Graph based Approach for Mining Frequent Sequential Access Patterns of Web pages

Dheeraj Kumar Singh
SoIT, R.G.P.V,
Bhopal (M.P.), India

Varsha Sharma
SoIT, R.G.P.V,
Bhopal (M.P.), India

Sanjeev Sharma
SoIT, R.G.P.V,
Bhopal (M.P.), India

ABSTRACT
The Internet has impacted almost every aspect of our society. Since the number of web sites and web pages has grown rapidly, discovering and understanding web users' surfing behavior are essential for the development of successful web monitoring and recommendation systems. To capture users' web access behavior, one promising approach is web usage mining which discovers interesting and frequent user access patterns from web logs. Sequential Web page Access pattern mining has been a focused theme in data mining research for over a decade with wide range of applications. The aim of discovering frequent sequential access (usage) patterns in Web log data is to obtain information about the navigational behavior of the users. This can be used for advertising purposes, for creating dynamic user profiles etc. We propose a new approach for mining the web usage data by creating graph using web access sequence of sorted web log and mining the useful sequential access pattern.

Keywords
Analysis on Web Usage Data, Graph Based Web Usage Mining, Mining Frequent Sequential Access Patterns From Web Log.

1. INTRODUCTION
The expansion of the World Wide Web has resulted in a large amount of data. The different types of data have to be managed and organized in such a manner that they can be accessed by different users efficiently [4]. Maintaining a website is as important as building it. To maintain website we need to improve its design. To improve the design of website we should find out how it is used [5]. To achieve this, web usage mining is a promising way to provide enhanced web services. Web usage mining, also known as web log mining, aims to evaluate interesting and frequent user browsing behavior from web browsing data that are stored in web server logs, browser logs, proxy server logs [11].

Web Usage mining [1] is the process of applying data mining techniques to the discovery of usage patterns from Web data, targeted towards various practical applications such as personalized web search and surfing, web recommendation systems. Data mining efforts associated with the Web, called Web mining, can be broadly divided into three classes, i.e. web content mining, web structure mining, and web usage mining .The primary task of Web Usage mining is pattern discovery & analysis, that is the only reason it is also called Web access pattern mining [7]. Many approaches [4-11] have been proposed for discovering sequential patterns from web data. However, most of the pervious works based on recursive process of candidate generation and testing the candidates, that is costly in terms of both time and space. Also there are some methods based on tree structure like FP-growth,MFP,that are suffering from problem of repetition of same item node resulting more space requirement to store many copies of same item. The main advantage of using graph is that there is only one node for a page or item. This requires less memory.

Specially for finding sequential access patterns, graph mining can be used to discover useful access patterns through complex user's browsing behavior. The structure of Web site can be modeled as a graph, in which, web pages represent by nodes, and edges represent hyperlinks between the pages. User's navigation on the Web site can be modeled as traversals on the graph. Capturing user's access patterns in such environments is referred to as mining usage patterns. Each traversal can be represented as a sequence of nodes. Once the graph is created, valuable information such as frequent sequential usage patterns can be discovered. For example, when a user visits several pages of a web site within a particular time limit called session, we can simulate the navigation path as a graph and apply graph mining to discover sequential web usage pattern [6].

2. MINING FREQUENT SEQUENTIAL ACCESS PATTERNS FROM WEB LOG
Frequent patterns are item sets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold called minimum support. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent item set. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a frequent sequential pattern. A substructure can refer to different structural forms, such as sub graphs, sub trees, or sub lattices, which may be combined with item sets or subsequences. If a substructure occurs frequently in a graph database, it is called a frequent structural pattern [2].

The sequential pattern mining problem was first introduced by Agrawal and Srikant [3]. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified minimum support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than minimum support. By applying these mining approaches on web data, we can get Frequent Sequential Web Access Patterns.

3. PROPOSED METHODOLOGY
The proposed algorithm constructs a graph to capture the user's web access behavior of a website and then uses the data mining steps in order to find out the Frequent Sequential Web Access Patterns.

In this approach, firstly we are applying the pre-processing step on Input Web server logs to get the list of accessed web page sequence within different sessions, and then will construct a Web Usage Graph using accessed web page sequence in all sessions. Finally applying pruning and mining steps to mine the useful sequential user access patterns. Figure 1 gives an overview of the proposed Graph-based web usage mining technique for mining Frequent Sequential Access Patterns of webpages for a particular website. The proposed approach consists of the following four steps:
(I) Preprocessing; (II) Constructing Web Usage Graph; (III) Pruning the Web Usage Graph; and (IV) Mining Frequent Sequential Web Access Patterns from Web Usage Graph.

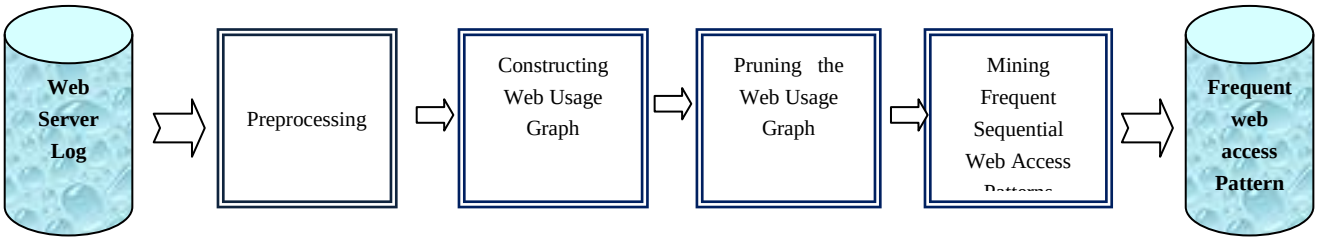

Figure 1: Overview of the Graph-based web usage mining approach.

3.1 Preprocessing

The preprocessing [4] task contains three separate phases. Firstly, the collected data must be cleaned, which means that graphic and multimedia entries are removed. Secondly, the different sessions belonging to different users should be identified. Third one is to identify the sessions from the raw data. A session is a group of activities performed by a user when he is navigating through a given site. For web server logs, all users' access activities of a website are recorded by the Web server of the website. Each user access record contains the client IP address, request time, requested URL, HTTP status code, etc. Users are treated as anonymous since the IP addresses are not mapped to any user-identifiable profile database. Web logs can be regarded as a collection of sequences of access events from one user or session in timestamp ascending order. Preprocessing tasks can be applied to the original web Log files to obtain all web access sessions [10].Table 1 shows the preprocessing step applied on a sample raw web log data shown in figure 2.Here we are defining different sessions according to their time stamp order, with a time interval of 1 hour for each session. For example, the Session S1 includes all the web pages accessed during time duration of 09:30-10:30 that are P1,P2,P1,P3,P4, similarly S2 has P1,P3,P5 page sequence and S3 has P4,P1 page sequence.

```
2012-01-15 00:09:33 192.168.15.194GET /P1.htm 200
2012-01-15 00:09:34 192.168.15.194GET /P2/images/t.gif200
2012-01-15 00:10:30 192.168.15.194GET /P4.php 200
2012-01-15 00:10:04 192.168.15.194GET / P1.htm 200
2012-01-15 00:10:04 192.168.15.194 GET /P3.php 200
2012-01-15 00:22:38 209.188.67.130 GET /P1.htm 304
2012-01-15 00:22:52 209.188.67.130 GET /P3.php 200
2012-01-15 00:23:00 209.188.67.130 GET /P5.php 304
2012-01-16 00:00:30 192.168.15.194GET /P4.php 200
2012-01-16 00:01:04 209.188.67.130GET / P1.htm 200
```

Figure 2: Example of Web log from a server

Table 1. Table of Accessed web page sequence with their respective sessions.

Session Id	Accessed web page sequence
S1	P1,P2,P1,P3,P4
S2	P1,P3,P5
S3	P4,P1

3.2 Constructing Web Usage Graph

Based on list of accessed web page sequence within different sessions, a Directed Graph can be constructed called web usage graph. Graph consists of vertices (nodes) and edges (links) in which, nodes are for web pages, and edges represent sequential access between the pages. The number of nodes in the graph is equal to number of distinct web pages accessed during all sessions. Each node in graph contains weight as node count that stores the number of occurrence of particular pages within different sessions and node id. Each edge in graph contains weight as link count, that represents the frequency of edge, edge id & session id list, list of sessions involved in that path or edge. Algorithm for constructing Web Usage Graph from given Table of Accessed web page sequence is given below.

3.2.1 Algorithm to create a Web Usage graph

Input: Web_Access_Sequence S – Table of Session id with accessed page list within that session.
Output: Web_Usage_graph<Node, Edge> G – Directed Sparse Multigraph where WebPages are represented by nodes and user navigation are represented by edges.

```
Create_graph (Web_Access_Sequence S)
 {
   For(Each Session Si in Web_Access_Sequence S)
      {
       For(Each webpage Pj in page sequence of Si)
          {
            If (i=j=1) {
                        create_node(Pj)
                      set node_count(Pj) =1
                                              }
       Else If (Pj does not exists in all nodes generated so far){
                          create_node(Pj)
```

```
                    set node_count(Pj) =1
                        }
            Else If (Pj does not exists in all nodes generated so far in page sequence of Si)
                    set node_count(Pj) = node_count (Pj) +1
   For(Each adjacent pair of  webpage Pj  in page sequence)
      {
        If (i=j=1) {
          create_edge(source node(Pj),destination node(Pj+1))
              set link_count= 1
              add session_id Si in session_id_list
               }
Else If (an edge with same source, destination does not exists in all edges generated so far) {
          create_edge(source node(Pj), destination node(Pj+1))
           set link_count= 1
           add session_id Si in session_id_list
}
      Else If (an edge with same source, destination does not exists in all edges generated so far in Si) {
                        link_count = link_count + 1
                         add session_id Si in session_id_list on the existing edge
                                        }
                                    }
                    }
For(All edges of graph, If there exist an ordered but not adjacent sequence between nodes in non existing session_id Si in session_id_list of that edge) {
         link_count = link_count + 1
        add session_id Si in session_id_list on the existing edge
         }
}
```

3.3 Pruning Web Usage Graph

In this phase we remove all the nodes having node count less than minimum support threshold and readjust the edges accordingly. While pruning graph if direct edge exists between two frequent nodes, then edge is kept intact. But if there exist edges between two infrequent nodes, then delete that edge .We remove all infrequent nodes and connect the frequent node with edge by which first infrequent node of sequence was connected. Also we check if there exists an edge having link count less than minimum support threshold in the pruned graph, then remove that edge. This step results a Pruned web usage graph that have only frequent nodes and edges. Following Algorithm is for pruning the input web usage graph.

3.3.1 Algorithm to prune the graph

Input: 1. Web_Usage_graph<Node, Edge> G - Directed Sparse Multigraph with,

Node : node containing weight as node_count and node id, and

Edge :edge containing weight as link_count ,edge id & session_id_list.

2. min_count - Threshold For node_count & link_count calculated through given percentage of minimum support,Mathematically expressed as :

min_count=total number of sessions × (% of minimum support)/100

Output: Pruned_Web_Usage_graph<Node,Edge> G'– A directed graph with only frequent nodes and edges.

```
prune_Graph(Web_Usage_graph G<N,E>, min_count)
  {
// This part of algorithm will find and remove all infrequent nodes of input web usage graph. //
For(Each nodes N in Web_Usage_graph G)
        {
  If(node_count(N) < min_count) {
          add node N to non-frequent_node list
          remove node N from Web_Usage_graph G
                        }
Return(Pruned_Web_Usage_graph G')
        }
// This part of algorithm will readjust the edges with their link_count and create new edges if required in Pruned_Web_Usage_graph G'. //
 For(Each nodes N in non-frequent_node list)
    {
       find all incident edges
       add them to incident_list
       find all out edges
       add them to out_list
    create an edge E' with  source node Form incident_list and destination from out_list
       If (an edge E with same source, destination exists as of E' in Pruned_Web_Usage_graph G) {
        link_count(E) = link_count(E) + 1
        add session_id of E' in session_id_list of the existing edge E
           }
          Else {
     add edge E' to Pruned_Web_Usage_graph G
     link_count(E') =1
     add session_id  in session_id_list of E'
                }
          }
// This part of algorithm will check and remove all infrequent edges of pruned web usage graph. //

For (Each edge E in Pruned_Web_Usage_graph G)
        {
    If(link_count(E) < min_count)
          remove edge E from Pruned_Web_Usage_graph G
      }
    }
}
```

3.4 Mining Frequent Sequential Web Access Patterns from Web Usage Graph

This process of mining begins with searching for biggest sequence of nodes, i.e. the longest path with maximum length. Where length is the total number of nodes involved in the sequence. By traversing nodes of Pruned Web Usage graph starting through each node, we get the frequently occurred sequence of nodes that represents frequent web access patterns. The frequency of the sequence will be the minimum link count of all the edges involved in that sequence. In such manner, traverse all the existing path in the Pruned Web Usage graph and enlist all frequent patterns along with their frequencies and arrange them by order of length. Frequent sequence with length of 1 are all the nodes in Pruned Web Usage graph itself, node count will represent their frequency. Following Algorithm is for mining the pruned web usage graph to get set of frequent sequential web access patterns.

3.4.1 Algorithm to mine Frequent Sequential Web Access Patterns

Input: Pruned_Web_Usage_graph <Node, Edge> G'.
Output: Frequent_Web_Access_Patern – Table of frequent_pattern with their respective frequency, arranged according to length of sequence.

```
Mining_graph(Pruned_Web_Usage_graph G')
{
For (All nodes & edges in Pruned_Web_Usage_graph G')
 {
Traverse the longest path
add the visited nodes in the path as sequence
set length=number of nodes in longest sequnce
while (length>1)
 {
Traverse all paths with number of nodes=length, in Pruned_Web_Usage_graph G'
add the visited nodes in the path as frequent_pattern
add frequent_pattern to frequent_pages_list
set frequency of frequent_pattern=minimum of link_count of visited edges in the frequent_pattern
length=length-1
Return (frequent_pattern, frequency, length)
                        }
                    }
If(length==1) {
For (All nodes in Pruned_Web_Usage_graph G')
 {
add the node to frequent_pattern
add frequent_pattern to frequent_pages_list
set frequency of frequent_pattern=node_count
Return (frequent_pattern, frequency, length)
            }
      }
}
```

4. PROBLEM STATEMENT

The problem of web usage access pattern mining is: Given a list of web access sequence of web server log of a website and minimum support threshold and we have to determine the complete set of frequent access patterns with their frequency of occurrence and length.

5. EXPERIMENT RESULT

The above algorithms were implemented in Java Programming language using Net Beans IDE 6.9.1.

Given a dummy set of accessed webpage sequence {ABDAC, EAEBCAC, BABFAE, AFBACFC}, where (A,B,C,D,E,F) are different pages of given website. Table 2, shows the result after session identification step of preprocessing on the given set, which contains a session id for each sequence.

Table 2. Web access sequence Table

Session ID	Web Access Pattern
S1	ABDAC
S2	EAEBCAC
S3	BABFAE
S4	AFBACFC

Using table 2 ,we have constructed a graph called Web Usage Graph, having total 6 nodes (A,B,C,D,E,F) representing distinct web pages accessed within all sessions, shown in figure 3.Each nodes and edges have their respective number of occurrence in distinct sessions.

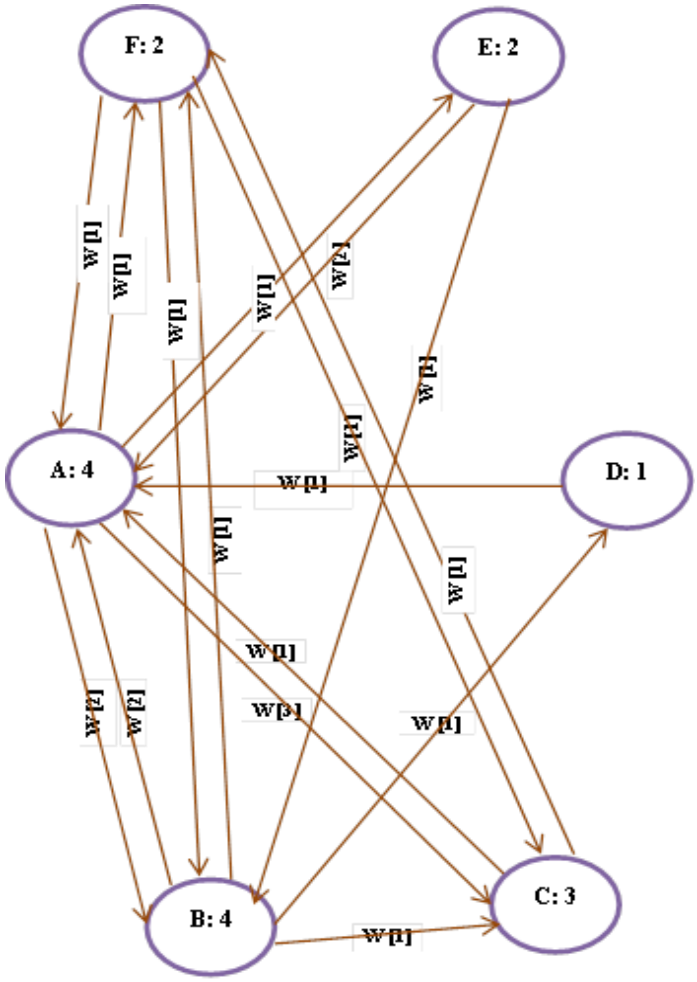

Figure 3: Web Usage Graph

Now we will remove all the infrequent nodes having frequency less than minimum support threshold and readjust the edges of web usage graph shown in figure 3.With minimum support of 75%, we calculated threshold for node count & link count using following formula:

Min count=total number of sessions × (% of minimum support)/100

i.e. Min count=3; So we have removed the nodes D:1,E:2, and F:2 and edges readjusted according to their previous connections. Since the link count of edge 'CA' is 1(i.e. less than 3) after edge readjustment in pruned graph, we have deleted it. Figure 4 shows the Pruned Web Usage Graph.

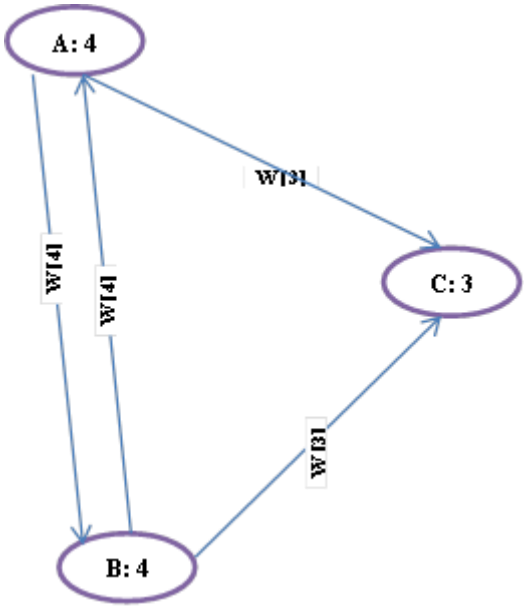

Figure 4: Pruned Web Usage Graph

Finally we traverse the entire existing path from each node in the pruned web usage graph shown in figure 4 and enlist frequent patterns (sub-sequences) with their respective frequencies, according to their length. Table 3 shows the result of mining the pruned web usage graph.

Table 3. List of Frequent Sequential Access Patterns

S.No	Length	Pattern	Frequency
1	4	ABAC	3
2	3	BAC	3
3	3	BAB	4
4	3	ABA	4
5	3	ABC	3
6	2	BC	3
7	2	BA	4
8	2	AB	4
9	2	AC	3
10	1	A	4
11	1	B	4
12	1	C	3

6. CONCLUSION AND FUTURE WORK

In this paper we have presented a graph based approach to mine the frequent sequential web page access patterns from web server logs. The contribution of the paper is to introduce a new way of web usage mining, and to show how frequent pattern discovery tasks can be accomplished by capturing complex user's browsing behavior in to a graph data structure in order to obtain hidden useful user's access patterns information. In future we can extend this algorithm as an efficient tool for generating web recommendations.

7. REFERENCES

[1] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan. 2000. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. SIGKDD Explorations, Vol. 1, No. 2.

[2] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan. 2007. Frequent pattern mining: current status and future direction. Springer Science+Business Media, LLC.

[3] Agrawal, R. and Srikant, R. Mining sequential patterns. 1995. Int. Conf. Data (ICDE'95), p.3–14.

[4] Renáta Iváncsy, István Vajk. 2006. Frequent Pattern Mining in Web Log Data. Acta Polytechnica Hungarica Vol. 3, No. 1.

[5] Mehdi Heydari, Raed Ali Helal, Khairil Imran Ghauth. 2009. A Graph-Based Web Usage Mining Method Considerind Client Side Data. International Conference on Electrical Engineering and Informatics 5-7 August 2009, Selangor, Malaysia.

[6] P. Deepa1 and Dr. V.Subbiah Bharathi. 2010. A Level-Wise Approach for Mining Frequent Web Usage Patterns. Proceedings of the Int. Conf. on Information Science and Applications.

[7] S.Vijayalakshmi, V.Mohan, S.Suresh Raja. 2010. Mining of User's Access Behaviour for Frequent Sequential Pattern from Web Logs.

[8] Jlawei Han, Jian Pei, Yiwen Yin, Runying Mao. 2001. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach.

[9] Liping Sun and Xiuzhen Zhang. Efficient Frequent Pattern Mining on Web Logs. School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia.

[10] D. Vasumathi, Dr. A. Govardhan. 2005 - 2009. Efficient Web Usage Mining Based on Formal Concept Analysis. Journal of Theoretical and Applied Information Technology.

[11] D. Vasumathi, Dr. A. Govardhan, K.Venkateswara Rao. 2005 - 2009. Performance Improvements and Efficent Approach for Mining Periodic Sequential Access Patterns. International Journal of Computer Science and Security, (IJCSS) Volume (3): Issue (5).