CLEOPATRA: Evolutionary Pattern-based Clustering of Web Usage Data

Qiankun Zhao[1] Sourav S Bhowmick[1] Le Gruenwald[2]*

[1]CAIS, Nanyang Technological University, Singapore
[2]University of Oklahoma, Norman, USA
qkzhao@pmail.ntu.edu.sg assourav@ntu.edu.sg
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 present an algorithm called CLEOPATRA (**CL**ustering of **E**v**O**lutionary **PA**t**T**e**R**n-based web **A**ccess sequences) to cluster web access sequences ($\mathcal{WAS}s$) based on their *evolutionary patterns*. In this approach, Web access sequences that have similar change patterns in their support counts in the history are grouped into the same cluster. The intuition is that often $\mathcal{WAS}s$ are event/task-driven. As a result, $\mathcal{WAS}s$ related to the same event/task are expected to be accessed in similar ways over time. Such clusters are useful for several applications such as intelligent web site maintenance and personalized web services.

1 Introduction

Recently, web usage mining has become an active area of research and commercialization [3, 6, 10]. Often, web usage mining provides insight about user behaviors 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 [1].

Generally, the web usage mining process can be considered as a three-phase process, which consists of *data preparation*, *pattern discovery*, and *pattern analysis* [10]. 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 [1]. Figure 1(a) 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 Figure 1(a) records the collection of $\mathcal{WAS}$s for a particular month. In the second phase, statistical methods and/or

* This material is based upon work supported by (while serving at) the National Science Foundation (NSF). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

S_ID	*WAS*s
1	<a, b, d, c, a, f, g>
2	<a, b, e, h, a, f, g>
3	<e, f, g, i, n>
4	< b, d, c, a, e>

(1) The first month

S_ID	*WAS*s
1	<a, b, d, c, a, f, g>
2	<b, d, c, x>
3	<e, f, g, i, n>
4	< b, e, h, b, d, c, n, f, g>

(2) The second month

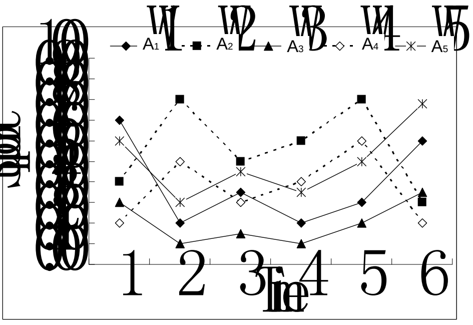
S_ID	*WAS*s
1	< b, d, e, a, f, g>
2	<b, e, h, b, d, c>
3	<e, f, g, i, n>
4	< e, f, g, i, n>

(3) The third month

S_ID	*WAS*s
1	< b, d, e, a, f, g>
2	<e, f, g, i, n>
3	< a, b, e, c, f, g>
4	<e, f, g, i, n>

(4) The fourth month

(a) Example of $\mathcal{WAS}$s



(b) Support of $\mathcal{WAS}$s over a time period

Fig. 1.

data mining techniques are applied to extract interesting patterns such as *Web Access Patterns* (WAPs)[7]. A WAP is a sequential pattern in a large set of $\mathcal{WAS}$s, which is visited frequently by users [7], 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 $\mathcal{A}$. Lastly, these patterns are used for further analysis in the third phase, which is application dependent.

From Figure 1(a), it is obvious 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. The dynamic behaviors of $\mathcal{WAS}$s can be attributed to various factors, such as changes to web content and users' interest, arrival of new web visitors, and effects of real life events.

In particular, the dynamic nature of $\mathcal{WAS}$ data leads to two challenging problems in the context of web usage mining: maintenance of web usage mining results and discovering novel knowledge [11]. In this paper, we focus on discovering novel knowledge from historical $\mathcal{WAS}$s. Particularly, we focus on clustering of $\mathcal{WAS}$s based on the characteristics of their evolution over time. The intuition behind this is that $\mathcal{WAS}$s are event/task driven. Consequently, $\mathcal{WAS}$s related to the same event/tasks are expected to be accessed in a similar way over time. For example, consider Figure 1(b), which depicts the support values (y-axis) of five $\mathcal{WAS}$s (denoted as A_1, A_2, A_3, A_4, and A_5) from time period 1 to 6 (x-axis). Note that i in the x-axis represents a time period (e.g., day, week, month etc.) and not a particular time point. It can be observed that evolutionary pattern of the supports for A_1, A_3, and A_5 are very similar over time (like the letter "W"). Similarly, the evolutionary patterns of supports for A_2 and A_4 are similar (like the letter "M"). However, the "W" and "M" clusters cannot be discovered by existing web usage mining techniques due to the fact that they focus only on knowledge discovery from snapshot data and maintenance of the knowledge with the changes to the data source. To extract those clusters, in this paper, we propose the Cleopatra (**CL**ustering of **E**v**O**lutionary **PAT**te**R**n-based web **A**ccess sequences) algorithm.

[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.

The CLEOPATRA clustering results can be useful in many applications, two of which are given below.

Intelligent Web Site Maintenance: With the massive amount of data on the web, it is critical to maintain a well-structured web site in order to increase customer loyalty. Recently web usage mining techniques have been successfully used as a key solution to this issue [3]. However, none of these techniques exploits the dynamic nature of $\mathcal{WAS}$s to restructure web sites. The CLEOPATRA clustering results can be used by web site administrators to maintain a well-structured web site. For example, consider the "W" cluster of $\mathcal{WAS}$s in Figure 1(b), which includes A_1, A_3, and A_5. By analyzing the evolutionary patterns, the web site administrator can figure out the possible reasons (such as promotions, release of new products, and holidays) for such patterns. Accordingly, the structure of the web site can be modified.

User Segmentation: User segmentation is to cluster web users based on the corresponding $\mathcal{WAS}$s to provide personalized services [4, 3]. Existing works either use sequence-based distance or probability models to measure the distance between $\mathcal{WAS}$s [4, 3]. However, none of them has taken the dynamic nature of $\mathcal{WAS}$s into account. For instance, two users may have the same list of $\mathcal{WAS}$s that belong to two topics, T_1 and T_2, having the same support. Using existing segmentation techniques, the two users will be grouped into the same cluster. However, they may have different preferences. For example, the first user may be currently interested in T_2 as most of the $\mathcal{WAS}$s about T_1 were accessed long time ago, while the second user may be currently interested in T_1 as most of the $\mathcal{WAS}$s about T_2 were also accessed long time ago. By taking the temporal information into account, the user segmentation can be more accurate as users in the same group are not only expected to have similar $\mathcal{WAS}$s but also evolutionary patterns of those $\mathcal{WAS}$s are expected to be similar as well.

The contributions of this paper can be summarized as follows:

- This is the first approach to cluster $\mathcal{WAS}$s based on the evolutionary patterns of their support counts.
- We proposed an algorithm called CLEOPATRA for clustering $\mathcal{WAS}$s based on the evolutionary patterns. Also, the performance of the algorithm is evaluated with real life web usage dataset.

2 Problem Statement

In general, web log data can be considered as sequences of web pages with *session identifiers* [1]. 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 $\mathcal{WAS}$.

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 $1 \leq j \leq k$ and p_i was visited between t_e and t_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 Figure 1(a) as an example. They can be partitioned into four $\mathcal{WAS}$ groups on a monthly basis, where $\mathcal{WAS}$s, the timestamps of which are in the same month, are partitioned into the same $\mathcal{WAS}$ group. The *size* of G, denoted as $|G|$, reflects the number of $\mathcal{WAS}$s in G.

Given a $\mathcal{WAS}$ group G, the *support* of a $\mathcal{WAS}$ A in G is $\Phi_G(A) = \frac{|\{A_i | A \subseteq A_i \in G\}|}{|G|}$. When the $\mathcal{WAS}$ group G is obvious from the context, the support is denoted as $\Phi(A)$. Similarly, when the $\mathcal{WAS}$ A 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 $\mathcal{WAS}$ A, let $H_A = \langle\ \Phi_1(A), \Phi_2(A), \Phi_3(A), \ldots, \Phi_k(A)\ \rangle$ be the sequence of support values of A in H_G. Then, the *degree of dynamic* (denoted as $\omega(A)$) and *version dynamic* (denoted as $\chi(A)$) of A are defined to summarize the changes of support values in the history (defined later in Section 3.1). Moreover, an *evolutionary pattern-based distance* (denoted as $\mathcal{D}$) is defined as the *Euclidian* distance between $\mathcal{WAS}$s based on their *version dynamic* values.

Given a collection of $\mathcal{WAS}$s, with an evolutionary pattern-based distance $\mathcal{D}$ and the degree of dynamic, the objective of the CLEOPATRA algorithm is to partition $\mathcal{WAS}$s into clusters such that $\mathcal{WAS}$s within the same cluster are more similar/closer to each other than to $\mathcal{WAS}$s in other clusters.

3 Representation of Historical $\mathcal{WAS}$s

Given a $\mathcal{WAS}$ denoted as $A = \langle p_1, p_2, p_3, \ldots, p_n \rangle$, in this paper, we use an unordered tree called $\mathcal{WAS}$ *tree* to represent the $\mathcal{WAS}$. 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; $N = \{p_1, p_2, \cdots, p_n\}$ is the set of nodes; and E is the set of edges in the maximal forward sequences of A. An example of a $\mathcal{WAS}$ tree is shown in Figure 2(a), which corresponds to the first $\mathcal{WAS}$ shown in Figure 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.

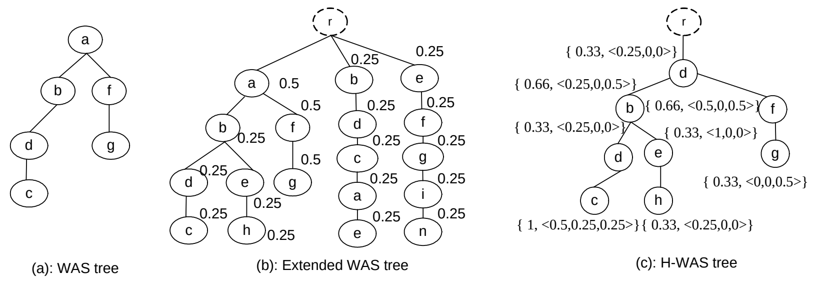


Fig. 2. Examples

Definition 1. **[Extended $\mathcal{WAS}$ Tree]** *Let G = [A_1, A_2, ..., 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, Θ), 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 Figure 1(a). The corresponding *extended $\mathcal{WAS}$ tree* is shown in Figure 2(b), where the value associated with each node is the Θ value. Next, we propose to merge the sequence of extended $\mathcal{WAS}$ trees into an historical $\mathcal{WAS}$ tree, called *H-$\mathcal{WAS}$ tree*.

Definition 2. **[H-$\mathcal{WAS}$ Tree]** *Let $H_G = \langle$ G_1, G_2, G_3, ..., 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, Θ). Then, the H-$\mathcal{WAS}$ tree is defined as H_G = (r, N, E, $\wp$), 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 $\wp$ 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 two metrics called *version dynamic* and *degree of dynamic* to summarize the history of support values.

Definition 3. **[Degree of Dynamic]** *Given a $\mathcal{WAS}$, A, with the corresponding support count sequence H_A =$\langle$ $\Phi_1(A)$, $\Phi_2(A)$, $\cdots$ $\Phi_n(A)$ $\rangle$, the degree of dynamic, denoted as $\omega(A)$, is defined as:*

$$\omega(A) = \frac{1}{n-1} * \sum_{i=1}^{n-1} d_i \quad where \quad d_i = \begin{cases} 1, \text{ } if \ \Phi_i(A) \neq \Phi_{i+1}(A); \\ 0, \text{ } otherwise \end{cases}$$

□

Definition 4. **[Version Dynamic]** *Given a $\mathcal{WAS}$, A, with the corresponding support count sequence H_A =$\langle$ $\Phi_1(A)$, $\Phi_2(A)$, $\cdots$ $\Phi_n(A)$ $\rangle$, the version dynamic, denoted as $\chi(A)$, is defined as a sequence $\chi(A)$=$\langle\chi_1(A)$, $\chi_2(A)$, $\cdots$, $\chi_{n-1}(A)\rangle$, where $\chi_i(A)$ =$\frac{|\Phi_i(A)-\Phi_{i+1}(A)|}{max\{\Phi_i(A),\Phi_{i+1}(A)\}}$, for $1 \leq i <$ n-1.* □

Figure 2(c) shows a part of an H-$\mathcal{WAS}$ tree, where the associated values are the corresponding degree of dynamic value, and the sequence of version dynamic values. The degree of dynamic measures how frequently the $\mathcal{WAS}$ changed and the version dynamic measures how significant are the changes in the history. Furthermore, based on the version dynamic metric, we propose an *evolutionary pattern-based distance* to measure the relationships between $\mathcal{WAS}$s.

Definition 5. [Evolutionary Pattern-based Distance] *Given two* $\mathcal{WAS}$*s (*A_1 *and* A_2*), the evolutionary pattern-based distance between* A_1 *and* A_2*, denoted as* $\mathcal{D}(A_1, A_2)$*, is defined as:*

$$\mathcal{D}(A_1, A_2) = \sqrt{(\chi'_1(A_1) - \chi'_1(A_2))^2 + \cdots + (\chi'_{n-k+1}(A_1) - \chi'_{n-k+1}(A_2))^2}$$

where $\chi'_i(A_j) = \frac{1}{k}\sum_i^{i+k-1}(\frac{\chi_i(A_j)-\overline{\chi(A_j)}}{\sigma(A_j)})$*,* k *is the user defined window size,* $\overline{\chi(A_j)}$ *and* $\sigma(A_j)$ *are the average support count value and standard deviation of* $\chi(A)$*.* □

Note that, the above evolutionary pattern-based distance measure is actually the Euclidean distance between the smoothed $\chi(A)$ sequence using the moving average. This distance measure can handle $\mathcal{WAS}$s with *different baseline*, *scale*, and *time offset*. Such properties are highly desired in this specific problem for the following reasons. Firstly, the average $\chi(A)$, which can be viewed as the baseline for the $\chi(A)$ sequence, for $\mathcal{WAS}$s that are related to the same event/task may vary a lot while their evolutionary patterns are similar. Secondly, the effects of event/task on different $\mathcal{WAS}$s can be different, which makes the scales of changes ($\chi(A)$) to those $\mathcal{WAS}$s different. Thirdly, there may be a different time delays for different $\mathcal{WAS}$s related to the same event/task, which may cause the time offset among $\chi(A)$ sequences.

4 Cleopatra Algorithm

The Cleopatra algorithm consists of three major phases: the *H-*$\mathcal{WAS}$ *tree construction* phase, the *node-based clustering* phase, and the *subtree-based clustering* phase. The objective of the *H-*$\mathcal{WAS}$ *tree construction* phase is to represent the $\mathcal{WAS}$*s* as trees and merge them into a single tree structure that records both the structural and temporal information. As the *H-*$\mathcal{WAS}$ *tree construction* has been discussed in [11], we focus on the clustering phases.

Node-based Clustering Phase: The objective of this phase is to categorize individual nodes with similar evolutionary patterns in the H-$\mathcal{WAS}$ tree into clusters. Note that individual nodes represent $\mathcal{WAS}$s from the root to the current nodes. *Hereafter, clustering individual nodes refer to clustering* $\mathcal{WAS}$*s that starts from the root and ends at the corresponding leaf nodes.* This algorithm is shown in Figure 3 and consists of two phases, a *two-level clustering* phase and an *iterative refinement* phase. In the first phase, given an H-$\mathcal{WAS}$ tree, firstly, it is *clustered* based on the *degree of dynamic* associated with the individual nodes. Then, using the *evolutionary pattern-based distance*, the degree of dynamic based clustering

```
Input: H-$\mathcal{WAS}$ tree: $H$
Output: a set of clusters $C$
1: $C'$=DBSCAN(H, $\omega(A)$)
2: for all Node pairs $(N_i, N_j)$ in cluster $c'_i \in C'$ do
3:     calculate $\mathcal{D}(N_i, N_j)$
4: end for
5: $C$ = DBSCAN ($c'_i$, $\mathcal{D}$), $\forall\ c'_i \in C'$
6: for Stop = False do
7:     $C'$=Split($C$)
8:     $C'$=Merge($C'$)
9: end for
10: Return($C$)
```

Fig. 3. Node-based Clustering Algorithm

```
Input: A set of clusters $C$, distance threshold $\epsilon$ for DBSCAN
Output: Refined clusters $C'$
1: for cluster $C_j \in C$ do
2:     calculate the centroid point $C(C_j)$
3: end for
4: for all $C_j, C_k \in C$ & $C(C_j) \neq C(C_k)$ do
5:     if $\mathcal{D}(C(C_j), C(C_k)) < 2 * \epsilon$ then
6:         merge them into a new cluster
7:         calculate the new centroid point
8:     end if
9: end for
10: Return clusters $C'$
```

Fig. 4. Merging Operation

results are further partitioned into smaller clusters. In the second phase, the iterative refinement phase, the *merging* and *splitting* algorithms are used to refine the quality of the clustering results. The reason is that in the first phase, the two metrics *degree of dynamic* and *evolutionary pattern-based distance* are used separately, when the merging and splitting operations converge, the results will be more accurate.

Note that we use the DBSCAN algorithm [2] to cluster the individual nodes in the H-$\mathcal{WAS}$ tree in this phase for the following reasons. First, the DBSCAN algorithm needs no prior knowledge about the number of clusters in the data collection. This is an advantage of the density-based clustering algorithms. Secondly, the naive DBSCAN approach has the time complexity of $O(N \log N)$, where N is the total number of points in the database, using spatial indexing techniques. Moreover, the DBSCAN algorithm is able to discover clusters with arbitrary shapes and is efficient for very large database. Notice that here the distances between nodes in the H-$\mathcal{WAS}$ tree are the Euclidean distances calculated based on the smoothed $\chi(A)$ sequence generated using the moving average.

In the first phase, the reason for designing a two-level clustering algorithm is to avoid computational cost. In the first level, the degree of dynamic values are used for producing a preliminary results as the degree of dynamic values are easier to obtain while the cost for calculating the evolutionary pattern-based distances are relatively more expensive. By doing this, the computational cost for calculating the evolutionary pattern-based distances for nodes that are not expected to be in the same cluster can be reduced.

In the second phase, the merging and splitting operations are proposed to refine the clustering results in the first phase. The intuition behind is that it is possible that the first level of *degree of dynamic* based clustering results may not fully reflect the *evolution pattern-based distances* between the nodes. Using this iterative merging and splitting operations, which will converge to certain results, we can guarantee that node-based clustering results are accurate, which is the foundation for the sub-tree based clustering in the next phase.

Specifically, merging operation is shown in Figure 4. Firstly, for each cluster a virtual centroid is obtained. Then, the distances between those centroids are calculated using the proposed evolutionary pattern-based distance measure. For clusters whose centroids are within a distance of $2*\epsilon$ will be merged together to form a new cluster, where ϵ is the radius parameter for the DBSCAN algorithm [2]. After that, the splitting operations is then performed on the new clustering results to split them into new clusters if possible. This splitting process is based on the DBSCAN algorithm as well.

Subtree-based Clustering Phase: The output of the node-based clustering phase is a set of clusters that consist of sets of individual nodes with similar change patterns. However, given a cluster, the relations between individual nodes are not captured. In this section, the individual nodes within clusters are merged together to form subtrees, which can represent higher level concepts or objects. Note that, the subtree construction process is guided by not only the links in the *H-$\mathcal{WAS}$ tree*, but evolution patterns of these nodes should be similar. For a given node in the cluster, to measure the number of nodes that have similar evolution patterns with it, the *evolutionary degree* is defined as follows.

Definition 6. [Evolutionary Degree] *Let $C = NodeClust(H)$ be a function that implements the node-based clustering phase where H is the H-$\mathcal{WAS}$ tree and C is the set of clusters returned by the function. Let $B(i,j) = Edge(n_i, n_j)$ be a function that takes in two nodes n_i and n_j and returns 1 if there exists or 0 if there does not exist an edge (n_i, n_j) in H. Let $C_x = \{n_1, n_2, \cdots, n_{|C_x|}\}$ and $C_x \in C$. Then, the evolutionary degree of $n_i \in C_x$ (denoted as $E_\bullet(n_i)$) is defined as follows: $E_\bullet(n_i) = \sum_{j=1}^{|C_x|} B(i,j)$, where $i \neq j$ and $0 < j \leq |C_x|$* □

From the above definition, it can be observed that nodes that have large evolutionary degree are expected to form large subtrees. In this section, we propose to extract the list of subtrees for each cluster. Firstly, nodes in each cluster are ranked based on the evolutionary degree in descending order. Then, to ensure that $\mathcal{WAS}$s in the same subtree have similar evolutionary patterns with each other, the *intra similarity* is defined as follows.

Definition 7. [Intra Similarity] *Let C = NodeClust (H) and $C = \{C_1, C_2, \cdots, C_n\}$. Let t_j be a subtree of H and $\mathcal{N}_t$ be the set of nodes in t_j. Let $\mathcal{K} = \{ K_1, K_2, \cdots, K_i \}$, where $K_r = |\mathcal{N}_t \cap C_r| \; \forall \; 0 \leq r \leq i$ and $r \leq n$. Then, the intra similarity of t_j, denoted as $\mathcal{IS}(t_j)$, is defined as: Max($\mathcal{K}$) / $|\mathcal{N}_t|$, where Max($\mathcal{K}$) is the maximum value in $\mathcal{K}$.* □

Definition 8. [Cluster Subtree] *Let $t_j = (N_j, A_j)$ be a subtree of H such that $N_j \subseteq C_x$ and $C_x \in C$ where $C = NodeClust(H)$. Then t_j is a cluster subtree if $\mathcal{IS}(t_j) \geq \beta$ where β is a user-defined threshold.* □

The algorithm for extracting subtree clusters is presented in Figure 5. The input of the subtree-based clustering algorithm is a set of clusters with sorted nodes. Firstly, the node with maximum evolutionary degree is selected and the corresponding subtree that includes all the nodes that are connected to that nodes is constructed and tested against the threshold value of $\mathcal{IS}$. If this subtree is a cluster subtree, then all the nodes in this subtree are eliminated from the list of subtrees in that cluster. Otherwise, if this subtree is not a cluster subtree, then the evolutionary degree of this node is set to *-1*. This process iterates till all the nodes in the subtree are tested.

```
Input: Clusters with sorted nodes $C$, $\mathcal{IS}$ threshold $\beta$
Output: Clusters of subtrees $CoS$
1: for all cluster $C_j \in C$ do
2:   for all node $n_x$ with the largest $E_\bullet(n_x)$ where $E_\bullet(n_x) > 0$ do
3:     prune all the leaf nodes that are in different cluster with $n_x$ iteratively
4:     calculate the $\mathcal{IS}$ of the subtree rooted at $n_x$
5:     if $\mathcal{IS}$ (Tree($n_x$)) $\geq \beta$ then
6:       insert this subtree into the CoS list
7:       prune all the leaf nodes in this subtree from this cluster
8:     else
9:       $E_\bullet(n_x) = -1$
10:    end if
11:  end for
12: end for
13: Return($CoS$)
```

Fig. 5. Subtree-based Clustering

Dataset	ϵ	k	β	H_{avg}	H_{min}	S_{avg}	S_{max}	$\|CoS\|$
UoS	0.05	30	0.8	0.81	0.16	0.21	0.46	46
UoS	0.10	60	0.8	0.79	0.13	0.23	0.51	38
UoS	0.15	60	0.75	0.78	0.17	0.19	0.48	34
UoS	0.20	90	0.75	0.78	0.14	0.20	0.46	36
Calgary	0.05	30	0.8	0.80	0.14	0.23	0.45	71
Calgary	0.10	60	0.8	0.79	0.15	0.16	0.38	68
Calgary	0.15	60	0.75	0.71	0.13	0.17	0.38	63
Calgary	0.20	90	0.75	0.75	0.06	0.13	0.32	62

Fig. 6. Experimental Results

5 Performance Evaluation

In this section, we evaluate our proposed clustering algorithm with two real datasets, the web log *UoS* and *Calgary*, obtained from the Internet Traffic Archive [5]. The *UoS* 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. The *Calgary* logs were collected from October 24, 1994 through October 11, 1995, a total of 353 days. There were 726,739 requests. Both of them have 1 second resolution. The web access patterns are transformed into a sequence of extended $\mathcal{WAS}$ trees with a duration of one day. All the following experiments are carried out on a PC with Intel Pentium 4, 1.7Ghz CPU, and 512MB RAM.

Our experiments focus on two aspects: the quality and novelty of the clustering results. To evaluate the quality of the our clustering results, two quality metrics, *Homogeneity* and *Separation* [9, 8], are used. Here we review the metrics: $H_{avg} = \frac{1}{M}\sum_{i<j,\ C(A_i)=C(A_j)} S(A_i, A_j)$; $H_{min} = \min_{C' \in C} \frac{2 * \sum_{i<j \in C'} S(A_i, A_j)}{|C'| * (|C'|-1)}$; $S_{avg} = \frac{2}{n(n-1)-2M}\sum_{i<j,\ C(A_i) \neq C(A_j)} S(A_i, A_j)$; and $S_{max} = \max_{C, C' \in C}$ $\sum_{A_i \in C,\ A_j \in C'} S(A_i, A_j)|C| * |C'|$, where n is the total number of $\mathcal{WAS}$ subtrees; A_i is the ith $\mathcal{WAS}$ subtree; M is the total number of node pairs that are within the same cluster; C is the set of clusters in the result and $|C|$ is the size of the set; $C(A_i)$ is the cluster to which A_i belongs. Note that, here we transform the evolutionary pattern-based distance to the similarity measure S such that we can use the above cluster quality metrics. That is, $S(A_i, A_j) = e^{-\mathcal{D}(A_i, A_j)}$. The larger homogeneity implies a better result, while a larger separation shows a worse result.

Figure 6 shows the quality of the clustering results with different parameters for the DBSCAN algorithm, size of moving window in the moving average, and the intra similarity threshold. The reason of using the above cluster quality metrics is that, due to privacy reasons, the original URLs of web pages in the web usage dataset are not available. Therefore, the ground truth of the clusters are not available. However, from the values in Figure 6, compared with the corresponding values in other applications that using above quality metrics, the quality of our results is comparable to the results in [9, 8].

Considering the novelty of our clustering results, although there is no quantified measures, we have the following observations. First, in the CLEOPATRA clustering results, we found many $\mathcal{WAS}$ pairs that are in the same cluster are very far away in the H-$\mathcal{WAS}$-tree while the evolutionary patterns are quite similar. Such clustering results can be useful for exploring the hidden factors that lead to the evolution of the corresponding $\mathcal{WAS}$s. Second, the overall structures of the clusters are quite similar in the CLEOPATRA clustering result. This means that suppose we have two clusters C_1 and C_2, where $C_1 = \{A_1, A_2, A_3\}$ and $C_2 = \{A_4, A_5, A_6\}$, although A_1, A_2, and A_3 may not be siblings or connected but pairs such as $\{A_1, A_4\}$, $\{A_2, A_5\}$, and $\{A_3, A_6\}$ are siblings or connected.

6 Conclusions

This work is motivated by the fact that existing web usage mining techniques only focus on mining snapshot web usage data and maintaining of the mining results incrementally. They do not consider the dynamic nature of web usage data. In this paper, we proposed the first approach of clustering historical $\mathcal{WAS}$s based on the evolutionary patterns. Experiments with real life datasets show CLEOPATRA can efficiently produce high quality clusters that cannot be discovered using existing web usage mining techniques.

Acknowledgements
We thank Dr Mukesh Mohania from IBM India Research Lab for the feedbacks on the initial draft of this paper.

References

1. M.-S. Chen, J. S. Park, and P. S. Yu. Efficient data mining for path traversal patterns. *TKDE*, 10(2):209–221, 1998.
2. M. Ester, H.-P. Kriegel, J. Sander and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pages 226-231, 1996.
3. S. Gunduz and M. T. Ozsu. A web page prediction model based on click-stream tree representation of user behavior. In *Proc. of SIGKDD*, pages 535–540, 2003.
4. X. Jin, Y. Zhou, and B. Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *Proceedings of ACM SIGKDD*, pages 197–205, 2004.
5. L. B. N. Laboratory. Internet traffic archive, http://ita.ee.lbl.gov/, 2004.
6. T. Li, Q. Yang, and K. Wang:. Classification pruning for web-request prediction. In *Proc. of WWW*, 2001.
7. J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu. Mining access patterns efficiently from web logs. In *Proc. of PAKDD*, pages 396–407, 2000.
8. R. Sharan, A. M. Katz, and R. Shamir. Click and expander: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.
9. R. Sharan and R. Shamir. Center click: A clustering algorithm with applications to gene expression analysis. In *ISMB*, pages 307–316, 2000.
10. J. Srivastava, R. Cooley, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *KDD Exploration*, 1(2):12–23, 2000.
11. Q. Zhao and S. S. Bhowmick and L. Gruenwald. WAM-Miner: In the Search of Web Access Motifs from Historical Web Log Data. In *Proceedings of CIKM* 2005.