
else
\( x = \bigcup \{i\} \);
end if

if \( |E_k(S_j)| = 1 \) then
\( L_r(i)++; \)
if \( (L_r(i) = 1) \) then
CreateEdge(i, i);
\( N(E)++; \)
\( f(e_c, i) = 1; \)
Label \( (e_c, i) = \langle prefix \rangle X : f(e_c, i) \);
prefix = \( \bigcup \{i\} \);
end if
if \( (L_r(i) > 1) \) then
\( f(e_c, i)++; \)
end if
od(i) ++;
id(i) ++;
\( InEdge(i) + +; \)
\( OutEdge(i) + +; \)
end if
if \( (\text{predecessor} \neq \phi) \) then
if \( (X \notin \text{any Label}(e_c, i)) \) then
CreateEdge \( \langle \text{predecessor}, i \rangle \);
\( N(E)++; \)
\( f(e_c, i) = 1; \)
Label \( (e_c, i) = \langle prefix \rangle X : f(e_c, i) \);
prefix = \( \bigcup \{i\} \);
od(\text{predecessor}) ++;
id(i) ++;
\( InEdge(i) + +; \)
\( OutEdge(\text{predecessor}) + +; \)
end if
if \( (X \in \text{any Label}(e_c, i)) \) then
\( f(e_c, i)++; \)
od(\text{predecessor}) ++;
id(i) ++;
end if
prefix = \( \bigcup \{i\} \);
predecessor = i;
end if
end for
end if
end for
return DGS

Figure 1: Pseudo code for constructing directed graph of a sequence database, DGS

For example, let the running example adopted in Pei, Han, Behzad Mortazavi, Pinto(2004) database be a sequence database $S^{17}$, shown in Table 1 and minimum support count = 2. The list of items in the sequence database is, $I=\{a, b, c, d, e, f, g\}$.

**Table 1 A sequence database**

<table>
<thead>
<tr>
<th>Sid</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>S10</td>
<td>$\langle a(abc)(ac)d(cf) \rangle$</td>
</tr>
<tr>
<td>S20</td>
<td>$\langle (ad)c(bc)(ae) \rangle$</td>
</tr>
<tr>
<td>S30</td>
<td>$\langle ef\rangle(ab)(df\rangle cb$</td>
</tr>
<tr>
<td>S40</td>
<td>$\langle eg(af)cbc\rangle$</td>
</tr>
</tbody>
</table>

Consider the sequence $\langle a(abc)(ac)d(cf) \rangle$ has 5 subsequence: (a), (abc), (ac), (d) and (cf) where item a and c show more than once in distinct subsequences. It is a 9-sequence since total 9 instances appearing in that sequence. Support count of a sequence can be calculated as follows: Item a occur 3 times in this sequence so it contributes 3 to the length of the sequence but the whole sequence $\langle a(abc)(ac)d(cf) \rangle$ contributes only one to the support of that sequence$^{16}$. Also, sequence $\langle a(bc)df \rangle$ is a subsequence of $\langle a(abc)(ac)d(cf) \rangle$. Since both sequences S10 and S30 contain subsequence $x = \langle (ab)c \rangle$, $x$ is a sequential pattern of length 3, that is, 3-pattern.

![Figure 2 DGS of a sequence database in Table 1](image)

III. MINING OF SEQUENTIAL PATTERNS FROM DGS

In this section, a method for mining all sequential patterns[12], $S_L$ from DGS is presented. For each node $i$, the method finds all sequential patterns ends with $i$. Suppose the number of edges ends on a node $i$ is 0 i.e., $InEdge(i)=0$, $i$ is the only pattern ends with $i$. Otherwise for each edge $e_i$ with $i$ as head, all subsets, $X$ of $Label(e_i, i)$ such that $|X| \geq 2$ and $i \in X$ are considered. Then the support count of $X$ is the total support counts of all the edges $e_i$ with distinct prefix and $i$ as head for which $X \subseteq Label(e_i,i)$. If this support count is greater than or equal to the minimum support $s$, then $X$ is included in the list of all sequential patterns. The pseudo code for extracting frequent patterns from a DGS is XoSP is shown in Figure 3.

A. Algorithm for Extracting Sequential Patterns, XoSP

The algorithm for extracting sequential patterns form the directed graph of the sequence database is presented as follows.

**Algorithm:** XoSP, Extraction of Sequential Patterns from DGS

**Input:** $s$: Minimum support threshold.

**DGS:** a model of sequence database $S$.

**Output:** $S_L$: List of sequential patterns extracted from DGS.

$m$: Number of sequential patterns.

**Method:**

$L=\phi, m=0$;

**for each** node$^{17} i, i \in DGS$ **do**

if ($f(i) \geq s$) then

$S_L = \cup \{i : f(i)\}$;

$m++$;

**end if**

if ($|\text{Edge}(i)| > 0$) then

$f = 0$;

**for each** $Label(e_i, i)$, as $i$ in last element of edge label do

$f = f(e_i, i)$;

$W = Label(e_i, i)$;

if ($W \notin S_L$) then // $S_L$ is the set of patterns already extracted

**for each** subsequence, $x$ ends with

$i, x \in W \& \& x \notin S_L$

**for each** $Label(e_y, i)$ do // other than $Label(e_i, i)$

if ((x $\in Label(e_y, i)$) &&

$(\text{prefix}(Label(e_y, i)) \neq \text{prefix}(Label(e_i, i)))$

$f = f(e_i, i)$;

**end if**

**end for**

**end for**

if ($f \geq s$) then

$S_L = \cup \{x : f\}$

$m++$;

**end if**

**end for**

**end for**

**end if**
Mining of Sequential Patterns using Directed Graphs

The algorithm DGS is illustrated with a sequence database consisting of 4 sequences and 7 items shown in Table 1: by applying XoSP to the DGS shown in Figure 2 all sequential patterns from the nodes of DGS have been extracted. The

```
begin if
end if
begin if
end for
return $S_L$;
```

Figure 3 Pseudo code of XoSP for extracting sequential patterns from DGS

The process of extracting sequential patterns using XoSP from a DGS is given below:

XoSP algorithm starts from the first node, that is, the node representing the first item in the list of items. For the directed graph DGS in Figure 3.13, XoSP starts from the node $a$. Frequency of node $a$, $f(a) = 4$ which is more than the minimum support, $s$. So $\langle a \rangle$ is a sequential pattern and is included in the set of all $f$ sequential patterns, $S_L$. The number of edges of the node $a$ can be calculated using the formula $|Edge(a)| = InEdge(i) + OutEdge(i) - L_i(i)$, where InEdge($i$) is the number in-edges of the node $a$ and OutEdge($i$) is the number of out-edges of the node $a$, and $L_i(i)$ is the number of loops at $i$. Thus $|Edge(a)| = 2 + 8 - 2 = 8$. The $Edge(a)$ with edge label, $\langle a(a) \rangle \not\subseteq S_L$ has frequency 1 and not yet been extracted as a sequential pattern. Let $w = \langle a(a) \rangle$. The subsequence $x \in W$, which ends with item $a$ are $\langle a \rangle$ and $\langle a(a) \rangle$. The subsequence $\langle a \rangle$ is already extracted and is contained in $S_L$. The sequence $\langle a(a) \rangle \not\subseteq S_L$. To get the net frequency of $\langle a(a) \rangle$ we have to consider total support count of the edges with different prefix in the edge label and contain $\langle a \rangle$ as the subsequence. One out-edge of $a$ with frequency 1, contains $\langle a(a) \rangle$ in its label as a subsequence with different prefix. Hence $f(\langle a(a) \rangle) = 1 + 1 = 2$, satisfying minimum support $s$, so $\langle a(a) \rangle$ is a sequential pattern that can be added to $S_L$.

For out-edge of node $a$ with label $\langle egf(a) \rangle$ and its prefix is $\langle ega \rangle \not\subseteq S_L$ and $f(ega) = 1$. Let $W = \langle ega \rangle$. The subsequences, $x \in W$ that ends with $a$ are $x = \{ \langle ea \rangle, \langle ga \rangle, \langle ega \rangle \}$. The subsequence $x_i = \langle ea \rangle \not\subseteq S_L$. We can find net frequency of the sub sequence $x_i = \langle ea \rangle$ by considering total support count of edges of $a$ which contain $\langle ea \rangle$ as a subsequence in its label with different prefix. One out-edge of node $a$ has $\langle ea \rangle$ as a subsequence in its label with frequency of the edge as 1. Hence the net frequency $f(\langle ea \rangle) = 1 + 1 + 2 = 5 \geq s$. So the subsequence $\langle ea \rangle$ is a sequential pattern with frequency 2 and can be added to $S_L$. For the next subsequence $x_i = \langle ea \rangle \not\subseteq S_L$, and cannot be a sequential pattern since the net frequency of $\langle ea \rangle$ is $f(\langle ea \rangle) = 1$ only. In this manner all the sequential patterns can be extracted from each node in the DGS. The set of all sequential patterns along with the respective frequencies mined from all the nodes of DGS is given in Table 2.

IV. EXPERIMENT RESULTS

The experimental results on the performance of DGS and comparisons with Prefixspan, GSP and Freespan [2][3][6][12] are discussed in this section. It shows that DGS producing better results than other previously proposed methods. This approach of mining sequential pattern is efficient and scalable in large databases. It is efficient and scalable for mining sequential patterns in large databases. In this paper, performance of our approach on a sample data set C1078S818[5][7][10] is provided. In this data set the number of items is set to 1000 and there are 10,000 sequences in the data set. The number of items and sequences taken for this data set is 1000 and 10,000 respectively.

![Image](image_url)
The average number of items and subsequences with in elements and sequence are set to 8 and is denoted as T8 and S8. We can increase the number of long sequential patterns when minimum support count is set at low level. The result of our experiment on given data set on scalability with different support count is shown in Figure 3.

450 400 350 300 250 200 150 100
Runtime (seconds)

Figure 3 Runtime comparisons among DGS, Prefixspan, GSP and Freespan

When we increase the minimum support count the number of sequential patterns generated is less. As the support threshold decreases the gaps between the runtime of four methods becomes clear. Both Freespan and Prefixspan are faster than GSP. But DGS method is more efficient and faster than and more scalable than Freespan and Prefixspan techniques$^{11,14,15}$

V. CONCLUSION

The algorithms for mining sequential patterns from big databases are studied and developed by using a new digraph model DGS. In this approach no candidates are generated and test approach such as GSP and Prefixspan are performed. This approach generates a directed graph DGS for sequence database and grows sequential patterns through the nodes of the DGS. The size of the DGS is always less than that of S, sequence database. This approach constructs a digraph model, DGS while scanning the sequence database S first time. No further scanning the sequence database S is necessary for extracting sequential patterns. In summary, DGS is more efficient and scalable than, GSP, Freespan and Prefixspan.

REFERENCES

17. dblab.cs.nccu.edu.tw
18. Jia-Wei Han, Jian Pei, Xi-Feng Yan. “From sequential pattern mining to structured pattern mining: A pattern-growth approach”, Journal of Computer Science and Technology, 2004

AUTHORS PROFILE

Dr. Sabeen S is presently working in the Department of Computer Science, SRM Institute of Science and Technology. He received his Doctoral degree from Anna University in the year 2012. He received his post graduate degree MCA in the year 2002 and M.Tech(CSE) in the year 2015. He has in total 18 years of teaching experience in various engineering colleges, Mohammed Sathak Engineering College, Nooral Islam College of Engineering, Jaya Engineering College and Sudhirth College of Engineering and Technology. He is having life membership in ISTE. His area of interest includes Data mining, IOT, Machine Learning. He has published many papers in international journals. He has guided more than 100 projects of post graduate students.
Dr. R. Arunadevi is presently working in the Department of Computer Science, Vidhyasagar Women’s College. She has completed MCA in 1996, M.Phil in 2005 and received her Ph.D. in 2017 from Manonmaniam Sundaranar University. She has got more than 23 years of teaching experience in Computer science. Her area of research includes Data Mining, Cloud computing and Network Security. She has guided many M.Phil. Scholars and provided project guidance to post graduate students and published many research articles.

Dr. B. Kanisha received her Ph.D. from Anna University, Chennai, India. She is working as an Associate Professor in the Department of Information Technology in Veltech Multitech Dr.RR &Dr.SR Engineering College, Chennai, India. She has 15 years of academic experience. She has published many papers in reputed journals and international conference proceedings. She has guided more than many undergraduate and post graduate students in their project works. Her research and teaching interests include human computer interaction, speech recognition, neural networks and data mining.

Dr. R. Kesavan is Professor of Department of Information Technology, Jaya Engineering College, Chennai, India. He received his Bachelor’s degree in Computer Science from Bharathidasan University, Trichy, and Master of Science in Computer Science from Bharathidasan University, Master of Philosophy from Manonmaniam Sundaranar University, Thirunelveli, and Tamilnadu, India. Master of Technology in Computer Science and Engineering from Dr. M G R University, Chennai. He completed his Doctorate Degree from St. Peter’s University, Chennai. He has been in teaching profession for the last 17 years. His research area is Mobile Adhoc Networks. He presented and published papers in conference proceedings and National and International Journals. He is a life member of Indian Society for Technical Education(ISTE), Member of International Association of Engineers(IAENG) and Editorial Board Member in IJCIS, IJN&A and IJSA International Journals.