Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.239 – 248

RESEARCH ARTICLE

Efficient Mining Web Navigation Pattern using an Efficient Graph Traverse Algorithm

Rajdeepa. B[1], Dr. Sumathi. P[2]

[1]Research Scholar, Chikkana Government Arts College, Tirupur & Assistant Professor, Dept. of Computer Science, PSG College of Arts & Science, Coimbatore, India
[2]Assistant Professor, PG & Research Department of Computer Science, Government Arts College, Coimbatore, India
[1] rajdeepab@gmail.com; [2] sumathirajes@hotmail.com

Abstract*— *In the modern world navigational behaviour of website visitor is an important role. In this paper implemented the web navigational pattern for college websites. Traditional method of web usage mining approach gives inefficient result for their web navigational pattern. So to overcome this, in this paper proposed an algortihm through-surfing pattern (TSP) from incremental database for college websites. The term TSP are well known to display the next visited Web pages in a browsing session. To record the information about navigation path of college website visitors by introducing path traversal graph and also applied the proposed efficient graph traverse algorithm depend on the proposed number of maximal references of path traversal pattern to discover the TSPS. The experimental results show the proposed mining method of TSP for college websites is provide high efficient and effective precision and recall based on minimum support compared to other approaches.

Keywords*— *Web Navigation Pattern, through-surfing pattern, Path Traversal Graph, Maximal References, Graph Traverse Algorithm, Incrementa Database

I. INTRODUCTION

Nowadays, the World Wide Web has becoming one of the most comprehensive information resources. It probably, if not always, covers the information need for any user. However, the Web demonstrates many radical differences to traditional information containers such as databases, in schema, volume, topic-coherence. Those differences make it challenging to fully use Web information in an effective and efficient manner. Web mining is right for this need [1].

Generally web Mining Data is classified into three types and it is shown in Fig. 1.

- Web content mining,
- Web structure mining
- and Web usage mining

Web usage mining is the process of extracting useful information from server logs i.e. users history. Web usage mining is the process of finding out what users are looking for on Internet.

Some users might be looking at only textual data, whereas some others might be interested in multimedia data. This technology is basically concentrated upon the use of the web technologies which could help for betterment [3].Web content mining is the mining, extraction and integration of useful data, information and knowledge from Web page contents. Content mining is the scanning and mining of text, pictures and graphs of a Web page to determine the relevance of the content to the search query. This scanning is completed after the clustering of web pages through structure mining and provides the results based upon the level of relevance to the suggested query. With the massive amount of information that is available on the World Wide Web, content mining provides the results lists to search engines in order of highest relevance to the keywords in the query. [2]

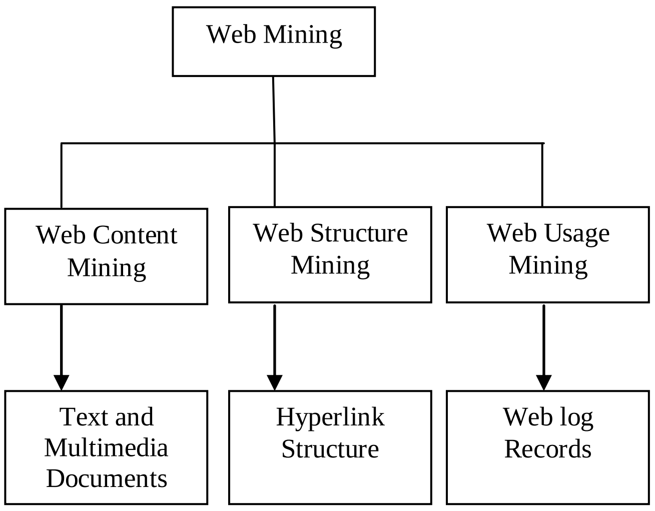

Fig. 1 Overview of Web Mining

Mining Web navigation patterns is useful in practice, and the extracted patterns can be used to predict and understand visitors'browsing behaviour and intentions. It is helpful in improving user experience, website configuration, and the efficiency and effectiveness of e-commerce. Website operators can apply Web navigation patterns to analyze and forecast user motivation, and thus provide better recommendations and personalized services for their customers [4,5].

The main task for the web structure mining is to handle the structure of the hyperlinks within the Web itself. One of the earliest research in this field is Link analysis. Nowadys, with the development of web mining, one of the part of web mining, structure mining research had increasing lot and these efforts had resulted in a newly emerging research area called Link Mining [6], which is located at the intersection of the work in link analysis, hypertext and web mining, relational learning and inductive logic programming, and graph mining. There is a potentially wide range of application areas for this new area of research, including Internet.

II. RELATED WORK

A Web navigation pattern referred to as a Web access pattern (also known as clickstream) is a path through one or more Web pages in a website that is extracted from the access logs of the Web server. A series of Web pages in a website requested by a visitor in a single visit is referred to as a session. The process of discovering patterns from access logs is known as Web usage mining or Web log mining [7].

Number of determination have to be developed towards mining various pattern from web logs are present in [8,9,10].

Lot of enemorous research was go through for the navigation behaviour of website visitors. In this research many of the methods are carried through the web access patterns, for instance successfully and enlightinh their efficiency of web pages, web visitors target groups are identified, web search performance are increased and the access efficiency of Web pages by the adaptive website system, reorganizing a website dynamically and predicting user behaviour patterns in mobile Web systems [11,12]. This type of research task to develop web access pattern and it is mainly foccussed on three main paradigms [13] they are association rules, sequential patterns, and clustering [14, 15].

Both the association rule and sequaential rule was minied by graph based approach and it is implemented in [16] in this the author given that the database is huge means that the algorith is inadequate. In [17] author implemented an incremental mining algorithm, termed DSM-PLW, to find the maximal reference sequences in one database scan.

To store the prevoius mining results are handled in [18] for incremental Web traversal patterns, here the author used the lattice structure for this type of storage. The patterns may be obtained rapidly when the database or the website structure is updated. Again, as stated in their conclusions, the size of the lattice structure may become too large to be loaded into the main memory.

The former mining algorithms suffer from either repetitive database scan or high memory load. For algorithms with a single database scan, they build special data structures to store the sequences in the database. However, it is impracticable to hold all sequences of the database in the data structure. On the contrary, our scheme for mining TSPs is realistic, in which the memory is loaded with the hyperlink structure of the website instead of the sequence database. Nevertheless, the TSPs are not the same as maximal frequent sequences or closed sequential patterns. The graph traverse algorithm proposed in this study is not directly comparable to those of mining sequential patterns, closed sequences, and Web traversal patterns. As we pay attention to the tracks of website visitors, the proposed method provides an efficient and effective way to realize what targets the visitors may reach and how they are achieved.

III. MINING TSP

In this paper propose an well organized mining approach to discover the throughout-surfing patterns. This throughout-surfing patterns are mined by the idea present in [12] that is maximal forward refernces. In this paper intially plan a compact structure called the path traversal graph, it develop to represent the tracks of Web navigation. Then the proposed throughout-surfing patterns are developed by an effective graph traverse algorithm. The proposed method is more successful and effectual than conventional sequential pattern mining algorithms, because it have the advantage of that it scans the database only once instead of generating candidate patterns.

The contributions of this paper are described as follows.

- The concept of TSP is introduced, which are efficient and effective to provide a big picture of the navigation paths for understanding the purposes of website visitors.
- A compact graph is devised to store the information of Web navigation paths. The information of Web browsing and hyperlinks between Web pages are kept in the graph. The edges in the path traversal graph record both incoming and outgoing hyperlinks and the via-links hold ‘‘from-to-via’’ information in the graph that are necessary to predict where a visitor will go at any vertex by the vertex he comes from.
- Then propose a graph traverse algorithm to find TSPs efficiently. A depth-first search (DFS) mechanism is adopted to traverse the path traversal graph.

A. Construction of path traversal graph

When user activities on a website are recorded in Web server logs, the data collected in the log files are further processed to create Web browsing sessions for pattern mining tasks. It is common to generate Web browsing sessions for mining Web navigation patterns.

TABLE I
WEB BROWSING SESSIONS

Session_ID	Web browsing session
So1	$\langle a, c, e, f, i, k \rangle$
So2	$\langle a, b, c, e, f, h, k \rangle$
So3	$\langle a, c, e, g, f, i, k \rangle$
So4	$\langle a, d, e, f, i, j \rangle$

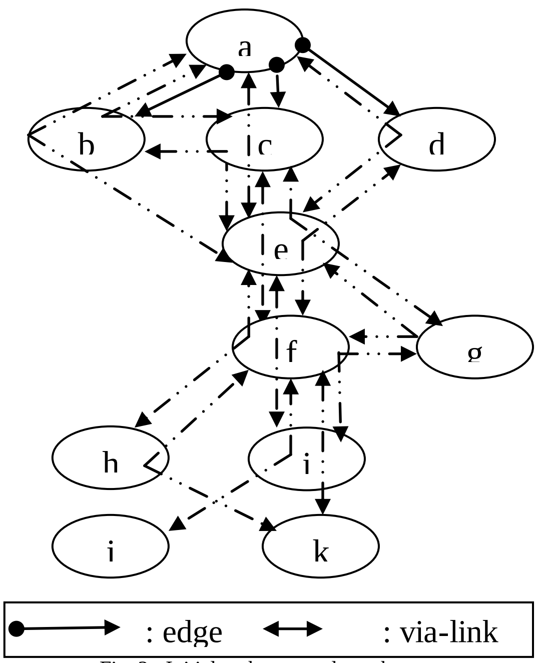

Fig. 2: Initial path traversal graph

In this study, a set of Web browsing sessions based on the maximal forward references [12] is the primary input to the mining method. To avoid scanning databases repeatedly as well as generating a huge amounts of candidate sequences, in this paper we propose a graph traverse approach to discover TSP. First, we devise a graph structure to retain the user navigation information. The information of Web browsing sessions is collected in the proposed path traversal graph. Then, the graph traverse algorithm is performed on the graph to find TSP.

TABLE II
VERTEX FOR PATH TRAVERSAL GRAPH

Vertex	Edge/ Via-link
a	<a,b>,<a,c>, <a,d>
b	<a,b,c>, <a,b,e>
c	<a,c,e>, <b,c,e>
d	<a,d,e>
e	<c,e,f>,<c,e,g>, <d,e,f>
f	<e,f,h>, <e,f,i>, <g,f,i>
g	<e,g,f>
h	<f,h,k>
i	<f,I,j>, <f,I,k>

For efficiency of implementation, the information of edges and via-links are associated with the vertices while constructing the data structure of the path traversal graph. For example, an edge <b, d> associated with vertex b indicates a Web navigation path from vertex b–d, and a via-link $\langle a, b, c\rangle$ associated with vertex b suggests that there is a path from vertex a–c via b.

While the path traversal graph is constructed, each Web browsing session, such as $\langle v_1, v_2, \ldots, v_n\rangle$ is decomposed to sets of edges

$\langle v_1, v_2\rangle, \langle v_2, v_3\rangle, \ldots, \langle v_{n-1}, v_n\rangle$ and via-links $\langle v_1, v_2, v_3\rangle, \langle v_2, v_3, v_4\rangle, \ldots, \langle v_{n-2}, v_{n-1}, v_n\rangle$ and then the edges and via-links are recorded on the path traversal graph. For example, the Web browsing session $\langle a, c, e, f, i, k\rangle$ in Table 1 can be decomposed to five edges $\langle a, c\rangle, \langle c, e\rangle, \langle e, f\rangle, \langle f. i\rangle$ *and* $\langle i, k\rangle$ plus four via-links $\langle a, c, e\rangle, \langle c, e, f\rangle, \langle e, f, i\rangle$ *and* $\langle f, i, k\rangle$. Based on the edge $\langle a, c\rangle$, vertex a and c are created and connected with the edge <a, c> in the path traversal graph. Then, vertex e and edge $\langle a, c\rangle$ are created and vertices a, c, and e are connected with via-link <a, c, e>. The following vertices are created and connected in the similar way.

Fig. 1 depicts the path traversal graph corresponding to the four Web browsing sessions in Table 1 where the notations represents the edges and via-links respectively. For simplicity, the edges of the vertices except vertex a are omitted. Suppose the minimum support is 50%. After all the edges and via-links with supports below the minimum support are removed and those vertices unconnected by any edge or via-link are deleted, the remainder is the frequent path traversal graph.

B. Graph traverse algorithm for mining throughout-surfing patterns

Present and analyze the graph traverse algorithm for mining throughout-surfing patterns. The algorithm discovers all throughout-surfing patterns by selecting suitable starting edges and traversing frequent path traversal graph in DFS order

In stage one, a vertex in the frequent path traversal graph is selected in step (g1) as the candidate for the starting vertex of a TSP. If the selected vertex has any edge that is not contained in any vialinks, that is, it is the starting vertex of some acyclic or partially cyclic TSPs, the mining process is proceeding to trace the edge and the follow-up via-links by calling the function trace() in step (g8).

The function trace() adopts a DFS approach to traverse the frequent path traversal graph. It uses two stacks for non-recursive operations. The UntracedStack is used to store the unselected vertices when trace() traces some throughout-surfing patterns with the same prefix. The other stack, BacktrackStack, is used to store the index values of vertices in throughout-surfing patterns that indicate the number of vertices backtracked in the depth-first search. In step (t1), trace() pushes the second vertex of the starting edge on UntracedStack and then pop it for further processing in step (t5).

After attaching the popped vertex w to eTSP in step (t6), the vialinks of w are checked if they are the follow-up paths of eTSP in steps (t9) and (t10). If there are k via-links $\langle x, w, y_j \rangle, 1 \le j \le k$. where x is the preceding vertex of w in eTSP, yj are pushed on UntracedStack for further processing. In addition, if $k \ge 2,$ the index value i of w in eTSP is pushed on BacktrackStack (k - 1) times in steps (t16)–(t19). It means that w is a diverging vertex in eTSP and there are (k - 1) throughout-surfing patterns with the same prefix of length i.

Algorithm: Graph traverse
Input: A frequent path traversal graph G
Output: All throughout- surfing patterns

```
//stage one: mining acyclic or partially cyclic throughout-surfing patterns
(g1) while (v=G.getunselectedvertex ()) {// for each vertex v in G
(g2) while (e=G.getEdge (v)) {//e=<v, u>
(g3) if (G.unTraced (e) &&! G.lastComponentVL (e) {// e is untraced and not contained in any via-link of v
(g4) G.markTraced (e); // mark e traced
(g5) TSP.intialized ();
(g6) TSP.push_back (e.front ()); // append v to TSP
(g7) G.markselected (v); //mark v selected
(g8) trace (e, TSP); //call trace ()
(g9)    }
(g10)   }
(g11)   }
// Stage two: mining fully cyclic throughout-surfing patterns
(g12) while (v=G.getvertex ()) {// for each vertex v in G
(g13) G.markUnselected (v); //mark v unselected
(g14)   }
(g15)
(g16) G.markTraced (l); // mark l untraced
(g17)   }
(g18) while (v=G.getunselectedvertex ()) {// for each vertex v in G
(g19) while (l=G.getViaLink (v)) {//l=<p, v, q>
(g20) if (G.unTraced (l)) {// l has not been traced in stage one
(g21) TSP.intialized ();
(g22) TSP.push_back (l.middle ()); // append v to TSP
(g23) G.markselected (v); // mark v selected
(g24)  e=l.getBackEdge (); //e=<v, q>
(g25) trace (e, TSP); //call trace ()
(g26)    }
(g27)}
(g28)}
```

Fig. 3 Graph Traverse algorithm

In step (t20), there are no added via-links by which eTSP can be extended, and therefore eTSP is ended at w. eTSP is outputted into the output buffer outPattern in step (t21). Then, in steps (t22) and (t23), an index value i in BacktrackStack is popped for backtracking eTSP if the BacktrackStack is not empty and the prefix of eTSP with length i is reused as a new TSP. Besides, the successive via-links and vertices of w along eTSP are marked untraced and unselected respectively in steps (t24)–(t28). The new eTSP can be extended by attaching a vertex popped from UntracedStack in the following iteration of trace().

When trace() returns, other TSPs starting at v are traced in the while loop of steps (g2)–(g10), and then unselected vertices are picked up for further processing in step (g1). After stage one, all acyclic or partially cyclic TSPs are found and the remainders are fully cyclic TSPs. A fully cyclic TSP can be traced from any via-link in the TSP. Steps (g18)–(g28) in stage two find all fully cyclic TSPs as similar to those steps in stage one except the starting vertices are selected from any via-link which has not been traced.

```
Void trace (Edge strartE, vector<char>eTSP) //startE=<v, u>
(t1) UntracedStack.push (startE. Back ()); //push u onto UntracedStack
(t2) x=startE. Front (); //x=v
(t3) while (! UntracedStack.empty ()) {//while UntracedStack is not empty
(t4) countviaLink=0;
(t5) w=UntracedStack.pop ();
(t6) eTSP.Push_back (w); //append w to eTSP
(t7) G.markselected (w); //mark w selected
(t8) e=new Edge(x, w); //e=<x, w>
(t9) while (l=G.getViaLink (w)) {//l=<x, w, y>, all untraced via-links with the same starting edge <x, w>
(t10) if ((l.getFrontEdge () ==e) && (! eTSP. Exist (e))) {// the starting edge of l is <x, w> and <x, w> is
not contained in eTSP
(t11) countviaLink++; //the number of via-links with the starting edge e
(t12) UntracedStack.push (l.back ()); //push y on UntracedStack
(t13) G.markTraced (l); //mark l traced
(t14)       }
(t15)   }
(t16)  if (countviaLink>=2) {// there are splitting nodes in eTSP
(t17) for (i=0; i<countviaLink-1; i++
(t18)   BacktrackStack.push (eTSP. Index (w)); // push the index of w in eTSP on BacktrackStack
(t19)   }
(t20) else if (countviaLink==0) {//eTSP cannot extend anymore
(t21)  out Pattern<<eTSP; //output eTSP
(t22) if (! Backtrack Stack. Empty ()) {// there exists another TSP with the same prefix o to eTSP
(t23) index=BacktrackStack.pop (); //the last splitting node
(t24) for (iter=eTSP. Begin () +index; iter<eTSP. End ()-2; iter++) {
(t25) G.unmarkSelected (*iter)
(t26) G.unmarkTraced (ViaLink (*iter, (*iter+1),*(iter+2)));
(t27)}//unmark the vertices and via-links I the suffix of eTSP
(t28) G.unmarkSelected (*iter, 2);
(t29) eTSP. Erase (index+1, eTSP.size ()-index-1); //retain the prefix
(t30) w=eTSP. Back ();
(t31)}
(t32)}
(t33) x=w;
(34)}
```

Fig. 4 trace () function

C. Proposed Efficient graph traverse algorithm

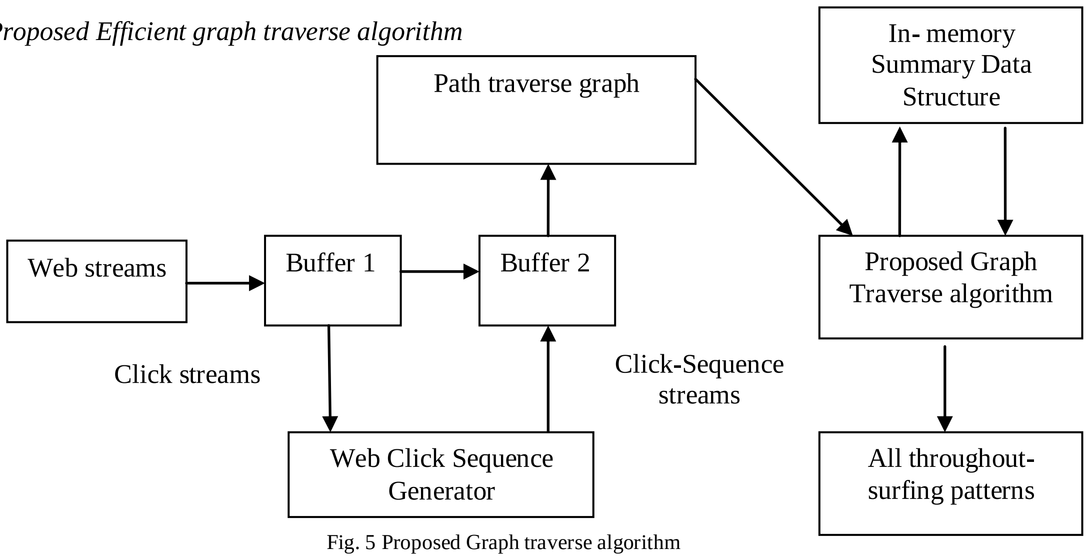
Path traverse graph
In- memory Summary Data Structure
Web streams
Buffer 1
Buffer 2
Proposed Graph Traverse algorithm
Click streams
Click-Sequence streams
Web Click Sequence Generator
All throughout-surfing patterns

Fig. 5 Proposed Graph traverse algorithm

Proposed graph traverse algorithm *consists of a* of a list of frequent references, such as$fr_1, fr_2, \ldots, fr_k$, where $fr_i.Nmfr \geq S.N$ and a set of Path traversal pattern tree of references fr_i, denoted by$fr_i.path - tree$,$\forall i = 1,2,\ldots,k$.

Each node in the$fr_i.path - tree, \forall i = 1,2,\ldots,k$ consists of four fields ifr_id, Nmfr, mfr_id and node-link. Where ifr_id is the identifier of the incoming forward reference, Nmfr registers the number of maximal forward references represented by a portion of the path reaching the node with the ifr_id, the value of cmfr_id assigned to a new node is the identifier of current maximal forward reference, and node-link links up a node with the next node with the same ifr_id in the proposed graph traverse algorithm.

Each entry $r_i, \forall i = 1,2,\ldots,k$ in the proposed graph traverse algorithm list consists of four fields: fr_id, Nmfr, cmfr_id, and head-link, where fr_id registers which forward reference identifier the entry represents, Nmfr records the number of maximal forward references in the stream so far containing the reference with identifier fr_id, cmfr_id assigned to a new entry is the identifier of the current maximal forward reference, and head-link is a pointer, and points to the root node of the fr_id.Path-tree.

```
Algorithm: proposed efficient graph traverse algorithm
Input: A stream of maximal forward references, $MFR_i, MFR_2, \ldots, MFR_N$ and a user-
defined minimum support threshold $s \in (0,1)$
Output: All throughout- surfing patterns.
1.  fr-list = {}; /* initialize the fr-list to empty*/
2. for each $MFR_i = < fr_1, fr_2, \ldots, fr_k >$ do /* $\forall i = 1,2,\ldots,N$ where N is the
identifier of current MFR*/
3. for each reference $fr_j \in MFr_i$ do / * $\forall j = 1,2,\ldots,k$ */
4. if $fr_j \nexists fr - list$
5.            create a new entry of form $(fr_j, 1, i, \rightarrow fr_j)$ into the fr-list;
6. else
.  $fr_j.Nmfr = fr_j.Nmfr + 1$;
    end if
9. MFR-projected $(MFR_i, = < fr_1, fr_2, \ldots, fr_j, \ldots, fr_m >)$
    for each reference $fr_j, \forall j = 1,2,\ldots, m$, in $MFR_i$ do
            Path tree maintenance$(fr_j | MFr_i, fr_j.path - tree, i)$;
    end
10 end for
11. end for
```

Fig. 5 Proposed Efficient Graph traverse algorithm

IV. Experimental Result

In this paper implemented the mining method of Web navigational pattern for the college websites. Incremental database are used here to perfrom the experimental analysis for college websites. The algorithm developed in this paper were implemented in MATLAB. Then conducted the experiments on a PC running Microsoft Windows 2000 Professional with an Intel/ 1.5GHz Pentium IV processor, 512MB of main memory, and 80GB of hard disk.

A. *Performance analysis*

In this paper perform a experiment to compare proposed TSP method with the modified Apriori and Prefixspan algorithm, these exisitng algorithm was implemented by (Lee and Yen 2007).

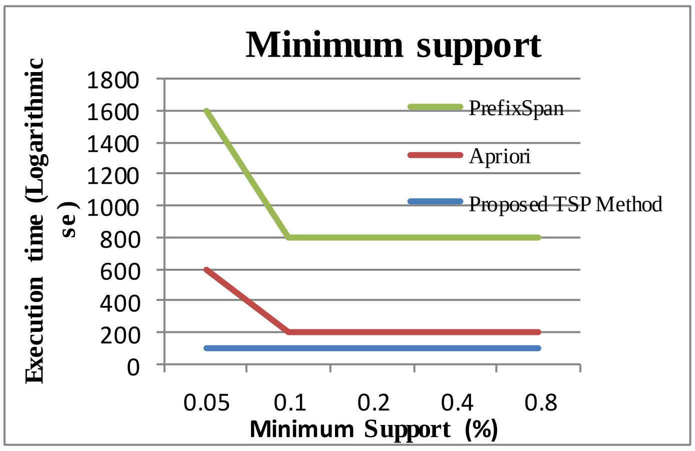

Fig. 6 Execution time for different settings of minimum support thresholds

Fig. 6 shows the result for different settings of minimum support thresholds. While the minimum support becomes lower, the number of frequent patterns increases and the frequent patterns get longer. From the above figure it is clearly observed that the execution time for proposed mining method for college websites remains almost at the same level because it discovers the TSP by traversing the path traversal graph and almost all the run time is spent on constructing the path traversal graph.

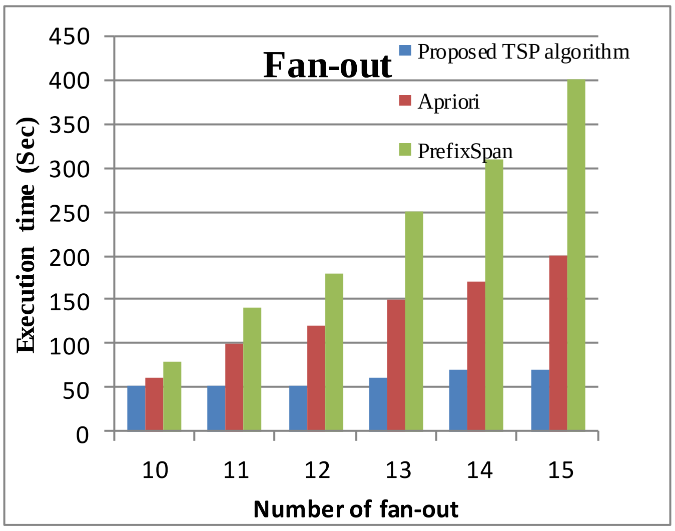

Fig. 7 Number of fan-out

That the number of fan-outs are varied from ten to fifteeen and it is shown in Fig. 7. The number of via-links associated with a vertex is proportional to the number of fan-outs. Therefore, the execution time grows with a larger number of fan-outs. From the above figure proposed method proves better result in exection time based on fan-out compared to other approaches.

TABLE IIII

PRECISION AND RECALL

Minimum Support (%)	Precision	Recall
0.02	0.654	0.424
0.10	0.789	0.729
0.14	0.930	0.937

Table III gives the precision and recall for proposed web navigation pattern for college websites. From the table,find out that low degree of precision and recalls are obtained for low minimum support and high degree of precision and recall are obtained for high minimum support.

V. CONCLUSION

In this paper investigate the problem of mining web navigation pattern for college website. Proposed mining web navigation pattern gives best result compared to other approaches. Here introduced the mining web navigation pattern from incremental database. To prove the effectiveness and efficiency of web navigation pattern for college web sites by introducing throughout- surfing patterns. To improve the efficieny of mining throughout-surfing petterns by efficient graph traverse algorithm. In addition, a path traversal graph structure is suitable for incremental mining of sequential patterns. The compact graph structurere tained in the main memory may be output to permanent storage.While mining patterns from the database with new added data, the path traversal graph is restored in the main memory and the new data is retrieved and appended to the graph. From the experimnetal result it is clearly observed that proposed web navigational pattern gives best result compared to other approaches.

REFERENCES

[1] Etzioni, O. “The World Wide Web: Quagmire or gold mine”, Communications of the ACM, 39(11):65-68, 1996.

[2] Raymond Kosala, Hendrik Blockeel, Web Mining Research: A Survey, ACM SIGKDD Explorations Newsletter, June 2000, Volume 2 Issue 1.

[3] Masand, B., Spiliopoulou, M., Srivastava, J and Zaiane, O. (2002) Proceedings of “WebKDD2002 –Web Mining for Usage Patterns and User Profiles”, Edmonton, CA, 2002.

[4] Arotariteia, D., Mitra, S. (2004). Web mining: A survey in the fuzzy framework. Fuzzy Sets and Systems, 148, 5–19.

[5] Pierrakos, D., Paliouras, GO., Papatheodorou, C., Spyropoulos, CD. (2003). Web usage mining as a tool for personalization: A survey. User Modeling and User-Adapted Interaction, 13, 311–372.

[6] Getoor, L (2003) “Link Mining: A New Data Mining Challenge”, SIGKDD Explorations, vol. 4, issue 2, 2003.

[7] Pei, J., Han, J., Mortazavi-Asl, B., & Zhu, H. (2000). Mining access patterns efficiently from Web logs. In Proceedings of the 4th Pacific-Asia conference on knowledge discovery and data mining, Kyoto, Japan (pp. 396_407).

[8] Cooley. R., Mobasher, B and Srivastava, J. (1999). “Data preparation for mining world wide web browsing patterns, in journal of knowledge and information systems, Vol.1, No.1.

[9] Spiliopoulou, M and Faulstich. L. (1998)”WUM: A Tool for Web Utilization Analysis”, the world web wide Databases, Lecture notes in computer science, Vol, e 1590, pp. 184-203.

[10] Zaiane, O., Xin, M and Jiawei, H. (1998). "Discovering Web access patterns and trends by applying OLAP and data mining technology on web logs", In proc. Advances in digital Libraries Conf, Melbourne, pp.144-158.

[11] Arayaa, S., Silvab, M and Weber, R. (2004). A methodology for Web usage mining and its application to target group identification. Fuzzy Sets and Systems, 148, 139–152.

[12] Chen, MS., Park, JS and Yu, PS. (1998). Efficient data mining for path traversal patterns. IEEE Transactions on Knowledge and Data Engineering, 10(2), 209–221.

[13] Facca, FM and Lanzi, PL. (2005). Mining interesting knowledge from Weblogs: A survey. Data and Knowledge Engineering, 53, 225–241.

[14] Abraham, A and Ramos, V. (2003). Web usage mining using artificial ant colony clustering and linear genetic programming. In IEEE congress on evolutionary computation, CEC2003 (pp. 1384_1391). Australia: IEEE Press.

[15] Ezeife, CI and Lu, Y. (2005). Mining Web log sequential patterns with position coded pre-order linked WAP-tree. Data Mining and Knowledge Discovery, 10, 5–38.

[16] Yen, SJ and Chen, ALP. (2001). "A graph-based approach for discovering various types of association rules", IEEE Transactions on Knowledge and Data Engineering, 13(5), 839–845.

[17] Li, HF., Lee, SY and Shan, MK. (2006). "DSM-PLW: Single-pass mining of path traversal patterns over streaming Web click-sequences", Computer Networks, 50, 1474–1487.

[18] Lee, YS and Yen, SJ. (2007). "Incremental and interactive mining of Web traversal patterns", Information Sciences, 178(2), 278–306.