FM-WAP Mining: In Search of Frequent Mutating Web Access Patterns from Historical Web Usage Data

Qiankun Zhao
Nanyang Technological University, Singapore
and
Sourav S. Bhowmick
Nanyang Technological University, Singapore

Recently, a large amount of work has been done in web access pattern (WAP) mining. Most of the existing techniques focus on mining WAP that occur frequently from snapshot web usage data collection. However, web usage data is dynamic in real life. The dynamic nature of web usage data leads to two challenging problems. The first problem is the maintenance of existing WAP mining results. The second problem is discovering hidden knowledge from the historical changes to WAP. As the frequent WAPs are useful in many applications, knowledge hidden behind the historical changes to WAPs, which reflects how WAPs change, is also critical to some applications such as adaptive web, web site maintenance, business intelligence, etc. In this paper, we propose a novel approach to discover hidden knowledge from historical changes to WAPs. We define a new type of knowledge, *Frequent Mutating WAP* (`FM-WAP`), based on the historical changes to WAPs. Moreover, a FM-WAP mining approach, which can discover all the FM-WAPs efficiently from historical web usage data, is presented. Firstly, the historical web usage data collection is partitioned into different groups according to the user-defined `calendar pattern`, where each group is represented as a `WAP forest`. Then, changes among the WAP forests sequence are detected and stored in the `global forest`. Lastly, the FM-WAP is extracted by traversing the global forest. Extensive experimental results show that our proposed approach is efficient with good scalability to very large web usage data collection. Moreover, this approach can produce new knowledge about web access patterns that cannot be discovered by existing techniques.

Categories and Subject Descriptors: H.2.8 [**Database Management: Database Applications**]: Data Mining

General Terms: Algorithm, Experimentation

Additional Key Words and Phrases: Web access pattern, Historical changes, FM-WAP

1. INTRODUCTION

Recently, with the evolution of the internet into the global information infrastructure, data on the web becomes the largest source of publicly available data. However, with the complex structures of the web sites and the overwhelming amount of data available on the web, it is really difficult for web surfers to access the desired information efficiently and for web masters to design and maintain a good web site structure as well. Under such circumstances, *web usage mining* was introduced to help both of them by automatically extracting useful knowledge from web log data collections, which record the history of interactions between web servers and web surfers [Cooley et al. 1997]. Such useful knowledge includes association rules [Srivastava et al. 2000], frequent web access patterns (WAPs) [Pei et al. 2000], statistics of web traffic [Spiliopoulou 2000], clusters and classes of web users [Fu

S_ID	Web Access Sequence
1	$\langle$ a, b, d, c, a, f, g $\rangle$
2	$\langle$ a, b, e, h, a, f, g $\rangle$
3	$\langle$ f, g, m, g, i $\rangle$
4	$\langle$ b, d, c, b, e, h $\rangle$

(a) The First Month

S_ID	Web Access Sequence
1	$\langle$ a, b, d, c, a, f, g, i $\rangle$
2	$\langle$ b, d, c, n $\rangle$
3	$\langle$ a, f, g, a, b, d, c, n $\rangle$
4	$\langle$ b, e, h, b, d, c, n $\rangle$

(b) The Second Month

S_ID	Web Access Sequence
1	$\langle$ b, d, c, a, f, g $\rangle$
2	$\langle$ b, e, h, b, d, c $\rangle$
3	$\langle$ a, f, g, m, g, i $\rangle$
4	$\langle$ a, b, e, h, a, f, g $\rangle$

(c) The Third Month

S_ID	Web Access Sequence
1	$\langle$ a, f, g, a, b, e, h, j $\rangle$
2	$\langle$ e, d, c, b, e, h, j $\rangle$
3	$\langle$ a, b, d, c, a, f, g $\rangle$
4	$\langle$ f, g, m, g, i, g, q $\rangle$

(d) The Fourth Month

Table I. Web access sequences of a web site.

et al. 1999]. Among them, mining of *frequent WAPs*, sequences of web pages that are frequently visited by web users in a fixed order, is the most fundamental and well-researched topic [Chen et al. 1998; Chi et al. 2004; Eirinaki and Vazirgiannis 2003; Pei et al. 2000; Xiao and Dunham 2001; Xiao et al. 2003]. The idea is to transform the web log data into sequences of events with user identifications and timestamps, and then extract the frequent sequential patterns from the events data with the statistic metric *support* [Pei et al. 2000]. It has also been shown that the frequent WAPs extracted from the web log data can be useful in a wide range of applications such as personalization, system improvement, site modification, business intelligence, and usage characterization, etc [Srivastava et al. 2000].

Generally, the web usage mining process can be considered as a three-phase process, which consists of *data preparation*, *pattern discovery*, and *pattern analysis* [Srivastava et al. 2000]. In the first phase, web log data are transformed into sequences of events based on the identification of users and sessions. In the second phase, statistical methods and data mining techniques are applied to extract interesting patterns such as frequent WAPs. Such patterns are stored for further analysis in the third phase. Since the pattern analysis phase is application dependant, let us illustrate the first two phases with an example. Given a web log archive that records the navigation history of a web site, by using some existing preprocessing techniques [Chen et al. 1998; Xiao and Dunham 2001], the raw log data can be transformed into a sequence of events as shown in Table I, where a, b, $\cdots$, z represent web pages and a sequence $\langle$ a, b, d $\rangle$ denotes a visiting pattern from page a to page b and finally to page d. Each sub-table in Table I records the web access sequence for a particular month. In the second phase, we can apply existing web usage mining techniques [Spiliopoulou 2000; Pei et al. 2000; Srivastava et al. 2000] to this dataset and extract the following types of knowledge. Note that only representatives of such knowledge are presented here. The reader may refer to [Srivastava et al. 2000] for details.

—*Association rules*: The association rule mining technique in [Agrawal et al. 1993] can be applied to the web access sequence data to extract the correlations between web pages that are accessed together during a user session. For example, $\{b, d\} \rightarrow c$ is an association rule with 50% *support* and 100% *confidence* according to the log data in Table I (a). It implies that whenever pages b and d are accessed, page c should also be accessed.

—*Frequent WAP*: Frequent WAPs are extensions of association rules. Beside correlations, they also imply the order of the web pages being visited within individual sessions. For example, suppose the threshold of support is 0.5. Then $\langle a, f, g\rangle$ can be extracted as a frequent WAP from Table I (a) (by using the technique described in [Pei et al. 2000]) since it has a support of 0.5 that is no less than the threshold. It implies that 50% of the user sessions follow the visiting pattern of $a \rightarrow f \rightarrow g$.

—*Statistical information of traffic*: An example of statistical information is that the web page b is the most frequently visited web page during the four months. Other statistics such as average page view, entry page, exit page, and geographical location of web surfers can also be obtained as shown in [Spiliopoulou 2000; Srivastava et al. 2000].

1.1 Motivation

From Table I, it is evident that the web usage data is dynamic. However, most of the above techniques focus only on discovering knowledge based on the statistical measures obtained from the static characteristics of web log data. They do not consider the dynamic nature of the web usage data. That is, with the changes to the content of the web site and users' familiarity to the web site structure, together with the introduction of new web visitors, WAPs may change over time. With the newly accumulated web usage data, some new frequent WAPs may emerge and some old frequent WAPs may become invalid. For instance, reconsider the previous example. The frequent WAP $\langle b, e, h \rangle$ in the first month becomes infrequent in the second month. Similarly, it is possible that the web page f, instead of b, becomes the most popular page in the next few months. To handle this issue, several techniques have been proposed to maintain and update the discovered knowledge [Baron and Spiliopoulou 2003; Baron et al. 2003; Ganti and Ramakrishnan 2002], and detect changes between two sets of mining results [Ganti et al. 1999; Liu et al. 2001]. **However, beside the updated version of such knowledge, interesting knowledge can be discovered from the sequence of changes to WAPs based on the history of the dynamic behaviors.** For instance, if we partition the above log data into four groups based on the timestamps as shown in Table I, the following novel knowledge can be discovered by mining the history of changes to WAPs.

- **Conserved Web Access Patterns (CWAP)**: Access patterns that never changed or seldom changed in terms of their support values in the histories are called *conserved WAPs (CWAPs).* They can be either frequent WAPs or infrequent WAPs (WAPs with a support less than the minimal support threshold). For instance, the frequent WAP $\langle a, b \rangle$ and the infrequent WAP $\langle g, i \rangle$ in

Frequent WAP	FM-WAP
$\langle a, f, g \rangle$	Yes
$\langle b, d, c \rangle$	Yes
$\langle a, b, e, f, g \rangle$	No
$\langle b, e, h \rangle$	No
$\langle \cdots \rangle$	$\cdots$

(a) Frequent WAP

FM-WAP	Frequent WAP
$\langle b, d, c \rangle$	Yes
$\langle b, e, h \rangle$	No
$\langle a, b, d \rangle$	No
$\langle f, g \rangle$	Yes
$\langle \cdots \rangle$	$\cdots$

(b) FM-WAP

Table II. Web access patterns.

Table I (with a minimal support threshold of 50%) did not change during the four months in terms of their support values.

- **Frequently Mutating WAP (FM-WAP)**: There are WAPs that change frequently in terms of their supports over a period of time. These WAPs are called *frequent mutating WAPs (FM-WAPs)*. Note that frequent WAP and FM-WAP are different. Frequent WAP is based on the support of the WAPs in the snapshot data, whereas FM-WAP is based on the *significance* and *frequency* of the changes to their support values, which we shall discuss later. For example, given the web log data shown in Table I, the set of frequent WAPs discovered by the WAP-mine [Pei et al. 2000] algorithm with the minimal support of 50% is shown in Table II (a). However, not all of these frequent WAPs are FM-WAPs. For instance, frequent WAPs $\langle a, f, g\rangle$ and $\langle b, d, c\rangle$ are FM-WAPs based on the *significance* and *frequency* of the changes to their supports. Whereas, the frequent WAPs $\langle a, b, d, c\rangle$ and $\langle b, e, h\rangle$ are not FM-WAPs as their supports do not change frequently and significantly. *Hereafter, we shall use the term "changes to WAPs" to mean the changes to WAPs in terms of their support values.*

Such novel knowledge can be useful in different applications such as the adaptive web, web site modification and maintenance, business intelligence, web site design, efficient web caching, etc. For instance, the CWAPs can be used as a guide for efficient web structure design. By extracting all the frequent CWAPs from the web sites in a specific domain, a well designed structure of a web site in the corresponding domain can be constructed. We call it a well designed web site structure because the frequent CWAPs reflect the most commonly used and stable patterns of visiting a web site for all the users, both familiar users and new users. At the same time, the infrequent CWAPs should be excluded from a well designed web site. It is because that infrequent CWAPs may represent parts of the web site that are very specific so that only a small group of fans or patient users will always follow such visiting patterns. Moreover, the CWAPs can be used for efficient promotion and marketing strategies in e-commerce. For instance, we can put the advertisements such as promotion and special offers along the way of the most frequent CWAPs since the users are expected to visit these pages with very high confidence and such patterns are not likely to change in the near future. With such high confidence of the correlation, the CWAPs can also be used for efficient web caching.

Similarly, the FM-WAPs can also be used for business intelligence. Suppose we obtained a set of FM-WAPs denoted as F. Then we can extract different types of

FM-WAPs from F. For example, we can extract a set of FM-WAPs I, $I \subseteq F$, which contains all FM-WAPs whose support values keep *increasing*. Consequently, I may represent the portions of the web site that are becoming more frequently visited. This may indicate that the corresponding parts of the web site are becoming more popular. With such semantic knowledge, trend-based business intelligence such as promotion, cross selling, advertisements, etc. can be adapted. For example, advertisements for the products can be placed through the access patterns in I. Moreover, by incorporating different properties of the web pages such as *content page*, *entry page*, and *exit page*, an efficient web site management strategy can be obtained. For example, if we find out that the page that is becoming more frequently visited is an exit page and it is not the right place for the users to leave the web site, the structure of that particular part will be modified to keep users on the web site longer and entice them to make more purchases or seek additional information. Similarly, the set of WAPs D and P, $D \subseteq F$ and $P \subseteq F$, which represent the sets of FM-WAPs whose supports keep *decreasing* and *changing periodically* in the history respectively, can be extracted. Such knowledge can be useful for building different intelligent business strategies and maintaining the web site structure as well.

To the best of our knowledge, the *CWAP*s and *FM-WAP*s cannot be discovered by existing web usage mining techniques. This is due to the fact that the existing techniques either focus on the snapshot data [Chen et al. 1998; Chi et al. 2004; Pei et al. 2000; Xiao and Dunham 2001] or on detecting the changes to the mining results [Baron and Spiliopoulou 2003; Baron et al. 2003; Ganti et al. 1999; Ganti and Ramakrishnan 2002]. None of these techniques considered the issue of *directly mining* the change patterns of WAPs from the original dataset. Even if we apply the frequent WAP techniques repeatedly to a sequence of snapshot data, they cannot generate such knowledge efficiently. Let us illustrate this with an example. Consider the example in Table I. If we apply WAP-mine [Pei et al. 2000] to the four partitions individually, only knowledge about the frequent WAPs in each month can be generated. For example, with a minimal support threshold of 50%, the frequent WAPs in the first month are shown in Table II (a). It is obvious that not all the support values for each WAP in the groups are recorded. Only for these WAPs that are frequent WAPs in all the groups, their historical support values are recorded. That means even if we want to use some post-processing techniques to extract the FM-WAPs and CWAPs from the mining results of frequent WAPs, the result is not complete at all. Table II (b) shows some of the FM-WAPs. It can be observed that not all FM-WAPs are frequent WAPs. To sum up, the approach of using exist frequent WAP mining techniques repeatedly, combined with the post processing on the mining results, has the following limitations.

- Firstly, as we shall see in Section 5.2, the process of repeatedly mining the frequent WAPs and post-processing the mining results is very expensive compare to our proposed mining approach. To extract the FM-WAPs and CWAPs, it is not necessary to mine the frequent WAPs since FM-WAPs and CWAPs are different from frequent WAPs. They are based on the percentage of changes to the support values and not the actual support values for the access patterns.
- Secondly, the result obtained by post-processing the frequent WAPs is not complete. It may include some of the CWAPs and FM-WAPs that are *frequent WAP*s

in each partition of the dataset. The change patterns of these infrequent WAPs, such as infrequent FM-WAPs and CWAPs, are lost. However, such knowledge is also important as we mentioned earlier. Details about the relationship between FM-WAPs and frequent WAPs are shown in Section 5.5.

- Lastly, this approach is not flexible. In the frequent WAP mining process, once the minimal support threshold changes, the whole process, which is expensive, has to be executed repeatedly.

1.2 Overview

In this paper, we propose a novel approach to extract the change patterns of WAPs from the historical web log data. Taking into account the dynamic nature of web log data, rather than updating the WAPs repeatedly, knowledge behind the changes to the WAPs is discovered. More specifically, we focus on discovering the frequent mutating WAPs and we call such mining process as *FM-WAP mining*. A short version of this paper appeared in [Zhao and Bhowmick 2004].

Similar to traditional web usage mining, FM-WAP mining has three phases: the *data preparation* phase, the *pattern discovery* phase, and the *pattern analysis* phase. In this paper we focus on the second phase. We assume that the web access log data has been properly preprocessed into separated session sequences using existing data preparation techniques [Xiao and Dunham 2001]. The pattern discovery phase can be further divided into the following sub-phases:

—*Construction of the WAP Forest*: The transformed web log data is partitioned into a sequence of partitions according to the user-defined *calendar pattern*. For each partition, a WAP forest is constructed by merging the set of *WAP trees*.

—*Detecting and Storing the Changes*: Changes between different WAP forests are detected by a modified tree comparison algorithm based on X-Diff [Wang et al. 2003]. The historical changes are stored in the proposed *global forest* structure.

—*FM-WAP Extraction*: The proposed mining algorithm is applied to the *global forest* to extract the FM-WAPs based on the *S-Value* and *F-Value* metrics.

We have implemented the FM-WAP mining algorithm and done extensive experiments. The experimental results show that our proposed algorithm can extract all the FM-WAPs, while repeatedly apply the existing frequent WAP mining techniques on the sequence of snapshot data and post-process the mining results can only extract part of the FM-WAPs (for the datasets we used, it can only extract around 30% of the FM-WAPs). Moreover, our proposed approach is more than three times faster than the above approach before we take into account the cost for the post-processing phase. The experimental results also show that our proposed approach is scalable to the size of the dataset. By comparing the FM-WAP mining results with existing frequent WAP mining results, it shows that more than half of the FM-WAPs cannot be discovered by existing web usage mining techniques. In summary, the major contributions of this paper are as follows.

- To the best of our knowledge, this is the first approach to discover useful change patterns of WAPs, frequent mutating WAPs (FM-WAPs), from the sequence of historical changes to web access patterns.
- By representing WAPs as unordered trees, which is a better representation of the underlying structural information for web access patterns than pure sequences,

we proposed an efficient algorithm for mining of FM-WAPs based on our proposed WAP forest and global forest data structures.

- Extensive experiments with both synthetic and real datasets have been conducted to demonstrate the efficiency and scalability of our algorithm.

The rest of this paper is organized as follows. In the next section, related research is discussed and compared with our approach. The overview of FM-WAP mining is presented in Section 3. In Section 4, details of the FM-WAP mining algorithm are described step by step. Section 5 presents the experimental results. Finally, the last section concludes this paper.

2. RELATED WORK

Our proposed FM-WAP mining is related to two existing research topics. They are frequent WAP mining and detecting changes to the mining results. In this section, we review some of the representative techniques.

Web access pattern mining is defined to extract hidden patterns from the navigation behavior of web users [Chen et al. 1998]. Different algorithms have been proposed to mine different types of access patterns from the web logs such as the maximal frequent forward sequence mining [Chen et al. 1998], the maximal frequent sequence mining with backward traversal [Xiao and Dunham 2001]. There are also algorithms for general web access pattern mining such as the WAP-Mine [Pei et al. 2000], the maximal and closed access pattern mining [Xiao et al. 2003; Chi et al. 2004], etc.

The maximal frequent forward sequence mining approach in [Chen et al. 1998] transforms the browsing patterns of users to maximal forward sequences. This work is based on statistically dominant paths. A maximal forward reference is defined as the sequence of pages requested by a user up to the last page before backtracking. Suppose the traversal log contains the following traversal path for a user: $\langle A, B, C, B, D \rangle$, after the transformation, the maximal forward sequences are $\langle A, B, C \rangle$ and $\langle A, B, D \rangle$. However, this approach may lose some useful information about the structure [Xiao and Dunham 2001]. Considering the above example, the structural information about the need to have a direct link from page C to page D is lost.

The maximal frequent sequence proposed in [Xiao and Dunham 2001] keeps backward traversals in the original sequences and thus preserve some structural information, i.e., the sibling relation within a tree structure. However, the maximal frequent sequence approach requires contiguous page access in the pattern, which limits the patterns to be found. Suppose there are two sessions $\langle A, B, C, A, E, A, D \rangle$ and $\langle A, B, C, A, D \rangle$. The access pattern of $\langle A, B, C, A, D \rangle$ may be missed since the access patterns $\langle A, B, C \rangle$ and $\langle A, D \rangle$ are not contiguous in the first session.

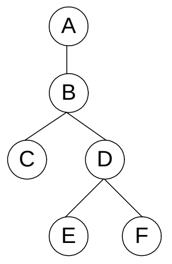
Two other approaches have been proposed in [Xiao et al. 2003; Chi et al. 2004] to overcome this limitation by considering the sessions as unordered trees. In [Xiao et al. 2003], the authors proposed a compact data structure, FST-Forest to compress the trees and still keeps the original structure. A PathJoin algorithm was proposed to generate all maximal frequent subtrees from the forest. However, this approach is neither scalable nor efficient. In [Chi et al. 2004], Chi et al. use the canonical form to represent the unordered trees. The authors proposed the CMTreeMiner algorithm to mine the maximal and closed frequent subtrees. It is claimed to be much faster than the PathJoin algorithm.

Another general algorithm is the WAP-Mine approach proposed in [Pei et al. 2000]. The authors proposed a compressed data structure, WAP-tree, to store data from the web logs. The WAP-tree structure facilitates the mining of frequent web access patterns. Different from the Apriori-based algorithms, the WAP-Mine algorithm avoids the problem of generating explosive numbers of candidates. Experimental results show that the WAP-Mine is much faster than the traditional sequential pattern mining techniques. Although the WAP-tree technique improved the mining efficiency, it recursively reconstructs large numbers of intermediate WAP-trees that require expensive operations of storing intermediate patterns.

The above web access pattern mining approaches focus on improving the efficiency of mining frequent WAPs and representing frequent WAPs in different format such as sequence and tree structure. However, the number of frequent WAPs can be huge. In particular, when the web log data accumulate over a period of time, the frequent WAPs have to be updated. Compare to the above approach, our proposed approach has the following advantages. Firstly, the entire web log data is grouped into sequences of partitions that are much smaller in size. The timestamps for the web log data within the same partition conforms to a particular calendar pattern. Consequently, our approach is more scalable. Secondly, our approach is not based on the actual support values for the access patterns. It is based on the changes to the support values. Lastly, our approach can provide more informative and concise knowledge about the dynamics of WAPs efficiently, while the above techniques cannot discover the complete set of FM-WAPs.

Considering the dynamic property of the datasets, there are several techniques proposed recently for maintaining and update previously discovered knowledge. They focus on two major issues. One is to actualize the knowledge discovered by detecting changes in the data such as the DEMON framework proposed by Ganti et al [Ganti and Ramakrishnan 2002]. Another is to detect interesting changes in the KDD mining results such as the FOCUS framework proposed by Ganti et al [Ganti et al. 1999], PAM proposed by Baron et al [Baron et al. 2003], and the fundamental rule change detection tools proposed by Liu et al [Liu et al. 2001]. However, these techniques have several limitations. Firstly, they were proposed for transactional database only, while the web access pattern should be represented as a tree structure to retain the structural knowledge of the web site [Xiao et al. 2003; Chi et al. 2004]. Secondly, they focus only on changes for the frequent itemsets while the evolution of infrequent itemsets can also be important. Thirdly, none of the above approaches has considered to mine the sequence of changes to obtain the knowledge about their change patterns.

More recently, we have proposed a similar approach, *FCS mining*, to extract frequent changing substructures from the historical structural changes to versions of XML documents [Zhao et al. 2004]. In *FCS mining*, we assume that versions of XML documents are available and we only focus on structural changes of the XML documents. However, FM-WAP mining is different in the following two aspects. Firstly, in FM-WAP mining, the raw web usage data is not stored as a sequence of versions. Rather, we proposed a calendar pattern based approach to partition the



(a)

Tree ID	Timestamp
T_{w1}	8:30, 1st, Feb, 2004
T_{w2}	10:30, 3rd, Mar, 2004
T_{w3}	16:30, 21st, Apr, 2004
T_{w4}	1:30, 6th, Feb, 2004
T_{w5}	9:30, 12th, Mar, 2004
T_{w6}	18:30, 23rd, Feb, 2004
T_{w7}	2:30, 16th, Apr, 2004

(b)

Fig. 1. An example.

raw web usage data into a sequence of groups, where the raw web usage data in each group is semantically linked together. Secondly, in FM-WAP mining, we focus on the changes to both the structure (WAP trees) and content (the support values for the WAP trees), while *FCS mining* focuses on only the structural changes.

To sum up, FM-WAP mining is proposed to overcome all the above mentioned limitations of existing related mining techniques and to produce novel and interesting knowledge from historical changes to WAPs.

3. OVERVIEW OF FM-WAP

In this section, we give an overview of FM-WAP. First, some preliminaries such as *WAP tree, calender schema, calendar pattern, WAP forest* are introduced. Next, the formal definitions of FM-WAP and FM-WAP mining are presented based on the preliminaries.

3.1 Preliminaries

As in most web access pattern mining approaches, web log can be considered as a sequence of events with user identifiers. There are different ways of preprocessing the web logs to user sessions with timestamp [Eirinaki and Vazirgiannis 2003]. We assume that the raw data has already been preprocessed into sessions using the technique in [Xiao and Dunham 2001], in which both forward traversal and backward traversal are considered. In our case, the user sessions are transformed into unordered trees called *WAP trees.* The advantage of considering the users sessions as unordered trees is that the unordered trees can capture more information about the structure of the web site since it does not require contiguous page access [Xiao et al. 2003]. Formally, the WAP tree, denoted as T_w, is defined as following.

Definition 3.1. **[WAP Tree]** Given a user session sequence $\langle P_1, P_2, \cdots, P_n \rangle$, where P_i is a web page, the corresponding *WAP tree* (T_w) for this sequence is represented as $T_w = (N,\ A,\ r)$, where r is the root of the tree that represent web page P_1; N is the set of nodes representing the web pages in $\{P_1, P_2, \cdots, P_n\}$; and A is the set of arches in the maximal forward sequences. ■

For instance, let $\langle\ A,\ B,\ C,\ B,\ D,\ E,\ D,\ F\ \rangle$ be an identified user session sequence, where $A,\ B,\ \cdots,\ F$ are identifiers of the web pages. The corresponding maximal forward sequences are $\langle\ A,\ B,\ C\rangle$, $\langle\ A,\ B,\ D,\ E\rangle$, and $\langle\ A,\ B,\ D,\ F\ \rangle$. According to our definition, this user session sequence can be modelled as an unordered tree as shown in Figure 1 (a). This kind of modelling method has been used in [Xiao et al. 2003; Chi et al. 2004]. After the transformation, the web logs are represented as a sequence of WAP trees. Each tree will be accompanied with a timestamp, i.e., 8:30 am, 1st Feb, 2004, which is the actual time when such access pattern is performed as shown in Figure 1 (b). To organize the WAP trees in a meaningful way based on the corresponding timestamps, we introduce the concepts of *calendar schema* and *calendar pattern.*

Definition 3.2. **[Calendar Schema]** A *calendar schema* is defined as a relational schema R = ($G_n : D_n$, G_{n-1}: D_{n-1}, $\cdots$, $G_1 : D_1$) with a valid constraint. Each G_i is a granularity name such as *year, month, day, hours*, etc. Each domain D_i is a set of positive integers that specifies the valid values of each granularity. ■

For example, there are 12 months in a year, so there are only 12 integers in the domain corresponding to the granularity *month.* The constraint *valid* is a Boolean function on $D_n \times D_{n-1} \times \cdots \times D_1$. It specifies which combinations of the values in $D_n \times D_{n-1} \times \cdots \times D_1$ are *valid.* The constraints can be used to exclude the combinations that users are not interested in or those that have no corresponding time intervals. For instance, if we are not interested in weekends, then we can let the *valid* value for such days to be *False.*

Definition 3.3. **[Calendar Pattern]** Given a calendar schema R = ($G_n : D_n$, G_{n-1}: D_{n-1}, $\cdots$, $G_1 : D_1$), a *calendar pattern* on R is a tuple of the form $\langle$ d_n, d_{n-1}, $\cdots$, d_1 $\rangle$, where each d_i is either in D_i or a wildcard symbol '*'. ■

Suppose we have a calendar schema $\langle$ *year, month, day* $\rangle$. The calendar pattern $\langle$ **, 12, 24* $\rangle$ means "the 24th of December of every year"; while $\langle$ *2004, *, 24* $\rangle$ means "the 24th of every month in 2004". Intuitively, each calendar pattern represents the time intervals specified by the constraint.

Given a set of WAP trees with timestamps, it can be further partitioned into smaller groups according to the corresponding timestamp and a user-defined calendar pattern. Suppose we have a set of WAP trees as shown in Figure 1 (b). Given a *month-based* calendar pattern ⟨*, month, *⟩ with the calendar schema $\langle$ *day, month, year* $\rangle$, the WAP trees can be partitioned into three groups { T_{w1}, T_{w4}, T_{w6}}, { T_{w2}, T_{w5} }, and { T_{w3}, T_{w7} }. Similarly, given another calendar pattern $\langle$ interval, *,* $\rangle$ with the calendar schema $\langle$ *interval, day, month*$\rangle$, where a day is divided into three intervals *0:00-8:00, 8:00-16:00*, and *16:00-0:00*, the same set of WAP trees will be partitioned into the following groups { T_{w1}, T_{w2}, T_{w5}}, {T_{w3}, T_{w6} }, and {T_{w4}, T_{w7} }. Each group of WAP trees can be merged together and represented as a *WAP forest* denoted as F_w. The WAP forest is formally defined as following.

Definition 3.4. **[WAP Forest]** Let W = $\{T_{w1}, T_{w2}, \cdots, T_{wn}\}$ be a set of WAP trees in a WAP group. Let N_i and A_i be the sets of nodes and arches in T_{wi} respectively. Then a *WAP forest* is defined as F_w = (*N, A, r, v*), where $N = N_1 \cup N_2 \cdots \cup N_n$; $A = A_1 \cup A_2 \cdots \cup A_n$; r is a virtual root of the forest; and v is a function that maps each node in N to an integer. ■

A WAP forest is a compact structure that represents a group of WAP trees by merging them into one structure. The WAP forests not only record the access

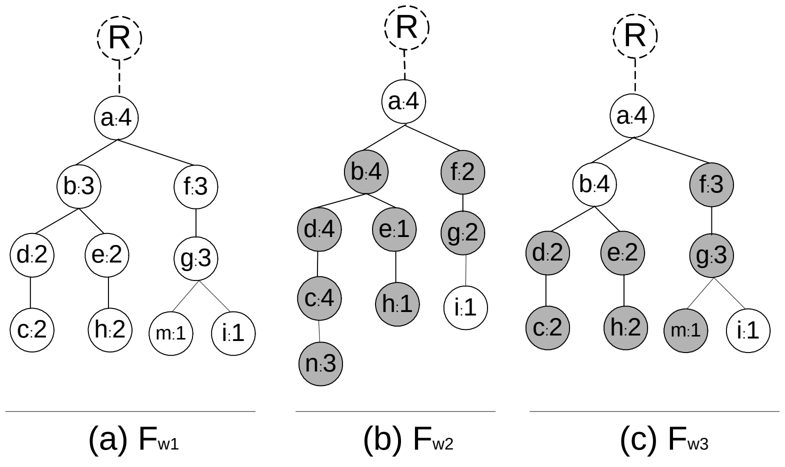


Fig. 2. WAP Forests.

patterns of the visitors but also how frequently the web pages were visited. The hierarchical structure of the forest represents the patterns of the pages being visited, while the integer value corresponding to each node represents the frequency of the web page being visited. The integer value is the *support count* of the corresponding node. Formally, the *support count* of a web page is defined as follows.

Definition 3.5. **[Support Count]** Let $\{T_{w1}, T_{w2}, \cdots, T_{wn}\}$ be a set of n WAP trees in WAP forest F_{wk}. Then the *support count* for a web page $P_i \in F_w$, denoted as *support(P_i)*, is defined as:

$$support(P_i) = \sum_{i=1}^{n} C_i, \ where\ C_i \ = \begin{cases} 1, & \text{if } P_i \in T_{wi} \\ 0, & \text{else} \end{cases}$$

■

The definition of support count is similar to the definition of support in transactional database [Agrawal et al. 1993]. Take the dataset in Table I (a) as an example. There are four WAP trees in the database. Two of the WAP trees contain the web page *d*. Consequently the support count for this web access pattern is 2. The support count for each web page is the *integer* recorded with the corresponding node in the WAP forest.

The construction of WAP forest is similar to the WAP-tree construction in [Pei et al. 2000]. The difference is that in the WAP-tree construction, each access session is modelled as a single path, while in the WAP forest construction each access session is taken as a tree structure. For example, for an access session $\langle$ *A, B, C, B, D, E, D, F* $\rangle$, in the WAP-trees construction it will be modelled as a single path starting from *A* to *F*. In our approach, this access pattern is modelled as an unordered tree as shown in Figure 1 (a). By doing this, the WAP forest represents the access patterns in a meaningful way such that the hierarchical relationship among web pages is preserved. Figure 2 shows the three WAP forests constructed based on the first three WAP groups in Table I. For instance, in Figure 2 (a), the integer "2" corresponding to node c means that the support count of this web page is 2 in the first group of WAP trees in Table I (a). We shall formally discuss the WAP forest construction algorithm in Section 4.

With the introduction of WAP forest, the set of WAP tree groups can be represented as a set of WAP forests. We order these WAP forests according to the corresponding timestamps to form a *WAP history* denoted as *H*. Formally, WAP history is defined as follows.

Definition 3.6. [**WAP History**] A WAP History is a sequence H= $\langle(t_1,\ F_{w1}),\ (t_2,\ F_{w2}),\ \cdots,\ (t_n,\ F_{wn})\rangle$, where F_{wi} is a WAP forest and t_i is a timestamp, for $i = 1 \cdots n$, and $t_i < t_{i+1}$ for $i = 1 \cdots n - 1$. The number of WAP forests in the WAP history H is defined as $|H|$, that is $|H| = n$. ■

Since the WAP forests in the WAP history are already ordered, here after, we represent the WAP history as $\langle F_{w1},\ F_{w2},\ \cdots,\ F_{wn}\rangle$ for simplicity. Based on this definition, the web log data is finally transformed into a WAP forest history according to the user defined calendar pattern.

3.2 FM-WAP Definition

Given a WAP history, as time goes by, some new web access patterns may be inserted; some old web access patterns may not exist any more; and for some others, their supports may change in the sequence. To measure the significance and frequency of the changes to the support of the WAPs, two metrics, *S-value* and *F-value*, are proposed. The *S-value* is defined to represent the significance of changes to the support values of a WAP, while the *F-value* is used to measure the frequency of changes to the support values of a WAP. Formally, they are defined as follows.

Definition 3.7. [**S-value**] Let F_{wi} and $F_{w(i+1)}$ be two WAP forests in the WAP history. Let P_k be a node in F_{wi}, P_{k+1} be the corresponding node of P_k in $F_{w(i+1)}$. The *S-value* of P_k from version i to i+1 is defined as

$$S_i(P_k) = \frac{|support(P_{k)}) - support(P_{k+1})|}{max(support(P_k),\ support(P_{k+1}))}$$

■

It can be observed that the *S-value* is between 0 and 1. A larger *S-value* implies a more significant change to the support count value. For example, consider the three WAP forests in Figure 2. The S-values for nodes b and f from the first WAP forest to the second WAP forest are 0.25 and 0.33 respectively. It is obvious that the change of support count to node f is more significantly than the change to node b. Note that the S-value is based on two consecutive WAP forests in the WAP history. Thus, for a WAP history with n forests, there is a sequence of *(n-1)* S-values for each web page.

Definition 3.8. [**Significant Change**] Let F_{wi} and $F_{w(i+1)}$ be two WAP forests in the WAP history. Let T_{wi} be a subtree of WAP forest F_{wi}, $T_{w(i+1)}$ be the corresponding WAP tree of T_{wi} in $F_{w(i+1)}$. Let α be the threshold for S-value.

Then it is a *significant change* for WAP tree T_{wi} from F_{wi} to $F_{w(i+1)}$ if $\forall\ P_i \in T_{wi}$, $S_i(P_i) \geq \alpha$. ■

This definition is based on the threshold of the S-value since in different domain the criterion of significant change can be different. Here the WAP tree can be a single web page or a set of connected web pages. Consider the previous example in Figure 2. If the value of the threshold α is set to 0.3, then web page f *changed significantly* from F_{w1} to F_{w2} since it has a S-value of 0.33. Similarly, we can say that the WAP tree rooted at node b *changed significantly* from F_{w1} to F_{w2} since every node in this subtree has a S-value no less than the S-value threshold (the S-values for nodes *b, d, c, n, e, h* are 0.33, 0.50, 0.50, 1, 0.50 and 0.50 respectively).

Definition 3.9. **[F-Value]** Let H be a WAP history where $|H| = n$ and T_{wi} is a WAP tree in the WAP forest F_{wj}. Let α be the threshold for the S-value, then the F-value for T_{wi} is defined as

$$F(T_{wi}, \alpha) = \frac{\sum_{i=1}^{n-1} f_i}{n-1}$$

$$where\ f_i = \begin{cases} 1, & \text{if it is a significant change for } T_{wi} \text{ from } F_{wi} \text{ to } F_{w(i+1)} \\ 0, & \text{else;} \end{cases}$$

■

The *F-value* for a WAP tree, T_{wi}, is defined to represent the percentage of times this WAP tree changed significantly against the total number of WAP forests in the sequence. Let us consider the WAP tree rooted at node b in Figure 2 as an example. From the previous calculation, this WAP tree changed significantly from F_{w1} to F_{w2}. Similarly, we found out that it did not change significantly from F_{w2} to F_{w3} since the S-value for node b from the second WAP forest to the third WAP forest is 0. According to the above formula, the F-value for this WAP tree rooted at node b is 0.5 (1/2). From this definition, it can also be observed that the *F-value* is between 0 and 1. It represents the frequency of significant changes to the WAP tree.

Based on the *S-value* and *F-value* metrics, *frequently mutating web access pattern*(FM-WAP) is defined as following.

Definition 3.10. **[FM-WAP]** Let H be a WAP history and $|H| = n$. Let α and β be the thresholds for the *S-value* and *F-value* respectively. Then a WAP tree $T_{wi} \subseteq F_{wm}$, where $1 \leq m \leq n$, is a **FM-WAP** if $F(T_{wi}, \alpha) \geq \beta$. ■

The FM-WAPs represent the access patterns that change significantly and frequently (specified by α and β) in the history. Consider the datasets in Table I. Some of the FM-WAPs are shown in Table II (b) when the thresholds for *S-value* and *F-value* are set to *0.2* and *0.25* respectively. In this table, we present the FM-WAPs as sequences of web pages, while in our final mining results the FM-WAPs are modelled as unordered tree structures. The FM-WAPs can be any subtree in the WAP forests. They can be rooted at any node and they may include one or more connected nodes. Some FM-WAP examples are shown in Figure 3. The FM-WAPs can be single nodes in the WAP forests such as node b and d in Figure 3 (a); they can be single paths such as ⟨ a, f, g ⟩ as shown in Figure 3 (b); also, they

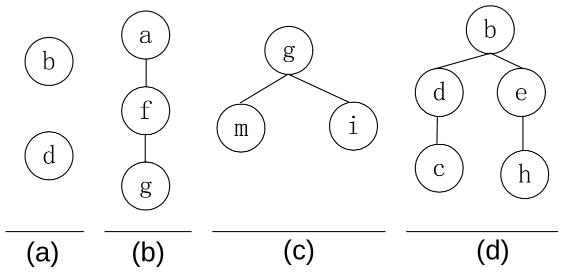


Fig. 3. FM-WAP examples.

can be subtrees such as ⟨ b, d, c, b, e ⟩ and ⟨ g, m, g, i ⟩ in Figures 3 (c) and (d) respectively.

Based on the above definitions of FM-WAPs, the problem of FM-WAP mining is defined as follows.

Definition 3.11. [**FM-WAP Mining**] Let M be a collection of web usage data, which is represented as a WAP history $H = \langle F_{w1}, F_{w1}, \cdots, F_{wn} \rangle$. Let α, β be the thresholds for S-value and F-value. The problem of FM-WAP mining is to discover the complete set of WAP trees M, where M= $\{T_{w1}, T_{w2}, \cdots, T_{wk}\}$ such that $T_{wi} \subseteq F_{wj}$ and F$(T_{wi}, \alpha) \geq \beta$, $\forall\ 0 < i \leq k$, and $0 < j \leq n$. ■

4. MINING FM-WAP

In this section, we present the algorithm for FM-WAP mining. The mining process consists of three phases, the *construction of WAP forest* phase, the *detecting and storing the changes* phase, and the *FM-WAP extraction* phase. The overview of the FM-WAP mining algorithm is shown in Figure 4. We elaborate on each phase in turn.

4.1 Phase 1: Construction of the WAP Forest

Given a collection of web log data, the set of WAPs can be identified by existing preprocessing techniques [Chen et al. 1998; Xiao and Dunham 2001]. For each WAP, there is a timestamp indicating when the visiting pattern is performed. First, the WAPs are transformed into WAP trees arranged in ascending order according to the timestamps. Next, according to the user-defined calendar pattern, these WAP trees are partitioned into WAP groups. For each WAP group, a WAP forest is constructed. Figure 5 shows the detail algorithm for building the WAP forest given a group of WAP trees. The WAP forest is initialized as the first WAP tree in the group with a virtual root node. Then, the next tree is compared with the existing WAP forest to merge them together as shown in Figure 5 (Lines from 3 to 17). For the rest of WAP trees in the group, their root nodes are compared with the current WAP forest using breadth first search. If the root nodes do not exist, the corresponding WAP trees are attached to the WAP forest as subtrees. Otherwise, the WAP trees are merged into the subtree that rooted at the node identical to the root of the WAP tree and the support counts for the affected nodes in the WAP forest are updated. This process iterates for all the WAP trees in the WAP group.

```
Input:
  A collection of web usage data: $W$
  A user defined calendar pattern: $cp$
  Threshold for S-value and F-value: $\alpha$ and $\beta$
Output:
  $F_w$: the complete set of FM-WAPs
Description:
 1: Construct the sequence of WAP forests $H$ from $W$ // Phase 1
        1.1 Represent the web access patterns as WAP trees
        1.2 Order the WAP trees according to the timestamps
        1.3 Partition them into groups according to the calendar pattern $cp$
        1.4 Merge the WAP trees within each group
        1.5 Output the WAP history $H$
 2: Detect and store the changes into the global forest $GF$ from $H$ // Phase 2
        2.1 Initialize the $GF$
        2.2 Detect the changes and merge them into $GF$
        2.3 Output the updated $GF$
 3: Extract the list of FM-WAPs $F_w$ from $GF$ // Phase 3
        3.1 Traverse the $GF$
        3.2 Compare the F-value and S-value against $\alpha$ and $\beta$
        3.3 Incorporate the FM-WAPs into the list $F_w$
        3.4 Output the final list of $F_w$
```

Fig. 4. Overview of FM-WAP Mining.

Figure 6 is an example of constructing a WAP forest from two WAP trees. Firstly, the WAP forest is initialized as the first WAP tree in the group with a virtual root node (Line 2) in Figure 5. After that, the second WAP tree is traversed (Lines 4 to 16 in Figure 5) and merged into the WAP forest. In this example, as the root of the second WAP tree exists in the WAP forest, this WAP tree is merged with the existing WAP forest and the support count of the root node a is increased by 1. Next, all the other nodes in this WAP tree are compared with the corresponding subtree in the WAP forest. For example, node b also exists in the corresponding subtree, its support count is increased by 1, while node e does not exist, this node is inserted into the WAP forest as a child node of b. The support count for node e is set to 1. This process iterates for all the other nodes and the final WAP forest is presented in Figure 6.

The WAP forest is proposed to record the statistics about the access patterns. Different from existing WAP mining approaches, we do not discard the WAPs that are infrequent. The frequent and infrequent WAPs are considered as equally important in our approach. This is because even infrequent WAPs may have certain patterns in their change histories, and such changes patterns are useful as mentioned in Section 1. The advantage of constructing a WAP forest is to store the shared prefix of the WAP trees only once so that the collection of WAP trees can be represented in a compact data structure. At the same time, by using the unordered tree representation, the hierarchical relationship among web pages are preserved. Since the WAP trees are unordered, the WAP forest is also unordered. However, to make the process of merging WAP trees more efficient, nodes in the WAP forest are ordered according to their support count values. In this paper, the order among siblings are in descending order with respect to their support count values.

```
Input:
  A set of WAP trees: { $T_{w1}, T_{w2}, \cdots, T_{wn}$ }
Output:
  $F_w$: the WAP forest
Description:
 1: Create a virtual root node for $F_w$
 2: Initialize $F_w$ as the first WAP tree in the group with a virtual root node.
 3: for all $i = 2$ to $n$ do
 4:   if the root of $T_{wi}$ does not exist in $F_w$ then
 5:     attach $T_{wi}$ as a subtree of $F_w$
 6:   else
 7:     //merge $T_{wi}$ to the subtree with the same root node
 8:     for all nodes $N_j$ in WAP tree $T_{wi}$ do
 9:       //traversed in BFS manner
10:       if $N_j$ exists in the current subtree of $F_w$ then
11:         increase the support count of $N_j$
12:       else
13:         create a new child node $N_j$ under the current node
14:       end if
15:     end for
16:   end if
17: end for
18: Return($F_w$)
```

Fig. 5. WAP Forest Construction.

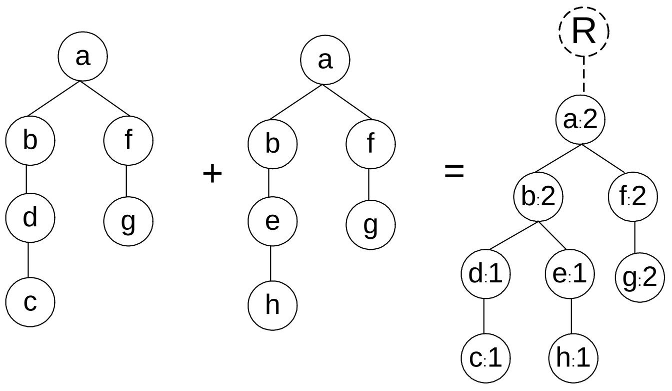


Fig. 6. Construction of the WAP Forest.

4.2 Phase 2: Detecting and Storing the Changes

Given the WAP history $H = \langle F_{w1}, F_{w2}, \cdots, F_{wn} \rangle$, the second step is to detect the changes between any two consecutive WAP forests in H and store them for FM-WAP extraction in the next phase. The changes in the forests can be obtained by modifying existing change detection techniques for tree structured data such as XyDiff [Cobena et al. 2002] and X-Diff [Wang et al. 2003]. Since we consider the web access patterns as unordered, we adapt the X-Diff approach that is proposed to detect changes for unordered XML documents.

Each WAP forest is considered as an XML document, where each web page is considered as an element and the corresponding support count is considered as an attribute. Rather than transforming the WAP forests into XML documents and then parse the XML documents, the WAP forests are modelled as trees that are similar to parsed XML documents. The change detection technique is then used to obtain the changes among them. In our case, there are only three types of changes: insertion of a new node, deletion of an old node, and updating the attribute value of a node. As X-Diff [Wang et al. 2003], a signature based change detection algorithm, detects all types of changes, we modify this algorithm to detect only the three types of changes we mentioned above. Moreover, the following two modifications are made. First, rather than keeping all the unchanged parts of the trees, only parts of the forest that have changed and their ancestor nodes are returned in the results. Second, we integrate that change detection process with the storing process. Rather than outputting the change results and parsing them again, the changes are directly stored into the *global forest* data structure. The global forest is defined as following.

A *global forest* is a 4-tuple $GF = (N, A, v, r)$, where N is a set of object identifiers; A is a set of labelled arcs (p, l, c) where $p, c \in N$ and l is the label of the arc; v is a function that maps each node $n \in N$ to a set of real values C_s, which represent the historical changes; r is a virtual node in N called *root*.

The global forest is a compact structure that records all the necessary statistics for FM-WAP mining. Since the FM-WAPs are defined based on the significance and frequency of the changes to the support count, we should be able to obtain the significance of the changes for each node and frequency of the changes from the global forest. There are different ways of representing the historical changes in the global forest.

4.2.1 *Naive Approach*

The naive approach is to record the sequence of S-values for each node in the global forest, from which the F-value can be obtained and hence the FM-WAPs can be extracted. In this approach, for each node there will be a sequence of *n-1* S-values in a WAP history H of n WAP forests. An example of the global forest constructed based on this naive approach is shown in Figure 8 (a). In the example that for each nodes in the global forest, a sequence of *n-1* S-values are stored. For instance, for node b, the sequence of S-values $\langle S_1(b), S_2(b), \cdots, S_{n-1}(b)\rangle$ are stored.

The advantage of this approach is that it is flexible since the sequence of values attached for the nodes are independent from the user defined parameters such as the thresholds for S-value and F-value. Such kind of representations and models are desirable in most data mining cases since the user defined parameters are subject to change with respect to the mining results. In this case, when the users are not satisfied with the mining results and want to adjust the parameters, there is no need to re-construct the global forest from scratch.

As we mentioned before, the global forest is a compact tree structure since the common prefixes for different changes to WAPs are represented only once in this structure. However, the space requirement for the global forest depends on not only the number of nodes in the global forest but also the size of the sequence of values attached to each node. In this approach, even the number of nodes in the

```
Input:
  A WAP history: $H$
  $\alpha$: the threshold of S-value
Output:
  $GF$: the global forest
Description: //This outline is the same for both the naive approach and the binary approach
 1: Initialize $GF$ as an empty forest with a virtual root
 2: Let current_node point to the root of $GF$
 3: for all i = 2 to $n$ do
 4:   Obtain the changes between two forests $F_{w(i-1)}$ and $F_{wi}$, denoted as $\triangle_i$
 5:   for all nodes $N_j$ in the delta $\triangle_i$ do
 6:     if the $S$ value for this node from $F_{wi}$ and $F_{w(i-1)}$ is no less than $\alpha$ then
 7:       if current_node has a child $N_j$ then
 8:         Update the value of $C_s$
            //this step is different for the naive approach and the binary approach
 9:       else
10:         Create a new child node $N_j$ and make the current_node point to $N_j$
11:       end if
12:     end if
13:   end for
14:   Let current_node point to the root of $GF$
15: end for
16: Return($GF$)
```

Fig. 7. Global Forest Construction.

global forest is much smaller than the total number of nodes in the WAP forests, the space requirement for the global forest can be huge. This is due to the fact that when the number of WAP forests in the history is very large, the size of the sequence of values corresponding to each node can be huge.

4.2.2 *Binary Approach*

To make the global forest compact, we proposed to use the binary approach to represent the historical changes. Given the threshold α for the *S-value*, C_s is a binary string that records the versions in the history where the support for the corresponding node has changed significantly. For instance, a binary string of '10011101' means that the support for this node has changed significantly in WAP forests *2, 5, 6, 7*, and *9* in the history.

Compared with the naive approach, the global forest in this approach will be smaller in terms of space requirement since the binary sequence is much smaller than the sequence of real values. However, this approach is not flexible since the values in C_s depend on the threshold for the S-value. Every time the users change the threshold for the S-value, the global forest will be re-constructed, while in the naive approach the global forest is independent from the threshold for the S-value. To sum up, the binary approach is more compact and suitable for very large dataset while the naive approach is more efficient for interactive data mining, where the thresholds change from time to time.

The algorithm for constructing the global forest is shown in Figure 7. First of all, the global forest, *GF*, is initialized as an empty forest with a virtual root node (Line 1). Next, for the second forest, the changes between the current forest and the

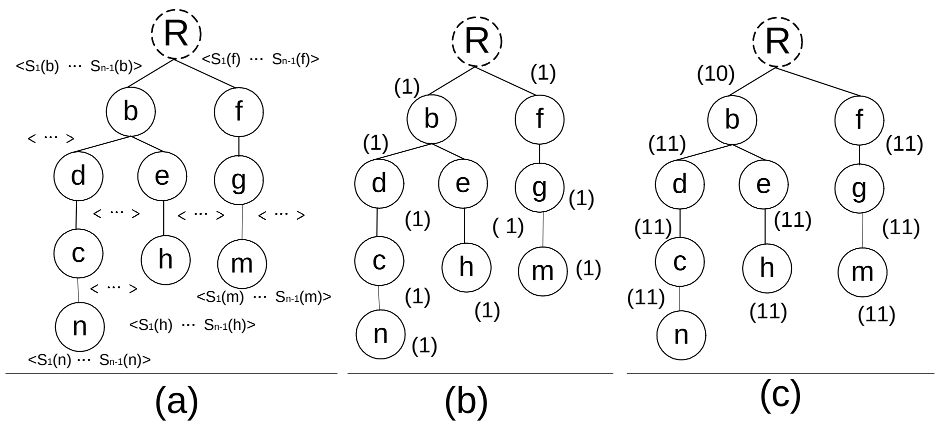


Fig. 8. Construction of the Global Forest.

previous forest are detected and incorporated into GF (Lines 3 to 15). This process iterates for all the WAP forests in the history and finally returns the updated forest GF. For the two approaches we mentioned above, the schema of construction the global forest is the same. The only difference is how to update the values in C_s. In the naive approach, it is straightforward. The new S-value for each nodes will be appended into C_s version by version. In the binary approach, the update process is guided by following rules.

- If the *S-value* of a node from F_{wi} to $F_{w(i+1)}$ is no less than the predefined threshold α, then
  1. If this node does not exist in the global forest, then the corresponding WAP tree in the change is inserted. The values of C_s are updated to the value of '000· · ·1' where there are (*i-1*) 0s in the string.
  2. If this node already exists in the global forest, the value of C_s is updated by setting the ith digit of the binary string as 1, and the empty digits before it are set to 0s. Continue the incorporating process for the remaining nodes in the changes.
- Else, skip this node and continue to check other nodes in the changes.

Based on the algorithm for global forest construction in Figure 7, we can conclude that the hierarchical structure and all the nodes in the global forest for both naive approach and binary approach are the same. The only difference the values in C_s. Since it is straightforward in the naive approach, where C_s is just a sequence of S-values, we illustrate the global forest construction process using the binary approach. An example of constructing the global forest based on the binary approach is shown in Figures 8 (b) and (c). This example is based on the three WAP forest in Figure 2. The changes between WAP forests are shown in Figure 2 with grey nodes. The changes are detected by our modified change detection technique. The first global forest is constructed based on the first two WAP forests in Figure 2, while the second global forest is constructed by incorporating the changes between the second and the third WAP forests in Figure 2. Suppose the threshold for *S*-

value is 0.2. The *S-value* can be calculated based on the supports and changes of the supports. Based on the *S-value*, it can be concluded that nodes a and i did not change significantly from the first WAP forest to the second WAP forest. After initialize the GF as a empty tree, all the nodes that change significantly from the first WAP forest to the second WAP forest are inserted into the GF with the corresponding value for C_s. For example, for node b and d the C_s value is "1", which represents that this node changed significantly from WAP forest 1 to WAP forest 2. The global forest that represents the changes from WAP forest 1 and WAP forest 2 is shown in Figure 8 (b). When the next WAP forest comes, the significant changes are detected and compared with the existing global forest. The value of C_s for the nodes in the global forest are updated according to the above rules. For nodes whose *S-value*s are less than the threshold, C_s value will be updated by appending a 0. This process iterates until there is no more WAP forest in the history. The final global forest constructed based on the three WAP forests in Figure 2 is the last global forest shown in Figure 8 (c).

4.3 Phase 3: FM-WAP Extraction

4.3.1 *Preliminaries*

Once the global forest is constructed, we proceed to the mining process where the FM-WAPs are extracted. A distinguishing character of this problem is that there is no heuristic relationship between the ancestor and descendant nodes in the WAP forest in terms of the changes to their support counts. Such characteristic makes our mining problem more challenging since the heuristic of tree structure mining, which is proved useful for pruning during the mining process [Xiao et al. 2003; Chi et al. 2004], cannot be used in our case. That is, the parent and child relationships have no effect to the individual nodes in terms of the changes to support counts. We cannot infer the changes to the parent nodes from the changes to the child nodes. For example, from the first WAP forest to the second WAP forest in Figure 8, it can be observed that nodes *b, f*, and g changed significantly, while their parent node (a, the parent node of node b) and child node (i, the child node of node g) did not change significantly. However, based on the definition of FM-WAP, there is a heuristic similar to the Apriori property in association rule mining [Agrawal et al. 1993]. For any FM-WAP T_{wi}, the *size* of the FM-WAP denoted as $|T_{wi}|$, is defined as the total number of nodes in this FM-WAP. A FM-WAP with the size value of n is called a *n-node FM-WAP*. For example, consider the FM-WAPs in Table II (b). The FM-WAP $\langle b, d, c\rangle$ is a *3-node FM-WAP* that consists of three *1-node FM-WAPs*: b, d, and c. *Hereafter, we call FM-WAPs with only one node as a 1-node FM-WAPs and FM-WAPs with more than one node as n-node FM-WAP.* From the definition of F-value, FM-WAPs, and the relationship between 1-node FM-WAPs and n-node FM-WAPs, we derived the following lemma.

Lemma 4.1. *Let T_{wi} be a n-node FM-WAP. Then $\forall\ P_j$ ($P_j \in T_{wi}, 1 \leq j \leq n$), P_j is a 1-node FM-WAP.*

Proof. Let T_{wi} be a *n-node FM-WAP*. According to the definition of FM-WAP and F-value, $F(T_{wi}, \alpha) = \frac{\sum_{i=1}^{n-1} f_i}{n-1} \geq \beta$. Here f_i is *1* if and only if $\forall\ P_j \in T_{wj}$, $S_i(P_j) \geq \alpha$. It implies that $\forall\ P_j \in T_{wj}$, $F(P_j, \alpha) \geq F(T_{wi}, \alpha) \geq \beta$. Consequently, P_j is a *1-node FM-WAP* according to the definitions. □

```
Input:
  $GF$: the global forest
  $\alpha, \beta$: the threshold of S-value and F-value
Output:
  $L$: the list of frequently changing web access patterns
Description:
 1: for all nodes $N_i$ in GF do
 2:   if $F(N_i, \alpha) < \beta$ then
 3:     mark all the links to this specific node as deleted
 4:     update GF
 5:   else
 6:     mark the node as 1-node FM-WAP; $L = L \cup ST$
 7:   end if
 8: end for
 9: for all 1-node FM-WAP $N_i$ in the updated GF do
10:   for all nodes connected with $N_i$ do
11:     if the node is evolutionarily connected with $N_i$ then
12:       a link between the node and $N_i$ will be created/preserved in the global forest
13:     else
14:       mark the link as deleted if there exists one between the node and $N_i$ in the global forest
15:     end if
16:   end for
17: end for
18: for all 1-node FM-WAP $N_i$ with a degree value greater than 1 in the GF do
19:   choose the 1-node FM-WAP with the largest degree value $i$
20:   if there are $i$ evolutionarily connected nodes then
21:     form a substructure $St$
22:     if F($St$) $\geq \beta$ then
23:       $L = L \cup ST$
24:     end if
25:   end if
26: end for
27: Return($L$)
```

Fig. 9. FM-WAP Extraction.

The algorithm of FM-WAP extraction is shown in Figure 9. The idea is to use the bottom-up traversal strategy to check the properties of all the possible FM-WAPs. There are two steps. The first step is to extract all the *1-node FM-WAPs* and prune the global forest. After that, the *n-node FM-WAPs* are extracted based on the pruned global forest by checking the potential combinations of 1-node FM-WAPs. This mining strategy is based on LEMMA 4.1.

4.3.2 *1-node FM-WAP Generation*

In this phase, all the 1-node FM-WAPs are generated by directly calculating the S-values and F-values and comparing them against the user defined threshold. For each node in the global forest, the *F-value* can be calculated as $\frac{n}{m}$, where $|H| = m$ and n is the number of times the corresponding structure has changed significantly (the corresponding S-value is no less than the threshold). If the F-value is no less than the threshold, then, the corresponding node is a 1-node FM-WAP and returned as FM-WAPs. Otherwise, this node is not a 1-node FM-WAP, all the links to its parent and children will be marked as deleted. This process iterates in

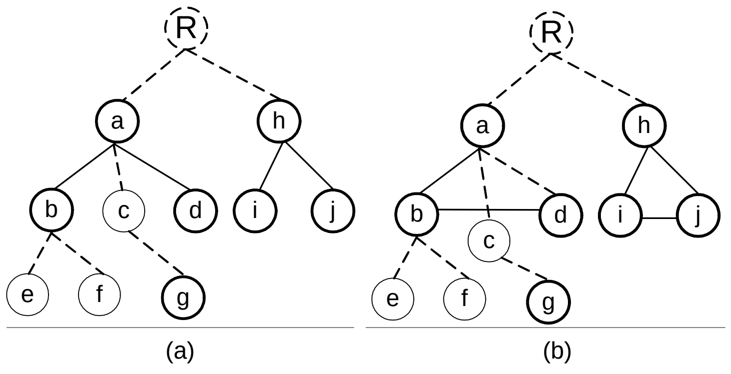


Fig. 10. FM-WAP Extraction Example.

a bottom-up manner for all the nodes until it reaches the root of the global forest.

At the end of this phase, we have a list of 1-node FM-WAPs and a modified global forest. An example of the updated global forest after the first step is shown in Figure 10(a). In this fictitious example, there are 7 1-node FM-WAPs, which are shown as bold nodes. Here since node c is not a 1-node FM-WAP, all the links to this node is marked as deleted (indicated by the dotted links).

4.3.3 *n-node FM-WAP Generation*

In this phase, the n-node FM-WAPs are generated based on the modified global forest generated in the first phase. Firstly, we define some useful terms that will be used to extract the n-node FM-WAPs.

Definition 4.2. [**Evolutionarily Connected Pair**] Let T_{wp}, T_{wq} be two 1-node FM-WAPs in a WAP history H, where $|H| = n$ and $0 \leq p, q \leq n$. Then $\{T_{wp}, T_{wq}\}$ is an *evolutionarily connected pair* if

$$\frac{\sum_{i=1}^{n-1} E_i}{n-1} \geq \beta,\ E_i = \begin{cases} 1, \text{ if } S_i(T_{wp}) \geq \alpha \text{ and } S_i(T_{wq}) \geq \alpha\ ; \\ 0, \text{ else;} \end{cases}$$

where n is the total number of versions and β is the threshold for F-value. ■

Based on Definitions 3.9, 3.10, and 4.2, the following lemmas can be derived.

LEMMA 4.3. *Let T_{wi} be a n-node FM-WAP. Then $\forall$ $\{P_m, P_n\}$ $(P_m, P_n \in T_{wi})$, $\{P_m,\ P_n\}$ is a evolutionarily connected pair.*

PROOF. According to the definition of FM-WAP, $F(T_{wi}, \alpha) = \frac{\sum_{i=1}^{n-1} f_i}{n-1} \geq \beta$. Here f_i is *1* if and only if $\forall$ $P_j \in T_{wj}$, $S_i(P_j) \geq \alpha$. That is, $\forall$ $P_m \in T_{wi}, P_n \in T_{wi}$, $S_i(P_m) \geq \alpha$ and $S_i(P_n) \geq \alpha$. Since $F(T_{wi}, \alpha) \geq \beta$, according to the definition of *evolutionarily connected* P_m and P_n is an evolutionarily connected pair. The heuristic is that if all of nodes in a WAP tree change together significantly, any two of them will surely change together significantly. □

LEMMA 4.4. *Let T_{wi} be a n-node FM-WAP. Then $\forall$ P_j $(P_j \in T_{wi})$, $\exists$ (n-1) other 1-node FM-WAPs that are evolutionarily connected to P_j.*

PROOF. As we mentioned before, a n-node FM-WAP consists of n 1-node FM-WAPs. According to the Definition 3.9 and LEMMA 4.3, all the 1-node FM-WAPs in the n-node FM-WAP should be evolutionarily connected to each other. Consequently, each 1-node FM-WAP should be evolutionarily connected to at least *n-1* other 1-node FM-WAPs. □

To extract all possible n-node FM-WAPs from the modified global forest, based on the above lemmas, we need to check the number of evolutionarily connected 1-node FM-WAPs for all the existing 1-node FM-WAPs. In this approach, we focus on finding connected n-node FM-WAPs, which consists of multiple 1-node FM-WAPs that are connected in the global forest. Here, a 1-node FM-WAP is connected to another 1-node FM-WAP if there exists a path between them and none of the edge has been marked at deleted. For example, 1-node FM-WAPs a and b, b and d are connected in Figure 10 (a).

The first step is to check all pairs of connected 1-node FM-WAPs in the modified global forest. If they are evolutionarily connected then a link between them is preserved (for direct parent and child 1-node FM-WAPs) or created (for 1-node FM-WAPs that are not directly connected in the global forest). To avoid repeatedly checking the evolutionarily connected property for the same pair of 1-node FM-WAPs, the modified global forest is taken as an ordered tree. For each 1-node FM-WAP, the evolutionarily connected properties to all the connected nodes that have larger values of pre-order number are calculated. For example, consider the global forest in Figure 10 (a). We will check the evolutionarily connected status for 1-node FM-WAP pairs $\{a,\ b\ \}$, $\{a,\ d\ \}$, $\{b,\ d\ \}$ but not for 1-node FM-WAP pairs $\{b,\ a\ \}$, $\{d,\ a\ \}$ *and* $\{d,\ b\}$. Suppose $\{a,\ b\ \}$ and $\{b,\ d\ \}$ are evolutionarily connected, $\{a,\ d\ \}$ is not evolutionarily connected. Then the link between a and d will be marked as deleted. At the same time, the links between $a,\ b$ and $b,\ d$ are preserved and created respectively. This process iterates for all the 1-node FM-WAPs. From Figure 10 (a), this phase will output the global forest as shown in Figure 10 (b).

For each 1-node FM-WAP, there is a set of 1-node FM-WAPs that are evolutionarily connected to it. We call this number as *evolutionary degree*. For example, in Figure 10 (b) the *evolutionary degree* for 1-node FM-WAP h is 2. Based on the *evolutionary degree* values, all the n-node FM-WAPs can be extracted. We process the 1-node FM-WAPs in descending order with respect to their *evolutionary degree* value. Based on LEMMA 4.4, it can be inferred that for each 1-node FM-WAP in a n-node FM-WAP will have an *evolutionary degree* value no less than *n-1* and there will be at least n such nodes. Based on these criteria, the first n-node FM-WAP is extracted and the process of extract such n-node FM-WAPs iterates for all the rest of the 1-node FM-WAPs. Consider the example in Figure 10, the n-node FM-WAP will be the 3-node FM-WAP $\langle\ h,\ i,\ h,\ j\ \rangle$.

Finally, the output of this extraction phase will include all the 1-node FM-WAPs generated in the first sub-phase and all the n-node FM-WAPs generated in the second sub-phase.

THEOREM 4.5. ***[Completeness of the Algorithm]*** *The FM-WAP mining approach can generate all the valid FM-WAPs.*

Proof. Let f be any valid FM-WAP in a sequence of n WAP forests with the thresholds for S-value and F-value are α and β. We prove that our FM-WAP mining approach can discover this WAP as a valid FM-WAP in two cases. Case 1: f is a 1-node FM-WAP; case 2: f is a n-node FM-WAP.

Case 1: When f is a 1-node FM-WAP, according to the definition of F-value and FM-WAP, the corresponding F-value should be no less than β and this FM-WAP can be easily generated by our algorithm as shown in Lines 2-7 in Figure 9.

Case 2: When f is a n-node FM-WAP, according to Lemma 4.1, all the nodes in this substructure should be 1-node FM-WAPs. Therefore, all the nodes in f should be included in the updated global forest after the 1-node FM-WAPs generating process. This is why our algorithm generates all the n-node FM-WAPs based on the 1-node FM-WAPs. Also based on Lemma 4.3, all the 1-node FM-WAPs in f should be evolutionarily connected, which should satisfy the evolutionarily connected status checking as shown in Lines 9-17 in Figure 9. Finally, the F-value of f is checked against the threshold β. According to the definition of FM-WAP, f is returned as a valid FM-WAP.

Thus, for any FM-WAP f, no matter it is a 1-node FM-WAP or n-node FM-WAP, our FM-WAP mining approach can discover it as a valid FM-WAP. The theorem is proved. □

4.4 Algorithm Analysis

In this section, we analyze the time complexity and space complexity of FM-WAP mining algorithm. In phase 1, the web log data that is represented as WAP trees are partitioned into n groups based on the calendar pattern. For each group, a WAP forest is constructed. Suppose the average number of nodes in the WAP trees is i and each group includes j WAP trees. The time complexity for this phase will be $O\ (n \times j \times i^2)$. In phase 2, the global forest construction and change detection are integrated. The average number of nodes in the WAP forest is $i \times j \times c$, where c is a constant. The time complexity for this phase is $O((n\text{-}1)\{i \times j\}^2 \times max\{deg(F_{wi}), deg(F_{wj})\} \times \log_2 max\{deg(F_{wi}), deg(F_{wj})\})$, where F_{wi} and F_{wj} are any two of the WAP forests in the history, according to the analysis in [Wang et al. 2003]. In phase 3, the global forest is traversed to extract all the FM-WAPs. The corresponding time complexity is $O(n \times i^2 \times j \times \log i)$. From the above analysis it is obvious that the dominant cost of FM-WAP mining is the cost of phase 2. We shall validate this experimentally in Section 5.

From the three mining phases, it is obvious that the space complexity of FM-WAP mining depends on the size of the global forest, which is the largest data structure. Based on the algorithms, we observed that the compactness of the global forest depends on the overlaps among the changes between WAP forests. However, the upper bound of the number of nodes in the global forest is $O(i \times j \times (n-1))$. The actual size of global forest is significantly smaller than the total number of nodes in the WAP trees as we shall see in the next section.

4.5 Summary

We proposed a global forest data structure to record the historical changes of WAPs and an efficient mining algorithm. It can extract all FM-WAPs with one scan over the original log data and one scan of the global forest. Our approach has following

Dataset	N	M	F	D	T
D25	100	10000	20	25	100000
F15	500	10000	15	25	500000
T2M	500	10000	10	25	1000000
T4M	1000	20000	30	40	10000000

Table III. Synthetic datasets.

advantages. We do not check all the combinations of the existing FM-WAPs since the cost is exponentially large. Our mining process is guided by the global forest. This forest is actually a pruned structure representation of the corresponding WAPs. Another advantage is that there is no need to scan the original data to get the support for the combinations of FM-WAPs. Their supports can be easily calculated by using the *AND* operation on all their descendant nodes.

5. PERFORMANCE EVALUATION

We performed four sets of experiments to evaluate the performance of our proposed FM-WAP mining algorithm. All experiments were conducted on a P4 1.7 GHz PC with 512Mb main memory running Windows 2000 professional. The algorithm is implemented in Java based on the two types of global forest we proposed. In this section, detail of the datasets and experimental results will be discussed respectively.

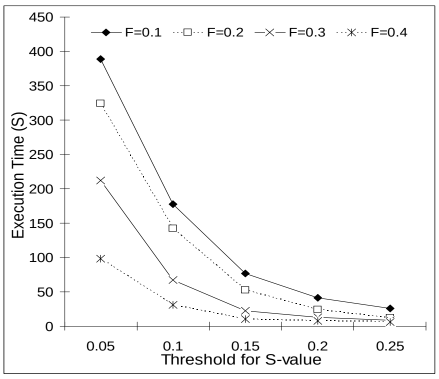
5.1 Datasets

Both synthetic and real web log data are used in the experiments. The real data is the web log *UoS* obtained from the Internet Traffic Archive [http://ita.ee.lbl.gov/]. It records the historical visiting patterns for University of Saskatchewan from June 1, 1995 to December 31, 1995, a total of 214 days. In this seven month period there were 2,408,625 requests. Timestamps have 1 second resolution. There are 2,981 unique URLs. After preprocessing the size of the dataset is 222MB. Using the calendar pattern $\langle$ *, week, * $\rangle$, it is transformed into a sequence of 30 WAP forests. The synthetic data set is generated using the synthetic tree generation program used in [Zaki 2002]. The characteristics of the synthetic data we used are shown in Table III. The program first constructs a master web site browsing tree, W, based on parameters supplied by the user. These parameters include the maximum fan out of a node (denoted as F), the maximum depth of the tree (denoted as D), the number of nodes in each tree (denoted as M), the total number of nodes in the sequence of tree (denoted as T), and the number of node labels (denoted as N). For the basic synthetic datasets, there are 30 WAP forests in the history.

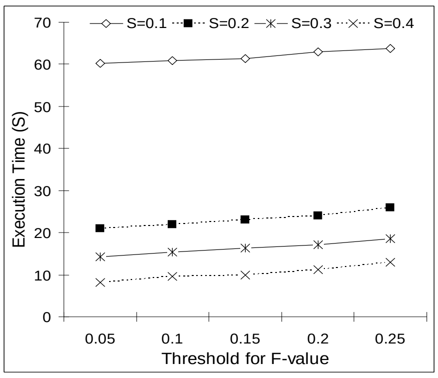
5.2 Time Efficiency Evaluation

In this section, we evaluate the time efficiency of FM-WAP algorithm by varying the two thresholds for *S-value* and *F-value.* Figures 11 (a) and (b) show the execution time of our algorithm with different thresholds of *S-value* and *F-value.* The real *UoS* web log dataset is used in the first two sets of experiments. In our experimental results, we find out that the time efficiency of the algorithm is the same for both the naive approach and the binary approach. Hence, for experimental results in this section, the running time is for both approaches if not specified.

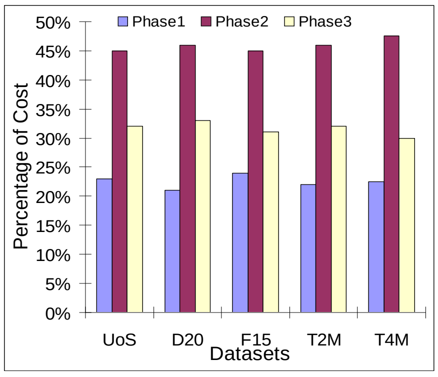
From Figure 11 (a) it can be observed that as the threshold of *S-value* increases the execution time decreases. This is due to the fact that the number of nodes



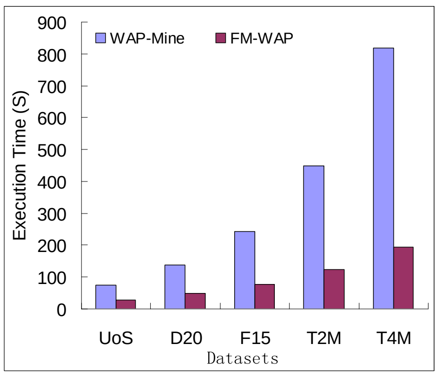
(a) Variation of S-value threshold α



(b) Variation of F-value threshold β



(c) Percentage of Execution Time



(d) WAP-Mine vs FM-WAP

Fig. 11. Experiment results [Efficiency].

that have changed significantly decreases as the threshold of *S-value* increases. Consequently, the size of the global forest will be smaller and the FM-WAP mining process will be faster. It also shows that with the same *S-value*, when the threshold for *F-value* is large, the execution time is smaller.

However, changes to the threshold for *F-value* does not affect the execution time much with a fixed threshold for *S-value* as shown in Figure 11 (b). This is because there is no closure property in the global forest and all the possible structures in the forest should be tested against the threshold of *F-value*. The threshold of *F-value* only affects the deleting process that is relatively cheap. Notice that there is a huge gap between the first two curves in this figure. This is because of the characteristic of the dataset we used (the S-values for most WAPs are less than 0.2).

Figure 11 (c) shows the cost of each phase in the FM-WAP mining process. The cost for partitioning the web log data is incorporated into the cost of constructing the WAP forest. It can be observed that the process of detecting and storing the changes in the global forest is the most expensive process. This is due to the fact the change detection for unordered trees is a time-consuming problem [Wang et al. 2003]. The process of FM-WAP extraction is very efficient. Experimental results show that the ratio of cost for each phase is the same for both the naive approach and binary approach based algorithms.

Figure 11 (d) shows the execution time of FM-WAP mining and repeated WAP-mine [Pei et al. 2000] for the five different datasets. The WAP-mine algorithm used in this paper is download from http://www.cs.ualberta.ca/∼tszhu/software/wap.zip. Our literature survey shows that most citations use this version of implementation [Gunduz and Ozsu 2003]. Here, the thresholds for the *S-value* and *F-value* are set to 0.2, the minimal support threshold for frequent WAP is set to 0.2. Note that, the cost for WAP-Mine does not include the cost of post-processing as mentioned in Section 1. Even in this case, from this figure it is obvious that the FM-WAP mining is more than three times faster than WAP-Mine. This is because of large size of the tree structure generated in the WAP-Mine algorithm and expensive cost of constructing the conditional WAP-tree. Considering the post-processing cost, we can conclude that the FM-WAP cannot be efficiently extracted by using existing frequent WAP mining approaches.

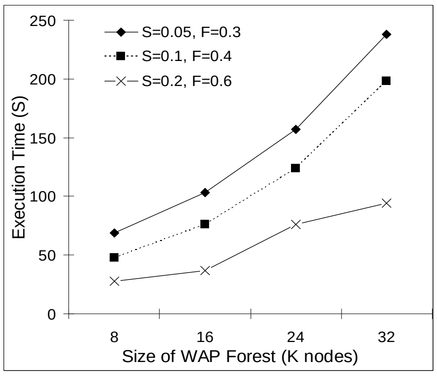
5.3 Scalability Evaluation

The scalability of FM-WAP mining is evaluated by varying the characteristics of the datasets such as the *size of each WAP tree*, the *number of WAP forests* in the history, the *percentage of changes* between WAP forests, etc. In this set of experiments, synthetic dataset *T4M* is used. The first three set of results are for both the naive approach based and the binary approach based global forest, where the running time for both approaches is the same. The last set of results is the comparison of the naive approach and the binary approach in terms of the memory usage.

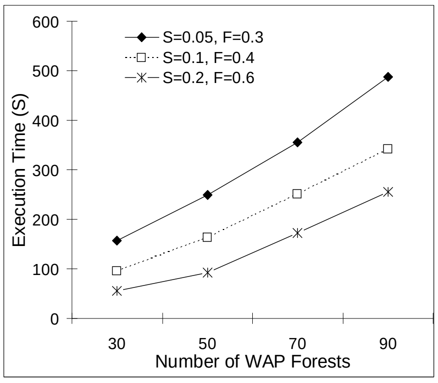
Figure 12 (a) shows the execution time of our algorithm as the size of the WAP forest increases. The percentage of changes between two consecutive WAP forests in this dataset is set to 10%. 30 WAP forests are used in this experiment. It can be observed that as the size of the WAP forest increases, the execution time also increases. This is because the size of the global forest increases as the size of the WAP forests increase. Moreover, as the size of the WAP forest increases, the cost for detecting the changes between WAP forests increases as well. However, the overall execution time increases in a near log manner in our experiments.

Figure 12 (b) shows the execution time of our algorithm as the number of WAP forests in the history increases. The percentage of changes between two consecutive WAP forests in this dataset is set to 10%. The average number of nodes in the WAP forests is 9000. It can be observed that as the number of WAP forests in the history increases, the execution time also increases. The reason behind is the same as the previous experiment of varying the size of the WAP trees.

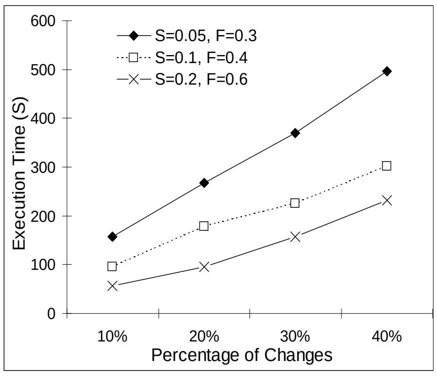
Figure 12 (c) shows the execution time of our algorithm as the percentage of changes between WAP forests increases. The average number of nodes in the WAP forest is 9000, and there are 30 WAP forests in each dataset. It can be observed that as the percentage of changes increases, the execution time also increases. This is



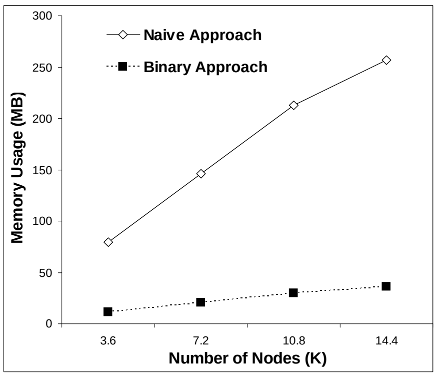
(a) Variation of WAP Forest Size



(b) Number of WAP Forest



(c) Variation of Perc. of Change

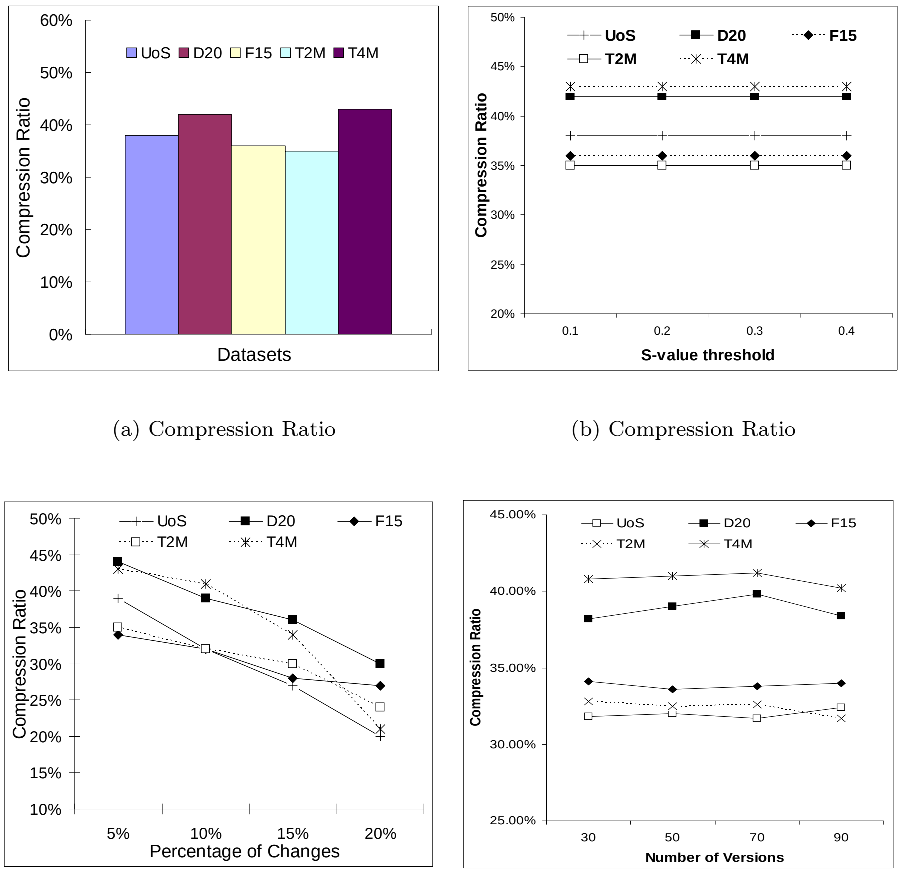


(d) Memory Usage

Fig. 12. Experiment results [Scalability].

because the size of the global forest increases as the percentage of changes increases.

Figure 12 (d) shows how the memory usage of the global forest changes as the average number of nodes in the each WAP forest increases. In this dataset, there are 80 WAP forests in the history and the percentage of changes among them is 10%. Here the memory usage depends on both the compactness of the global forest structure and the size of the values affiliated to each node. We compare the memory usage of the global forest for two different approaches, the naive approach and the binary approach. From the global forest construction algorithm in the previous section, we can conclude that the total number of nodes in the global forest is the same for both approach, the difference is the size of the sequence of numbers that are used to represent the change history for each node. From the experimental results, it is obvious that the memory usage for the binary approach is much smaller



(a) Compression Ratio (b) Compression Ratio

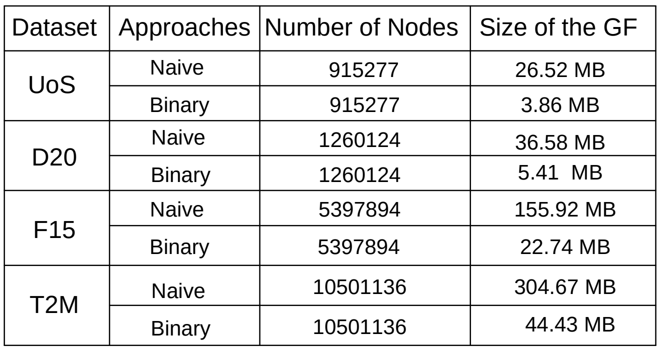
(c) Compression Ratio (d) Compression Ratio

Fig. 13. Experiment results [Compression].

than the naive approach. In most cases, the naive approach has a memory usage more than 6 times larger than the binary approach. In the implementation of the naive approach, we use the *shortint* to represent the sequences of S-values. By doing so the memory usage is reduced, but, as a tradeoff this approach will only have a precision of 0.01 in terms of the S-value and the threshold. We believe this will not significantly affect the final result. In the binary approach, there is no approximation in terms of the S-value and the threshold.

5.4 Compression Efficiency Evaluation

In this section, we evaluate the compression efficiency of our proposed global forest by varying the thresholds for *S-value* and the percentage of changes between WAP forests. The synthetic datasets in Table III and the real *UoS* dataset are used in the following experiments. In the first three set of experiments, we evaluate the



Dataset	Approaches	Number of Nodes	Size of the GF
UoS	Naive	915277	26.52 MB
	Binary	915277	3.86 MB
D20	Naive	1260124	36.58 MB
	Binary	1260124	5.41 MB
F15	Naive	5397894	155.92 MB
	Binary	5397894	22.74 MB
T2M	Naive	10501136	304.67 MB
	Binary	10501136	44.43 MB

Fig. 14. Comparison of Naive and Binary Approaches.

compression efficiency use the *compression ratio* measure. The *compression ratio*, denoted as R, is defined as $\frac{c}{o}$, where c and o are the number of nodes in the global forest and the total number of nodes in the original web log data. As we can see from the global forest construction algorithm in Figure 7, the number of nodes in the global forest for both the naive and binary approach are the same. Thus, the compression ratios for both approaches are the same.

Figure 13 (a) shows the compression ratio of our proposed global forest. For the five datasets used in our experiments, in which the percentage of changes is 10% and the threshold for S-value is 0.1, the size of the global forest is only less than half the size of the original forest sequences. This is due to the fact that the overlapping part is represented only once in the global forest. It shows that our global forest is a compact data structure for storing historical changes to web access patterns.

Figure 13 (b) shows how the compression ratio of global forest changes when the threshold of *S-value* changes. The percentage of changes is set to 10% and there are 30 versions of WAP forest in the history. The size of the global forest does not change at all as the threshold value of *S-value* increases. This is because the global forest records all the nodes that have changed in the WAP history, no matter how significant the changes are. As a result, the compression ratio for the global forest is irrelevant with the threshold for the *S-value*.

From Figure 13 (c), we can observe that the compression ratio of global forest changes when the percentage of changes between WAP forests changes. For this experiment, the threshold for the *S-value* is set to 0.2. The size of the global forest increases as the percentage of changes increases. This is because when the percentage of changes increases, the number of WAP trees that change significantly will increase. As a result the global forest will be larger.

Figure 13 (d) shows how the compression ratio changes as the total number of WAP forests in the history increases. For this experiment, the percentage of changes is fixed to 10%. We can observe that the compression ratio of global forest does not change much when the total number of WAP forests in the history changes. It is because that when the number of WAP forests in the sequence increases to a certain value, the global forest will cover almost all the possible nodes in the WAP forest that are likely to change. After that, when the number of WAP forest increases, it does not change the structure of the global forest too much. Notice that the compression ratio shown in this figure is the ratio with respect to the number of nodes.

Figure 14 is a table that shows the comparison of the naive approach and the binary approach for the four different datasets. In this set of experiments, the percentage of changes is fixed to 10% and there are 30 versions of WAP forests in the history. The results in this table verified that the number of nodes in the global forests for both the naive and binary approaches are the same, while the actual size of the global forest are different. It can also be observed that the size of storage for each node, which can be derived from the number of nodes and the size of the global forest, in the same approach is the same for different datasets. Moreover, the ratio of the size of the storage for the naive approach against the size of the storage for the binary approach is the same for all the datasets (in the above experimental results, it is around 6.8). It is because the historical information is stored as binary sequence in the binary approach, while the historical information is stored as short integer sequence in the naive approach.

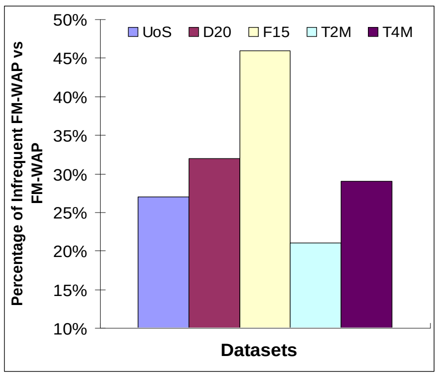
From the above experimental results about efficiency, scalability, and compression, we can summarize the difference between the naive approach and the binary approach as following. The advantage of the naive approach is that there is no need to re-construct the global forest even if the users change the threshold for the S-value. However, the global forest constructed based on this approach requires a lot of memory usage. For the binary approach, it has much less requirement for the memory usage. However, it is necessary to re-construct the global forest every time the threshold for S-value changes. As we can see from Figures 11 (c) and 12 (d), the time for re-constructing the global forest in both approaches is almost 1/3 of the total mining cost and the memory requirement for the naive approach is almost 6 times larger than the binary approach. In real life, users can choose the appropriate approach based on different applications and requirements.

5.5 Novelty of Knowledge Discovered

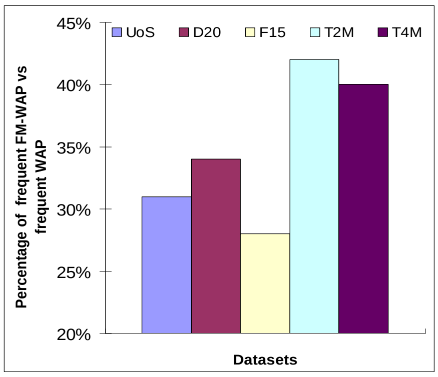
As mentioned earlier, not all the frequent WAPs are FM-WAPs and not all the FM-WAPs are frequent WAPs either. In this section, we evaluate the novelty of the knowledge discovered by FM-WAP mining. We have done experiments to show the relation between frequent WAPs and FM-WAPs. The synthetic dataset in Table III and the real dataset *UoS* are used in the following experiments.

Figure 13 (e) shows the percentage of FM-WAPs that are not frequent WAPs. The frequent WAPs are generated by using the WAP-mine algorithm in [Pei et al. 2000] to the five datasets. The minimal support threshold for frequent WAP is 10%. The threshold for *S-value* and *F-value* are 0.2 and 0.5 respectively. This is used to demonstrate that not all the FM-WAPs are frequent WAPs.

Our experimental results also show that not all the frequent WAPs are FM-WAPs. Figure 13 (f) shows the percentage of frequent WAPs that are FM-WAPs. The minimal support threshold for frequent WAP is 10%, while the threshold for *S-value* and *F-value* are 0.1 and 0.4 respectively. This is used to demonstrate that not all frequent WAPs are meaningful and useful in terms of their historical change patterns.



(a) Infrequent FM-WAP



(b) Non-Freq FM-WAP

Fig. 15. Experiment results [Novelty].

5.6 Summary

From the above experimental results, we can conclude that our proposed FM-WAP mining algorithm is efficient. The mining cost grows in a near log manner when the total number of nodes in the dataset increases. It is more efficient compared with the approach of re-applying existing frequent WAP mining algorithms and post-processing the mining results. The experiments also show that our proposed data structure *global forest* is compact for storing historical changes of WAPs and the FM-WAP mining algorithm is scalable. By comparing the mining results of existing frequent WAP mining algorithms with our FM-WAP mining results, it can be observed that our proposed FM-WAPs are novel knowledge that cannot be discovered by existing frequent WAP mining and post-processing techniques efficiently and completely.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel approach to extract hidden knowledge from the history of changes to WAPs. We proposed a global forest data structure to represent and store the historical changes to WAPs. Using this structure, our algorithm can discover the FM-WAPs by only two scans over the web log data. Experiments based on both synthetic and real datasets shown that our algorithm is efficient and scalable, and it can produce novel knowledge that cannot be discovered even by post-processing the results of existing frequent WAP mining algorithms. We have also shown how such novel knowledge can be useful in different applications.

Our future work will be identifying specific change patterns such as increasing, decreasing, or periodic changes from the set of FM-WAPs. Also, we want to cluster different parts of a web site based on the change patterns to WAPs and explore the semantic meanings of such clusters for business intelligence.

REFERENCES

AGRAWAL, R., IMIELI;SKI, T., AND SWAMI, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. Washington, D.C., 207–216.

BARON, S. AND SPILIOPOULOU, M. 2003. Monitoring the evolution of web usage patterns. In *Proceedings of European Web Mining Forum, Workshop at European Conference on Principles and Practice of Knowledge Discovery in Databases.*

BARON, S., SPILIOPOULOU, M., AND GNTHER, O. 2003. Efficient monitoring of patterns in data mining environments. In *Proceedings of the 7th East-European Symposium on Advances in Databases and Information Systems.* Springer-Verlag, 253–265.

CHEN, M.-S., PARK, J. S., AND YU, P. S. 1998. Efficient data mining for path traversal patterns. *IEEE Transaction on Knowledge and Data Engineering 10,* 2 (April), 209–221.

CHI, Y., MUNTZ, R., XIA, Y., AND YANG, Y. 2004. CMTreeMiner: Mining both closed and maximal frequent subtrees. In *Proceedings of the Pacific-Asia conference on Knowledge Discovery and Data mining.* 63–73.

COBENA, G., ABITEBOUL, S., AND MARIAN, A. 2002. Detecting changes in XML documents. In *Proceedings of the 18th International Conference on Data Engineering.* 41–52.

COOLEY, R., MOBASHER, B., AND SRIVASTAVA, J. 1997. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of International Conference on Tools with Artificial Intelligence.* 558–567.

EIRINAKI, M. AND VAZIRGIANNIS, M. 2003. Web mining for web personalization. *ACM Transactions on Internet Technology 3,* 1, 1–27.

FU, Y., SANDHU, K., AND SHIH, M.-Y. 1999. Clustering of web users based on access patterns. In *ACM KDD International Workshop on Web Usage Analysis and User Profiling (WEBKDD'99).*

GANTI, V., GEHRKE, J., AND RAMAKRISHNAN, R. 1999. A framework for measuring changes in data characteristics. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM Press, 126–137.

GANTI, V. AND RAMAKRISHNAN, R. 2002. Mining and monitoring evolving data. In *Handbook of massive data sets.* Kluwer Academic Publishers, 593–642.

GUNDUZ, S. AND OZSU, M. T. 2003. A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, 535–540.

HTTP://ITA.EE.LBL.GOV/. Internet traffic archive.

LIU, B., HSU, W., AND MA, Y. 2001. Discovering the set of fundamental rule changes. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM Press, 335–340.

PEI, J., HAN, J., MORTAZAVI-ASL, B., AND ZHU, H. 2000. Mining access patterns efficiently from web logs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining.* 396–407.

SPILIOPOULOU, M. 2000. Web usage mining for web site evaluation. *Communication of ACM 43,* 8, 127–134.

SRIVASTAVA, J., COOLEY, R., DESHPANDE, M., AND TAN, P.-N. 2000. Web usage mining: Discovery and applications of usage patterns from web data. *Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining 1,* 2, 12–23.

WANG, Y., DEWITT, D. J., AND CAI, J. Y. 2003. X-Diff: An effective change detection algorithm for XML documents. In *Proceedings of the IEEE International Conference of Data Engineering.* 519–530.

XIAO, Y. AND DUNHAM, M. H. 2001. Efficient mining of traversal patterns. *Data and Knowledge Engineering 39,* 2, 191–214.

XIAO, Y., YAO, J.-F., LI, Z., AND DUNHAM, M. H. 2003. Efficient data mining for maximal frequent subtrees. In *Proceedings of the IEEE International Conference on Data Mining.* 379–386.

Zaki, M. J. 2002. Efficiently mining frequent trees in a forest. In *Proceedings of Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 71–80.

Zhao, Q. and Bhowmick, S. S. 2004. Mining history of changes to web access patterns. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*. Pisa, Italy.

Zhao, Q., Bhowmick, S. S., Mohania, M., and Kambayashi, Y. 2004. FCS mining: Discovering frequently changing structures from historical XML structural deltas. In *Proceedings of ACM CIKM International Conference on Information and Knowledge Management.*