International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-11, September 2019

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 pattern[3] 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 $x \subseteq s \in S$ 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 $\sup(x) = \frac{|s \in S : x \subseteq s|}{|S|}$. Thus $\sup(x)$ is the percentage of sequences contains x. If a threshold *min_supp* is fix, then any subsequence x of s with $\sup(x) \geq min\_supp$ is called a sequential pattern". "An association rule is an implication of the form $x \Rightarrow y$ where $x, y \subseteq S$ and $x \cap y = \phi$. The support of rule $x \Rightarrow y$ is defined to be $\sup(x \cup y)$. The confidence of the rule is defined as $conf\,(x \Rightarrow y) = \frac{\sup(x \cup y)}{\sup(x)}$.

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 $e_1, e_2, \ldots$, and e_k. Each element e_j is an itemset denoted as $(i_1, i_2, \ldots, i_k)$ where i_j is an item, $i_j \in I$. An edge (i_j, 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 $\langle a(ab) \rangle$ and also with values $\langle ef\,(ab) \rangle$ indicating that it occurs once as part of the sequences S_{10} and S_{30}.

Manuscript published on 30 September 2019.
***Correspondence Author(s)**
Dr. Sabeen S, Department of Computer Science, SRM Institute of Sceince and Technology, Kattankulathur, Tamilnadu, India. Email: sabeens@srmist.edu.in
Dr. R. Arunadevi, Department of Computer Science, Vidhya Sagar Women's College, Chengalpettu,Tamilnadu, India. .Email: arunaa_2008@yahoo.com
Dr. B. Kanisha, Department of Information Technology, Veltech Multitech Dr.RR & Dr.SR Engineering College, Chennai, India. Email:johnnykanisha@gmail.com
Dr. R. Kesavan, Department of Information Technology, Jaya Engineering College, Thiruninravur, Chennai, Tamilnadu, India. Email: kesavanmcajec@gmail.com

Retrieval Number: K22420981119/19©BEIESP
DOI: 10.35940/ijitee.K2242.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

These values are stored in memory dynamically. The algorithm for developing the data model DGS is given in Figure1.

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*.
$L_c(i)$: 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 with head *i* 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 items in instances of S_j;
DGS: The directed graph of the sequence database.
Method:

```
for each item i ∈ I, 1 ≤ i ≤ n , do
    CreateNode(i);
      f (i) = 0;
        od (i) = 0;
        id (i) = 0;
        InEdge (i) = 0;
    OutEdge (i) = 0;
    L_c (i) = 0; // 1 ≤ i ≤ n
    K = 0
end for
K = 0;
for each sequence S_j ∈ S, 1 ≤ j ≤ 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 = φ ; prefix = φ ; N(S_j^c) =0;
for each element in S_j , E_k(S_j) ∈ E(S_j),1 ≤ k ≤ N(S_j) do
    predecessor = φ; X = φ;
        for each item i ∈ E_k(S_j)
    N(S_j^c) ++;
if (i ∉ item) then
  f (i)++;
  item = ∪{i};
  X = ∪{i};
else  X = ∪{i};
end if
if ( |E_k(S_j)| = 1)  then
    L_c (i)++;
    if ( L_c (i)= 1) then
  CreateEdge(i,i);
  N(E)++;
  f(e_c ,i) = 1;
  Label(e_c ,i) = ⟨prefix⟩ X : f(e_c, i) ;
  Prefix = ∪{i};
   end if
      if ( L_c (i) > 1) then
  f (e_c ,i)++;
   end if
     od (i) + +;
     id (i) + +;
     InEdge (i) + +;
     OutEdge (i) + +;
          end if
          if (predecessor ≠ φ ) then
             if (X ∉ any Label(e_c ,i)) then
                CreateEdge (predecessor, i);
                N (E) + +;
                f (e_c ,i) = 1;
     Label (e_c ,i) = ⟨prefix⟩ X : f(e_c, i) ;
     prefix = ∪{i};
          od (predecessor) + +;
          id (i) + +;
          InEdge (i) + +;
          OutEdge (predecessor) + +;
       end if
     if (X ∈ any Label(e_c ,i)) then
          f (e_c ,i)++;
          od (predecessor) + +;
         id (i) + +;
       end if
          prefix = ∪{i};
          predecessor = i ;
     end for
  end for
  I (S_j) = item;
  L(S_j)= N(S_j^c)
  if(N(S_j^c) > K ) then
     K= N(S_j^c) ;
  end if
```


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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[9] be a *sequence database S*[17], shown in Table1 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[18]

Sid	Sequence
S10	$\langle a(abc)(ac)d(cf)\rangle$
S20	$\langle (ad)c(bc)(ae)\rangle$
S30	$\langle (ef)(ab)(df)cb\rangle$
S40	$\langle eg(af)cbc\rangle$

Consider the sequence $\langle a(abc)(ac)d(cf)\rangle$ has 5 sub sequence: (*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 sub sequence $x=\langle (ab)c\rangle$, *x* is a sequential pattern of length 3, that is, *3-pattern*.

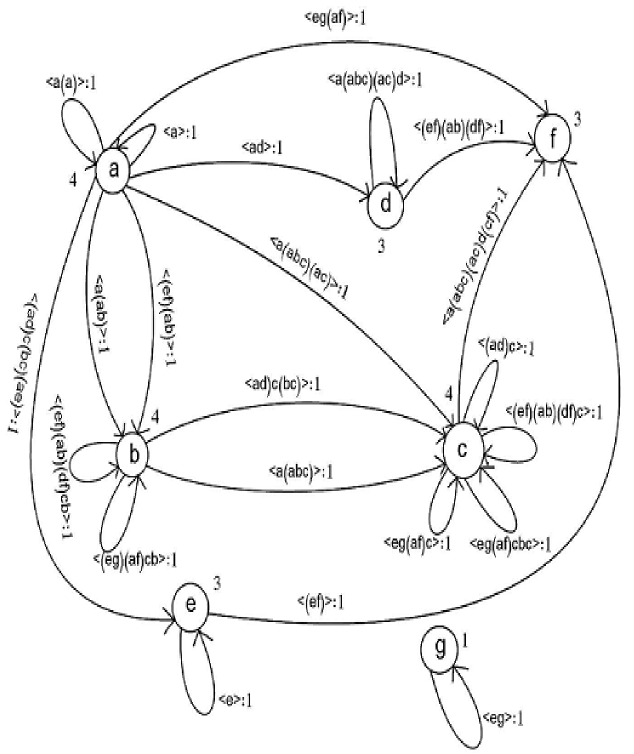

Figure 2 DGS of a sequence database in Table 1

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_c with *i* as head, all subsets, *X* of *Label*(e_c, *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_c with distinct prefix and *i* as head for which $X \subseteq Label(e_c, i)$. 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. Agorithm 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[12] *i*, $i \in DGS$ ***do***
if $(f(i) \geq s)$ ***then***
$S_L = \bigcup \langle i : f(i)\rangle;$
m++;
end if
if $(|Edge(i)|) > 0$ ***then***
f = 0;
for each *Label*(e_c ,*i*), as *i* in last element of edge label ***do***
$f = f(e_c, i);$
$W = Label(e_c, i);$
if $(W \not\subseteq S_L)$ ***then*** // S_L is the set of patterns already extracted
for each subsequence, *x* ends with $i, x \in W \,\&\&\, x \not\subseteq S_L$ ***do***
for each $Label(e_y, i)$ ***do*** // other than *Label* (e_c ,*i*)
if
$((x \in Label(e_y, i))$ &&
$(prefix(Label(e_y, i)) \neq prefix(Label(e_c, i))))$
$f = + f(e_y, i);$
end if
end for
end for
if $(f \geq s)$ ***then***
$S_L = \bigcup \langle x : f\rangle$
m++;
end if

Retrieval Number: K22420981119/19©BEIESP
DOI: 10.35940/ijitee.K2242.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

end if
end for
end if
end for
return S_L**;**

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

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

Node Id	Sequential Patterns	Frequency of the Patterns	Node Id	Sequential Patterns	Frequency of the Patterns
a	$\langle a \rangle$	4	C	$\langle abc \rangle$	4
	$\langle aa \rangle$	2		$\langle (ab)c \rangle$	2
	$\langle aba \rangle$	2		$\langle (ab)dc \rangle$	2
	$\langle ba \rangle$	2		$\langle ac \rangle$	4
	$\langle ca \rangle$	2		$\langle acc \rangle$	3
	$\langle aca \rangle$	2		$\langle adc \rangle$	2
	$\langle a(bc)a \rangle$	2		$\langle bc \rangle$	4
	$\langle (bc)a \rangle$	2		$\langle (bc) \rangle$	2
	$\langle ea \rangle$	2		$\langle bdc \rangle$	2
b	$\langle b \rangle$	4		$\langle dc \rangle$	2
	$\langle ab \rangle$	4		$\langle eac \rangle$	2
	$\langle (ab) \rangle$	2		$\langle ebc \rangle$	2
	$\langle acb \rangle$	2		$\langle ec \rangle$	2
	$\langle cb \rangle$	3		$\langle efc \rangle$	2
	$\langle db \rangle$	2		$\langle fbc \rangle$	2
	$\langle dcb \rangle$	2		$\langle fc \rangle$	2
	$\langle eab \rangle$	2	D	$\langle d \rangle$	3
	$\langle eacb \rangle$	2		$\langle bd \rangle$	2
	$\langle eb \rangle$	2		$\langle ad \rangle$	3
	$\langle ecb \rangle$	2		$\langle (ab)d \rangle$	2
	$\langle efb \rangle$	2	E	$\langle e \rangle$	3
	$\langle efcb \rangle$	2	F	$\langle f \rangle$	3
	$\langle fb \rangle$	2		$\langle af \rangle$	3
	$\langle fcb \rangle$	2		$\langle (ab)f \rangle$	2
c	$\langle c \rangle$	4		$\langle bf \rangle$	2
	$\langle cc \rangle$	3		$\langle ef \rangle$	2
	$\langle a(bc) \rangle$	2			

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.[19] 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 allf sequential patterns, S_L. The number of edges of the node *a* can be calculated using the formulae $|Edge(i)|$ = *InEdge*(*i*) + *OutEdge*(*i*) - $L_c(i)$, 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* $L_c(i)$ is the number of loops at *a*. Thus $|Edge(a)|$ = 2 + 8 - 2 = 8. The *Edge* (*a*) with edge label, $\langle a(a) \rangle \notin S_L$ 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 $\langle a(a) \rangle \notin S_L$. 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 $\langle a\ (a) \rangle$ 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 $\langle eg\ (af\ ) \rangle$ and its prefix is $\langle ega \rangle \notin S_L$ 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 $x_1 = \langle ea \rangle \notin S_L$. We can find net frequency of the sub sequence $x_1 = \langle ea \rangle$ 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 $\geq$ *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 $x_2 = \langle ea \rangle \notin S_L$, 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 P*refixspan, 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 C10T8S8I8[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.

Retrieval Number: K22420981119/19©BEIESP
DOI: 10.35940/ijitee.K2242.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

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.

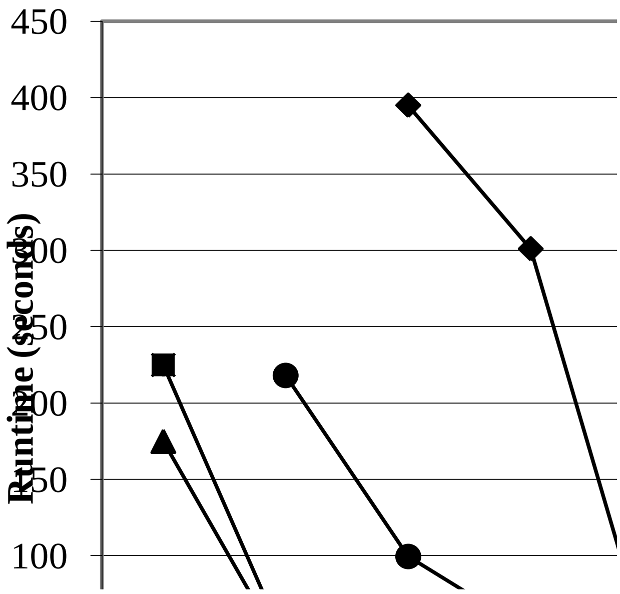

Figure 3 Runtime comparisons among DGS, Prefixspan, GSP and Freespan

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[9,11,14,15].

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

1. Agrawal R and Srikant R., Mining Sequential Patterns. In Proc. 1995 Int.Conf. Data Engineering (ICDE'95), pages 3–14, Taipei, Taiwan, Mar. 1995.
2. Agrawal R, Mannila H., Srikant R., Toivonen H., and Verkamo A.I., "Fast Discovery Of Association Rules" In U.M. Fayyad, G. Piatetsky - Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp. 307–328. MIT Press, 1996
3. Bettini. C, Sean Wang X., and Jajodia S., Mining Temporal Relationships with Multiple Granularities In Time Sequences, Data Engineering Bulletin, 21:32–38, 1998.
4. Garofalakis M, Rastogi R., and Shim K. Spirit: Sequential pattern mining with regular expression constraints. In Proc. Int. Conf. Very Large Data Bases (VLDB'99), pages 223–234, Edinburgh, UK, Sept. 1999.
5. Han J, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In Proc. Int. Conf. Data Engineering (ICDE'99), pages 106–115, Sydney, Australia, April 1999.
6. Lu. H, Han J., and Feng L. Stock movement and n-dimensional intertransaction association rules. In Proc. 1998 SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD'98), pages 12:1– 12:7, Seattle, WA, June 1998.
7. Mannila. H, H Toivonen, and Verkamo A. I. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259–289, 1997.
8. Ozden. B, Ramaswamy S., and Silberschatz A. Cyclic association rules. In Proc. 1998 Int. Conf. Data Engineering (ICDE'98), pages 412–421, Orlando, FL, Feb. 1998.
9. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., ... Hsu, M. C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering, 16(11), 1424-1440. https://doi.org/10.1109/TKDE.2004.
10. Ramaswamy S., Mahajan S., and Silberschatz A. On the discovery of interesting patterns in association rules. In Proc. Int. Conf. Very Large Data Bases (VLDB'98), pages 368–379, New York, NY, Aug. 1998.
11. Sabeen. S., "Classification of Message in Dynamic Notice Board using Android", European Journal of Scientific Research, Vol. 92, No. 4, December 2012.
12. Arumugam, S., Sabeen, S. "Association Rule Mining using Path Systems in Directed Graphs", INT J COMPUT COMMUN, ISSN 1841-9836, 8(6):791-799, December, 2013.
13. Srikant. R and Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In Proc. 5th Int. Conf. Extending Database Technology (EDBT'96), pages 3–17, Avignon, France, Mar. 1996.
14. Wang. J, Chirn G., Marr T., B. Shapiro, Shasha D., and K. Zhang. Combinatiorial pattern discovery for scientific data: Some preliminary results. In Proc. 1994 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'94), pages 115–125, Minneapolis, MN, May, 1994.
15. Zaki M. J. Efficient enumeration of frequent sequences. In Proc. 7th Int. Conf. Information and Knowledge Management (CIKM'98), pages 68–75, Washington DC, Nov. 1998.
16. Jun Ai, Linzhi Huang, Fei Wang, Jiaming Wang. "Research on Relations between Software Network Structure and Fault Propagation", 2016 International Conference on Software Analysis, Testing and Evolution (SATE), 2016.
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
19. "RESEARCH ON THE PROJECTION POSITIONBASED SEQUENTIAL PATTERN MINING ALGORITHM", Control and Intelligent Systems, 2014.

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, Mohamad Sathak Engineering College, Noorul Islam College of Engineering, Jaya Engineering College and Sidharth Institute 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. ArunadeviSecond 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 IJISA International Journals.

Retrieval Number: K22420981119/19©BEIESP
DOI: 10.35940/ijitee.K2242.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)