World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:10, 2009

Concurrency in Web Access Patterns Mining

Jing Lu, Malcolm Keech, and Weiru Chen

***Abstract*—**Web usage mining is an interesting application of data mining which provides insight into customer behaviour on the Internet. An important technique to discover user access and navigation trails is based on sequential patterns mining. One of the key challenges for web access patterns mining is tackling the problem of mining richly structured patterns. This paper proposes a novel model called Web Access Patterns Graph (WAP-Graph) to represent all of the access patterns from web mining graphically. WAP-Graph also motivates the search for new structural relation patterns, i.e. Concurrent Access Patterns (CAP), to identify and predict more complex web page requests. Corresponding CAP mining and modelling methods are proposed and shown to be effective in the search for and representation of concurrency between access patterns on the web. From experiments conducted on large-scale synthetic sequence data as well as real web access data, it is demonstrated that CAP mining provides a powerful method for structural knowledge discovery, which can be visualised through the CAP-Graph model.

***Keywords*—**concurrent access patterns (CAP), CAP mining and modelling, CAP-Graph, web access patterns (WAP), WAP-Graph, Web usage mining.

I. Introduction

WEB data mining is the discovery of patterns from the web when viewed from its various perspectives [1]. It is natural to consider the application of general data mining techniques in the context of the web, while recognising that web-based data will sometimes require pre-processing to a suitable format. According to analysis targets, web mining can be divided into three different types, namely web content mining, web structure mining and web usage mining [2,3]. Web content mining pursues the search for useful information from the web contents, its data and documents; web structure mining is used to identify the relationship between web pages associated by information or direct link connection; web usage mining focuses on techniques that could predict customer behaviour while the user interacts with the web.

Web access patterns mining is a type of usage mining that looks at web page visit history. The frequent web access patterns mined from web log files are essential for web masters and developers to improve the design of their web sites further [4]. Mining frequent web access patterns from very large databases (e.g. using click-stream analysis) has been studied intensively and there are a variety of approaches. Most of the previous studies have adopted a sequential patterns mining technique – which aims to find sub-sequences that appear frequently in a sequence database – on a web log access sequence. In web server logs, a visit by a client is recorded over a period of time and the discovery of sequential patterns allows web-based organisations to predict user visit patterns, which helps in targeting advertising aimed at groups of users based on these patterns.

Traditional sequential patterns mining approaches such as Apriori-based algorithms [5,6] encounter the problem that multiple scans of the database are required in order to determine which candidates are actually frequent. Pei et al. introduced a compressed data structure called Web Access Pattern tree (or WAP-tree), which facilitates the development of algorithms for mining access patterns from pieces of web logs [7]. Since then, many modifications were proposed in order to further improve efficiency, by eliminating the need to perform any re-construction of intermediate WAP-trees during mining; for example the Position Coded Pre-order Linked Web Access Pattern mining algorithm [8,9], Conditional Sequence mining algorithm [10] and the modified Web Access Pattern (mWAP) algorithm [11].

Typically the methods described above mine the complete set of web access patterns and, in many cases, a large set of access patterns is not intuitive and not necessarily very easy to understand or use. Also, questions that can be asked about web access patterns mining are: What is the inherent relation among web access patterns? Is there a general representation of the access patterns? And are there any other novel patterns that can be discovered based on these access patterns? These questions point out some challenges for web access patterns mining methods and indicate further research directions in web data modelling.

Sequential Patterns Graph (SPG) has been proposed as the minimal representation of a collection of sequential patterns as well as describing the inherent relationship among sequences [12]. The SPG approach is extended in this work to model all access patterns and called Web Access Patterns Graph (WAP-Graph). The primary focus of this paper is on the search for concurrency in web access patterns through Concurrent Access Patterns (CAP) mining. The WAP-Graph approach itself is then extended to model all concurrent access patterns and called CAP-Graph.

Related work is highlighted in the next section to provide

Dr Jing Lu is a Research Fellow at the School of Computing and Communications in Southampton Solent University, East Park Terrace, Southampton, SO14 0YN, UK (e-mail: Jing.Lu@solent.ac.uk).

Dr Malcolm Keech, Faculty of Creative Arts, Technologies and Science, University of Bedfordshire, Park Square, Luton, LU1 3JU, UK (Malcolm.Keech@beds.ac.uk).

Professor Weiru Chen, Faculty of Computer Science and Technology, Shenyang Institute of Chemical Technology, Shenyang, 110142, P.R. China (WillC@china.com).

relevant background on sequential patterns post-processing and web access patterns mining. The modelling of access patterns is presented in section III through WAP-Graph and its construction algorithm. The idea of concurrency is introduced to the web access context in section IV, and both the mining and modelling methods are proposed for concurrent access patterns, culminating in the novel CAP-Graph representation. An experimental evaluation using synthetic and real datasets is given in section V which showcases the results of CAP mining and modelling. The paper draws to a close by suggesting a framework for web access patterns post-processing while making brief conclusions.

II. RELATED WORK

This section will describe two types of related work to provide background and further motivation – one of them is from the authors' previous research on sequential patterns post-processing.

A. Sequential Patterns Post-Processing

Frequent patterns mining is one of the most important knowledge discovery techniques, searching for sub-structures that appear *frequently* (i.e. more than a given support threshold). For example, frequent itemset mining [13] aims to find frequent *itemsets* in a transaction database and sequential patterns mining [5,6,14] aims to find *sub-sequences* that appear frequently in a sequence database.

With the successful implementation of efficient and scalable algorithms for mining frequent itemsets and sequential patterns, it was natural to consider extending the scope of previous study to more structured data mining, for example through post-processing. There are two aspects to such sequential patterns post-processing here: the modelling of sequential patterns in a graphical way [12] and discovering new structured patterns beyond these sequences [15].

In sequential patterns mining, given a customer sequence database and user-specified minimum support (*minsup*), a set of sequential patterns (i.e. frequently occurring sub-sequences within the database) can be discovered. All sequential patterns under the specified *minsup* can be generated from the maximal sequence set – sequential patterns which are not contained in any other sequential patterns. Thus, a directed acyclic graph called Sequential Patterns Graph (SPG) was defined to represent the maximal sequence set [12].

SPG can be viewed as the graphical representation of the relationship among sequential patterns. Nodes (i.e. items or itemsets) of SPG correspond to elements in a sequential pattern and directed edges were used to denote the sequence relation between two elements. Two special types of nodes called a start node (represented by double circles) and a final node (represented by a bold circle) were defined to indicate the beginning and end of maximal sequences. Any path from a start node to a final node corresponds to one maximal sequence. Fig. 1 gives an example of a SPG which shows the graphical components used.

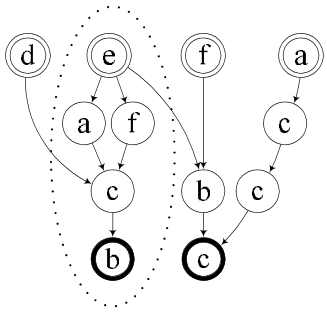

Fig. 1 A sequential patterns graph

The segment circled by a dotted line in Fig. 1 represents maximal sequences *eacb* and *efcb* in particular, while also including all of the other sequential patterns *e*, *a*, *c*, *f*, *b*, *ea*, *ef*, *ec*, *eb*, *ac*, *fc*, *ab*, *fb*, *cb*, *eac*, *efc*, *eab*, *efb* and *ecb*.

The significance of SPG is not limited to the minimal representation of a collection of sequential patterns. It also motivates the discovery of further relationships among sequential patterns and this led to the novel approach to sequential patterns post-processing called Post Sequential Patterns Mining or PSPM [15]. Some sequential patterns may be supported by the same data sequence, and these have been called *concurrent* patterns; while some others may not possibly occur in the same data sequence, and these have been called *exclusive* patterns. Furthermore, some sequential patterns may occur more than once in a data sequence, such that an iterative relationship can be expressed, and this was called an *iterative* pattern. *Structural Relation Patterns* is the general designation of patterns [16] that consists of sequential patterns, concurrent patterns, exclusive patterns, iterative patterns and their composition.

PSPM does not mine structural relation patterns directly from the data, as it first takes advantage of existing sequential patterns mining methods. Further analysis of the inherent relationships behind these sequential patterns resulted in the identification of new structures, such as concurrent patterns, and a corresponding data mining method was proposed in [16].

B. Web Access Patterns Mining

Web access patterns have been defined by Pei et al. based on the problem statement of sequential patterns mining [7]. In general, a web log can be regarded as a sequence of user identifier and event pairs. Each piece of web log is a sequence of events from one user or session in timestamp ascending order. Pei et al. modelled pieces of web logs as sequences of events and mined the sequential patterns beyond a certain support threshold. For convenience, we first introduce some definitions and notation for representing the user's action when visiting a web site [7].

Let $P=\{p_1,\ldots, p_q\}$ be a set of q items (e.g. web pages). An *Access Sequence* $S=<as_1,\ldots, as_l>$ is an ordered list of items (web pages), where $as_i \in P$, $i \in \{1,\ldots, l\}$ and l is called the *length* of the access sequence. An access sequence of length l is also called an l-sequence.

An access sequence $S_1=<X_1, X_2,\ldots, X_m>$ is *contained* in another sequence $S_2=<Y_1, Y_2,\ldots, Y_n>$ if $m\le n$ and there exist integers $1\le i_1<i_2<\ldots<i_m\le n$ such that $X_j=Y_{i_j}$ $(1\le j\le m)$, and it is denoted by $S_1\angle S_2$. If sequence S_1 is contained in sequence S_2, then S_1 is called a sub-sequence of S_2 and S_2 a super-sequence of S_1.

A *Web Access Sequence Database* (WASD) is a set $\{S_1, S_2, \ldots, S_u\}$, where each S_i $(1\le i\le u)$ is an access sequence. The *support* in WASD of any given access sequence S is defined as $\mathrm{Sup}_{\mathrm{WASD}}(S)=|\{S_i:S\subseteq S_i\}|/u$, where $|\ldots|$ denotes the number of access sequences. Given a fraction *minsup* $(0<minsup\le 1)$ as the minimum support threshold, S is called an *Access Pattern* in WASD if $\mathrm{Sup}_{\mathrm{WASD}}(S)\ge minsup$. An access pattern is called a *maximal access pattern* if it is not contained in any other access patterns.

Problem Statement. The problem of web access patterns mining is: given a web access sequence database WASD and a minimum support threshold, *minsup*, mine the complete set of access patterns in WASD.

Due to the importance of its application, web access patterns mining has been extensively studied in the literature and there exists a diversity of algorithms. Most of them are either Apriori-based algorithms or WAP-tree algorithms. Without referring to any specific algorithms, Example 1 illustrates the nature of web access patterns mining using the notation and definitions above.

Example 1. Given a small web log that recorded user access to seven web pages labelled as $\{a, b, c, d, e, f, g\}$ respectively. Let WASD={<*abdac*>, <*eaebcac*>, <*babfaec*>, <*afbacfc*>} with a *minsup* of 50%. Fig. 2 shows the set of all access patterns in this case, presented at levels of the same length of sequences.

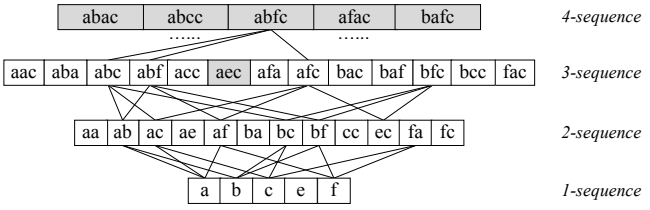

Fig. 2 Web access patterns with 50% *minsup*

The lines in the figure represent the containing relationships between access patterns; for example, *abc*∠*abfc*, *afc*∠*abfc* and *ab*∠*abf* etc. The access patterns within the shadow boxes are maximal access patterns, i.e. *abac*, *abcc*, *abfc*, *afac*, *bafc* and *aec*. This means that at least 50% of users visited web pages in the above sequences.

III. WEB ACCESS PATTERNS MODELLING

Almost all of the studies related to web access patterns mining focus on improving the efficiency of mining methods and there is little work on post-processing web access patterns; for example on how to visualise these access patterns or how to discover structural knowledge based on these access patterns. Therefore, a novel model called Web Access Patterns Graph (WAP-Graph) is proposed as a graphical representation of web access patterns as well as providing the means to represent the inherent relations among the access patterns.

A. Web Access Patterns Graph

The web can be modelled naturally as a directed graph, consisting of a set of abstract nodes (web pages) joined by directional edges (hyperlinks). The idea of sequential patterns graph can also be used for the modelling of web access patterns and we introduce the following definition.

Definition 1. Given a Maximal Access Patterns Set (MAPS) – a collection of access patterns that is not contained by other access patterns – Web Access Patterns Graph (WAP-Graph) is defined as the graphical representation of the MAPS. It is a *5-tuple* expressed as *WAP-Graph*=(*V, E, S, F, δ*), where

1) *V* is a nonempty set of nodes (web pages). Each element of an access pattern in MAPS corresponds to one node in V and each node in WAP-Graph corresponds at least to one element of an access pattern in MAPS.
2) *E* is a set of directed edges (hyperlinks). The relation of any two adjacent elements in an access pattern of MAPS corresponds to the directed edge of two nodes in WAP-Graph. Any one directed edge corresponds to the sequential relation of at least one pair of adjacent elements in an access pattern of MAPS.
3) *S* is a set of start nodes, S⊆V, and S≠∅. There are no start nodes that have the same label in WAP-Graph (if there are, they should be considered as the same node).
4) *F* is a set of final nodes, F⊆V, and F≠∅. There are no final nodes that have the same label in WAP-Graph (if there are, they should be considered as the same node).
5) *δ* is a function from a set of directed edges to a set of pairs of nodes. *δ* can also be defined as a map function of V→V, which indicates the relations between any two nodes.

For any node in WAP-Graph, the subsequent paths of it cannot be the same, and the ancestor paths of it cannot be the same either. For each pair of different nodes in WAP-Graph, if they have same label, there must be different ancestor paths or subsequent paths of them.

Example 2. The maximal access patterns from Example 1 are *aec*, *abac*, *abcc*, *abfc*, *afac* and *bafc*. Fig. 3 shows a WAP-Graph that corresponds to the MAPS and, therefore, the complete set of access patterns in Fig. 2.

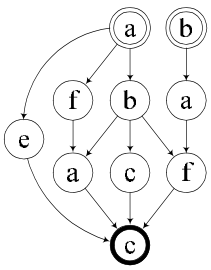

Fig. 3 WAP-Graph for access patterns in Fig. 2

With reference to Fig. 3, it is seen that nodes of WAP-Graph correspond to elements in an access pattern and directed edges are used to denote the access relation between two web pages. As with SPG, a start node (represented by double circles) and a final node (represented by a bold circle) are defined to indicate the beginning and end of maximal access sequences. Any path from a start node to a final node corresponds to one maximal access pattern.

B. WAP-Graph Construction

The approach to construction of a WAP-Graph is described below.

1) *Initialisation*. Determine the longest length *l* of access patterns in MAPS – represent one of the longest access patterns by a directed graph G – sort the remaining patterns in order of length.
2) *Construction*. For the next available access pattern *ap* in MAPS, find any common prefix and/or postfix with G – if they share a common prefix/postfix, then use the Algorithm below to construct the next transitional graph model G – otherwise represent *ap* by a separate graph G' and set G=G∪G'.
3) *Iteration*. For the remaining access patterns in MAPS, which have the same or shorter length – i.e. *l*, *l*-1, *l*-2, etc. – repeat Step 2 incrementally until there are no access patterns left in MAPS. The final result G is the Web Access Patterns Graph, WAP-Graph.

Algorithm 1 WAP-Graph Construction Algorithm

Input: An access pattern *ap* from a maximal access patterns set MAPS and a transitional graph model G
Output: New directed graph G after incremental construction
Procedure:

preS=common prefix of *ap* and G
postS=common postfix of *ap* and G
elemS=*ap*-*preS*-*postS*
Represent *elemS* by the directed graph G'
If *preS* is not empty
    Add a directed edge from the last node of *preS* in G to the first node of G'
If *postS* is not empty
    Add a directed edge from the last node of G' to the first node of *postS* in G
This new directed graph includes a new pattern *ap* and is called G.

Using the modelling method and construction algorithm above, the WAP-Graph for MAPS={*aec*, *abac*, *abcc*, *abfc*, *afac*, *bafc*} can be constructed step-by-step. Fig. 4 is a graphical illustration of this procedure and highlights the above approach, where the dotted lines represent the new edges in the transitional model.

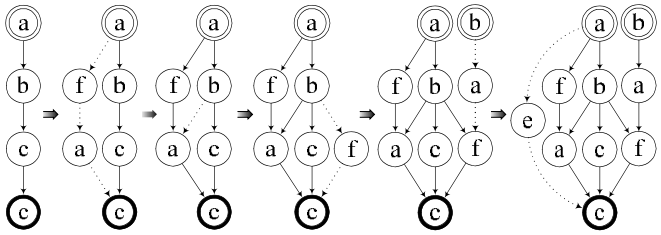

Fig. 4 WAP-Graph construction for MAPS={*aec*, *abac*, *abcc*, *abfc*, *afac*, *bafc*}

It is worth noting that the initialisation and construction phases select the first/next available access pattern, where there is not always a unique choice. For this reason, the result of access patterns modelling can yield different but equivalent graphical representations.

WAP-Graph can be viewed as the visual embodiment of the relationship among access patterns on the web. It is the minimal representation of a collection of access patterns. The significance of WAP-Graph is not limited to graphical representation of web access patterns mining results, as the inherent relationships among access patterns can also be represented.

As a further example, it can be seen from Fig. 2 that some access patterns share the same prefix, e.g. *ab*, *ae* and *af* share the same prefix *a*, which means at least 50% of users visited web pages in the sequence *ab*, *ae* or *af*. Here web pages *b*, *e*, and *f* may not have linkage directly, but they were *all* accessed through *a*. Is there any relationship among access patterns which share the same prefix/postfix? Are users who visited web page *a* likely to visit all of the web pages *b*, *e*, and *f* subsequently? If so, then *b*, *e* and *f* are potentially in a concurrent relationship.

IV. CONCURRENT ACCESS PATTERNS MINING AND MODELLING

Concurrency is an important aspect of some system behaviour. For example, discovering patterns of concurrent behaviour from traces of system events is useful in a wide variety of software engineering tasks, including user interaction modelling and software process improvement, and this can be extended to user interaction on the web. This section will propose a corresponding concurrent access patterns mining method with associated modelling.

A. Concurrent Access Patterns

The notion of *concurrence* will be developed first in this sub-section, which leads to *concurrent access patterns* and the related theorem. For the following definitions, the set of access sequences $\{S_i\}$ $(1 \le i \le u)$ in the web access sequence database WASD is used and it is assumed that $\{ap_1, ap_2, \ldots, ap_k\}$ is a set of access patterns AP which are not contained in each other.

Definition 2. The *concurrence* of access patterns ap_1, $ap_2, \ldots, ap_k$ is defined as the fraction of access sequences $\{S_i\}$ that contain *all* of the access patterns. This is denoted by

$$concurrence(ap_1, ap_2, \ldots, ap_k) = |\{S_i: ap_j \angle S_i, \forall j\ (j=1,2,\ldots,k)\}|/u$$

where $ap_j \angle S_i$ represents access pattern ap_i contained in access sequence S_i.

The user-specified minimum support threshold (i.e. *minsup*) has been used as the frequency measurement for mining frequent itemsets, sequential patterns and web access patterns. Another percentage value, minimum *concurrence* threshold ($0<mincon\leq 1$), is introduced below to check the concurrency of access patterns within the whole web access sequence database.

Definition 3. Let *mincon* be the user-specified minimum concurrence. If

$$concurrence(ap_1, ap_2, \ldots, ap_k) \geq mincon$$

is satisfied, then $ap_1, ap_2, \ldots, ap_k$ are called *Concurrent Access Patterns*. This is represented by $CAP_k=[ap_1+ap_2+\ldots+ap_k]$, where k is the number of access patterns which occur together and the notation '+' represents the concurrent relationship between them.

Definition4. A concurrent access pattern $CAP_k=[a_1+a_2+\ldots+a_k]$ is *contained* in concurrent access pattern $CAP_{(k+m)}=[b_1+b_2+\ldots+b_{k+m}]$ if $a_i \angle b_j$, for $1\leq i\leq k$ and $1\leq j\leq (k+m)$. This is denoted by $CAP_k \angle CAP_{(k+m)}$. Concurrent access patterns are called *maximal* if they are not contained in any other concurrent access patterns.

Example 3. Consider WASD={<*abdac*>, <*eaebcac*>, <*babfaec*>, <*afbacfc*>} from Example 1 and assume a *mincon* of 50%. Since both access sequences <*babfaec*> and <*afbacfc*> support access patterns *abac* and *abfc*, then: *concurrence*(*abac*, *abfc*)=2/4=50%. Therefore, they constitute a concurrent access pattern given by $CAP_2=[abac+abfc]$, where *abac* and *abfc* share the same prefix *a* and postfix *c* within CAP_2.

The following theorem, derived from Lu et al. [17], gives a more concise CAP representation.

Theorem 1. If k access patterns from a set AP make up a concurrent access pattern $CAP_k=[x\alpha_1 y+x\alpha_2 y+\ldots+x\alpha_k y]$, where ($x\alpha_i y \in$ AP, $1\leq i\leq k$; $\alpha_i \in$ AP; $x,y \in$ AP or $x,y=\varnothing$), then it can be further represented as $x[\alpha_1+\alpha_2+\ldots+\alpha_k]y$ and $\alpha_1,\alpha_2,\ldots,\alpha_k$ called the *k branches* of CAP_k.

For example, by having another look at $CAP_2=[abac+abfc]$ in Example 3, one can take out the common prefix *ab* and postfix *c* to yield the modified representation *ab*[*a*+*f*]*c*.

Note that in a concurrent access pattern such as *ab*[*a*+*f*]*c*, the order of branches *a* and *f* is indefinite. Therefore *ab*[*a*+*f*]*c* can appear in a web access sequence database in the form of *abafc* or *abfac*. Also note that, while *abafc* or *abfac* cannot be discovered from traditional web access patterns mining with a *minsup* of 50%, they make up the new pattern *ab*[*a*+*f*]*c* under a *mincon* of 50%.

B. CAP Mining

Using the above definitions, the problem of concurrent access patterns mining can be stated as follows: given a web access sequence database WASD and web access patterns mining results (i.e. web access patterns which satisfy a minimum support threshold), *concurrent access patterns mining* aims to discover the set of all concurrent access patterns beyond a given user-specified minimum concurrence, *mincon*.

The three main steps for mining concurrent access patterns are described below.

1) *Calculation of Access Patterns Supported by Access Sequences (SuppAP).* Access patterns which are supported by a given access sequence S ($S \in$ WASD) under *minsup* are computed and denoted by:

   SuppAP(S)={ap: $ap \in AP \wedge ap \angle S$}.

   The union of SuppAP(S_1), SuppAP(S_2),…, SuppAP(S_n), where $S_i \in WASD$ ($1\leq i\leq n$, n is the number of access sequences), is the set of access patterns supported by WASD.
2) *Determination of Concurrent Access Patterns (CAP).* Each SuppAP(S_i) can be viewed as a transaction, i.e. the unordered set of access patterns supported by access sequence S_i. Thus, the problem of finding the concurrent access patterns (under user-specified *mincon*) from WASD becomes one of mining frequent itemsets from a transaction database under *mincon*=*minsup*. Therefore, the traditional frequent itemset mining approach can be adapted for this step.
3) *Finding Maximal Concurrent Access Patterns (MaxCAP).* According to the containing relationship among sequences, the CAPs need to be simplified in order to obtain the maximal concurrent access patterns. This can be achieved using Definition 4 by:

   i) Deleting the concurrent access patterns CAP_k which are contained by other concurrent access patterns $CAP_{(k+m)}$.

   ii) Deleting the access patterns in CAP_i contained by other access patterns within the same CAP_i.

Example 4. An example is given to explain how to mine concurrent access patterns based on traditional web access patterns mining. In order to do this, the web access sequence database WASD in Example 1 and access patterns in Fig. 2 are considered again for illustration, with a *mincon* of 50%. The following steps correspond to the three described above.

Step 1. Using the results from Fig. 2, SuppAP are calculated for every access sequence in order. These access patterns are shown in Table I.

Step 2. Access patterns sets which are supported by at least two web access sequences in WASD is the requirement for concurrency here, given the *mincon* of 50%. The second and third access sequences support access patterns *a, b, c, e, aa, ab, ac, ae, ba, bc, ec, aac, aba, abc, aec, bac* and *abac*, which therefore constitute concurrent access patterns.

Step 3. Containing relationships exist among the above concurrent access patterns however, e.g. $a \angle aa \angle aac \angle abac$; $b \angle ba \angle bac \angle abac$; $e \angle ec \angle aec$; $c \angle bc \angle abc$ … (as indicated by the lines in Fig. 2). Therefore the sub-sequences *a, aa, aac, b, ba, bac, e, ec, c, bc, abc, aba* can be deleted, which results in the maximal concurrent access pattern $CAP_2=[aec+abac]$.

TABLE I
WEB ACCESS PATTERNS SUPPORTED BY EACH SEQUENCE

Web Access Sequence	SuppAP
<*abdac*>	*a*, *b*, *c*, *aa*, *ab*, *ac*, *ba*, *bc*, *aac*, *aba*, *abc*, *bac*, *abac*
<*eaebcac*>	*a*, *b*, *c*, *e*, *aa*, *ab*, *ac*, *ae*, *ba*, *bc*, *cc*, *ec*, *aac*, *aba*, *abc*, *acc*, *aec*, *bac*, *bcc*, *abac*, *abcc*
<*babfaec*>	*a*, *b*, *c*, *e*, *f*, *aa*, *ab*, *ac*, *ae*, *af*, *ba*, *bc*, *bf*, *ec*, *fa*, *fc*, *aac*, *aba*, *abc*, *abf*, *aec*, *afa*, *afc*, *bac*, *baf*, *bfc*, *fac*, *abac*, *abfc*, *afac*, *bafc*
<*afbacfc*>	*a*, *b*, *c*, *f*, *aa*, *ab*, *ac*, *af*, *ba*, *bc*, *bf*, *cc*, *fa*, *fc*, *aac*, *aba*, *abc*, *abf*, *acc*, *afa*, *afc*, *bac*, *baf*, *bcc*, *bfc*, *fac*, *abac*, *abcc*, *abfc*, *afac*, *bafc*

Using the same approach to simplification, another two concurrent access patterns can be determined which are supported by (i) the second & fourth access sequences and (ii) the third & fourth access sequences respectively, namely CAP_2=[*abac*+*abcc*] and CAP_3=[*abac*+*abfc*+*afac*+*bafc*].

C. CAP Modelling

The use of graphical representation has led to the development of a sequential patterns model (SPG) and web access patterns model (WAP-Graph) that explore the inherent relationships among sequential patterns and web access patterns respectively. The idea is adapted now for modelling concurrent access patterns. The definition of WAP-Graph is extended to define Concurrent Access Patterns (CAP) Graph and followed by an example for illustration.

Definition 5. CAP-Graph is a graphical representation of concurrent access patterns denoted by a 7-tuple expressed as *CAP-Graph*=(*V, E, S, F, δ, I, O*), where the 5-tuple in Definition 1 is extended to CAP-Graph with two special types of nodes:

6) *I* is a set of *in-link* nodes, I⊆V, with two or more incoming access relations applied to concurrent paths to allow no more than one outgoing access relation.
7) *O* is a set of *out-link* nodes, O⊆V, allowing independent execution between concurrent paths, modelled by connecting two or more outgoing access relations.

These new graphical components are shown in Fig. 5, where the '+' represents concurrency in the CAP-Graph.

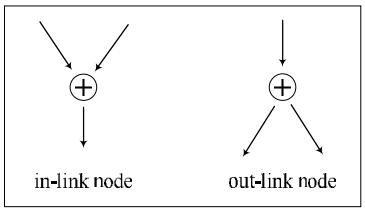

Fig. 5 CAP-Graph components

In-link nodes synchronise incoming access relations, while out-link nodes represent a fork to multiple outgoing relations.

Theorem 1 indicates the direction for a more concise representation of CAPs. The features of WAP-Graph make it straightforward to model the common prefix/postfix elements; therefore concurrent access patterns modelling can use a similar construction method and algorithm. The underlying difference is that the WAP-Graph procedure considers an access pattern from MAPS each time, while CAP-Graph construction considers them from a maximal concurrent access patterns set. In addition, there are two extra steps needed to mark in-link and out-link nodes, which are shown in bold in Algorithm 2 below.

Algorithm 2 CAP-Graph Construction Algorithm

Input: An access pattern *ap* from a maximal concurrent access patterns set MaxCAP and a transitional graph model G
Output: New directed graph G after incremental construction
Procedure:
preS=common prefix of *ap* and G
postS=common postfix of *ap* and G
elemS=*ap*-*preS*-*postS*
Represent *elemS* by the directed graph G'
If *preS* is not empty
{Add a directed edge from the last node of *preS* in G to the first node of G';
Mark the last node of *preS* as an *out-link* node}
If *postS* is not empty
{Add a directed edge from the last node of G' to the first node of *postS* in G;
Mark the first node of *postS* as an *in-link* node}
This new directed graph includes a new pattern *ap* and is called G.

Example 5. Using the extension of the WAP-Graph method and the CAP-Graph construction algorithm above to model the concurrent access pattern CAP_3=[*abac*+*abfc*+*afac*+*bafc*].

1. *Initialisation*. Determine the longest access patterns in CAP_3 – they all have the same length for this example. Represent one of them by a directed graph G – e.g. *abfc* – see Fig. 6(i).

2. *Construction*. For the next available access pattern, *abac* in CAP_3, find any common prefix *preS* with G – this is *ab*; and find any common postfix *postS* – this is *c*. Taking out *preS* and *postS* from *abac*, the remaining part *elemS*=*a* can be represented by a directed graph G'. Add a directed edge from the last node of *preS* in G (i.e. *b*) to the first node of G' (i.e. *a*) and mark *b* as an *out-link* node with '+'. Also add a directed edge from the last node of G' (i.e. *a*) to the first node of *postS* (i.e. *c*) and mark *c* as an *in-link* node with '+'. The result of this step is the graph shown in Fig. 6(ii), where the dotted lines represent the new edges in the transitional model.

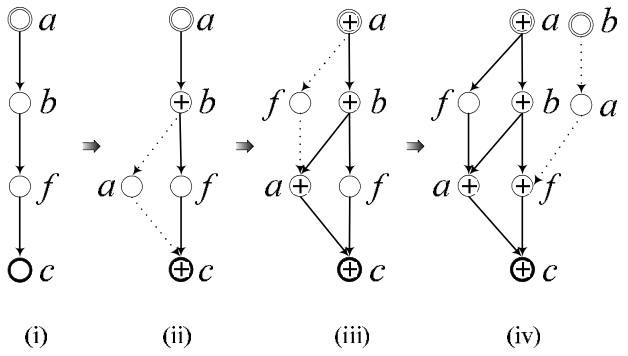

Fig. 6 Modelling of CAP_3=[*abac+abfc+afac+bafc*]

3. *Iteration*. For the remaining access patterns in turn, i.e. *afac* and *bafc*, construct new graphs in a similar manner: e.g. for *afac*, find any common prefix *preS* with G – this is *a*; and find any common postfix *postS* – this is *ac*. Taking out *preS* and *postS* from *afac*, the remaining part *elemS*=*f* can be represented by a directed graph G'. Add a directed edge from the last node of *preS* in G (i.e. *a*) to the first node of G' (i.e. *f*) and mark *a* as an *out-link* node with '+'. Also add a directed edge from the last node of G' (i.e. *f*) to the first node of *postS* (i.e. *a*), marking *a* as an *in-link* node to give Fig. 6(iii). The final result or CAP-Graph is represented in Fig. 6(iv).

V. EXPERIMENTS

We examine the effects of the proposed WAP-Graph construction, CAP mining and modelling on synthetic and real datasets. The method and algorithms are implemented using Microsoft Visual C++ where, to mine the web access patterns, we use the Prefix*Span* algorithm [14]. The executable code is available from the *IlliMine system package*, a partially open-source data mining package: http://illimine.cs.uiuc.edu/, last accessed 7 June 2009.

A. Synthetic Dataset

To evaluate the scope of WAP modelling and concurrent access patterns mining, we first performed experiments on large-scale synthetic datasets. Synthetic sequence data was drawn from the IBM Almaden data generator – http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html, last accessed 7 June 2009 – which has been used in many sequential patterns mining and web access patterns mining studies [7,14,18].

This synthetic dataset generator produces a database of sequences whose characteristics can easily be controlled by the user. It requires the specification of the average number of transactions in a sequence |C|, the average number of items in a transaction |T|, the average length of maximal potentially large sequences |S|, the number of different items |N| and the number of data sequences |D|.

We generated four datasets for testing and performed web access patterns mining on each; the results from modelling access patterns for one of them, C10-T8-S8-N1K-D10K, are presented below in Table II. This dataset contains 10,000 data sequences and 1,000 different items, where the average number of items in a transaction (i.e. event) is set to 8 and the average number of transactions per data sequence is set to 10. It can be seen that the number of WAP-Graph components – nodes and directed edges – increases significantly as *minsup* decreases.

TABLE II
WAP-GRAPH COMPONENTS FOR SYNTHETIC DATASET

minsup	APs	MAPS	Nodes	Edges
5%	481	470	450	40
4.5%	557	536	484	88
4%	666	638	530	157
3.5%	881	831	600	317
3%	1268	1187	664	662
2.5%	2135	2003	755	1462
2%	4210	3983	955	3438
1.5%	9718	9400	1328	8663

The CAP mining method was tested on the same dataset using seven different minimum supports (i.e. *minsup* ranging from 2% to 5%) and, for each *minsup*, the minimum concurrence (*mincon*) was set in a corresponding range. Table III shows the number of concurrent access patterns mined under different *minsup* and *mincon* combinations. It demonstrates that CAP mining can be used to discover concurrency between access patterns, beyond that which web access patterns mining is designed to achieve.

TABLE III
CAP NUMBERS UNDER DIFFERENT MINSUP AND MINCON

mincon	*minsup* =2%			*minsup* =2.5%			*minsup* =3%			*minsup* =3.5%			*minsup* =4%			*minsup* =4.5%			*minsup* =5%		
	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4	CAP_2	CAP_3	CAP_4
2%	13731	4417	31	13828	4417	31	13795	4417	31	13761	4417	31	13695	4424	31	13598	4428	31	13446	4439	31
2.5%	7581	1328	2	7581	1328	2	7807	1328	2	7785	1328	2	7738	1328	2	7707	1328	2	7679	1329	2
3%	4535	408		4535	408		4535	408		4649	408		4612	408		4583	408		4563	408	
3.5%	2859	150		2859	150		2859	150		2859	150		2891	150		2872	150		2847	150	
4%	1835	50		1835	50		1835	50		1835	50		1835	50		1842	50		1827	50	
4.5%	1204	18		1204	18		1204	18		1204	18		1204	18		1204	18		1210	18	
5%	841	6		841	6		841	6		841	6		841	6		841	6		841	6	

Table III highlights the relationship between *mincon* and *minsup*: only when *mincon*≤*minsup* does a concurrent access pattern make sense. This is because both the *mincon* and *minsup* thresholds can be considered as *frequencies* for general frequent patterns mining – *mincon* for mining concurrent access patterns and *minsup* for mining web access patterns – and the former are derived from the latter. Note that, for a range of *mincon* thresholds under a given *minsup*, the concurrent access patterns may be different.

Fig. 7 uses a logarithmic scale to illustrate that the number of CAPs decreases exponentially with the increase in *mincon* when (e.g.) *minsup* is 5%.

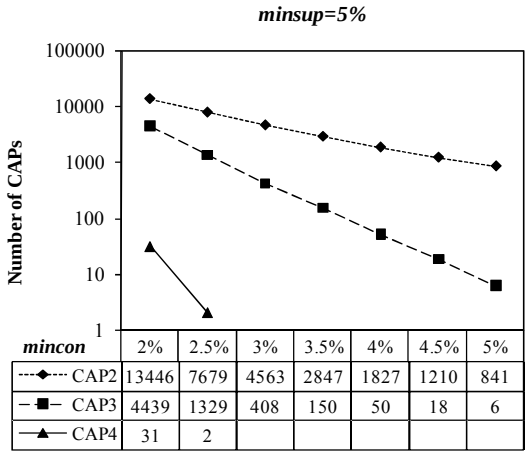

Fig. 7 Logarithmic curves for CAP2/3/4 numbers

B. Real Dataset

A real-world dataset pertaining to customer web access data is used to evaluate the effectiveness of CAP mining and modelling under various concurrence thresholds. This dataset is obtainable from Blue Martini's Customer Interaction System at http://cobweb.ecn.purdue.edu/KDDCUP/, last accessed 7 June 2009.

Three categories of data, i.e. Customer, Orders and Click-stream information, are collected by the Blue Martini application server and further details are provided in [19]. In general, customers can have multiple sessions. Each session can have multiple page views and multiple orders, while each order can have multiple order lines, and each order line is a purchase record of one product with a quantity of one or more. We focus on click-stream analysis here.

The web page access dataset, *Clicks*, was selected from the original Blue Martini information and used in the experiments. It comprises *Request Date, Request Sequence*, *Content ID* (page ID) and *Request Template* (web page name), which have been extracted from the click-stream information. This dataset is 72.8MB in total and contains 234,489 customers, 776,985 click-streams and 77 web pages. Table IV lists the format and sample content of the *Clicks* data.

TABLE IV
FORMAT/CONTENT OF CLICKS DATASET

Request Date	Sequence	Request Template	Content ID
30/01/2000	1	main/home.jhtml	1389
30/01/2000	2	main/lifestyles.jhtml	8171
30/01/2000	3	main/assortment.jhtml	9897
30/01/2000	1	main/ home.jhtml	1389
30/01/2000	2	main/vendor.jhtml	8263
30/01/2000	3	main/vendor.jhtml	8263
30/01/2000	4	main/vendor.jhtml	8263
30/01/2000	5	main/search_results.jhtml	1425
30/01/2000	6	main/departments.jhtml	9901
30/01/2000	7	main/search_results.jhtml	1425
30/01/2000	8	products/productDetailLegwear.jhtml	10771
30/01/2000	9	products/productDetailLegwear.jhtml	10771
…	…	…	…
01/02/2000	1	main/home.jhtml	1389
01/02/2000	2	main/vendor.jhtml	8263
01/02/2000	3	main/vendor.jhtml	8263
01/02/2000	4	main/vendor.jhtml	8263
01/02/2000	5	main/vendor.jhtml	8263
01/02/2000	6	main/vendor.jhtml	8263
01/02/2000	7	main/assortment.jhtml	9897
01/02/2000	8	products/productDetailLegwear.jhtml	10771
01/02/2000	9	main/vendor.jhtml	8263
…	…	…	…
01/02/2000	22	main/boutique.jhtml	8267
01/02/2000	23	main/boutique.jhtml	8267
…	…	…	…

Access Patterns Mining

Table V illustrates the relationship between *minsup* and the number of web access patterns found in the *Clicks* dataset following mining. The table shows the sequences of length *k* where, e.g. when *minsup*=4.5%, there are eight access patterns with a unique item, fifteen access patterns with two items and four access patterns with three items.

TABLE V
ACCESS PATTERNS UNDER VARIOUS MINSUP

minsup	Number of *k*-sequences				
	1-sequence	2-sequence	3-sequence	4-sequence	5-sequence
4.5%	8	15	4		
4%	9	19	8		
3.5%	10	21	14	2	
3%	10	25	18	4	
2.5%	14	34	32	7	1
2%	17	43	54	22	4

Concurrent Access Patterns Mining

Table VI shows the extent of the concurrent access patterns mining results across a range of *mincon* values, which are set equal to *minsup* in each case.

TABLE VI
CAP MINING ON THE WEB PAGE ACCESS DATASET

CAP_k	*mincon=minsup*					
	4.5%	4%	3.5%	3%	2.5%	2%
CAP_2	5	7	7	16	33	75
CAP_3	1		1	4	9	23
CAP_4					2	5

It can be seen from Table VI that, under the *s*ame *mincon* and *minsup* (i.e. 4.5% here), five CAP_2 and one CAP_3 are discovered after performing concurrent access patterns mining; these are:

[<9901> + <1389> <8267>]
[<8267> + <1389> <9901>]
[<1389> <10771> + <1389> <8267>]
[<1389> <9897> + <9897> <9897>]
[<1389> <9897> + <1389> <10771>]
[<1425> + <9901> + <10771>]

Concurrent Access Patterns Modelling

Using the CAP modelling method, the concurrent access patterns above can be represented graphically and three of them are shown in Fig. 8.

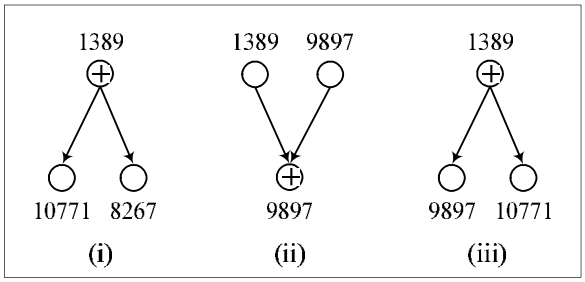

Fig. 8 CAP-Graphs for CAP_2 results

The out-link nodes and in-link node are marked with a '+' and connect the concurrent paths. For example, Fig. 8(i) shows that at least 4.5% of users requested web pages in this behaviour by first accessing page 1389, and then accessing pages 10771 and 8267 concurrently. With respect to Fig. 8(ii), at least 4.5% of users requested web pages in this behaviour – they accessed pages 1389 and 9897 concurrently, then accessed page 9897 again in other sessions.

This type of information was not readily available when access patterns mining was performed on the web data, where the following access patterns were obtained: <1389> <10771>, <1389> <8267>, <1389> <9897> and <9897> <9897>. However, when CAP mining is applied, hidden patterns are obtained which can be represented further as in Fig. 9, cross-referencing with Table IV.

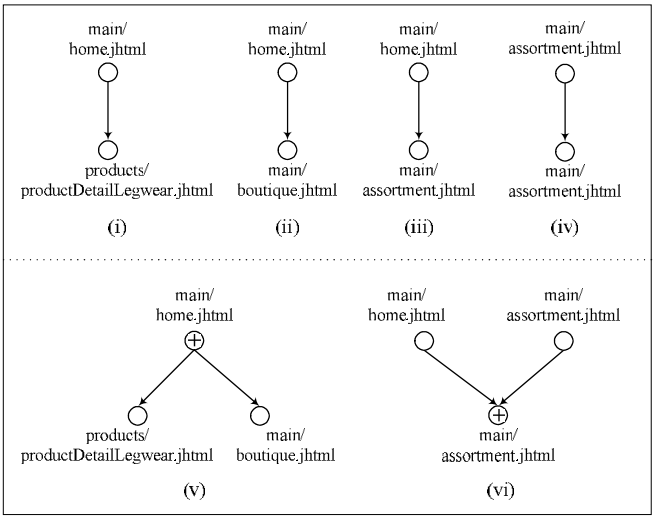

Fig. 9 Comparison of WAP-Graph (top) and CAP-Graph (bottom) for the web page access dataset

Fig. 9 shows regular activity when web pages are accessed. Fig. 9(i) to (iv) are the result of access patterns mining – in the form of WAP-Graph; Fig. 9(v) and (vi) correspond to CAP modelling following concurrent access patterns mining – in the form of CAP-Graph. It is important to note that there is no direct linkage between web pages *main/boutique.jhtml* and *products/productDetailLegwear.jhtml* for example, but they were both accessed through the *main/home.jhtml* page to form a concurrent relationship – these three web pages make up the new concurrent access pattern as modelled in Fig. 9(v).

It can be seen that CAP mining and modelling can be applied in web analysis to search for and represent user navigation trails associated with the most frequently accessed patterns which display concurrency beyond certain thresholds. This demonstrates the potential of concurrent access patterns mining, where more complex structural relationships can be discovered and modelled from web usage data.

VI. CONCLUSIONS

Web data mining aims to discover useful information from web pages, hyperlink structures and usage logs [1]. Consequently, web mining tasks are usually categorised into three main types: web content mining, web structure mining and web usage mining. Searching for access patterns is one aspect of web usage mining and the work presented here has developed this further in the context of *web access patterns post-processing*, where we outline a corresponding framework below.

Traditional web access patterns mining is performed first on the web access sequence database and the resulting access patterns are taken as input to web access patterns post-processing – boxed area of Fig. 10.

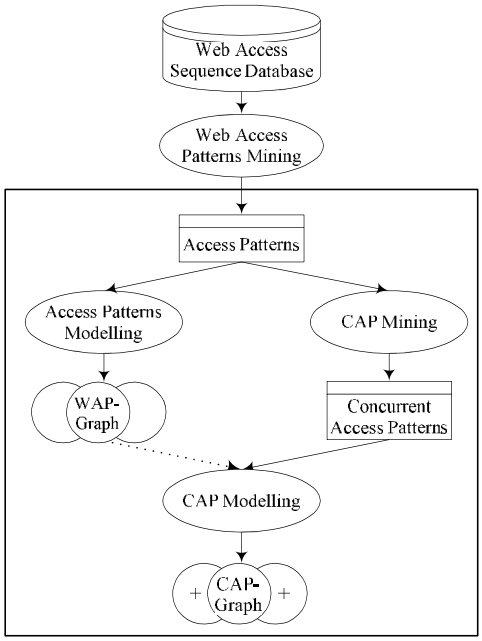

Fig. 10 Web access patterns post-processing framework

One branch of the post-processing is access patterns modelling, which constructs the graphical representation for web access patterns called WAP-Graph (section III). The other branch pursues mining of concurrency relationships from access patterns and the result is called Concurrent Access Patterns (CAP). Fig. 10 shows that WAP-Graph provides a bridge to CAP modelling, where the former construction approach is extended to represent the concurrent access patterns and called CAP-Graph (section IV). The experiments on synthetic sequence and real web access data have demonstrated that CAP mining and modelling are suitable for predictive tasks in web usage mining (section V).

There is not necessarily a boundary between web content mining, web structure mining and web usage mining. For example, web structure can be treated as part of web content and web usage can be considered to belong to web structure [4]. It follows that web content, structure and usage mining can all be pursued separately or combined together in a single application. Thus, the approach proposed in this paper can potentially be applied in web structure mining to deal with the structure of hyperlinks across the Internet. The *objects* in this case are web pages and *links* are either in-, out- or co-citation, where two web pages are linked to from the same page [20]. Co-citation can be represented by a concurrency relationship in CAP modelling; while *in-* is equivalent to the in-link node of CAP-Graph; and *out-* corresponds to the out-link node.

The potential remains for the extension of web access patterns post-processing to embrace other structural relation patterns [16] where, for example, the concept of *exclusive access patterns* may have a specific application. This would imply further data mining and modelling techniques to discover novel patterns from the web through its content, structure and usage.

REFERENCES

[1] B. Liu, *Web Data Mining – Exploring hyperlinks, contents and usage data*. Book series: Data-Centric Systems and Applications. Springer Berlin/Heidelberg, 2007, ch. 1, 12.

[2] R. Kosala and H. Blockeel, "Web Mining Research: a survey," *ACM SIGKDD Explorations Newsletter*, vol. 2 Issue 1, June 2000.

[3] J. Srivastava, R. Cooley, M. Deshpande and P-T. Tan, "Web Usage Mining: Discovery and applications of usage patterns from web data," *SIGKDD Explorations*, 2000, 1(2):12-23.

[4] J. Wang, Y. Huang, G. Wu and F. Zhang, "Web Mining: Knowledge discovery on the Web," Systems, Man and Cybernetics, *IEEE SMC '99 Conference* Proceedings, (Tokyo, Japan, 1999), IEEE, vol. 2, 137-141.

[5] R. Agrawal and R. Srikant, "Mining sequential patterns," Proceedings of *the 11th International Conference on Data Engineering*, (Taipei, Taiwan, 1995), IEEE Computer Society Press, 3-14.

[6] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and performance improvements," Proceedings of *the Fifth International Conference on Extending Database Technology*, (Avignon, France, 1996), Springer-Verlag, vol. 1057, 3-17.

[7] J. Pei, J. Han, B. Mortazavi-asl and H. Zhu, "Mining access patterns efficiently from web logs," In Proceedings of *the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (Kyoto, Japan, 2000), Springer, 396-407.

[8] C. I. Ezeife and Y. Lu, "Mining web log sequential patterns with position coded pre-order linked WAP-tree," *International Journal of Data Mining and Knowledge Discovery*, 2005, 10, 5-38.

[9] W. Wang and P. T. Cao-Thai, "Novel position-coded methods for mining web access patterns," *IEEE International Conference on Intelligence and Security Informatics*, 2008, 194-196.

[10] X. Tan, M. Yao and J. Zhang, "Mining maximal frequent access sequences based on improved WAP-tree," Proceedings of *the Sixth International Conference on Intelligent Systems Design and Applications*, IEEE Computer Society Press, 2006, vol. 1, 616-620.

[11] J. D. Parmar and S. Garg, "Modified web access pattern (mWAP) approach for sequential pattern mining," INFOCOMP – *Journal of Computer Science*, June, 2007, 6(2): 46-54.

[12] J. Lu, X. F. Wang, O. Adjei and F. Hussain, "Sequential patterns graph and its construction algorithm," *Chinese Journal of Computers*, 2004, 27(6): 782-788.

[13] R. Agrawal, T. Imielinski and A. Swami, "Mining association rules between sets of items in large databases," Proceedings of the 1993 ACM SIGMOD, 207-216.

[14] J. Pei, J. W. Han, B. Mortazavi-Asl and H. Pinto, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," Proceedings of *the 17th International Conference on Data Engineering*, (Heidelberg, Germany, 2001), IEEE Computer Society Press, 215-224.

[15] J. Lu, O. Adjei, W. R. Chen and J. Liu, "Post Sequential Patterns Mining: A new method for discovering structural patterns," Proceedings of *the Second International Conference on Intelligent Information Processing*, (Beijing, China, 2004), Springer-Verlag, 239-250.

[16] J. Lu, W. R. Chen, O. Adjei and M. Keech, "Sequential patterns post-processing for structural relation patterns mining," *International Journal of Data Warehousing & Mining*, 2008, 4(3): 71-89.

[17] J. Lu, W. R. Chen and M. Keech, "Graph-based modelling of concurrent sequential patterns," *International Journal of Data Warehousing & Mining*, to appear.

[18] P. Tang, and M. P. Turkia, "Mining frequent web access patterns with partial enumeration," Proceedings of *the 45th Annual Southeast Regional Conference*, (Winston-Salem, North Carolina, USA, 2007), ACM, 226-231.

[19] R. Kohavi, C. Brodley, B. Frasca, L. Mason and Z. J. Zheng, "KDD-Cup 2000 Organizers' Report: Peeling the onion," *SIGKDD Explorations*, vol. 2, Issue 2, 86-98, 2000.

[20] L. Getoor, "Link Mining: a new data mining challenge," *SIGKDD Explorations*, vol. 4, Issue 2, 2003.