Applications of Concurrent Access Patterns in Web Usage Mining

Jing Lu[1], Malcolm Keech[2], and Cuiqing Wang[3]

[1] Southampton Solent University, Southampton UK, SO14 0YN
[2] University of Bedfordshire, Park Square, Luton UK, LU1 3JU
[3] Shenyang University of Chemical Technology, Shenyang China, 110142
Jing.Lu@solent.ac.uk, Malcolm.Keech@beds.ac.uk,
wangcuiqing@126.com

Abstract. This paper builds on the original data mining and modelling research which has proposed the discovery of novel structural relation patterns, applying the approach in web usage mining. The focus of attention here is on concurrent access patterns (CAP), where an overarching framework illuminates the methodology for web access patterns post-processing. Data pre-processing, pattern discovery and patterns analysis all proceed in association with access patterns mining, CAP mining and CAP modelling. Pruning and selection of access patterns takes place as necessary, allowing further CAP mining and modelling to be pursued in the search for the most interesting concurrent access patterns. It is shown that higher level CAPs can be modelled in a way which brings greater structure to bear on the process of knowledge discovery. Experiments with real-world datasets highlight the applicability of the approach in web navigation.

Keywords: web access patterns (WAP) post-processing, concurrent access patterns (CAP), CAP mining and modelling, WAP pruning, knowledge discovery.

1 Introduction

Web usage mining is the process of applying data mining techniques to the discovery of usage patterns from web data to understand and better serve the needs of user navigation on the Internet [1]. The method and algorithms for web usage mining are normally divided into three stages: data collection and pre-processing, pattern discovery and patterns analysis. Depending on the ultimate goals and the desired outcomes, there are different types of pattern discovery and analysis techniques such as association rules generation, sequential patterns mining, cluster analysis and visitor segmentation, and classification and prediction based on web/user transactions [2].

Association rules generation aims to discover unordered correlation among the *frequent* items in a transaction database. In the context of web usage mining, a transaction is a group of web page accesses and an item is a single web page. Therefore, association rules refer to the sets of pages that are accessed together under a minimum support threshold, even though they may not be directly linked with each other.

L. Bellatreche and M.K. Mohania (Eds.): DaWaK 2013, LNCS 8057, pp. 339–348, 2013.

Sequential patterns mining attempts to identify frequently observed sequential occurrence of items across *ordered* transactions (or sequence databases) over time [3]. This technique has been applied in the web usage context to discover web access patterns (WAP) and to capture frequent navigation paths among user trials. WAP-tree was introduced to facilitate development of algorithms for mining access patterns from pieces of web logs [4]. Analysis of these access patterns allows Internet-based organisations to understand user preferences and predict future visit patterns.

Sometimes the scale and level of detail of access patterns found makes it difficult to identify specific navigation trails in web logs. Structural relation patterns have been introduced to extend the search for more complex patterns hidden behind large sequences of data [5]. Discovering these patterns is based on post-processing of sequential patterns mining results – those sequential patterns supported by the same data sequence have been called *concurrent patterns* and could be useful for analysing clickstreams. While much of the research in web mining has its focus on *efficiency*, this paper will pursue concurrent access patterns (CAP) mining and modelling methods which both discover and represent new structures not found elsewhere.

Some related work is highlighted in the next section to provide the relevant background on web access patterns mining and structural relations, culminating in the formal definition of concurrent access patterns. Following the novel framework for WAP post-processing in section 3, a CAP mining and graph construction methodology is presented with a worked example to illustrate the approach taken. An experimental evaluation using real datasets is given in section 4 which showcases the effectiveness of CAP mining and modelling at generating new and stimulating results. The paper draws to a close by summarising and making brief conclusions.

2 Related Work

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

2.1 Web Access Patterns

Web access patterns have been defined by Pei et al. based on the problem statement of sequential patterns mining [4]. 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 chronological order.

Let $P = \{p_1, p_2, \ldots, p_t\}$ be a set of t items (e.g. web pages). An *access sequence* $AS = \langle as_1, as_2, \ldots, as_m \rangle$ is an ordered list of itemsets (web pages), where $as_i \in P$, $1 \leq i \leq m$. The number of items in a sequence is known as the *length* of the access sequence. A sequence $AS_1 = \langle X_1, X_2, \ldots, X_u \rangle$ is *contained* in $AS_2 = \langle Y_1, Y_2, \ldots, Y_v \rangle$ if $u \leq v$ and $X_i \subseteq Y_j$ for all i, $1 \leq i \leq u$ and corresponding j, $1 \leq j \leq v$, and it is denoted by $AS_1 \angle AS_2$.

A *Web Access Sequence Database* (WASD) is a set $\{AS_1, AS_2, \ldots, AS_n\}$, where each AS_i $(1 \leq i \leq n)$ is an access sequence. The *support* in WASD of any given AS is defined as $Sup_{\mathrm{WASD}}(AS) = |\{AS_i: AS \subseteq AS_i\}|/n$, where $|\ldots|$ denotes the number of sequences. AS is

called an *Access Pattern* in WASD with respect to a minimum support threshold *minsup* ($0<minsup\leq 1$) if $Sup_{WASD}(AS)\geq minsup$. Web access patterns mining thus discovers a set of patterns from a given WASD under a user-specified *minsup*.

Example 1. Given a small web log that recorded user access to seven web pages labelled {*a*, *b*, *c*, *d*, *e*, *f*, *g*} respectively and let WASD={<*abcfge*>, <*adcbef*>, <*abfe*>, <*bfg*>}. Table 1 shows the set of all access patterns mined with a *minsup* of 50% and this will be used as a running worked example throughout the paper.

Table 1. Access patterns from a sample WASD

Sequence	Access Patterns Supported by each Sequence (SuppAP), *minsup*=50%
AS_1=*abcfge*	*a,b,c,e,f,g,ab,ac,ae,af,be,bf,bg,ce,cf,fe,fg,**abe**,**abf**,**ace**,**acf**,afe,bfe,bfg,abfe*
AS_2=*adcbef*	*a,b,c,e,f,ab,ac,ae,af,be,bf,ce,cf,**abe**,**abf**,**ace**,**acf***
AS_3=*abfe*	*a,b,e,f,ab,ae,af,be,bf,fe,abe,abf,afe,bfe,abfe*
AS_4=*bfg*	*b,f,g,bf,bg,fg,bfg*

2.2 Structural Relations and Concurrent Access Patterns

Structural relation patterns have been defined as a general designation of patterns that consists of sequential patterns, *concurrent* patterns, *exclusive* patterns, *iterative* patterns and their composition [5]. This sub-section provides the necessary background for concurrent patterns in the web access context.

Following the notation used in the previous sub-section, new ordering relationships based on access patterns can be predicated as follows: given a *Web Access Sequence Database* WASD = {AS_1, AS_2, …, AS_n}, let α, β be two of the access patterns mined from WASD with minimum support threshold *minsup* and assume that α, β are not contained in each other. With regard to a particular sequence $AS\in$ WASD, access patterns α and β have a *concurrent* relationship if and only if both of them have occurred in *AS*, i.e. $(\alpha\angle AS)\wedge(\beta\angle AS)$ is true. This is represented by $[\alpha+\beta]_{AS}$, where the notation '+' represents the concurrent relationship [5].

Definition 1 (Concurrence). *The **concurrence** of access patterns α and β is defined as the fraction of sequences that contains both of the access patterns. This is denoted by* $concurrence(\alpha,\beta) = |\{AS_k:(\alpha\angle AS_k)\wedge(\beta\angle AS_k)\}|/n$, *where* $AS_k\in WASD$, $1\leq k\leq n$ *and n is the total number of access sequences.*

The user-specified *minsup* provides the threshold for frequency measurement when mining frequent itemsets, sequential patterns and web access patterns. Another fractional value, the minimum *concurrence* threshold, *mincon* ($0<mincon\leq 1$) is used to check the concurrent relationships of access patterns.

Definition 2 (Concurrent Access Patterns). *Let mincon be the user-specified minimum concurrence. The concurrence of access patterns* $ap_1, ap_2, \ldots, ap_r$ *is defined as* $concurrence(ap_1, ap_2, \ldots, ap_r) = |\{AS_k:[ap_1+ap_2+\ldots+ap_r]\}|/n$, *where* $AS_k\in WASD$ *and* $1\leq k\leq n$. *If* $concurrence(ap_1, ap_2, \ldots, ap_r)\geq mincon$ *is satisfied, then* $ap_1, ap_2, \ldots$ *and* ap_r *are called **concurrent access patterns**. This is represented by* $CAP_r = [ap_1+ap_2+\ldots+ap_r]$, *where there is no particular order for the access patterns.*

Example 2. Consider WASD={<*abcfge*>, <*adcbef*>, <*abfe*>, <*bfg*>} from Example 1 and assume a *mincon* of 50%. For the access patterns *abe*, *abf*, *ace* and *acf* shown in bold in Table 1, according to Definition 1, *concurrence*(*abe*,*abf*,*ace*,*acf*) = 2/4 = 50%. Therefore, together they constitute the concurrent access pattern given by CAP_4 = [*abe*+*abf*+*ace*+*acf*].

Definition 3 (Maximal CAPs). *The concurrent access patterns represented by* $CAP_k = [a_1+a_2+\ldots+a_k]$ *are contained in* $CAP_{k+m} = [b_1+b_2+\ldots+b_{k+m}]$ *if* $a_i \angle b_j$ *for all i,* $1 \leq i \leq k$ *and corresponding j,* $1 \leq j \leq (k+m)$. *This is denoted by* $CAP_k \angle CAP_{k+m}$. *Concurrent access patterns are called* ***maximal CAPs*** *if they are not contained in any other concurrent patterns.*

Note that the CAP_4 from Example 2 is also maximal, where this type of pattern represents a navigation trail associated with the most frequently accessed patterns, displaying concurrency beyond the user-specified threshold.

3 Web Access Patterns Post-processing

With the successful implementation of efficient and scalable algorithms for mining web access patterns, it is natural to consider extending the scope of previous study to more structured data mining for enhanced knowledge discovery in web usage.

3.1 Framework

Fig. 1 introduces a novel framework for *web access patterns post-processing* which can be described through its four rows. First, depending on the nature of the log files, data pre-processing involves different types of tasks such as data fusion and cleaning, user/web page identification and data transformation [2]. The 1st row completes with access patterns mining using traditional sequential patterns mining techniques.

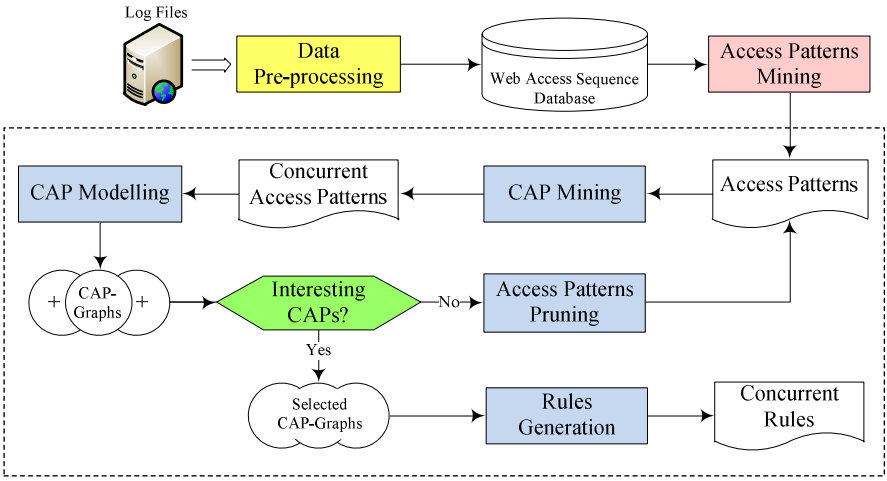

Fig. 1. WAP post-processing framework

The initial cycle of concurrent pattern discovery proceeds through CAP mining and modelling (2nd row of Fig. 1), which yields CAP-Graphs for analysis (3rd row, see sub-section 3.3). If the CAPs are deemed uninteresting, then access patterns pruning is undertaken before another cycle of pattern discovery is instigated. If at this stage the patterns analysis is successful, through selection of more interesting CAP-Graphs, then concurrent rules can be generated (4th row) and the process completes.

3.2 From WAP to CAP Mining

The method used to mine concurrent access patterns is derived from that for concurrent sequential patterns [5] and has been highlighted previously in the web mining context [6]. It is illustrated step-by-step below with the running worked example.

Step 1. Calculation of Access Patterns Supported by each Sequence.

Access patterns which are supported by a sequence AS_i (i.e. $AS_i \in$ WASD, $1 \le i \le n$) are computed and denoted by: $SuppAP(AS_i) = \{ap: ap \in AP \wedge ap \angle AS_i\}$.

The results of this for Example 1 are shown in the second column of Table 1.

Step 2. Determination of Concurrent Access Patterns.

Each $SuppAP(AS_i)$ can be viewed as a transaction, i.e. the unordered set of access patterns supported by data sequence AS_i. Thus, the problem of finding the concurrent access patterns which satisfy the specified minimum concurrence (*mincon*) becomes one of mining frequent itemsets under *minsup=mincon*.

To calculate CAPs with *mincon*=50% means finding the groups of access patterns which occur together in at least 50% of the data sequences. In Example 1, both data sequences AS_1 and AS_2 support the common set of access patterns which constitutes CAP = [*a+b+c+e+f+ab+ac+ae+af+be+bf+ce+cf+abe+abf+ace+acf*].

Step 3. Finding Maximal Concurrent Access Patterns.

According to the containing relationship among sequences, the CAPs need to be simplified in order to deduce the maximal concurrent patterns. Using Definition 3, these can be obtained by deleting the concurrent access patterns which are contained by other CAPs, then deleting the access patterns in particular CAPs (in turn) which are contained by other access patterns within the same CAP.

For example, for the CAP from the previous step, the following contained relationships exist: $a \angle af \angle acf$, $b \angle ab \angle abe$, $c \angle ce \angle ace$, ... ; therefore this concurrent access pattern can be reduced to the maximal CAP_4 = [*abe+abf+ace+acf*].

3.3 CAP Modelling and Knowledge Representation

The use of graphical models in data mining has motivated the development of a sequential patterns graph, SPG that explores the inherent relationships among these patterns. The idea was adapted in [7] for modelling concurrent sequential patterns.

Each page view can be represented as a node in a graph and the directed edge between two nodes indicates a sequential relation, i.e. user navigation. The first node for each path is called a *start* node and the last is called a *final* node. A node with two or more incoming sequential relations applied to the paths is called a *synchronizer* node, while *fork* nodes allow independent execution between concurrent paths.

Without dwelling on the detail, in this context concurrent access patterns graph (CAP-Graph) is thus a graphical representation of CAPs which maps a function from a set of directed edges to a set of pairs of nodes (see [7]). The definition of CAP-Graph is an extension of that for sequential patterns graph and therefore the method to construct SPG can be adapted for CAP modelling.

For the running worked example, and following initialisation, the intermediate construction and iteration phases proceed as illustrated in Fig. 2. By taking sequential patterns in turn, nodes and directed edges are added step-by-step, culminating in the final CAP-Graph for CAP_4=[*abe+abf+ace+acf*].

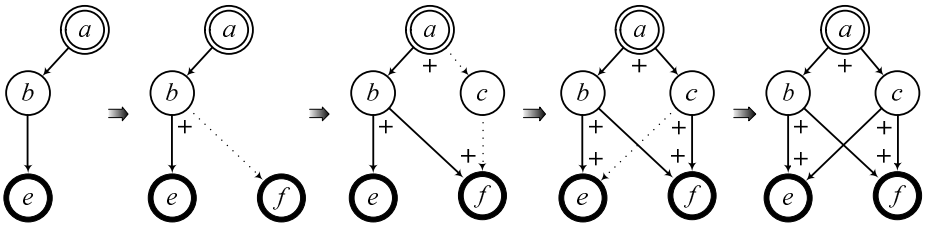

Fig. 2. Modelling CAP_4=[*abe+abf+ace+acf*] using adapted SPG method

We conclude WAP post-processing by indicating another way to simulate knowledge representation here through *concurrent rules*. For example, the following rules can be generated based on the above CAP-Graph: $a \Rightarrow [b+c] \Rightarrow [e+f]$. This may be used to predict visit patterns: after accessing page *a*, visitors are likely to access pages *b* and *c* (in no particular order), followed by further visiting pages *e* and *f*.

4 Experiments and Evaluation

The empirical analysis of the proposed CAP mining and modelling methods was performed on real-world datasets to test their effectiveness as well as to further illustrate the framework and process for web access patterns post-processing.

4.1 Pre-processing Real Datasets

The initial focus is on pre-processing real datasets available from published sources, namely *BMS-WebView-1* and *msnbc.com*.

BMS-WebView-1 was the first of the three KDD CUP 2000 datasets, which is sometimes called "Gazelle" [8]. This dataset contains clickstream data from a former web retailer, *Gazelle.com*, and has been used widely to assess the performance of frequent patterns mining. The particular file here contains 59,602 sequences and has been pre-processed already, which is convenient, although the inherent meaning of the 497 items no longer features in the competition web site.

The msnbc.com anonymous web dataset was drawn from [9]. The original data comes from Internet Information Server (IIS) logs for msnbc.com and contains 989,818 data sequences in total for one day. Each sequence in the dataset corresponds

to page views of a user during that 24-hour period and each event (item) in the sequence corresponds to a user's request for a page.

For web access patterns mining, the original ordering of the pages is important. If a page appears twice in succession in the same sequence, then only the first request will remain following pre-processing. There are many such sequences for the msnbc dataset which contain repetitive/adjacent items or single items only. Therefore, pre-processing has been extended for the experiments which reduces data sequences by 60%. The final file has been divided fairly equally into four datasets called *msnbc1.dat* to *msnbc4.dat*, with each processed separately, to make computation more manageable.

4.2 BMS-WebView-1

All experiments have been conducted on a 2.5GHz Intel Core i5 processor with 4GB main memory running Windows 7. Appropriate use has been made of *PrefixSpan* for access patterns mining, an open-source data mining package available from [10].

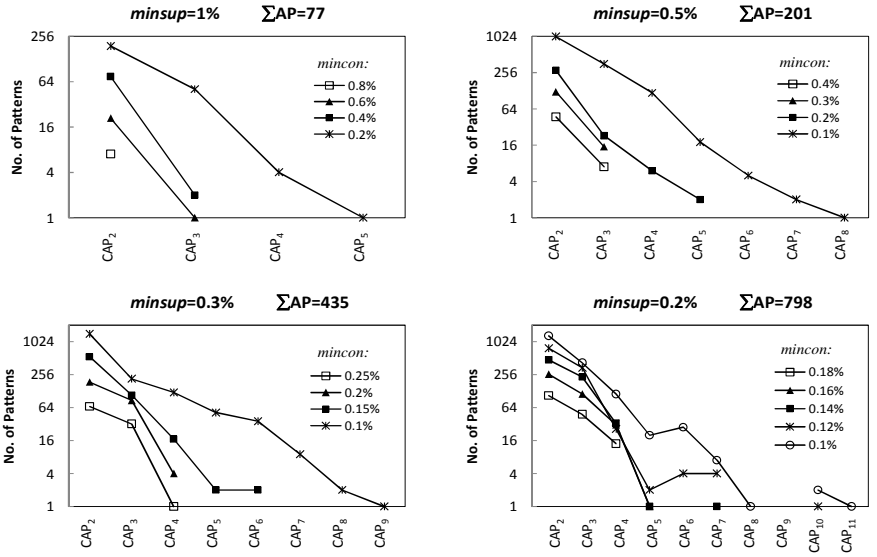

Fig. 3. CAPs mined from BMS-WebView-1 under various *minsup* and *mincon*

Several tests have been performed on BMS-WebView-1 across a range of *minsup* values – no CAPs were found beyond a 2% threshold. A summary of the nature of these results is shown in Fig. 3 when *minsup*=1%, 0.5%, 0.3% and 0.2%. It is noted that the number of patterns increases as *mincon* thresholds decrease within the same *minsup*, while no CAPs have been found for this dataset when *mincon*=*minsup*.

It is significant that *higher level* CAPs can be mined for BMS-WebView-1 under decreasing *minsup* and *mincon*, although there are limits for these experiments. For example, when *minsup*=0.1%, the number of access patterns mined reaches nearly 4,000 and run time grows exponentially for CAP mining under the smaller *mincon*. Likewise,

when *minsup*=0.3%, the number of CAPs exceeds 2,000 once *mincon*≤0.08, which suggests a natural cut-off point for presentational purposes here.

Two novel and interesting CAP-Graphs are selected from BMS-WebView-1 results and highlighted in Fig. 4, where CAP_9 and CAP_{11} patterns demonstrate that complex structural relations can be discovered and modelled from web usage data.

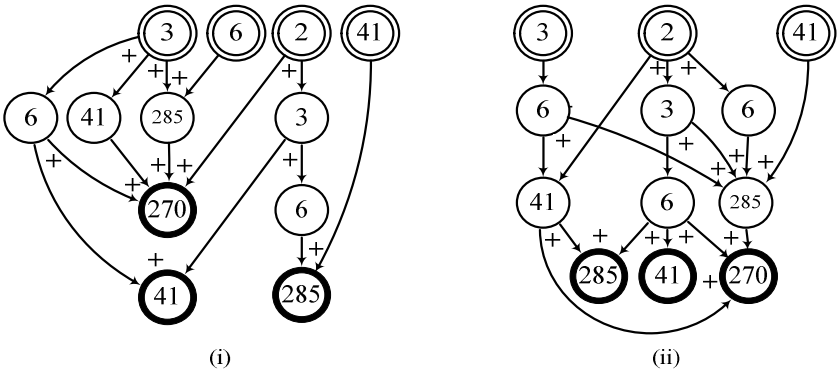

Fig. 4. Sample (i) CAP_9 when *minsup*=0.3% and (ii) CAP_{11} when *minsup*=0.2%; *mincon*=0.1%

4.3 msnbc.com

The second set of tests was performed on the msnbc dataset, where no CAPs were found for *minsup*≥7%. Following preliminary access patterns mining with *minsup*=5%, several CAPs were discovered with repetitive formats and modelling results confirmed these CAPs were not interesting. msnbc is a dense dataset containing only 17 different items and this causes significant repetition in the access patterns. Therefore, to improve the mining results, WAPs have been pruned by removing repetitive and adjacent items while deleting single length patterns not useful in the context of concurrency.

CAP mining thus resumes based on the pruned WAPs before CAP modelling once more helps to determine whether there are any interesting patterns. This sub-section presents selected results from the *msnbc1* dataset only, although experiments were also performed on *msnbc2-4* which were essentially equivalent. Initial tests with *minsup*=5% show a progression of the highest level patterns from a single CAP_2 to a pair of CAP_3 to a single CAP_4 then a single CAP_5 as *mincon* decreases from 4% to 1%.

Varying the *minsup* threshold below 5% increases the number of access patterns as well as the potential for CAP discovery. This is best illustrated for *minsup*=1%, where a representative sample of the highest level CAP-Graphs can be selected, as in Fig. 5.

The results are shown *with meaning* this time, where original codes are transposed into the corresponding page category. For *mincon*=1% there was a choice of 21 CAP_3 and the two highlighted in (i) and (ii) give a flavour for the concurrent patterns mined. For *mincon*=0.75% and 0.5% there were only single CAP_5 and CAP_6 respectively, as in (iii) and (iv), each displaying a more interesting structure. And for *mincon*=0.25% there was a choice of four CAP_8, where the two patterns in (v) and (vi) demonstrate that complex structural relations can again be found from real web data.

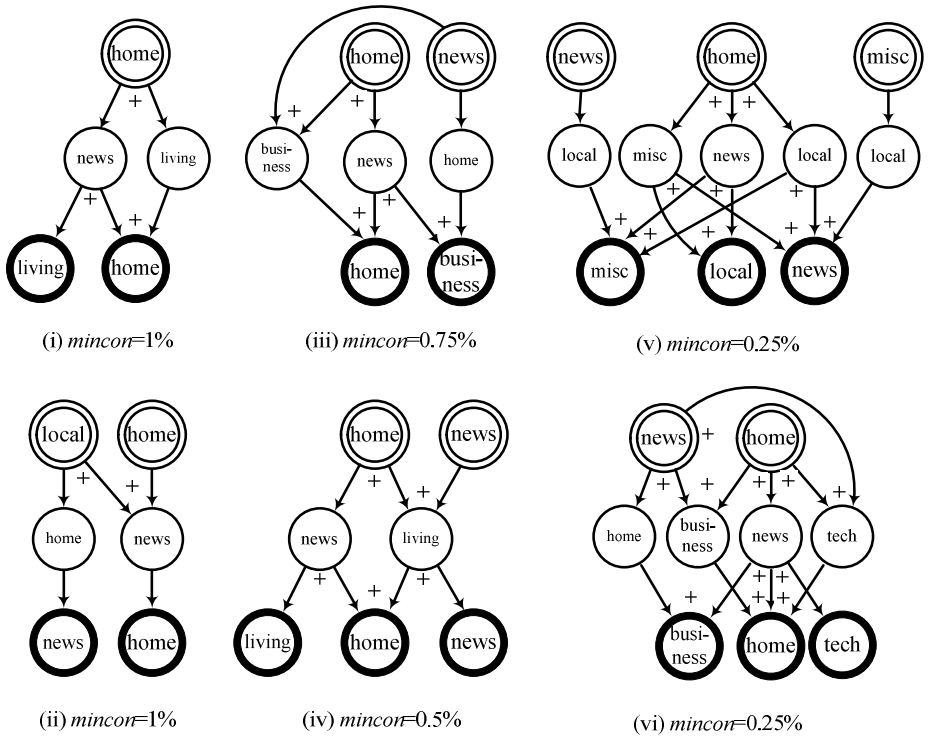

Fig. 5. Examples of CAP-Graphs for msnbc1.dat when *minsup*=1% under various *mincon*

5 Conclusion

The focus of this work has been on the application of original data mining and modelling research to the web mining context. In particular, concurrent access patterns are defined and explored through an approach to web access patterns post-processing which employs a novel framework for knowledge discovery and analysis in Fig. 1. Constituent data mining and modelling techniques yield concurrent access patterns and graphs allowing the question to be raised: are these CAPs interesting? This calls to some extent for a subjective judgement, but careful inspection of CAP-Graphs provides enough evidence to inform a suitable answer.

Experiments have been performed on a number of real-world datasets and two are reported here: BMS-WebView-1 and msnbc.com. Our clear motivation is to evaluate the effectiveness of the techniques while illustrating the stages of the WAP post-processing approach. Pre-processing is required for one of the datasets and initial access patterns mining undertaken for both. The results of CAP mining and modelling are always interesting up to a point, but there can still be many CAP-Graphs to consider and they sometimes exhibit unhelpful repetition. This is where access patterns pruning can come in, which was especially useful for the second dataset.

The extent to which higher level CAPs can be discovered is first examined for BMS-WebView-1, where results are reported in Fig. 3 for $0.2\% \leq minsup \leq 1\%$. The connectivity and structure of the sample CAP-Graphs shown in Fig. 4 highlights the potential for knowledge representation through concurrent patterns, albeit with no attached meaning here for this web retail dataset. Pre-processing is required for msnbc.com before it is divided into four for individual investigations across $1\% \leq minsup \leq 5\%$. When the lower *minsup* value is explored in tandem with decreasing *mincon* then, following access patterns pruning, the progressive nature of structural relations modelled unfolds in Fig. 5. The higher level CAPs convey the increasingly rich navigation of real pages viewed in this media context.

Future work is being considered in three directions: first, while concurrent rules are suggested in this paper, a specific study of their role and generation would be useful. Second, while CAP modelling presents a stimulating visualisation of mining results, there can be many CAP-Graphs to view for large-scale datasets – a more automated approach to CAP selection would be worthwhile. And finally, while the WAP post-processing framework does not limit the number of iterations in pursuit of the most interesting CAPs, a single additional cycle is illustrated in experiments here. Criteria could be developed for further access patterns pruning informed by other types of constraint relevant to concurrency and applied more widely.

References

1. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. SIGKDD Explorations 1(2), 12–23 (2000)
2. Liu, B.: Web Data Mining – Exploring Hyperlinks, Contents, and Usage Data. Book series: Data-Centric Systems and Applications. Springer, Heidelberg (2011)
3. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: 11th International Conference on Data Engineering, pp. 3–14. IEEE Computer Society Press, Taipei (1995)
4. Pei, J., Han, J., Mortazavi-asl, B., Zhu, H.: Mining Access Patterns Efficiently from Web Logs. In: Terano, T., Liu, H., Chen, A.L.P. (eds.) PAKDD 2000. LNCS, vol. 1805, pp. 396–407. Springer, Heidelberg (2000)
5. Lu, J., Chen, W.R., Adjei, O., Keech, M.: Sequential Patterns Post-Processing for Structural Relation Patterns Mining. International Journal of Data Warehousing and Mining 4(3), 71–89 (2008)
6. Lu, J., Keech, M., Chen, W.R.: Concurrency in Web Access Patterns Mining. In: International Conference on Data Mining, vol. 58, pp. 600–609. WASET, Venice (2009)
7. Lu, J., Chen, W.R., Keech, M.: Graph-based Modelling of Concurrent Sequential Patterns. International Journal of Data Warehousing and Mining 6(2), 41–58 (2010)
8. Kohavi, R., Brodley, C., Frasca, B., Mason, L., Zheng, Z.: KDD-Cup 2000 Organizers' Report: Peeling the Onion. SIGKDD Explorations 2(2), 86–98 (2000)
9. UCI KDD Archive, `http://kdd.ics.uci.edu/databases/msnbc/msnbc.html`
10. IlliMine System Package, `http://illimine.cs.uiuc.edu/`