Cluster Comput (2016) 19:269–277
DOI 10.1007/s10586-015-0507-z

Frequent pagesets from web log by enhanced weighted association rule mining

S. P. Malarvizhi[1] · B. Sathiyabhama[2]

Received: 12 March 2015 / Revised: 30 October 2015 / Accepted: 31 October 2015 / Published online: 21 November 2015

Abstract Mining frequently visited web pages from web logs have become an imminent need for web usage mining to understand the behavior of users. Frequent pageset mining and association rule mining (ARM) algorithms existing in the literatures suffer from storage and run time issues. It is because these algorithms mine all of the frequent pagesets based on minimum support threshold and all possible association rules based on minimum confidence threshold. Hence for analyzing the usage level of the web, a more quality oriented and useful mining can be performed by means of weighted ARM (WARM) on web logs. WARM in fact reduces the storage and run time, as it mines the frequent pages based on weighted support and association rules based on weighted confidence. Proposed T+weight tree algorithm gives importance to the dwelling time of the pages visited by the users. Pages are assigned with weights based on dwelling time which shows that these pages may have some significance and attracted the users' interest. T+weight tree algorithm finds frequent pagesets based on weights in a single scan of the database. Empirical results show that, proposed T+weight tree method takes lesser computational time than the other methods in the literature because it produces lesser number of more significant pagesets.

Keywords Frequent pattern mining · Weight estimation · Weighted minimum support · WARM · T+weight tree · Web logs

✉ S. P. Malarvizhi
malarvizhisp@gmail.com

[1] Anna University, Chennai, Tamil Nadu, India

[2] Sona College of Technology, Salem, India

1 Introduction

With the ever-growing use of computing techniques for different applications, the consequences of data mining have also got a rapid swift recently. A number of data mining capabilities have been discovered in the literature. One of the most common issues in the research of data mining is association rule mining (ARM) [1].

Pattern extraction from larger transactional databases uses ARM algorithms. The goal of these algorithms is to fetch inherent relationships among the items. It brings out all the association rules from the database, satisfying a predefined minimum support and confidence threshold. This problem has got many applications in marketing, decision analysis and business management [2].

The hub of this problem is in the mining of frequent itemsets. Frequent itemset mining determines those items which appears together commonly in transactional databases. This corresponds to a large number of customers. If this number is above a certain user defined threshold, then this itemset is considered frequent. Algorithms like Apriori and FP-growth have been used for mining frequent itemsets. Based on these basic algorithms, many algorithms have evolved in research literature [2].

Conventional ARM model is modified to hold WARM problems where weight is assigned to every individual item [3]. By means of weight assignment, target itemsets are given a priority for selection according to their importance, rather than their occurrence rate.

Web documents consist of huge information and hence attract large numbers of researchers to focus on the application of web mining techniques. Web content mining, web structure mining and web usage mining are the three types of web mining used for knowledge and information discovery. The first two types utilize the real or primary data on the web and are used for analyzing the web page contents. Web usage mining mines the secondary data which is a textual log data stored in the web servers, derived from the interactions of the users with the web. It generates data based on the browsing patterns or behaviors of the users [4].

Web usage mining also recognized as web log mining has got four stages in it. They are data collection, Data preprocessing, pattern discovery and pattern analysis [5]. Web log mining is a potential tool for discovering knowledge about the behavioral patterns of the users and website usage information that can be used for different website design tasks. The web content management and link connectivity are improved using the user behavioral patterns for enhanced services of websites.

This paper provides several data preprocessing techniques and algorithms that can be used to convert raw web server logs into user session files in order to carry out web usage mining.

Rest of the paper is structured as follows. Section 2 reviews related works of Frequent Patterns and WARM for Web log. Section 3 details the proposed system of how to preprocess the web server logs, weight estimation techniques, T+weight tree structure and WARM algorithm based on weights. Section 4 carries experimental evaluation. Section 5 provides conclusion.

2 Related work

FP-growth algorithm is used for acquiring the frequent access patterns from the web log data and to provide valuable information about the users [6].

An efficient mining methodology for WARM is proposed by Wang et al. [7]. WARM assigns numerical attribute for every item. This judges the weight of the item in a particular weight domain. The issues of discovering significant binary relationships in transaction datasets in a weighted setting is addressed by Tao et al. [3] where each item is allowed to have a weight. This uses the idea of WARM [7] which is both scalable and efficient in discovering significant relationships in weighted settings.

A new algorithm called combined frequent pattern mining (CFPM) is proposed by Sun and Zhang [8] to cater for web log data specifically.

Recently used frequent web pages may be kept in a server's cache to speed up web access [9]. WARM technique finds pages of the user's current interest and caches them to give faster internet access. This approach confines both user's habit and interest as compared to other approaches where emphasis is only on habit.

To get the useful information, web log files are examined which helps to improve the services offered by web portals and information access and retrieval tools, giving information on problems arising to the users. Mining frequent patterns from web log data can help to optimize the structure of a web site and improve the performance of web servers. Web users are benefitted from these frequent patterns. Frequent pattern mining techniques are discussed for discovering different types of patterns in a web log [10].

The conventional models in existing literature do not take into account the difference between the transactions, and the WARM does not work on databases with only binary attributes. A new measure using link-based models called w-support takes the quality of transactions into consideration rather preassigned weights [11].

An enhanced algorithm for mining the frequently visited groups of web pages is proposed by Yiling Yang et. al. with the consideration of both the site topology and the content of web pages [12]. Here calculation of support includes another two arguments, the content-link ratio of the web page and the inter-linked degree of the page group. Hence, frequently visited web pages with more user interestingness are delivered.

ARM model assumes that all items have the same significance without considering their weights into account. But, WARM requires each item to be given a weight value to reflect their importance to the users. The weights may correspond to special promotions on some products, or the profitability of different items [13].

Proposed system mines the web log for frequent pagesets using an enhanced WARM involving a tree data structure called T+weight tree.

3 Proposed system

Proposed system is intended to achieve weighted rule mining from web logs using weights for each web page visited by users. Weights are assigned based on various factors like frequent access, page's significance, dwelling time and interest of visitors. T+weight tree algorithm is proposed to discover the frequent and significant web pages from web log data.

Few pages may be visited only by few visitors. They may have dwelled for much longer time on that page which shows the importance of those pages. In the existing literature, dwelling time of users which may communicate the importance of the contents of that page are never taken into consideration. They take into account only the number of times a page is visited even if the time spent on that particular page is less, which shows that the page may be of less significance. Out of the visited pages, some pages might have taken more time because of more useful information than others. Proposed T+weight tree solves this problem. Figure 1 shows the process involved in weighted rule mining. Preprocessing the web log is essential before weight assignment to web pages. Cleaning the user log obtained from web server

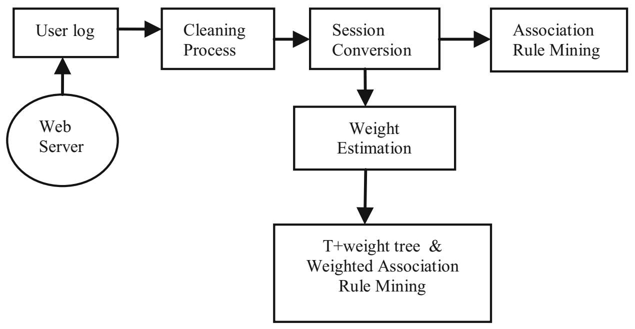

Fig. 1 System Architecture

Table 1 Access sequence details for one session ID of an IP address

Sl.No	Requested date and time	Page URL
1.	05/April/2012 02:13:32	/mech.html
2.	05/April/2012 02:17:47	/edudetails.html
3.	05/April/2012 02:23:09	/civil.html
4.	05/April/2012 02:31:18	/dept%10brochure/brochure.html

is the first and foremost work in preprocessing. It removes the irrelevant and redundant data. Cleaned log is processed for each IP in order to get the session wise details. Next, T+weight tree is constructed for frequent pageset mining and WARM. Here, the page dwelling time is considered as weight.

3.1 Preprocessing

Data preprocessing plays an essential role in web Usage Mining process. It performs a series of processing of web log file covering data cleaning, user identification, session identification, path completion and transaction identification. It removes the invalid data and successfully improves the eminence of mining model and can also reduce the mining time. The data preprocessing task is more complex in web usage mining. However, it serves to offer structural, reliable and integrated data source for pattern discovery. Raw log files contain unimportant details like image accessed, failed entries, etc., which will affect the accuracy of pattern discovery and analysis [14]. Almost 80 % of mining efforts are often depleted to improve the quality of data. Hence, web log analysis and data preprocessing are more essential.

The web log data will be in the form of relational database model. Weight of visited pages is considered as a function of page dwelling time of visitors. Thus, dwelling time of URLs in the web log must be present as one of the attributes for each page visited. This arrangement makes the mining process easier in order to obtain the significant pages with weights.

The web page request details are preserved in the web logs. Attributes like Page URL, IP address, Session ID, Requested time and Dwelling time are updated for each page request in the log list.

Web log is preprocessed to obtain the backend database in a relational form for mining as shown in Table 1 (access sequence details) and Table 2 (session details). The details of the table belong to the web log of an educational institution, which is considered as the dataset for the proposed work. Each page request is restructured as a separate entry with unique session ID for the user [15]. Session details carry for a particular duration, the total number of sessions in the web log and the relevant total number of pages for all the sessions.

Dwelling time can be included as one of the attribute in Table 1 by making use of the following Eq. (1).

$$T_{dp} = \sum_{p=1}^{m} (R_{p+1} - R_p) \tag{1}$$

Since requested time is available for each page in the table, page dwelling time of pth page is calculated by subtracting the requested time of pth page R_p from the requested time of p+1th page R_{p+1} as given in Eq. (2) using time scale. Otherwise, by means of cookies in the working node, we can calculate the total dwelling time of all pages separately and can send all the cookies to the web log when the session closes [16]. This will help to have the page dwelling time directly from the log.

Table 3 shows one such sample Visitor's log derived from the web log. For instance, it is assumed that in a day, 5 visitors (based on total number of session IDs) have visited total of 4 pages. Every element of each record in Table 3 represents page dwelling time of the respective pages.

Table 2 Session details

Sl.No.	Session ID	IP Addr.	Page1	Page2	…	Page n
1.	2358105	60.93.19.86	/eee.html		…	
2.	2358106	87.212.41.3	/mark.html	/edufback….	…	
3.	2358107	125.98.7.109	/rank.html	/corresmessa….	…	/toointeres…
4.	2358108	215.193.336...	/ece.html		…	
5.	2358109	77.249.61.125	/admission.html	/feedback.html	…	

Table 3 Sample Visitor's log

Visitors (or) session ID	Page dwelling time (min)			
	p1	p2	p3	p4
1	8	3	0	0
2	0	2	4	0
3	4	0	4	0
4	6	2	5	0
5	0	7	0	15

If user defined minimum support is 2, at least 2 visitors should have visited that page. Otherwise the particular page is not said to be frequent. Hence, page p4 in Table 3 is not considered to be frequent, even though it has got more dwelling time than other pages and may be more informative and desirable. Therefore, dwelling time plays an important role, without which loss of information may arise. Due to this reason the proposed system considers and gives due credits to the weighted support.

3.2 Weight estimation

3.2.1 Weighted attributes

Tao et al. [17] have considered weight settings for a super market domain. A (a_1, a_2, …, a_k) are the attributes chosen to compute weights. As the proposed algorithm's domain is web log mining, visitor's page dwelling time is considered to be an essential attribute in the log. There are two categories of weights namely the *page weight* and the *pageset weight*.

3.2.2 Page weight

Page weight is a value attached to a page URL visited, representing its significance and denoted as w(p). It is a function of selected weighting attributes denoted as w(p) = f(a). The weight related attribute here is user's average dwelling time on that page.

3.2.3 Pageset weight

Based on the page weight w(p), the weight of a pageset, denoted as w(ps), can be obtained from the weights of the pages visited by a particular user. To calculate the average value of the page weights, the following Eq. (2) is used.

$$w(ps) = \frac{\sum_{x=1}^{|ps|} w(p_x)}{|ps|} \tag{2}$$

It is possible that in a web log database some pages may be visited by more people in shorter times, while some pages may be visited by few people but for much longer times. Both the cases are significant. Formal case shows the usual and frequently repeated pages which have attracted visitor's interest, whereas the latter case shows the importance of informative pages. This has motivated the authors to find a formula for calculating the weight based on dwelling time. Equation (2) has steered to obtain the weight of m-pageset ps given by,

$$w(ps) = \frac{n_{ps}}{n}\left(\sum_{q=1}^{n}\sum_{p=1}^{m} T_{pq}\right) \tag{3}$$

Hence, pageset weight is the product of number of visitors who visited a pageset for a particular duration out of total number of visitors of that duration and sum of dwelling time of all the visitors of that pageset. Where n_{ps} and n are the number of visitors who visited a particular pageset and total number of visitors respectively for a fixed duration. m is the total number of pages. T_{pq} represents the dwelling time of a page pεps of a qth visitor. The proposed methodology is based on weighted minimum support. If a pageset assures user defined weighted minimum support $w_{min-sup}$, then it is considered to be a frequent pageset. In case of 1-pageset, total pageset weight is equal to total page weight and hence $n_{ps} = n_p$.

In case of WARM, the downward closure property may not hold properly. It is not essential that all the subsets of a frequent pageset should be frequent, because the sense of frequent pageset is adapted to handle weighted support. But this is overruled in the proposed T+weight tree method. Every page in a pageset has to be individually frequent with respect to the records of that pageset.

3.3 T+weight tree

The abstraction of the database is stored in memory and used to mine the frequent pagesets based on $w_{min-sup}$.

For Table 3 let $w_{min-sup}$=2. By applying weight definition given in Eq. (3), pages p1, p2, p3 and p4 are all frequent 1-pagesets, because individual page weight is the ratio of the total dwelling time of all the visitors who have visited that page to the total number of visitors, both for a stipulated duration. Individual page weights for our sample database are: w(p1)=(3*18)/5=10.8, w(p2)=(4*14)/5=11.2, w(p3)=(3*13)/5=7.8 and w(p4)=(1*15)/5=3. 2-pageset {p2,p3} is also frequent as weight({p2,p3}) =. 2*((2+4)+(2+5))/5=(2*13)/5=5.2. Number of visitors who have visited both p2 and p3 are alone considered for adding the dwelling time. w{p2} = (2*4)/5 = 1.6, which is less than $w_{min-sup}$. Even though p2 appears with less weight, {p2,p3} is frequent.

This is the basic motivation behind the proposed T+weight tree method. It's a tree based data structure for finding the frequent pagesets based on weight in a single scan of the database.

An m-pageset ps is frequent and satisfies downward closure property, if it satisfies both the conditions:

(i) Individual page weights in pageset ps with respect to only the set of records of ps is $\geq w_{\text{min-sup}}$.
(ii) Total weight of ps is $\geq w_{\text{min-sup}}$.

The objective of the proposed system is to discover the output as weighted association rules for the given input web log database D using T+weight tree method. To obtain the weighted association rules, the following steps are to be incorporated.

1. T+weight tree is built with every element of the record in D.
2. All frequent pagesets using $w_{\text{min-sup}}$ are discovered.
3. Applying ARM based on user defined minimum confidence, weighted association rules are discovered [2].

Weighted association rules are the rules which carry frequent pages with weights in both antecedent and consequent part of an association rule.

This method of finding the frequent pagesets using T+weight tree method is seen better than Weighted tree method [18] as applied to web logs, for finding the frequent pagesets and weighted association rules.

3.3.1 T+weight tree node structure

T+weight tree has two nodes as in Fig. 2.

1. *Page node* It has a page URL with two pointers. One is directed to the nodes having session IDs and weights and the other is directed to next page URL.
2. *SID node*It has two parts, Session ID (SID) and Weight as shown in Fig. 2b. Weight indicates the time dwelled in a page in that session. As many similar nodes as there are SIDs consisting of that page URL is coupled to the respective page node by means of pointers.

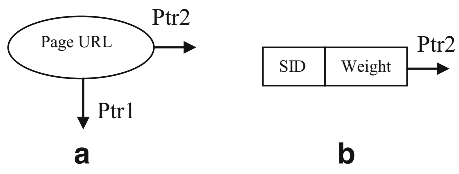

Fig. 2 **a** Page node. **b** SID node

Database is scanned only once to obtain the T+weight tree. While scanning, a separate page node is formed for each new page URL undergone and the SID nodes for the particular page visited are affixed to the respective page nodes along with dwelling time as weights.

3.4 WARM algorithm based on weights

The T+weight tree method of WARM has the following steps.

(i) Creating T+weight tree.
(ii) Eliminating infrequent Page URLs in the tree.
(iii) Finding frequent pagesets based on weights.
(iv) Performing WARM.

These steps can be achieved by means of the following algorithms (Algorithm A to Algorithm E).

Algorithm A: Creation of T+weight tree.

```
Input   : Database D
Output  : T+weight tree
Method:
for each attribute or page URL 'p' with non zero
dwelling time T in records of D do
begin
    create SID nodes with weight as the dwelling time T
    and attach these nodes to relevant Page nodes.
end
```

Algorithm B: Reducing the T+weight tree

```
Input   : $w_{min\text{-}sup}$
Output  : {F1} (set of Frequent 1-pagesets or pages ) and
          T+weight tree without infrequent Page URLs.
Method:
for each page URL 'p' in T+weight tree do
begin
    if [($n_p$/n)(sum(dwelling time of all SID nodes of p))]<
        $w_{min\text{-}sup}$
    then remove p's branch from the T+weight tree
    else {F1} = {F1} U {p}
end.
```

Algorithm C: Finding S (set of all non empty subsets of F1)

Total number of subsets of an n element set is 2^n. By mathematical "combinations" one can get nC_r number of r element subsets out of n element set by using,

$$nC_r = \frac{n!}{r!(n-r)!} \quad (4)$$

Since, single element subsets and Null set are not required, only a total of 2^n-nC_1-1 (Since nC_1 is n, 2^n-n-1) subsets are needed to be obtained.

```
Input  : {F1}(set of Frequent 1 pages )
Output : {S} (Set of all non empty subsets of {F1}except
          one element subsets i.e. frequent 1- pagesets)
Method:
Main( )
{ // input {F1}
   for i = 0 to size //size is no. of elements n in power set.
   subset (0,0,size)}
subset (start, index, N)
  if (index-start+1==N) do
  {
   for j=index to Size
     for i=start to index
       output({a[i],a[j]});// output {S}
  if (start!=N)   // N is no. of subsets
     subset (start+1, start+1, N)
  }
 else subset (start, index+1,N)
end
```

Algorithm D: Discovering frequent pagesets

Let {x} be the set of attributes or pages and is an element of {S} and |R| is number of records of {R} (visitors of a particular pageset n_{ps}).

```
Input  : Reduced T+weight tree, $w_{min-sup}$, n (total number of
          Visitors during a period)
Output :{$F_{all}$}(set of all frequent pagesets, except frequent 1-
          pagesets)
Method:
for each {x} in {S} do
   begin
    R={SIDs of first element in {x }}
    for each element 'a' in {x} other than first element do
       begin
         R={R} $\cap$ {SIDs in which 'a' is present}
       end
    if R is non empty then
     for each element 'e' in {x} do
       begin
         if[(|R| /n)(sum(dwelling time of 'e' in every record r
             in R ))]< $w_{min-sup}$
         then go to next {x} in {S} and repeat from start
         else next element 'e'
       end
    if [(|R| /n)( sum (dwelling time of elements of R w.r.to
             {x}))]$\geq$ $w_{min-sup}$
       then {$F_{all}$}= {$F_{all}$} U {x}
  end
```

Algorithm E: WARM

```
Input  : {$F_{all}$}(set of all frequent pagesets, except frequent 1-
           pagesets), $c_{min}$ (user defined minimum confidence).
Output: Weighted association rules
Method:
for each {x} in {$F_{all}$}do
    begin
      for each subset {s} in {x} do
         begin
           if (w({x})/w({s})) $\geq$ $c_{min}$
          then  output a rule of the form {s} =>{(x -s)}
         end
    end
```

3.5 Illustration for the sample Visitor's log

The T+weight tree for the sample Visitor's log of Table 3 is in Fig. 3.

The ellipses are the pages in the database and the rectangular boxes are the session IDs with weight. If $w_{min-sup}$=2, by this algorithm, since all the 1-pagesets are frequent as shown in Sect. 3.3, there won't be any branch reduction in the tree. For this tree {F1} = {p1, p2, p3, p4} and {S} is set of subsets of {F1} except frequent-1 and null sets. For a 4 element set there will be 2^4 subsets. Leaving frequent-1 and null sets 2^4-$4C_1$-1=11 subsets are generated. Hence,

$$\{S\} = \{\{p1, p2\}, \{p1, p3\}, \{p1, p4\}, \{p2, p3\}, \{p2, p4\}, \{p3, p4\}, \{p1, p2, p3\}, \{p1, p2, p4\}, \{p1, p3, p4\}, \{p2, p3, p4\}, \{p1, p2, p3, p4\}\}$$

Every element set of {S} is {x}. By algorithm (iv), first let us consider the element set {p1,p2}.

{x} = {p1, p2}

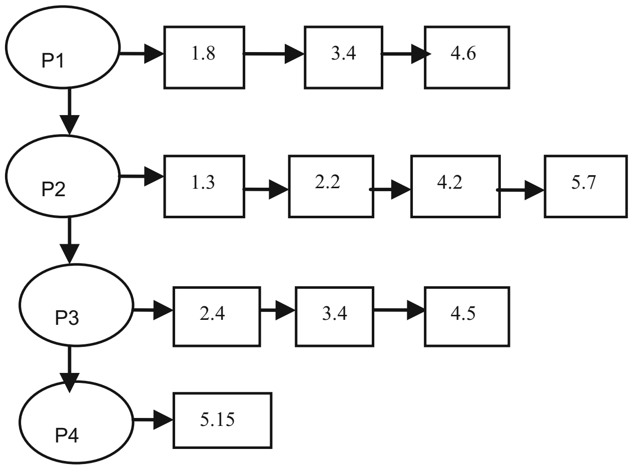

Fig. 3 T+weight tree for the sample database

R = {1,3,4}
a = p2 (other than first element);
R = {1,3,4}∩{1,2,4,7} = {1,4}
R is non empty.

For each element ‘e’ in {x},

e = p1; $2(8+6)/5=5.6 \geq w_{min-sup}$
e = p2; $2(3+2)/5=2 \geq w_{min-sup}$

If $w_{min-sup}$=3, then p2 will not be frequent and hence {p1,p2} is not a frequent pageset.
Then check weight for pageset {x}
$2(8+3+6+2)/5 = 7.6 \geq w_{min-sup}$.
Hence {x} = {p1,p2} is a frequent 2-pageset and becomes an element of {F_{all}}

All the attributes (pages) of {x} in records of R has to be individually frequent. If at least one attribute is infrequent and if it does not meet the user defined $w_{min-sup}$, then no need to check for other attributes of {x}. Next element of {x} in {S} is considered next.

Similarly checking for all the subsets of {S}, {F_{all}} = {{p1,p2},{p1,p3}}. Remaining sets of {S} are infrequent. Total frequent pagesets, hence will be individual elements of {F1} which is a set of frequent 1-pages and {F_{all}}. To mine less number of more significant pages, $w_{min-sup}$ has to be raised.

With the help of algorithm (v), weighted association rules have to be mined. Let us for example consider one frequent 2-pageset from {F_{all}}, say {x} = {p1,p2} and let us have user defined minimum confidence c = 50 % i.e. 0.5.

$$\{s\} = \{p1\}$$
$$w(\{x\})/w(\{s\}) = w\{p1, p2\}/w\{p1\} = 7.6/10.8$$
$$= 0.7 \geq c_{min}$$

Hence the rule {p1} -> {p2} is outputted as it is a strong association rule satisfying minimum confidence. Similarly, for other frequent pagesets, strong association rules have to be mined using WARM.

4 Experimental evaluation

For assessing the performance of proposed method, experiments are conducted on two different datasets. One is EDI (educational institution) dataset and the other is msnbc dataset available in UCI machine learning repository from Internet Information Server (IIS) logs for msnbc.com. Comparison has been carried out between Weighted tree [18] and proposed T+weight tree, both in terms of speed (CPU execution time in seconds) and space (Number of frequent pagesets generated) for various $w_{min-sup}$. For comparison sake, both the data are considered for one full day duration. i.e for twenty four hours. For calculating the speed in terms of CPU execution time, stubs were included in the program.

Experiments were performed on an Intel Core I5, 3.2 GHz processor machine with 2GB RAM and 500 GB hard disk with Windows XP platform. T+weight tree algorithm is implemented in Java.

The proposed method shows better performance because of the difference in the formula used for calculating the pageset weight. Experimental results are provided below from Figs. 4, 5, 6, 7, and 8. Tables 4 and 5 show the comparison of execution time and number of frequent pagesets generated with respect to Weighted tree and T+weight tree methods for msnbc and EDI datasets. The empirical results confirm that for the datasets, the proposed T+weight tree method takes lesser execution time than Weighted tree method, since it produces lesser number of more significant pagesets comparatively.

Comparison of the results for execution time of both the methods is shown in Figs. 4 and 5 and that for comparison of number of pagesets generated by both the methods is given in Figs. 6 and 7. It is seen that, as $w_{min-sup}$ increases the execution time and space reduces more for T+weight tree than for Weighted tree.

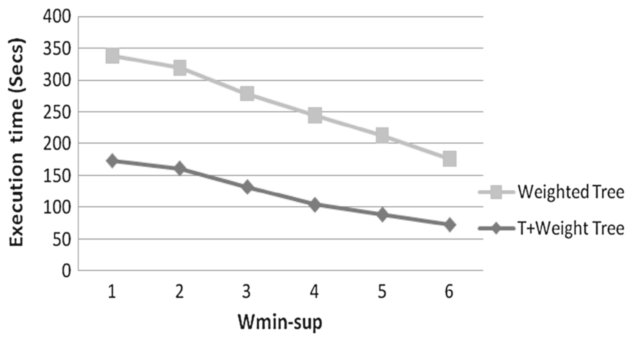

Fig. 4 Comparison of Execution time for Weighted tree and T+weight tree for msnbc dataset

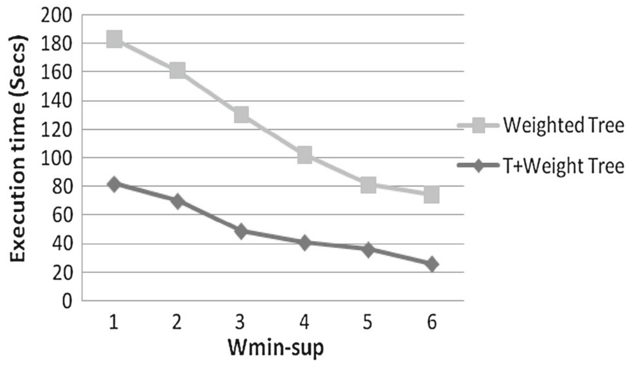

Fig. 5 Comparison of Execution time for Weighted tree and T+weight tree for EDI dataset

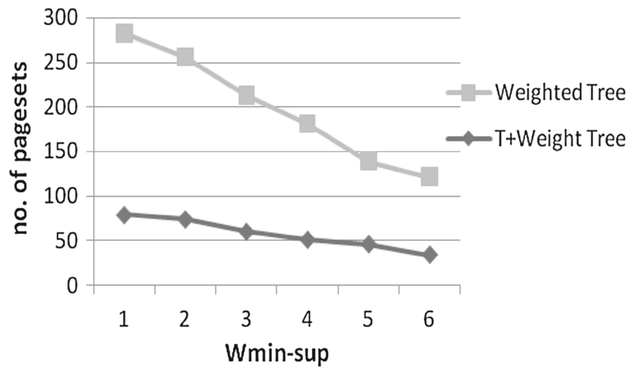

Fig. 6 Comparison of Number of pagesets generated by Weighted tree and T+weight tree for msnbc dataset

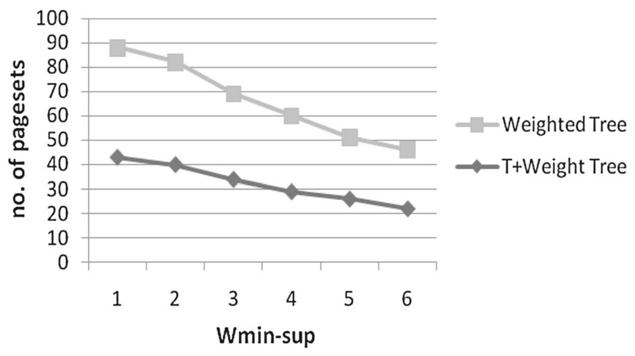

Fig. 7 Comparison of Number of pagesets generated by Weighted tree and T+weight tree for EDI dataset

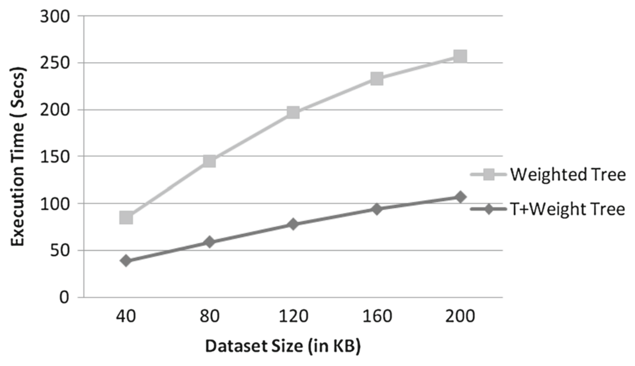

Fig. 8 Scalability of T+weight tree on EDI dataset

When it comes to comparison among EDI and msnbc datasets, execution time and number of frequent pagesets mined are more for msnbc dataset than EDI dataset as it is a globalized news repository. The news that affected the people most, whether positively or negatively and its impact are clearly visualized by mining the msnbc dataset. The change in trends can also be envisioned.

T+weight tree is scalable because it performs well even when the input dataset size increases. For this purpose both the datasets are splitted into smaller datasets each consisting of different sizes for experimental purposes. Dataset range is changed from 40 to 200 KB and the results are produced in comparison with Weighted tree for EDI dataset in Fig. 8 for a $w_{min-sup}$ of 6. It shows that the proposed method scales up well with respect to increase in dataset size and that it can efficiently work on larger datasets. From Fig. 8 we infer that T+weight tree is linearly scalable and has got an improved scalability than Weighted tree.

Table 4 Execution time and number of frequent pagesets generated for msnbc dataset

$w_{min-sup}$	Speed (execution time in s)		Space (number of frequent pagesets)	
	Weighted tree	T+weight tree	Weighted tree	T+weight tree
1	339	173	283	79
2	320	161	256	74
3	279	132	213	60
4	245	104	181	51
5	213	89	139	46
6	176	73	121	34

Table 5 Execution time and number of frequent pagesets generated for EDI dataset

$w_{min-sup}$	Speed (execution time in s)		Space (number of frequent pagesets)	
	Weighted tree	T+weight tree	Weighted tree	T+weight tree
1	183	82	88	43
2	161	70	82	40
3	130	49	69	34
4	102	41	60	29
5	81	36	51	26
6	74	26	46	22

5 Conclusion

Method for mining frequent pagesets from web log based on page dwelling time as weights is discussed. This employs T+weight tree arrangement and from experimental evaluation it is found to be more efficient than Weighted tree method, by both time and space. The time elapsed for finding all the subsets of a set is more. If an efficient way is found out to reduce this time, then this method would be much more beneficial to find the frequent pagesets.

The system so far discussed gains importance in mining the frequent pagesets and association rules based on weighted minimum support and confidence. But those frequent pagesets may have frequently occurred in the earlier stage of the respective duration than in the later stage. In order to identify most recently and less recently accessed frequent web pages, the log may be divided into n windows and weights may vary for several windows ranging from high value for the recently accessed window to low value for that of least recently accessed [9]. The pages in a particular window carry the weight of the window multiplied by its dwelling time. Then T+weight tree method may be preceded.

T+weight tree method of WARM gains a credit in mining the web logs of educational institutions, to find the frequent pages which shows the behavior of the students. The access of those pages which spoil them can be denied and those which are informative can be taken into account for taking future decisions.

The browsing behavior of users cannot be accurately predicted in advance for any organization and it may randomly vary day-to-day. It depends upon the need of the person and the trend. In our example, we have considered the data for one full day from both the datasets, which may give various results if the data considered is for some other day in the dataset. Hence as a future work some stochastic optimization models can be included.

References

1. Zhao, Q., Bhowmic, S.S.: Association Rule Mining: A Survey Technical Report, CAIS, Nanyang Technological University, Singapore. No. 2003116 (2003)
2. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2004)
3. Tao, F., Murtagh, F., Farid, M.: Weighted association rule mining using weighted support and significance framework, SIGKDD 2003
4. Wang, H., Yang, C., Zeng, H.: Design and implementation of a web usage mining model based on upgrowth and prefixspan. Commun. IIMA **6**(2), 71–86 (2006)
5. Chitraa, V., Davamani, D., Selvdoss, A.: A survey on preprocessing methods for web usage data. Int. J. Comput. Sci. Inf. Secur. **7**(3), 78–83 (2010)
6. Mishra, R., Choubey, A.: Discovery of frequent patterns from web log data by using FP growth algorithm for web usage mining. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **2**(9), 311–318 (2012)
7. Wang, W., Yang, J., Yu, P.: Efficient mining of weighted association rules (WAR), In: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 270–274 (2000)
8. Sun, L., Zhang, X.: Efficient frequent pattern mining on web logs, In: APweb 2004, LNCS 3007, pp. 533–542. Springer, Berlin (2004)
9. Srivastava, A., Bhosale, A., Sural, S.: Speeding up web access using weighted association rules. PReMI 2005. Lecture Notes in Computer science, vol. 3776, pp. 660–665. Springer, Berlin (2005)
10. Iváncsy, R., Vajk, I.: Frequent pattern mining in web log data. Acta Polytech. Hung. **3**(1), 77–90 (2006)
11. Sun, K., Bai, F.: Mining weighted association rules without preassigned weights. IEEE Trans. Knowl. Data Eng. **20**(4), 489–495 (2008)
12. Yang, Y., Guan, X., You, J.: Enhanced Algorithm for Mining the Frequently Visited Page Groups, Shanghai Jiaotong University, Shanghai
13. Velvadivu, P., Duraisamy, K.: An optimized weighted association rule mining on dynamic content. Int. J. Comput. Sci. Issues **7**(2), 16–19 (2010)
14. Kewen, L: Analysis of preprocessing methods for web Usage Data, In: 2012 International conference on measurement, Information and Control (MIC), School of Computer and Information Engineering, Harbin University of Commerce, Harbin
15. Malarvizhi, S.P., Sathiyabhama, B.: Enhanced reconfigurable weighted association rule mining for frequent patterns of web logs. Int. J. Comput. **13**(2), 97–105 (2014)
16. Matthew, M.: ASP.NET The Complete Reference. Tata Mcgraw Hill Education Private. Ltd., Berkeley (2002)
17. Tao, F., Murtagh F., Farid, M: Weighted Association Rule Mining using Weighted Support and Significance Framework, In: SIGKDD (2003)
18. Kumar, P., Ananthanarayana, SV.: Discovery of Weighted Association Rules Mining, 978-1-4244-5586-7/10/$26.00 C 2010 IEEE, vol. 5, pp.718–722

S.P. Malarvizhi received the B.E. degree in Electrical and Electronics Engineering from Annamalai University in 1994 and the M.E. degree in Computer Science and Engineering from Anna University of Technology, Coimbatore in 2009. Currently she is the Research Scholar of Anna University, Chennai, Tamilnadu, India. She is a member of the ISEEE, ISTE India and NC member of Computer Society of India.

B. Sathiyabhama received her Ph.D. degree in Computer Science and Engineering from National Institute of Technology, Tiruchirappalli in 2009. Currently she is a Professor and Head in the department of Computer Science and Engineering at Sona College of Technology, Salem, India where she has been since 1998. Her research interests include Bioinformatics, Compilers, Algorithm Analysis, Big Data Analytics and Data mining with a focus on optimization based clustering techniques and. She is a member of the IEEE, ISTE India and Computer Society of India.