WAM-Miner: In the Search of Web Access Motifs from Historical Web Log Data

Qiankun Zhao[1] Sourav S Bhowmick[1] Le Gruenwald[2]
[1]Nanyang Technological University, Singapore. {pg04327224, assourav}@ntu.edu.sg
[2]University of Oklahoma, Norman, USA. ggruenwald@ou.edu

ABSTRACT

Existing web usage mining techniques focus only on discovering knowledge based on the statistical measures obtained from the *static* characteristics of web usage data. They do not consider the dynamic nature of web usage data. In this paper, we focus on discovering novel knowledge by analyzing the *change patterns* of historical web access sequence data. We present an algorithm called WAM-MINER to discover *Web Access Motifs* (WAMs). WAMs are web access patterns that never change or do not change *significantly* most of the time (if not always) in terms of their support values during a specific time period. WAMs are useful for many applications, such as intelligent web advertisement, web site restructuring, business intelligence, and intelligent web caching.

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

General Terms: Algorithm, Design, Experimentation.

Keywords: Web Access Motif, Dynamic Pattern, Web Usage Mining.

1. INTRODUCTION

Web Usage Mining (WUM) – the application of data mining techniques to discover usage patterns from web data – has been an active area of research and commercialization [19]. Existing web usage data mining techniques include statistical analysis [19], association rules [11], clustering [15, 12], classification [13], sequential patterns [18], and dependency modeling [10]. Often, such mining provides insight that helps optimizing the website for increased customer loyalty and e-business effectiveness. Applications of web usage mining are widespread, ranging from usage characterization, web site performance improvement, personalization, adaptive site modification, to market intelligence.

Generally, the web usage mining process can be considered as a three-phase process, which consists of *data preparation*, *pattern discovery*, and *pattern analysis* [19]. Since the last phase is application-dependent, let us briefly describe the first two phases. In the first phase, the web log data are transformed into sequences of events (called *Web Access Sequences* ($\mathcal{WAS}$s)) based on the identification of users and the corresponding timestamps. For example, given a web log archive that records the navigation history of a web site, by using some existing preprocessing techniques [6, 21], the raw log data can be transformed into a set of $\mathcal{WAS}$s. Table 1 shows an example of such $\mathcal{WAS}$s. Here S_ID represents a sequence id and a $\mathcal{WAS}$ such as $\langle a, b, d, c, a, f, g\rangle$ denotes a visiting sequence from web page a to pages b, d, c, a, f and finally to page g. Each sub-table in Table 1 records the collection of $\mathcal{WAS}$s for a particular month. In the second phase, statistical methods and/or data mining techniques are applied to extract interesting patterns such as *Web Access Patterns* (WAPs)[18]. A WAP is a sequential pattern in a large set of $\mathcal{WAS}$s, which is visited frequently by users [18]. That is, given a support threshold ξ and a set of $\mathcal{WAS}$s (denoted as $\mathcal{A}$), a sequence W is a WAP if W appears as a *subsequence*[1] in at least $\xi \times |\mathcal{A}|$ web access sequences of $\mathcal{A}$. For clarity, in this paper we call such a WAP a *frequent* WAP. Consequently, a sequence that appears in fewer than $\xi \times |\mathcal{A}|$ web access sequences of $\mathcal{A}$ is called an *infrequent* WAP. These patterns are stored for further analysis in the third phase.

(a) The first month

S_ID	$\mathcal{WAS}$s
1	$\langle a,b,d,c,a,f,g\rangle$
2	$\langle a,b,e,h,a,f,g\rangle$
3	$\langle e,f,g,i,n\rangle$
4	$\langle b,d,c,a,e\rangle$

(b) The second month

S_ID	$\mathcal{WAS}$s
1	$\langle a,b,d,c,a,f,g\rangle$
2	$\langle b,d,c,x\rangle$
3	$\langle e,f,g,i,n\rangle$
4	$\langle b,e,h,b,d,c,n,f,g\rangle$

(c) The third month

S_ID	$\mathcal{WAS}$s
1	$\langle b,d,e,a,f,g\rangle$
2	$\langle b,e,h,b,d,c\rangle$
3	$\langle e,f,g,i,n\rangle$
4	$\langle e,f,g,i,n\rangle$

(d) The fourth month

S_ID	$\mathcal{WAS}$s
1	$\langle b,d,e,a,f,g\rangle$
2	$\langle e,f,g,i,n\rangle$
3	$\langle a,b,e,c,f,g\rangle$
4	$\langle e,f,g,i,n\rangle$

Table 1: Example of $\mathcal{WAS}$s

1.1 Motivation

From Table 1, it is evident that web usage data is dynamic in nature. For instance, the $\mathcal{WAS}$ $\langle$ *b, d, e, a, f, g* $\rangle$ did not exist in the first and second months but appeared in the third and fourth months. Similarly, the $\mathcal{WAS}$ $\langle$ *a, b, d, c, a, f, g* $\rangle$ occurred in the first and the second months but disappeared after that. The $\mathcal{WAS}$ $\langle$ *e, f, g, i, n* $\rangle$ became increasingly popular as it occurs only once in the first three

[1]If there are two $\mathcal{WAS}$s $A_1 = \langle B, E, A\rangle$ and $A_2 = \langle A, B, C, E, A\rangle$, then A_1 is a subsequence of A_2.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

months, but in the fourth month, it occurs twice. Note that the above dynamic behaviors of $\mathcal{WAS}$s can be attributed to various factors, such as changes to the content of the web site, users' familiarity to the web site structure, arrival of new web visitors, and effects of sudden occurrences of important real life events.

Such dynamic nature of web usage data poses both challenges and opportunities to the web usage mining community. Existing web usage mining techniques focus only on discovering knowledge based on the statistical measures obtained from the *static* characteristics of web usage data. They do not consider the dynamic nature of web usage data. In particular, the dynamic nature of $\mathcal{WAS}$ data leads to the following two challenging problems.

1. **Maintenance of WUM results:** Take the $\mathcal{WAS}$s in Table 1 as an example. The knowledge discovered (e.g., frequent WAPs) in the first month using existing techniques will not include the $\mathcal{WAS}$s, the timestamps of which are in the second month and beyond. Hence, the mining results of existing techniques have to be updated constantly as $\mathcal{WAS}$ data changes. This requires development of efficient incremental web usage mining techniques.
2. **Discovering novel knowledge:** Historical collection of $\mathcal{WAS}$ data contains rich temporal information. While knowledge extracted from snapshot $\mathcal{WAS}$ data is important and useful, interesting and novel knowledge describing temporal behaviors of $\mathcal{WAS}$s can be discovered based on their historical *change patterns.* Note that in this paper, the term *change patterns* of a $\mathcal{WAS}$ (or WAP) indicates the change to the *popularity* of the $\mathcal{WAS}$ (or WAP) in the historical $\mathcal{WAS}$ database. The *popularity* is measured by *support.* Traditionally, *support* has been defined as the percentage of times a sequence occurred in a data collection [2]. In our context, *support* represents the percentage of times a $\mathcal{WAS}$ (or WAP) occurred in a given collection of $\mathcal{WAS}$s (called a $\mathcal{WAS}$ *group*). *Hereafter, changes to the* $\mathcal{WAS}$*s (or WAPs) refer to the changes to the support of the* $\mathcal{WAS}$*s (or WAPs).*

In this paper, we focus on discovering novel knowledge by analyzing the change patterns of historical $\mathcal{WAS}$ data. Different types of novel knowledge can be discovered by mining the history of changes to $\mathcal{WAS}$s. Particularly, in this paper we focus on discovering *Web Access Motifs*[2] (WAMs). WAMs are WAPs that never change or do not change *significantly* most of the time (if not always) in terms of their support values during a specific time period. For example, consider Figure 1, which depicts the support values (y-axis) of four WAPs (denoted as W_1, W_2, W_3, and W_4) from time period t_1 to t_5 (x-axis). Note that t_i in the x-axis represents a time period (e.g., day, week, month etc.) and not a particular time point. The support values of W_1 do not change significantly (varying between *0.7* and *0.8*), hence, W_1 can be considered as a WAM. Similarly, most of the support values of W_2 hover around 0.1 (except for that at t_4), therefore, it can be considered as a WAM. However, the supports of W_3 and W_4 change significantly (e.g., support of W_3 changed from 0.8 to 0.4 during the transition from t_1 to t_2) and, thus,

[2]The term "motif" is inspired by the notion of *motifs* in biology. *Motifs* in biology are certain patterns in DNA or protein sequences that are strongly conserved by evolution. Note that strongly conserved does not mean completely conserved.

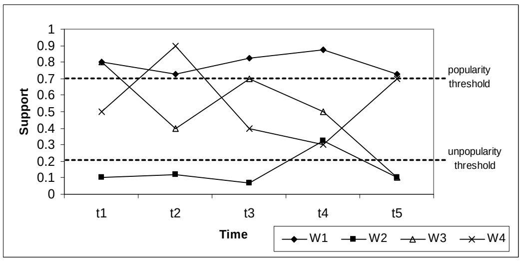

Figure 1: Support of WAPs over a time period

these two WAPs are not WAMs. As we shall see later, the degree of changes is measured using a metric called *conservation rate*. WAMs are useful for many applications such as intelligent web advertisement, web site restructuring, business intelligence, and intelligent web caching (discussed in Section 2).

We present techniques to discover two types of web access motifs: *popular* and *unpopular* WAMs. Given a *popular support threshold* α, a WAM is considered *popular* if most of the time its support is greater than α during a specific time period. For example, reconsider Figure 1. Let $\alpha = 0.7$. Then, W_1 is a popular WAM as its support value is always greater than or equal to 0.7 during the time period of t_1 to t_5. A popular WAM represents a sequence of pages that are consistently popular to the website's visitors over a duration of time. Similarly, given an *unpopular support threshold* β, a WAM is considered *unpopular* if most of the time it has low support (i.e., its support values are less than β) during a specific time period. An unpopular WAM thus represents a sequence of web pages that are rarely accessed by web users over a time period. For example, consider W_2 in Figure 1. Let $\beta = 0.2$, then most of the support values of W_2 are less than 0.2 during the period of t_1 to t_5 (except at t_4); hence, W_2 is an unpopular WAM. As discussed in Section 2, both popular and unpopular WAMs can be beneficial to many applications.

At the first glance, it may seem that the above types of WAMs can be discovered by postprocessing the results of existing frequent WAP mining techniques [18, 21]. However, to the best of our knowledge, the complete set of WAMs cannot be efficiently discovered by those existing techniques for the following reasons (even if we apply them repeatedly to a sequence of snapshot data):

- First, existing techniques focus either on snapshot data [6, 18, 21] or on detecting the changes to the mining results [4, 7, 8]. None of these techniques considers the issue of *directly mining* the *change patterns* of WAPs from the original data set to discover novel knowledge (e.g., popular and unpopular WAMs).
- Second, as we shall see in Section 6, the process of repeatedly mining frequent WAPs at different time points and post-processing the mining results to discover WAMs is expensive and may not discover the complete set of popular WAMs. To extract popular WAMs, it is not necessary to mine frequent WAPs since WAMs are different from frequent WAPs. WAMs are based on the changes to the support counts of the access patterns over a specific time period. WAPs, on the other hand, is based on the overall support counts of the access patterns at a particular timepoint.

ID	WAMs
W1	< a, b, d, a, f >
W2	< g, m, f, k >
W3	< g, m, c, e, a>

(i) Popular WAMs

ID	WAMs
W4	< g, m, t, u, x >
W5	< t, u, b, d, u >

(ii) Unpopular WAMs

(a) Time t1

ID	WAMs
W1	< a, b, d, a, f >
W2	< g, m, f, k >
W6	< a, b, e, h, f >

(i) Popular WAMs

ID	WAMs
W4	< g, m, t, u, x >
W3	< g, m, c, e, a >
W5	< t, u, b, d, u >

(ii) Unpopular WAMs

(b) Time t2

Figure 2: WAM examples

- Third, the frequent WAP mining process only discovers frequent WAPs [18] or maximal contiguous sequences (MCS) [21]. However, a WAM can be either a frequent (popular) or infrequent (unpopular) WAP. Consequently, unpopular WAMs cannot be discovered using these techniques.

1.2 Contributions

In summary, the major contributions of this paper are as follows.

- We introduce an approach that, to the best of our knowledge, is the first one to discover popular and unpopular Web Access Motifs (WAMs) from the sequence of historical changes to web access patterns. We show with illustrative examples that WAMs are useful for many real life applications.
- We present a technique to represent changes to Web Access Patterns (WAPs) in term of their support counts. We also propose two metrics called *conservation rate* and *support range* to quantitatively measure the *significance* of changes to support counts of WAPs.
- We propose an efficient algorithm called WAM-MINER for discovering popular and unpopular WAMs based on the above metrics.
- We present the results of extensive experiments with both synthetic and real datasets that we have conducted to demonstrate the efficiency and scalability of our algorithm. We also conduct experiments to determine the *quality* of our results.

The rest of this paper is organized as follows. In Section 2, we present some representative applications of WAMs. In Section 3, we describe the problem formally and illustrate it using an example. Section 4 introduces a model to represent the changes to WAPs and metrics used to detect WAMs. In Section 5, the WAM mining algorithm is described. Section 6 presents the experimental results. Section 7 reviews the related works. Finally, the last section concludes this paper.

2. APPLICATIONS OF WAMS

Knowledge of WAMs can be useful in many applications, such as web advertisement, web site restructuring, business intelligence, and web caching. We now elaborate on some of these applications.

Intelligent Web Advertisement: It has been claimed that 99% of all web sites offer standard banner advertisements [5], underlying the importance of this form of online advertising. For many web-based organizations, revenue from advertisements is often the only or the major source of income (e.g., Yahoo.com, Google.com) [3]. The most commonly used pricing schemes employed in banner advertisements is the *cost-per-thousand impressions* (CPM) model where the cost is associated with the amount of exposure of the advertisement. Several sites also use the *cost-per-click* (CPC) model, where the advertiser pays the publisher each time the advertisement banner is clicked on. These two models indicate that one of the ways to maximize revenues for the party who owns the advertising space is to design intelligent techniques for the selection of an appropriate set of advertisements to display in appropriate web pages.

Consequently, there have been several recent research efforts on scheduling banner advertisements on the web [3]. Selection of banner advertisements is currently driven by the nature of the banner advertisement, Internet knowledge of the target market, relevance of the web page contents, and popularity of the web pages [3, 9]. *However, none of these techniques consider the evolution of web access patterns for the advertisement selection problem.* In particular, WAMs can be useful for designing more intelligent advertisement placement strategies. Let us illustrate this with a simple example. Consider the popular WAMs in Figure 2 as extracted by our WAM-MINER algorithm at times t_1 and t_2 where $t_2 > t_1$. Observe that W_1 and W_2 remained as popular WAMs at t_1 and t_2. This indicates that the sequences of web pages in W_1 and W_2 consistently received a large number of visitors during the specified time period from t_1 to t_2 and are expected to continue this trend in the near future. Hence, it makes sense to put relevant banner advertisements on these pages in order to maximize revenues. Note that our approach can easily be integrated with any existing advertisement selection techniques and does not call for any drastic change to the existing frameworks.

Web site restructuring: It is well known that ill-structured design of web sites prevents the users from rapidly accessing the target pages. Recently web usage mining techniques have been successfully used as a key solution to this issue [15]. However, none of these techniques exploits the evolving nature of WAPs to restructure web sites. Results of WAM mining can be used by web site administrators to restructure their web sites according to the historical access characteristics of web site visitors. Let us illustrate the usefulness of WAMs in this context with an example. Reconsider the WAMs in Figure 2. The following information can be gleaned which can be used to restructure web sites.

- Consider the WAMs, W_2 and W_4. Both of these WAMs share the same prefix (pages g and m). However, W_2 is a popular WAM whereas W_4 is an unpopular WAM during the specified time period from t_1 to t_2. Hence, web site administrators may reorganize the pages in W_4 in order to improve the number of visitors to these pages.
- The WAM W_3 was discovered as a popular WAM at time t_1 but became unpopular to web visitors after t_1. This may be due to various reasons such as changes to the web content, currency of the information in the web pages, poorly structured information, and presence of banner(s) that the consumers perceive to be out of place with the web pages. Web site administrators can further investigate the reasons behind this phenomenon and restructure the site if necessary.
- Observe that W_5 consistently remained unpopular to web visitors. This may be due to various reasons. One of them is that web pages are not easily reachable from pages in popular WAMs. Hence, web site administrators may restructure the web site in a way so that

pages in W_5 are in close vicinity of pages in popular WAMs.

Intelligent web caching: Web caching has been used by many business organizations to reduce the time that their customers must wait for their web search results. One of the most difficult issues in web caching is to identify which web pages to cache. The discovery of WAMs provides a solution to this problem. For example, pages in the popular WAMs W_1, W_2, W_3, and W_6 in Figure 2 can be cached for future access because their support counts are large and are not expected to change.

3. PROBLEM STATEMENT

In general, web log data can be considered as sequences of web pages with *session identifiers* [21]. Formally, let $P = \{p_1, p_2, \ldots, p_m\}$ be a set of web pages. A *session* S is an ordered list of pages accessed by a user, i.e., $S = \langle (p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n)\rangle$, where $p_i \in P$, t_i is the time when the page p_i is accessed and $t_i \leq t_{i+1}\ \forall\ i = 1, 2, 3, \ldots, n-1$. Each session is associated with a unique identifier, called session ID. A *web access sequence* ($\mathcal{WAS}$), denoted as A, is a sequence of consecutive pages in a session. That is, $A = \langle p_1, p_2, p_3, \ldots, p_n\rangle$ where n is called the *length* of the $\mathcal{WAS}$. Note that it is not necessary that $p_i \neq p_j$ for $i \neq j$ in a $\mathcal{WAS}$. This is because a web page may occur more than once in a session due to backward traversals or reloads [21].

The access sequence $W = \langle p'_1, p'_2, p'_3, \ldots, p'_m\rangle$ is called a *web access pattern* (WAP) of a $\mathcal{WAS}$ $A = \langle p_1, p_2, p_3, \ldots, p_n\rangle$, denoted as $W \subseteq A$, if and only if there exist $1 \leq i_1 \leq i_2 \leq \ldots \leq i_m \leq n$ such that $p'_j = p_{i_j}$ for $1 \leq j \leq m$.

A $\mathcal{WAS}$ *group* (denoted as G) is a bag of $\mathcal{WAS}$s that occurred during a specific time period. Let t_s and t_e be the start and end times of a period. Then, $G = [A_1, A_2, \ldots, A_k]$ where p_i is included in $\mathcal{WAS}$ A_j for $1 < j \leq k$ and p_i was visited between t_e and t_s. The *size* of G, denoted as $|G|$, reflects the number of $\mathcal{WAS}$s in G. Note that, it is possible $A_i = A_j$ for $i \neq j$ in a bag of $\mathcal{WAS}$s. For instance, we can partition the set of $\mathcal{WAS}$s on a daily, weekly or monthly basis, where the timestamps for all the $\mathcal{WAS}$s in a specific $\mathcal{WAS}$ group are within a day, a week, or a month. Consider the $\mathcal{WAS}$s in Table 1 as an example. They can be partitioned into four $\mathcal{WAS}$ groups on a monthly basis, where $\mathcal{WAS}$s whose timestamps are in the same month are partitioned into the same $\mathcal{WAS}$ group.

Given a $\mathcal{WAS}$ group G, the *support* of a $\mathcal{WAS}$ A in G is $\Phi_G(A) = \frac{|\{A_i | A \subseteq A_i\}|}{|G|}$. When the $\mathcal{WAS}$ group is obvious from the context, the support is denoted as $\Phi(A)$. Similarly, when the $\mathcal{WAS}$ is obvious from the context, the support is denoted as Φ.

In our investigation, the historical web log data is divided into a sequence of $\mathcal{WAS}$ groups. Let $H_G = \langle\ G_1, G_2, G_3, \ldots, G_k\ \rangle$ be a sequence of k $\mathcal{WAS}$ groups generated from the historical web log data. Given a WAP W, let $H_W = \langle\ \Phi_1(W), \Phi_2(W), \Phi_3(W), \ldots, \Phi_k(W)\ \rangle$ be the sequence of support values of W in H_G. Then, *maximum popularity support* of W (denoted as M_W) is defined as $M_W = \Phi_i$ where $\Phi_i \geq \Phi_j\ \forall\ 0 \leq j \leq k$ and $i \neq j$. Similarly, *minimum unpopularity support* of W (denoted as U_W) is Φ_r where $\Phi_r \leq \Phi_j\ \forall\ 0 \leq j \leq k$ and $r \neq j$. The pair (M_W, U_W) is called the *support range* of W (denoted as $\mathcal{R} = (M_W, U_W)$). Furthermore, the *conservation rate* of W is denoted as $\mathcal{C}_W = F(H_W)$ where F is a function (defined later in Section 4) that returns the *rate of change* of support values of W in H_W and $0 \leq \mathcal{C}_W < 1$.

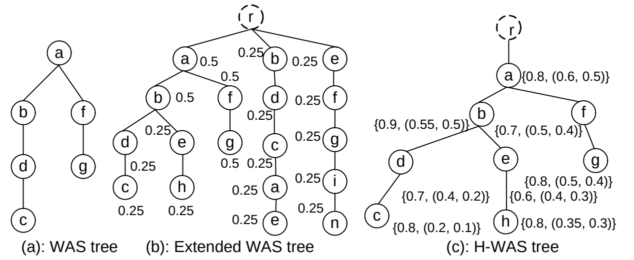

Figure 3: Examples

Given the *popularity threshold* α and a *conservation threshold* μ, a WAP W is a *popular WAM* if and only if $\forall\ W' \subseteq W$, $C_{W'} \geq \mu$ and $M_{W'} \geq \alpha$. Similarly, given the *unpopularity threshold* β, a WAP W is a *unpopular WAM* if and only if $\forall\ W' \subseteq W$, $C_{W'} \geq \mu$ and $U_{W'} \leq \beta$. Our objective of WAM mining is to find all popular and unpopular WAMs in the historical web log data given some popularity and unpopularity thresholds, and conservation threshold.

4. MODELING HISTORICAL $\mathcal{WAS}$

In this section, the problem of how to model the historical $\mathcal{WAS}$s and measure their change patterns is discussed. We begin by discussing how a $\mathcal{WAS}$ Group is represented followed by the representation of $\mathcal{WAS}$ group history. Finally, we discuss the statistical summarization technique for the $\mathcal{WAS}$ group history.

4.1 Representation of $\mathcal{WAS}$ Group

Given a $\mathcal{WAS}$ denoted as $A = \langle p_1, p_2, p_3, \ldots, p_n\rangle$, in the literature, there are various ways to represent the relationship among web pages in the sequence [18, 22]. In [18], a $\mathcal{WAS}$ is represented as a flat sequence, while in [22] a $\mathcal{WAS}$ is represented as an unordered tree, which was claimed more informative with the hierarchical structure. In this paper, we adopt the unordered tree representation of $\mathcal{WAS}$. A $\mathcal{WAS}$ tree is defined as $T_A = (r, N, E)$, where r is the root of the tree that represents web page p_1; N is the set of nodes where $V = \{p_1, p_2, \cdots, p_n\}$; and E is the set of edges in the maximal forward sequences of A. An example of $\mathcal{WAS}$ tree is shown in Figure 3 (a), which corresponds to the first $\mathcal{WAS}$ shown in Table 1 (a).

As a result, a $\mathcal{WAS}$ group consists of a bag of $\mathcal{WAS}$ trees. Here, all occurrences of the same $\mathcal{WAS}$ within a $\mathcal{WAS}$ group are considered identical. Then the $\mathcal{WAS}$ group can also be represented as an unordered tree by merging the $\mathcal{WAS}$ trees. We propose an *extended* $\mathcal{WAS}$ *tree* to record the aggregated support information about the bag of $\mathcal{WAS}$s within a $\mathcal{WAS}$ group. The *extended* $\mathcal{WAS}$ *tree* is defined as follows.

DEFINITION 1. **[Extended $\mathcal{WAS}$ Tree]** *Let $G = [A_1, A_2, \ldots, A_k]$ be a bag of $\mathcal{WAS}$s, where each $\mathcal{WAS}$ A_i, $1 \leq i \leq k$, is represented as a tree $T_{A_i} = (r_i, N_i, E_i)$. Then, the extended $\mathcal{WAS}$ is defined as $T_G = (r, N, E, \Theta)$, where $N = N_1 \cup N_j \cdots \cup N_k$; $E = E_1 \cup E_j \cdots \cup E_k$; r is a virtual root; and Θ is a function that maps each node in N to the support of the corresponding $\mathcal{WAS}$.* □

Consider the first $\mathcal{WAS}$ group in Table 1. The corresponding *extended* $\mathcal{WAS}$ *tree* is shown Figure 3 (b), where the value associated with each node is the Θ value. It can be observed that the common prefix for different $\mathcal{WAS}$ trees is presented only once in the extended $\mathcal{WAS}$ tree. For example, the common prefix of $\langle$ *a, b, d, c, a, f, g* $\rangle$ and $\langle$ *a, b, e, h, a, f, g* $\rangle$ is $\langle$ *a, b, a, f, g* $\rangle$, which is presented once in the extended $\mathcal{WAS}$ tree. Details of how to construct the extended $\mathcal{WAS}$ tree will be discussed in Section 5.

4.2 Representation of $\mathcal{WAS}$ Group History

The simplistic method of representing $\mathcal{WAS}$ Group History is to merge the sequence of extended $\mathcal{WAS}$ trees together to form an historical $\mathcal{WAS}$ tree (called H-$\mathcal{WAS}$ tree) in a similar way as we have merged the $\mathcal{WAS}$ trees in the previous section. However, the H-$\mathcal{WAS}$ tree and extended $\mathcal{WAS}$ tree are different in several aspects. Firstly, all occurrences of the same $\mathcal{WAS}$ tree in one $\mathcal{WAS}$ group are considered to be equal, while occurrences of the same extended $\mathcal{WAS}$ tree in a sequence of $\mathcal{WAS}$ groups may have different support values. Secondly, the order of extended $\mathcal{WAS}$ trees is important in the construction of the H-$\mathcal{WAS}$ tree, while the order of $\mathcal{WAS}$ trees is not important in the construction of the extended $\mathcal{WAS}$ tree. Moreover, the purpose of the extended $\mathcal{WAS}$ tree is to record the support values in a specific $\mathcal{WAS}$ group, while the purpose of the H-$\mathcal{WAS}$ tree is to record the history of support values of the $\mathcal{WAS}$s. As a result, the historical support values in the H-$\mathcal{WAS}$ tree are represented as a time series, where the i^{th} element represents the support values of the $\mathcal{WAS}$ in the i^{th} $\mathcal{WAS}$ group.

Definition 2. **[H-$\mathcal{WAS}$ Tree]** *Let $H_G = \langle G_1, G_2, G_3, \ldots, G_k \rangle$ be a sequence of k $\mathcal{WAS}$ groups, where each $\mathcal{WAS}$ group G_i, $1 \leq i \leq k$, is represented as an extended $\mathcal{WAS}$ tree, $T_{G_i} = (r_i, N_i, E_i, \Theta)$. Then, the H-$\mathcal{WAS}$ tree is defined as $H_G = (r, N, E, \Lambda)$, where r is a virtual root; $N = N_1 \cup N_j \cdots \cup N_k$; $E = E_1 \cup E_j \cdots \cup E_k$; and Λ is a function that maps each node in N to the sequence of historical support values of the corresponding $\mathcal{WAS}$.* □

Note that, in the H-$\mathcal{WAS}$ tree, there is a sequence of support values for each node; while there is only one support value for each node in the extended $\mathcal{WAS}$. In this paper, rather than using the entire sequence of support values, we propose a metric called *conservation rate* that summarizes the history of support values and make the H-$\mathcal{WAS}$ tree more compact.

4.3 Summarization of Support History

Given a $\mathcal{WAS}$ A and sequence of support values $H_A = \langle \Phi_1(A), \Phi_2(A), \Phi_3(A), \ldots, \Phi_k(A) \rangle$, the sequence of support values can be considered as a time series because the support values of a $\mathcal{WAS}$ may change over time in real life. Then, we propose to model the sequence of support values using the following linear regression model.

$$\Phi_t(A) = \Phi_0(A) + \lambda t, \ where\ 1 \leq t \leq k$$

Here the idea is to find a "best-fit" straight line through the data points $\{(\Phi_1(A), 1), (\Phi_2(A), 2), \ldots, (\Phi_k(A), k)\}$, where $\Phi_0(A)$ and λ are constants called *support intercept* and *support slope* respectively. The most common method for fitting a regression line is the method of least-squares [20]. By applying the statistical treatment known as linear regression to the data points, the two constants can be determined using the following formula [20].

$$\Phi_0(A) = \frac{k\sum_{i=1}^{k}(i * \Phi_i(A)) - (\sum_{i=1}^{k}\Phi_i(A))(\sum_{i=1}^{k} i)}{k\sum_{i=1}^{k}(\Phi_i(A))^2 - (\sum_{i=1}^{k}\Phi_i(A))^2}$$

$$\lambda = \frac{\sum_{i=1}^{k} i - (\Phi_0(A) * \sum_{i=1}^{k}\Phi_i(A))}{k}$$

Besides the two constants, there is another measure to evaluate how the regression fits the data points actually. It is the correlation coefficient, denoted as r.

$$r = \frac{k\sum_{i=1}^{k}(\Phi_i(A) * i) - (\sum_{i=1}^{k}\Phi_i(A))(\sum_{i=1}^{k} i)}{\sqrt{[k\sum_{i=1}^{k}(\Phi_i(A))^2 - (\sum_{i=1}^{k}\Phi_i(A))^2][k\sum_{i=1}^{k} i^2 - (\sum_{i=1}^{k} i)^2]}}$$

The correlation coefficient, r, always takes a value between -1 and 1, with 1 or -1 indicating perfect correlation. The square of the correlation coefficient, r^2, represents the fraction of the variation in $\Phi_t(A)$ that may be explained by t. Thus, if a correlation of, say 0.8, is observed between them, then a linear regression model attempting to explain the changes to $\Phi_t(A)$ in terms of t will account for 64% of the variability in the data [20].

Based on the above linear regression-based model for support history we now propose the metric *conservation rate*.

Definition 3. **[Conservation Rate]** *Let $\langle \Phi_1(A), \Phi_2(A), \ldots, \Phi_k(A) \rangle$ be the sequence of historical support values of a $\mathcal{WAS}$ A, where $\Phi_i(A)$ represents the i^{th} support value for A and $1 \leq i \leq k$. The conservation rate of $\mathcal{WAS}$ A is defined as $C_A = r^2 - |\lambda|$.* □

Note that the larger the absolute value of the slope, the more significantly the support changes over time. At the same time, the larger the value of r^2, the more accurate is the regression model. Hence, the larger the conservation rate C_A, the support values of the $\mathcal{WAS}$ change less significantly. In other words, the support values of a $\mathcal{WAS}$ are more *conserved* with the increase in the conservation rate. Also from the regression model, it can be inferred that $|\lambda| < \frac{1}{k}$ as $0 \leq \Phi_t(A) \leq 1$. In real life the value of k can be huge, thus $|\lambda| \ll r \leq 1$. Consequently, we can guarantee that $0 \leq C_A \leq 1$. When $C_A = 1$, the support of $\mathcal{WAS}$ A is a constant where $r^2 = 1$ and $\lambda = 0$.

Based on the above notion of conservation rate, we can define the Λ function in the H-$\mathcal{WAS}$ tree as follows.

Definition 4. **[Λ Function]** *Given an H-$\mathcal{WAS}$ tree, $H_G = (r, N, E, \Lambda)$, where r is a virtual root; $N = N_1 \cup N_j \cdots \cup N_k$; $E = E_1 \cup E_j \cdots \cup E_k$; the Λ function is defined to map each node $n \in N$ to a pair $(C_A, \mathcal{R})$ where C_A is the conservation rate and $\mathcal{R} = (M_A, U_A)$ is the support range of a $\mathcal{WAS}$ A whose last page is represented by n.* □

Example 1. Figure 3 (c) shows a part of an H-$\mathcal{WAS}$ tree, where the associated values are the corresponding conservation rate, unpopular support value, and popular support value in turn. In this example, the WAPs $\langle$ a, b, e, h $\rangle$ and $\langle$ a, f, g $\rangle$ are popular WAMs, given the thresholds for conservation rate, popular support threshold, and unpopular support threshold are 0.6, 0.3, and 0.05 respectively.

5. ALGORITHM WAM-MINER

In this section, we proposed an algorithm called WAM-MINER to discover the two types of WAMs from the historical web usage data. The mining process consists of two phases: the *H-$\mathcal{WAS}$ tree construction* phase and the *WAM extraction* phase. We discuss these phases in turn.

Algorithm 1 Extended $\mathcal{WAS}$ tree Construction.

Input: A $\mathcal{WAS}$ Group: $G = [T_{A_1}, T_{A_2}, \ldots, T_{A_n}]$
Output: T_G: the extended $\mathcal{WAS}$ tree

```
Create a virtual root node for T_G
Initialize T_G as the first WAS tree
for all  i = 2 to n  do
  if the root of T_{A_i} does not exist in T_G then
    attach T_{A_i} as a subtree of T_G and update Φ_i((N_j)
  else
    for all nodes N_j in WAS tree T_{A_i} do
      if N_j exists in the current subtree of T_G then
        Update Φ_i((N_j)
      else
        create a new child node N_j under the current node
      end if
    end for
  end if
end for
Return(T_G)
```

Algorithm 2 WAM Extraction

Input: The H-$\mathcal{WAS}$ tree: H_G
Thresholds: μ, α, and β
Output: The popular and unpopular WAMs: W_P and W_U

```
for all node n_i ∈ H_G do
  if M_{n_i} ≥ α then
    if C_{n_i} ≥ μ then
      W_P = n_i ⋃ W_P
    end if
  else
    if U_{n_i} ≤ β then
      if C_{n_i} ≥ μ then
        W_U = n_i ⋃ W_U
      end if
    end if
  else
    prune n_i
  end if
end for
Return(W_P, W_U)
```

5.1 Phase 1: H-$\mathcal{WAS}$ Tree Construction

Given a collection of web log data, we assume that it is represented as a set of $\mathcal{WAS}$s with corresponding timestamps. This phase consists of two steps. First, the sequence of extended $\mathcal{WAS}$ tree is constructed. Then, the H-$\mathcal{WAS}$ tree is built. Both algorithms for extended $\mathcal{WAS}$ tree construction and H-$\mathcal{WAS}$ tree construction are similar. The basic idea is to match the trees and merge the common prefix to make the representation compact. As the only difference between the extended $\mathcal{WAS}$ tree construction and H-$\mathcal{WAS}$ tree construction is the attributes associated with the nodes, in this section, only details of the extended $\mathcal{WAS}$ tree construction are presented.

The extended $\mathcal{WAS}$ tree construction algorithm is shown in Algorithm 1. Given a $\mathcal{WAS}$ group, firstly, the extended $\mathcal{WAS}$ tree is initialized as the first $\mathcal{WAS}$ tree in the group with a virtual root node. Then, the next tree is compared with the existing extended $\mathcal{WAS}$ tree to merge them together. That is, if a $\mathcal{WAS}$ tree or part of a $\mathcal{WAS}$ tree does not exist in the extended $\mathcal{WAS}$ tree, they will be inserted into the extended $\mathcal{WAS}$ tree. Otherwise, the $\mathcal{WAS}$ trees are merged into the subtrees that rooted at the node identical to the root of the $\mathcal{WAS}$ trees. For both the extending and merging process, their support values are updated accordingly. This process iterates for all the $\mathcal{WAS}$ trees in the $\mathcal{WAS}$ group.

Similarly, given a sequence of extended $\mathcal{WAS}$ trees, the H-$\mathcal{WAS}$ tree is constructed. Note that, the extending and merging process follows the same rules as the above rules for constructing the extended $\mathcal{WAS}$ tree. However, the attributes in the H-$\mathcal{WAS}$ tree are different from the attributes in the extended $\mathcal{WAS}$ tree. For example, in the extended $\mathcal{WAS}$ tree, there are only one support values associated with each node as shown in Figure 3. In the H-$\mathcal{WAS}$ tree, initially there will be a sequence of support values for a $\mathcal{WAS}$, which is associated with the last node. In the H-$\mathcal{WAS}$ tree construction process, for each $\mathcal{WAS}$, the sequence of support values are transformed into the conservation rate and support range using the linear regression model we discussed before.

5.2 Phase 2: WAM Extraction

Given the H-$\mathcal{WAS}$ tree, with the user-defined threshold for conservation rate (μ), popularity threshold (α), and unpopularity threshold (β), the WAM extraction phase is actually a traversal over the H-$\mathcal{WAS}$ tree. The algorithm of WAM extraction is shown in Algorithm 2. Here, the support range is first compared with α and β to determine the potential groups of popular WAMs and unpopular WAMs to which the corresponding WAP belongs to. If $M_A \geq \alpha$ and $U_A \leq \beta$ then, the conservation rate is further compared with the threshold μ. These WAPs whose conservation rate is no greater than μ are assigned to the popular WAMs and unpopular WAMs accordingly. Lastly, the sets of popular and unpopular WAMs are returned.

EXAMPLE 2. Let us take the H-$\mathcal{WAS}$ tree in Figure 3 (c) as an example. Let $\alpha = 0.3$, $\beta = 0.05$, and $\mu = 0.7$. First, we check the root of the H-$\mathcal{WAS}$ tree, its $M_r > 0.3$ and $C_r > 0.7$, then node a is included in the popular WAMs. Then, nodes b, d, c are checked in a similar way. In this example, node e is pruned out but its child node h is included, then node e is directly linked to node b in the final result.

6. PERFORMANCE EVALUATION

In this section, we present experimental results to evaluate the performance of our proposed WAM-MINER algorithm. All experiments were conducted on a P4 1.80 GHz PC with 512Mb main memory running Windows 2000 professional. The algorithm is implemented in Java.

Both real and synthetic web log datasets are used in the experiments. The real data is the web log *UoS* obtained from the Internet Traffic Archive [1]. It records the historical visiting patterns for University of Saskatchewan from June 1, 1995 to December 31, 1995. There were 2,408,625 requests with 1 second resolution and 2,981 unique URLs. The synthetic data set is generated using the synthetic tree generation program used in [23]. The characteristics of the synthetic data we used are shown in Table 2. The program first constructs a tree representation of the web site structure based on two parameters, the maximum fan out of a node (denoted as F) and the maximum depth of the tree (denoted as D). Based on the web site structure, a collection of $\mathcal{WAS}$s with the corresponding timestamps are generated by mimicking the user behaviors. In Table 2, $\overline{S}$ is the average size of the $\mathcal{WAS}$s and N is the number of $\mathcal{WAS}$s in the corresponding datasets.

6.1 Scalability and Efficiency

As the size of the web usage data collection can be affected by two factors: the number of $\mathcal{WAS}$s and the average size

Dataset	N	$\overline{S}$	F	D
D_1	10000	15	15	30
D_2	20000	15	15	30
D_3	30000	15	15	30
D_4	40000	15	15	30
D_5	50000	15	15	30
D_6	20000	10	10	25
D_7	20000	20	10	30
D_8	20000	25	15	35
D_9	20000	30	20	35

Table 2: Synthetic datasets

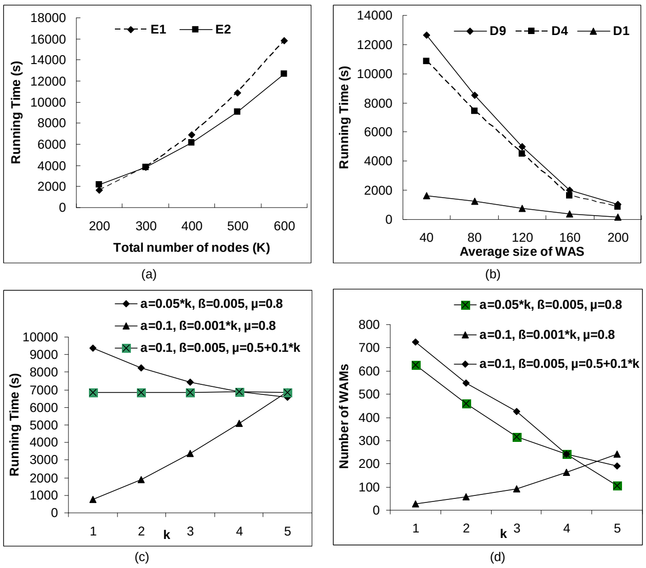

Figure 4: Experiment results

of each $\mathcal{WAS}$, two sets of experiments have been conducted to evaluate the scalability of our proposed algorithm. In the first set of experiments, denoted as E_1 in Figure 4 (a), synthetic datasets D_1, D_2, D_3, D_4, and D_5 are used, where the average size of each $\mathcal{WAS}$ is fixed while the number of $\mathcal{WAS}$s is varied. In the second set of experiments, denoted as E_2 in Figure 4 (a), synthetic datasets D_2, D_6, D_7, D_8, and D_9 are used, where the number of $\mathcal{WAS}$s is fixed while the average size of each $\mathcal{WAS}$ varies.

Figure 4 (a) shows the running time of the algorithm as the total number of nodes in the dataset increases. The user defined time interval, α, β, μ are set to 12 hours, 0.01, 0.005, and 0.8 accordingly. The running time increases as the total number of nodes increases from 100k to 600k. The reason is that with more nodes, both the cost of constructing the trees and the traversal over the H-$\mathcal{WAS}$ tree becomes more expensive. However, we observed that even for the same total number of nodes, the running time is much expensive when the number of $\mathcal{WAS}$s is large and the average size of each $\mathcal{WAS}$ is small. This is because the cost of calculation of Φ_0 and the conservation rate is quite expensive when the number of extended $\mathcal{WAS}$ trees is large. Note that for the same user-defined time interval, a larger number of $\mathcal{WAS}$s indicates that there are more extended $\mathcal{WAS}$ trees.

Besides the size of the datasets, experiments are also conducted to show how the parameters such as: user-defined time intervals, conservation rate, popularity threshold, and unpopularity threshold, affect the efficiency of the mining algorithm. Figure 4 (b) shows how the user-defined time interval affects the running time using D_1, D_4 and D_9. We set $\alpha = 0.1$, $\beta = 0.005$, and $\mu = 0.8$. Here, we use the average number of $\mathcal{WAS}$s in the $\mathcal{WAS}$ groups to represent the size of the time interval. It can be observed that the running time decreases as the size of the user-defined time interval increases. The reason is that the number of extended $\mathcal{WAS}$ trees is small as the average size of the $\mathcal{WAS}$ group increases. As a result, the computation cost of calculating the support range and conservation rate decreases.

Figure 4 (c) shows the relationship between the running time and the thresholds using D_9. There are three variables in this figure, the x-axis k changed from 1 to 5, and the values of α, β, and μ are dependent on k. For example, in the first set of experiment, $\beta = 0.005$ and $\mu = 0.8$; while $\alpha = 0.05 \times k$. Similarly, the values of β and μ are changed in a similar way in the remaining two experiments. It can observed that when α increases, the running time decreases because the number of popular WAMs decreases accordingly. When β increases, the running time increases because there are more unpopular WAMs. When μ increases, the running time is almost stable, which is because of the computation cost is independent of the threshold of conservation rate.

6.2 Quality of Popular and Unpopular WAMs

As there are four parameters, the user-defined time interval, α, β, and μ, in our algorithm, in this section, we investigate how the four parameters affect the quality of the mining results. By varying one parameter and fixing the values for the other three parameters, the effects of each parameter are evaluated in the following experiments. Note that the size of the time interval is measured by the average number of $\mathcal{WAS}$ in each $\mathcal{WAS}$ group. In the following experiments, the *UoS* real dataset is used.

In the first set of experiments, α, β and μ are fixed to 0.1, 0.005, and 0.8 respectively, the user-defined time interval varies from 40 to 200. Table 3(a) shows the number of popular WAMs and unpopular WAMs with different user-defined time interval. We observed that as the time interval increases, the number of popular and unpopular WAMs increases. By looking into the the results, we observed that popular and unpopular WAMs with smaller user-defined time intervals are also popular and unpopular WAMs with larger user-defined time intervals. We also compare the number of popular WAMs extracted by our WAM-MINER with the number of popular WAMs extracted by repeatedly using WAP-Mine[3][18]. We observed that the WAP-Mine based approach cannot extract all the popular WAMs. Note that, in the WAP-Mine-based popular WAM extraction approach, the conservation rate is calculated using the number of times a WAP is frequent in the sequence of $\mathcal{WAS}$ groups divided by the total number of $\mathcal{WAS}$ groups.

By fixing the user-defined time interval to 40, the effects of the other three parameters are evaluated in similar ways. Figure 4(d) shows how the total number of popular and unpopular WAMs changes with different α, β, and μ. Here, we introduce a variable, k, as the x-axis. Then, the values of α, β, and μ are represented using k. For example, in the first set of experiments, $\beta = 0.005$ and $\mu = 0.8$; while $\alpha = 0.05 * k$. It can be observed that the total number of WAMs increases as β increases, α decreases, or μ decreases.

Table 3(b) shows the quality of the regression-based model for extracting WAMs. In this experiment, the UoS dataset is partitioned into 30 $\mathcal{WAS}$ groups and is divided into two parts, denoted as P_1 and P_2. P_1 is used to construct the

[3]Downloaded from http://www.cs.ualberta.ca/∼tszhu

(a) Number of WAMs

Size of G	40	80	120	160	200
Popular WAMs	67	138	253	306	327
Unpopular WAMs	106	219	237	342	395
WAP-Mine	21	26	32	36	48

(b) Prediction Accuracy

$\|P_1\|$	$\|P_2\|$	Accuracy	α	β	μ
10	20	0.94	0.4	0.05	0.8
10	20	0.93	0.3	0.05	0.8
10	20	0.95	0.4	0.01	0.7
15	15	0.93	0.4	0.05	0.8
15	15	0.94	0.4	0.05	0.6
15	15	0.93	0.4	0.01	0.6
20	10	0.93	0.4	0.05	0.8
20	10	0.94	0.3	0.05	0.8
20	10	0.93	0.3	0.05	0.9

Table 3: Experimental Results

regression model and P_2 is used to evaluate the *accuracy* of the model. That is, we extract the popular and unpopular WAMs in P_1 using the regression model and check whether these are still popular/unpopular WAMs in P_2. The *accuracy* is defined as the percentage of popular/unpopular WAMs obtained from P_1 that are still popular/unpopular WAMs in P_2. Formally, let R_1 and R_2 be the sets of popular and unpopular WAMs returned by the WAM-MINER using P_1. Let Z_1 and Z_2 be the sets of popular and unpopular WAMs based on the entire dataset. Then accuracy is denoted as $\frac{1}{2}(\frac{|R_1 \cap Z_1|}{|Z_1|} + \frac{|R_2 \cap Z_2|}{|Z_2|})$. The results show that the accuracy of our model is quite high for different size of P_1. Furthermore, the quality of the model is robust and not affected by the user-defined thresholds as here we only identify whether a WAM is still popular/unpopular in P_2.

7. RELATED WORK

Web access sequence mining is defined to extract hidden patterns from the navigation behavior of web users [6]. In the existing web access pattern mining approaches, the sequential pattern mining algorithms are employed to extract the frequent access patterns such as WAP [18, 2], maximal forward frequent sequence [6], maximal frequent sequence with backward traversal [21], maximal and closed access pattern mining [22]. The focus of the existing works is to propose different data structures such as WAP-tree [18], H-struct [17], and prefix tree [16], that can make the subsequence mining problem more efficient and scalable. These approaches focus on mining patterns based on the overall support counts of the access patterns at a particular timepoint. These approaches ignore the changes to the support values of the $\mathcal{WAS}$. Differing from these approaches, our work is based on the changes to the support counts of the access patterns over a specific time period. The change patterns of the support values are expected to reflect the historical behaviors of the 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 [8]. Another is to detect interesting changes in the KDD mining results such as the FOCUS framework proposed by Ganti et al [7], PAM proposed by Baron et al [4], and the fundamental rule change detection tools proposed by Liu et al [14].

Our effort differs from the above approaches in the following ways. First, these techniques are proposed either for updating the mining results or detecting the changes to the mining results with respect to the changes to the data sources. Second, in previous approaches, only the order within web access sequences are considered, while we also consider the timestamps corresponding to each occurrence of the same web access sequence. Lastly, unlike the above techniques, we also extract unpopular WAMs which can be useful in many applications as discussed in Section 2.

8. CONCLUSIONS

This work is motivated by the fact that existing web usage mining techniques focus only on discovering knowledge based on the statistical measures obtained from the static characteristics of web usage data. They do not consider the dynamic nature of web usage data. We focus on discovering novel knowledge by analyzing the change patterns of historical web access sequence data. Specifically, we propose an algorithm called WAM-MINER that extracts popular and unpopular Web Access Motifs (WAMs) from historical web usage data. WAMs are WAPs that never change or do not change significantly most of the time (if not always) in terms of their support values during a specific time period. WAMs are useful for many applications, such as intelligent web advertisement, web site restructuring, business intelligence, and intelligent web caching. Experimental results on both synthetic data and real datasets show that WAM-MINER is efficient and scalable. More importantly, it can extract novel knowledge that cannot be discovered by existing web usage mining approaches.

9. REFERENCES

[1] http://ita.ee.lbl.gov/. Internet traffic archive.
[2] R. Agrawal and R. Srikant. Mining sequential patterns. *ICDE*, 3–14, 1995.
[3] A. Amiri and S. Menon. Efficient scheduling of internet banner advertisements. *ACM TOIT*, 3(4):334–346, 2003.
[4] S. Baron, M. Spiliopoulou, and O. Gnther. Efficient monitoring patterns in data mining environments. *ADBIS*, 253–265, 2003.
[5] C. Buchwalter, M. Ryan, and D. Martin. The state of online advertising: data covering 4^{th}Q 2000. TR AdRelevance, 2001.
[6] M.-S. Chen, J. S. Park, and P. S. Yu. Efficient data mining for path traversal patterns. *TKDE*, 10(2):209–221, 1998.
[7] V. Ganti, J. Gehrke, and R. Ramakrishnan. A framework for measuring changes in data characteristics. *PODS*, 1999.
[8] V. Ganti and R. Ramakrishnan. Mining and monitoring evolving data. *Handbook of massive data sets*, 593–642, 2002.
[9] J. Garofalakis, P. Kappos, and D. Mourloukos. Web site optimization using page popularity. *IEEE Internet Computing*, 3(4):22–29, 1999.
[10] S. Gunduz and M. T. Ozsu. A web page prediction model based on click-stream tree representation of user behavior. *SIGKDD*, 535–540, 2003.
[11] X. Huang, A. An, N. Cercone, and G. Promhouse. Discovery of interesting association rules from livelink web log data. *ICDM*, 763–766, 2002.
[12] N. Labroche, N. Monmarche, and G. Venturini. Web sessions clustering with artificial ants colonies. *WWW*, 2003.
[13] T. Li, Q. Yang, and K. Wang:. Classification pruning for web-request prediction. *WWW*, 2001.
[14] B. Liu, W. Hsu, and Y. Ma. Discovering the set of fundamental rule changes. In *Proc. of SIGKDD*, 335–340, 2001.
[15] B. Mobasher, R. Cooley, and J. Srivastava. Creating adaptive web sites through usage-based clustering of URLs. *IEEE KDEX Workshop*, 1999.
[16] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: mining sequential patterns by prefix-projected growth. *ICDE*, 215–224, 2001.
[17] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-Mine: hyper-structure mining of frequent patterns in large databases. *ICDM*, 441-448, 2001.
[18] J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu. Mining access patterns efficiently from web logs. *PAKDD*, 396–407, 2000.
[19] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.
[20] S. Weisberg. Applied linear regression. Wiley; 2 edition, July, 1985.
[21] Y. Xiao and M. H. Dunham. Efficient mining of traversal patterns. *DKE*, 39(2):191–214, 2001.
[22] Y. Xiao, J.-F. Yao, Z. Li, and M. H. Dunham. Efficient data mining for maximal frequent subtrees. *ICDM*, 379–386, 2003.
[23] M. J. Zaki. Efficiently mining frequent trees in a forest. *SIGKDD*, 2002.