Special Issue Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, www.ijtrd.com

Frequent Path Analysis of Organizational Website Using AFPT

Dr. C.Gomathi M.C.A.,M.Phil.,Ph.D.,
Professor, Department of MCA., Vidyaa Vikas College of Engineering and Technology, Tiruchengode, Namakkal Dist., Tamil Nadu, India

Abstract: Frequent path analysis discovers frequent path from web logs. An important application of Data Mining is Web usage mining for finding frequent path from web log accesses. Apriori like pattern mining technique requires expensive multiple scans of database. So now a days, Bpriori based algorithm is used. It is faster than traditional pattern mining techniques. In this, Kongu Arts and Science College (KASC) and KSR College of Engineering (KSRCE) web log files are taken after preprocessing for frequent path analysis. Here, an efficient **Advance Frequent Path Technique (AFPT**) has been proposed. This proposed algorithm modify the traditional approach for improving efficiency, especially when the support threshold becomes smaller and size of the database gets larger. The proposed algorithm totally eliminates the storage cost during mining and considerably reduces the execution time. The results of experiments show the efficiency of the improved algorithm.

The proposed algorithm is used to improve the design of web sites, analyzing the student's behaviors and developing web sites according to different usage scenarios.

I. INTRODUCTION

Apriori is an influential algorithm for mining frequent item sets for boolean association rules. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database**.**

II. ALGORITHM APRIORI

The following Figure1 shows the Apriori algorithm. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass k, consists of two phases. First, the large itemsets L_{k-1} found in the (k-1)th pass are used to generate the candidate itemsets C_k, using the apriori-gen function.

1) L1 = {large 1-itemsets ;}
2) for (k = 2; $L_{k-1} \neq 0$; k++) do begin
3) C_k = apriori-gen(L_{k-1}); // New candidates
4) forall transactions t € D do begin
5) C_t = subset (C_k, t); // Candidates contained in t
6) forall candidates c € C_t do
7) c:count++;
8) end
9) L_k = {c € C_k | c:count ≥ minsup;
10) end
11) Answer = $\bigcup_k L_k$;

Figure 1: Algorithm Apriori

Table 1: Notation

K - itemset	An itemset having K items
L_k	set of large K-itemsets (those with minimum support) Each member of this set has two fields : i) itemset and ii) support count
C_k	Set of candidate k-itemsets (portentially large itemsets) Each member of this set has two fields: i) itemset and ii) support count

III. PROPOSED ALGORITHM – ADVANCED FREQUENT PATH TECHNIQUE (AFPT)

The proposed AFPT algorithm takes the input from the preprocessed log file. The log file transactions consist of number of strings. To reduce the computation, the transactions are divided into two halves, namely left and right consecutive string using a center string. From left and right consecutive string, the path which occur the maximum number of times is declared as the frequent path.

Input : Preprocessed Log File (L)
Output : Frequent Path (FP)
Process :
Step 1: Read a Transaction (T) from Log File (L).
Step 2: For each Transaction (T), find center string CS[i], which has .html extension.
Step 3: Find the number of occurrences of each center string count (CS[i]).
Step 4: Find the maximum count (max) for all center string count.CS[i]..
Step 5: Scan Log File to find Frequent Path (FP) such as
i. Scan the all predecessor page of center string page.

The result produced by the algorithm is given below:

Input:
Sample transactions = 12

203.101.67.192 - - [27/Aug/2007:11:05:51 +0530] "GET /main.html HTTP/1.0" 200 6640 "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

210.211.182.11 - - [27/Aug/2007:11:05:50 +0530] "GET /recent.html HTTP/1.1" 200 39189 "http://www.kasc.ac.in/college.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; digit_may2002)"

203.101.67.192 - - [27/Aug/2007:11:05:58 +0530] "GET /cspgdept/cspg.html HTTP/1.0" 200 232 "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

203.101.67.192 - - [27/Aug/2007:11:05:58 +0530] "GET /cspgdept/cspg.html HTTP/1.0" 200 2038 "http://www.kasc.ac.in/cspgdept/cspg.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

203.101.67.192 - - [27/Aug/2007:11:05:58 +0530] "GET /cspgdept/cspg.html HTTP/1.0" 200 36026 "http://www.kasc.ac.in/cspgdept/cspg.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

203.101.67.192 - - [27/Aug/2007:11:09:07 +0530] "GET /departments.html HTTP/1.0" 200 4000 "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

210.211.182.11 - - [27/Aug/2007:11:10:30 +0530] "GET /college.html HTTP/1.1" 304 - "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; digit_may2002)"

210.211.182.11 - - [27/Aug/2007:11:10:30 +0530] "GET /main.html HTTP/1.1" 304 - "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; digit_may2002)"

210.211.182.11 - - [27/Aug/2007:11:10:37 +0530] "GET /cspgdept/cspg.html HTTP/1.1" 200 232 "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; digit_may2002)"

210.211.182.11 - - [27/Aug/2007:11:10:38 +0530] "GET /cspgdept/cspg.html HTTP/1.1" 200 2038 "http://www.kasc.ac.in/cspgdept/cspg.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; digit_may2002)"

203.101.67.192 - - [27/Aug/2007:11:15:49 +0530] "GET /forthcoming.html HTTP/1.0" 304 - "http://www.kasc.ac.in/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

203.101.67.192 - - [27/Aug/2007:11:19:27 +0530] "GET /mgtdept/management.html HTTP/1.0" 200 41022 "http://www.kasc.ac.in/departments.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"

Output:

No. of occurrences of center string cspg.html = 5
Left and Right consecutive string of center string cspg.html is
www.kasc.ac.in -> cspgdept -> cspg.html.

This is the frequent path for the above sample transactions.

The success of the Apriori algorithm and the Frequent Path tree (FP-tree) method has inspired to consider the possibility and appropriateness of applying their basic ideas and techniques to solve the Generalized Center String problem. Note that the two kinds of problems are different. While the problem of mining association rules requires the detection of frequent itemsets, the Generalized Center String problem asks for mining center strings by finding all of their mutated copies limited by a distance $d \geq 0$. In this chapter, the mutated copies of a center string are also called consensus strings, meaning that occur "quite often" in the input sequences. Any sequence containing a consensus string is called an origin of the string. Based on the observation that a center string defined in GCS also has downward closure property, and then it takes all substrings of the input sequences as seeds and use them iteratively to find longer and longer consensus strings by a level-wise search strategy. At each level of search, the center strings are obtained by enumerating all consensus strings. Using a level-wise search strategy gives a new exact and efficient algorithm named **Advance Frequent Path Technique (AFPT).**

IV. EXPERIMENTAL RESULTS

The proposed algorithm is implemented in VB.NET and all experiments were found on Intel Pentium running on Microsoft Window XP profession. The web server log files from KASC and KSRCE web server after preprocessing have been taken for the experiments. This KASC and KSRCE log file sizes are 5,498 and 6,200 records respectively. The proposed method had applied on this preprocessed web log files to prepare frequent pattern. The proposed mining algorithm is efficient than the existing, by considering each and every log transaction as a seed and identifies the center string. This eliminates the storage cost during mining and considerably reduces the execution time.

Figure 2 shows the frequent path visited from the 5,498 records after preprocessed. The datasets and algorithms are tested with minimum support 5% against the different size database 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500 and 5000 records. Sample screens are shown from Figures 2 to 7.

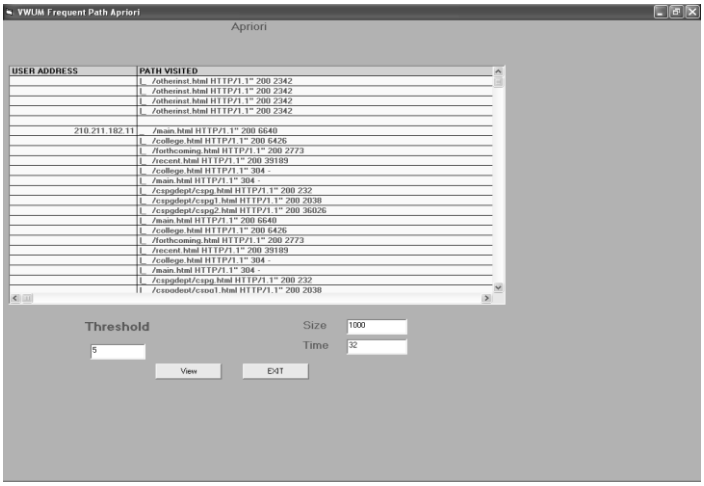

Figure 2 Frequent Path Analysis using Apriori with 1000 records for KASC

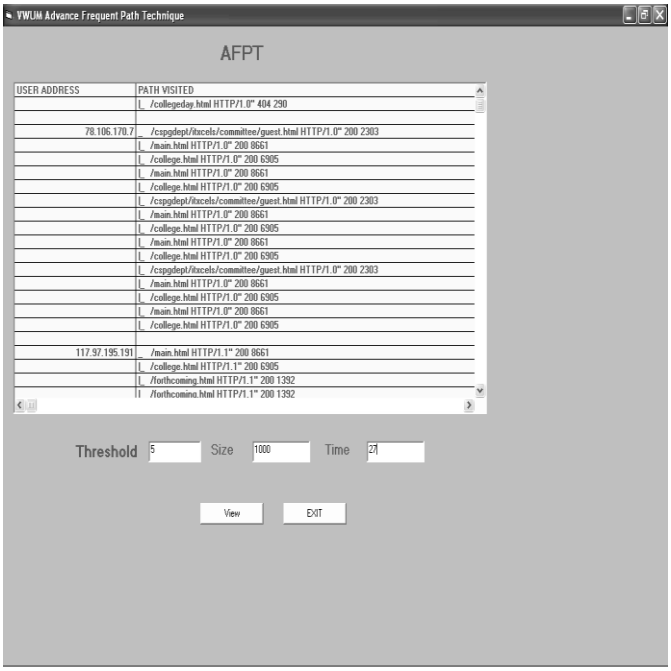

Figure 3 Frequent Path Analysis using AFPT with 1000 records for KASC

In Figure 2, 1000 records with minimum threshold 5% are taken to find frequent path. The Apriori algorithm takes 32 seconds as shown in Figure 2, where as the proposed AFPT algorithm takes 27 seconds as shown in Figure 3, to perform the execution. The percentage of AFPT over Apriori based on execution time is 84.4%.

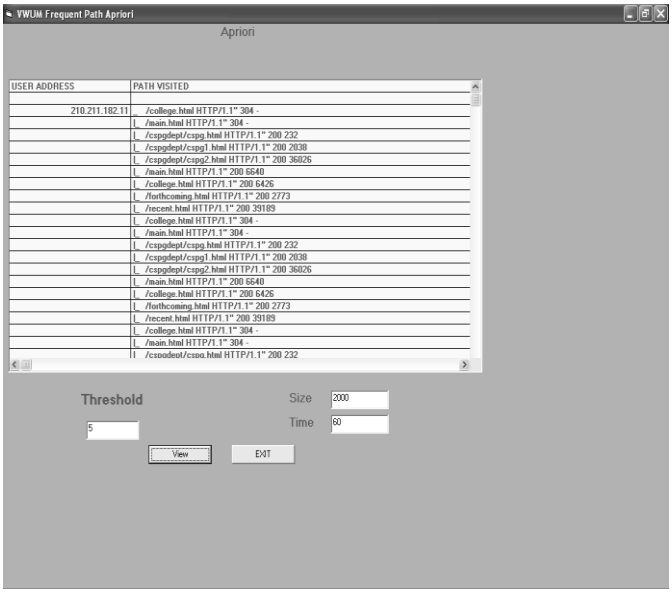

Figure 4 Frequent Path Analysis using Apriori with 2000 records for KASC

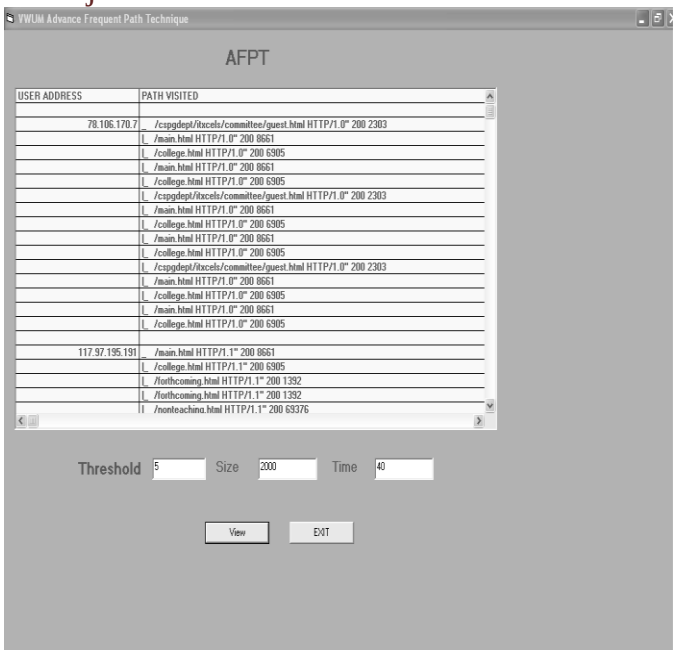

Figure 5 Frequent Path Analysis using AFPT with 2000 records for KASC

In Figure 4, 2000 records are considered to find frequent path. The novel system is executed with 2000 records and minimum threshold 5%. The Apriori algorithm takes 60 seconds as shown in Figure 4, where as the proposed AFPT algorithm takes 40 seconds as shown in Figure 5, to perform the execution. The percentage of AFPT over Apriori based on execution time is 71.2%.

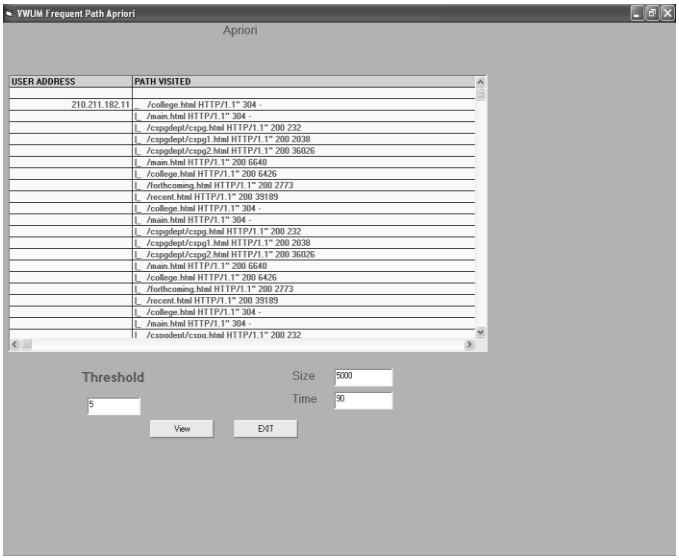

Figure 6 Frequent Path Analysis using Apriori with 5000 records for KASC

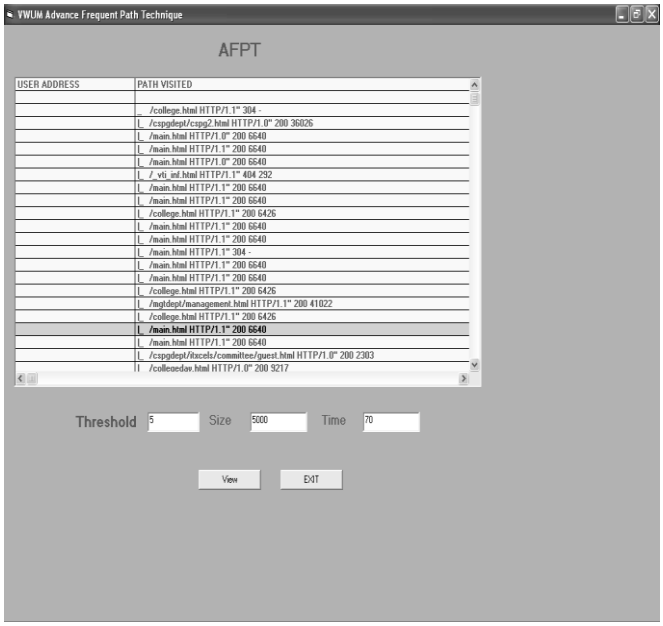

Figure 7 Frequent Path Analysis using AFPT with 5000 records for KASC

In Figure 6, 5000 records are considered to find frequent path. The Apriori algorithm takes 90 seconds as shown in Figure 6, where as the proposed AFPT algorithm takes 70 seconds as shown in Figure 7, to perform the execution. The percentage of AFPT over Apriori based on execution time is 77.8%.

Table 2 shows the execution time of AFPT and Apriori algorithm for KASC webserver. From Table 2 and Figure 8, it can be seen that the execution time increase as size of the database increase. The decrease in the size of the data reduces the execution time. In average, the percentage of AFPT over Apriori based on execution time is 81.71%.

Table: 2 Execution time comparison with fixed minimum support 5% for KASC

S.No.	Records	Apriori (time in Seconds)	AFPT (time in Seconds)	Percentage of AFPT over Apriori based on execution time (%)
1	500	18	16	88.9
2	1000	32	27	84.4
3	1500	48	36	75
4	2000	60	40	66.7
5	2500	68	59	86.8
6	3000	73	62	84
7	3500	78	64	82
8	4000	80	68	85
9	4500	84	69	82
10	5000	90	70	77.8
	Average			81.71

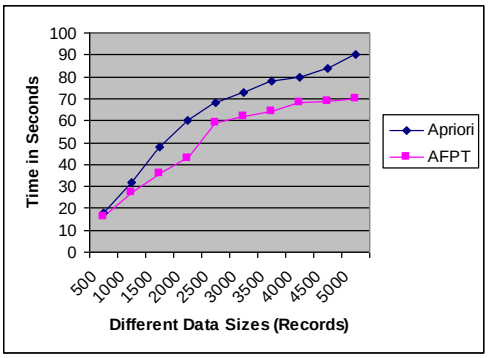

Figure 8 Execution time vs. different data sizes for KASC

Table 3 shows the execution time of AFPT and Apriori for KSRCE webserver. The percentage of execution time of AFPT over Apriori is 81.45%. Figure 9 shows the graphical representation of the execution time for different data sizes of KSRCE web server.

Table 3 Execution time comparison with fixed minimum support 5% for KSRCE

S. No.	Records	Apriori (time in Seconds)	AFPT (time in Seconds)	Percentage of AFPT over Apriori based on execution time (%)
1	500	20	17	85
2	1000	30	26	86
3	1500	61	40	65.6
4	2000	70	62	88.6
5	2500	76	64	84
6	3000	78	60	77
7	3500	80	67	83.6
8	4000	86	71	82.6
9	4500	91	75	82
10	5000	98	79	80.1
	Average			81.45

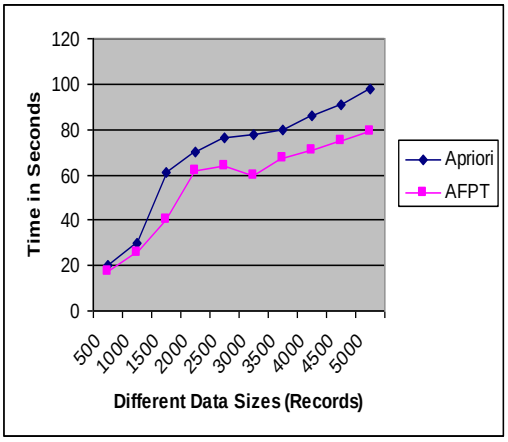

Figure.9 Execution time vs. different data sizes for KSRCE

For KASC and KSRCE webserver, the execution time of AFPT is less than the Apriori Algorithm.

CONCLUSION

In this chapter, AFPT mine tool has been developed using VB.NET to find the frequent path for KASC and KSRCE web log files. The proposed system analyzes the student's

behavior. Using this analyzed data set the web usage behavior of the students can be visualized. From this visualization, the new knowledge is extracted to improve the organization and the web site. The proposed system helps the organization in analyzing the student's behavior and also the proposed system reduces the memory space and execution time to 81.71% and 81.45% as shown in Table 2 and Table 3 respectively.

References

[1] Agrawal R., Imielinski T., Swami A: "Mining Association Rules between Sets of Items in Large Databases", *Proc. ACM SIGMOD '93 Int. Conf on Management of Data*, Washington D.C., 1993, pp. 207-216.

[2] R. Agrawal, and R. Srikant(1994) Fast Algorithms for Mining Association. *In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Santiago, Chile, pp. 487-499.

[3] Gomathi.C., Moorthi M., Duraiswamy K. (2008), Preprocessing of Web Log Files in Web Usage Mining. *The ICFAI journal of Information Technology*, Vol. 4, No. 1, pp. 55-66.

[4] Han,J., Kamber, M. (2001). Data Mining: Concepts and Techniques. Morgan-Kaufmann Academic Press, San Francisco.

[5] R. Kosala, and H. Blockeel(2000) Web Mining Research: A Survey. *In ACM SIGKDD Explorations*, Vol.2, pp. 1-15.

[6] Richards A.J.: "*Remore Sensing Digital Image Analysis. An Introduction*", Springer Verlag, 1983.

[7] J.Srivastava, R.Cooley, M. Deshpande, and P.-N. Tan (2000) Web Usage Mining:Discover and Applications of Usage Patterns from Web Data. *In ACM SIGKDD Explorations*, Vol. 1, No. 2, pp.12-23.