# Chapter 3
# Mining and Cyclic Behaviour Analysis of Web Sequential Patterns


**K. R. Venugopal and K. C. Srikantaiah**

**Abstract** Understanding Web users' behaviour is an important criterion for improving the overall experience of Web users. Web Pattern Mining is one such field that helps us to mine useful behavioural patterns and draw conclusions from them after careful analysis. Efficient Web pattern mining is a challenge taking into consideration the enormous quantities of raw Web log data and explosive growth of information in the Web. In this chapter, we propose a novel algorithm called Bidirectional Growth based mining Cyclic Behaviour Analysis of Web sequential Patterns (BGCAP) that effectively combines these strategies to generate prefetching rules in the form of 2-sequence patterns with Periodicity and threshold of Cyclic Behaviour that can be utilized to effectively prefetch Web pages, thus reducing the users' perceived latency. In other words, BGCAP grow patterns bidirectionally along both ends of detected patterns and allows faster pattern growth with fewer levels of recursion thus eliminating unnecessary candidates and support for efficient pruning of invalid candidates. Due to these facts, BGCAP requires only (log n+1) levels of recursion for mining n Web Sequential Patterns. Our experimental results show that the Web Sequential Patterns and in turn prefetching rules generated using BGCAP is 5–10% faster for different data sizes and generates about 5–15% more prefetching rules than TD-Mine.

## 3.1 Introduction

Data Mining is the process of extracting useful information from a large repository of data. Web mining is one of the types of data mining and can be defined as the process of discovery and analysis of useful information from the data corresponding to World

K. R. Venugopal (✉)
Bangalore University, Jnana Bharathi, Bengaluru 560056, India
e-mail: venugopalkr@gmail.com

K. C. Srikantaiah
SJB Institute of Technology, BGS Health and Education City, Uttarahalli Main Road, Kengeri, Bengaluru 560060, India
e-mail: srikantaiahkc@gmail.com



K. R. Venugopal et al., *Web Recommendations Systems*,
https://doi.org/10.1007/978-981-15-2513-1_3

Wide Web. There are three major types of Web mining: (i) Web Structure Mining, (ii) Web Content Mining and (iii) Web Usage Mining. Web Structure Mining is the process of using graph theory to analyse a Web graph, where the nodes represent Web pages and the edges represent the hyperlinks among them. According to the type of Web structural data, Web structure mining can be divided into two categories: (i) extracting patterns from hyperlinks in the Web and (ii) mining the document structure, i.e. analysis of the DOM tree structure of the pages to describe HTML or XML tag usage. Web Content Mining is the process of mining actual content (text, image or multimedia) from the Web pages of the World Wide Web for information. Web content mining can be divided into two broad categories:

**Agent-Based Approaches**: Agent-based Web mining systems can further be divided into the following three categories: (i) Intelligent Search Agents: These Web agents are tools that interact with and learn the structure of unfamiliar Web pages and retrieve information from a variety of such sites using only general information about the domain. (ii) Information Filtering/Categorization Tools: A number of Web agents use various information retrieval techniques and characteristics of open hypertext Web documents to automatically retrieve, filter and categorize them. Criteria for categorization can be semantic information embedded in link structures and document content to create cluster hierarchies of hypertext documents, and structure an information space. (iii) Personalized Web Agents: This category of Web agents learn user preferences and discover Web information sources based on these preferences and those of other individuals with similar interests.

**Database Approaches**: Database approaches in Web content mining focus on techniques for organizing the semi-structured data on the Web into more structured collections of resources, and using standard database querying mechanisms and data mining techniques to analyse it. There are two techniques in this approach:

(i) Multilevel Database Systems: The main idea behind this technique is that the lowest level of the database contains semi-structured information stored in various Web repositories, such as hypertext documents. At the higher level, metadata or generalizations are extracted from lower levels and organized in structured collections, i.e. relational or page-oriented databases.

(ii) Web Query Systems: Web-based query systems utilize standard database query languages such as SQL for structural information about Web documents.

Web Usage Mining is the automatic discovery of user access patterns from Web logs stored in different servers. Servers collect large volumes of data in their daily operations, generated automatically and collected in server access logs. Other sources of user information include referrer logs which contain information about the referring pages for each page reference, and user registration or survey data gathered *via* scripts. Analysing such data can help in understanding the user's behaviour so that the server can improve its services like recommendation and Web personalization. Most Web analysis tools provide mechanisms for reporting user activity in the servers and various forms of data filtering. Using such tools, it is possible to determine the number of accesses to the server and to individual files, the times of visits and the

domain names and URLs of users. These tools can be placed into two main categories such as follows:

**Pattern Discovery Tools**: These tools discover association rules and sequential patterns from server access logs. Sequential access patterns are essential for understanding and predicting the user's behaviour.

**Pattern Analysis Tools**: Once Web access patterns have been discovered, they need to be understood, visualized and interpreted so that the knowledge gained by the analysis can be utilized to improve the services offered by the Web servers.

The Internet is an extremely large collection of a network of networks, which in turn consist of Web servers which contain huge quantities of data, clients or end-user system which request information or services from the servers and finally client-side and server-side proxies which are additional systems that help and provide better communication amongst clients and servers. Such factors give rise to the necessity of creating intelligent systems that can effectively mine and analyse patterns and Web Usage Mining is one such technique. Mining from the Web includes integrating various data sources such as server access logs, referrer logs, user registration or profile information; and the importance of identifying user sessions or transactions from usage data, site topologies and models of user behaviour. Analysing such mined information results in predicting the users' behaviour and this knowledge in the form of prefetching rules is also extremely helpful in reducing users perceived latency and improving the quality of Web services.

WAP-Mine is one of the FP-growth-based algorithms for mining frequent Web access patterns from Web access database. But in the process of mining frequent Web access patterns, WAP-Mine produces many intermediate data which brings down the efficiency especially at lower support. TD-Mine, which is an extension of WAP-Mine, overcomes this problem and saves more space by reducing the amount of intermediate data generated. But, TD-Mine needs $n$ levels of recursions to mine a pattern of length $(n + 1)$.

In our approach, we first perform preprocessing upon the raw Web logs to generate the session database of each Web user, which lists the Web users (IPs) along with their corresponding sessions. Then, we utilize a bidirectional pattern growth algorithm called UpDown Directed Acyclic Graph (UDDAG) [1] to generate Sequential Web access patterns from this session database and analyse them using Cyclic Model Analysis [2] to find out the Periodicity and Cyclic Behaviour of the mined 2-sequence patterns. The cyclic behaviour analysis can be used to generate Web prefetching rules. Constrains on Sequential Pattern Mining like date and time are specified, so that it returns interesting, more desirable, useful navigational patterns instead of huge and unwanted patterns.

In this chapter, we propose Bidirectional Growth based mining Cyclic behaviour Analysis of Web sequential Patterns (BGCAP) algorithm that utilizes UDDAG based on bidirectional pattern growth approach, which in turn focuses the search on a restricted portion of initial database to avoid expensive candidate generation and test step and is better on mining longer patterns. The strategies used in UDDAG are partitioning, projection and detection. UDDAG only needs *(log n+1)* levels of

recursion thereby significantly reducing the execution time compared to TD-Mine. It takes minimal processing power for mining the complete set of sequential patterns in a large sequence database. These sequential patterns are analysed using Cyclic Model Analysis that deals with the tendency of certain sequential patterns to repeat themselves periodically after definite time intervals. This concept is greatly helpful in predicting future browsing patterns and prefetching Web pages aimed at certain user groups.

## 3.2 Related Works

Several techniques have been proposed for sequential pattern mining. They are mainly of two types: (i) A priori based (ii) Frequent Pattern growth (FP-growth) based. A priori based mining techniques such as a priori-all, GSP [3], SPADE [4], LAPIN-SPAM [5], LAPIN [6], scan the database multiple times. A $n$ size pattern requires $n$ scans of the database and hence these mining techniques are generally inefficient. FP-growth based mining techniques such as FreeSpan, BIDE [7], COBRA [8], PrefixSpan [9], UDDAG [1], etc., utilize a tree-based representation that reflects the original database and two scans are required to construct the tree. From this tree, the sequential patterns are derived without reference to the original database. Changes in the original database can easily be reflected in the tree by incremental analysis. These sequential pattern mining algorithms are used for mining Web access patterns from Web logs.

Cheng et al. [10] proposed an approach that combines a priori-all and clustering for sequential pattern mining to identify the user pattern and cluster users path patterns and make the similar users' cluster as one group in order to reduce the pattern that is effective for users in personalized service. But, it should be taken into account that not all patterns in a cluster may prove to be useful as they can produce erroneous conclusions after pattern analysis.

Gaol [11] explored habits of users using a priori-all algorithm, which first stores the original Web access sequence database for storing non-sequential data. This is based on the fact that the greater the number of combinations produced, the less likely the number of users who perform a combination of these and vice versa. While such an approach is simple and straightforward, such brute force tactics are obsolete as a priori-all algorithms are found to be least efficient with respect to sequential pattern mining.

Pei et al. [12] proposed Web Access Pattern tree (WAP-tree) for efficient mining of access patterns from Web logs. The Web access pattern tree stores highly compressed, critical information for sequential pattern mining. The WAP-tree registers all access sequence counts. There is no need for mining the original database any more as the mining process for all Web access patterns needs to work on the WAP-tree only. Therefore, WAP-mine needs to scan the access sequence database only twice. The height of the WAP-tree is one plus the maximum length of the frequent subsequences in the database. The width of the WAP-tree, i.e. the number of leaves of the tree,

is the number of access sequences in the database. The size of the WAP-tree is much smaller than the size of access sequence database. It is shown that WAP-mine outperforms and has better scalability than GSP.

Xiaoqiu et al. [13] proposed the Improved WAP-tree in the form of highly compressed access sequences and introducing a subtree structure to avoid generation of conditional WAP-tree repeatedly and to generate maximal sequences. Improved WAP-tree excels traditional WAP-tree in time and space and shows better stability as the lengths of patterns vary. Mining frequent access sequences based on WAP-tree needs to scan transaction database only twice.

Yang et al. [14] designed an efficient algorithm Top-Down Mine (TD-mine) which makes use of the WAP-tree data structure for Web access pattern mining. WAP-tree can be traversed both top-down and bottom-up for the extraction of frequent access patterns. In TD-mine, a header table is used to traverse the tree from the root to the leaf nodes and mine patterns where the nodes are frequently accessed.

Liu et al. [15] proposed the Breadth-First Linked WAP-tree (BFWAP-tree) to mine frequent sequences which reflects parent–child relationship of nodes. The proposed algorithm builds the frequent header node links of the original WAP-tree in a Breadth-First fashion and uses the layer code of each node to identify the parent–child relationships between nodes of the tree. It then finds each frequent sequential pattern through progressive Breadth-First sequence search, starting with its first Breadth-First subsequence event. BFWAP avoids re-constructing WAP-tree recursively and shows a significant performance gain.

Vijayalakshmi et al. [16] designed an extended version of PrefixSpan called EXT-Prefixspan algorithm to extract the constraint-based multidimensional frequent sequential patterns in Web usage mining by filtering the dataset in the presence of various pattern constraints. EXT-PrefixSpan then mines the complete set of patterns but greatly reduces the efforts of candidate subsequence generation. This substantially reduces the size of the projected database and leads to efficient processing. EXT-PrefixSpan can be used to mine frequent sequential patterns of multidimensional nature from any Web server log file in the light of obtaining the frequent Web access patterns. However, EXT-PrefixSpan does not specify any particular constraint for consideration, i.e. it is highly generic.

Wu et al. [17] proposed the CIC-PrefixSpan, a modified version of PrefixSpan that mines and generates Maximal Sequential patterns by combining PrefixSpan and pseudo-projection. First, preprocessing is done to categorize the user sessions into human user sessions, crawler sessions and resource-download user sessions for efficient Web sequential pattern mining by filtering out the non-human user sessions, leaving the human user sessions and finding the transactions using Maximum Forward Path (MFP). By utilizing CIC-PrefixSpan, the memory space is reduced and generating duplicate projections to find the most frequent path in the users access path tree is also avoided. It is shown that CIC-PrefixSpan yields accurate patterns with high efficiency and low execution time compared to GSP and PrefixSpan. However, the frequent substructures within a pattern cannot be mined by CIC-PrefixSpan.

Verma et al. [18] designed a new pattern mining algorithm called Single-Level Algorithm for extracting behaviour patterns. These patterns are used to generate

recommendations at run time for Web users. Single-Level Algorithm is designed keeping in mind the dynamic adaptation of focused websites that have a large number of Web pages. It combines preprocessing, mining and analysis to eventually predict the users' behaviour and hence is useful for specific websites and is highly scalable. It is shown to be more efficient than a priori algorithm. However, performing preprocessing on a very large Web log database can be time consuming and too cumbersome to be integrated with mining and analysis.

Nasraoui et al. [19] presented a framework for discovering and tracking evolving user profiles in real-time environment using Web usage mining and Web ontology. Preprocessing is first performed on the Web log data to identify user sessions. Then, profiles are constructed for each user and enriched with other domain-specific information facets that give a panoramic view of the discovered mass usage modes. This framework summarizes a group of users with similar access activities and consists of their viewed pages, search engine queries and inquiring and inquired companies. By mapping some new sessions to persistent profiles and updating these profiles, most sessions are eliminated from further analysis and focusing the mining on truly new sessions. However, this framework is not scalable.

Pitman et al. [20] modified the Bi-Directional Extension (BIDE) algorithm for mining closed sequential patterns in order to identify domain-specific rule sets for recommendation of pages and personalization for Web users in E-commerce. Individual supports are specified for each customer so that products can be recommended for individuals. Also, BIDE creates multidimensional sequences and further increases prediction for customers who do not explicitly specify their needs by using search functionality. However, additional strategies must be explored for identifying the most relevant sequential patterns without an exhaustive exploration of the search space bounded only by minimum support.

Masseglia et al. [21] proposed a Heuristic-based Distributed Miner (HDM), a method that allows finding frequent behavioural patterns in real time irrespective of the number of Web users. Navigational schemas, that are completely adaptable to the changing Web log data, are provided by HDM for efficient frequent sequence pattern mining. Based on a distributed heuristic, these schemas provide solutions for problems such as (i) discovering interesting zones (a great number of frequent patterns concentrated over a period of time) (ii) discovering super-frequent patterns and (iii) discovering very long sequential patterns and interactive data mining. However, the quality of the schemas can further be improved by adapting the candidate population.

Zhou et al. [22] designed an intelligent Web recommender system known as Sequential Web Access based Recommender System (SWARS) for sequential access pattern mining. Conditional Sequence mining (CS-mine) algorithm is used to identify frequent sequential Web access patterns. The access patterns are then stored in a compact tree structure, called Pattern tree, which is then used for matching and generating Web links for recommendations. SWARS has shown to achieve good performance with high satisfaction and applicability.

Yen et al. [23] address the issue of re-discovery of dynamic Web logs due to the obsolete Web logs as a result of deletion of users log data and insertion of new logs.

Incremental mining utilizes previous mining results and finds new patterns from the updated (inserted or deleted) part of the Web logs. A new incremental mining strategy called Incremental Mining of Web Traversal Patterns (IncWTP) is proposed, which makes use of an incremental updating algorithm to maintain the discovered path traversal patterns when entries are inserted or deleted in the database. This is achieved by making use of an extended lattice structure, which is used to store the previous mining results. However, the changes made to the website structure is not reflected in the lattice structure.

Zhang et al. [24] applied the Galois lattice to mine Web sequential access patterns by representing the paths traversed using graphs and compares the performance with that of a priori. Since the a priori-like algorithms frequently scan the entire transaction database to generate candidate patterns, Galois lattice reduces time complexity of closed sequential pattern mining as it needs only one scan.

Jain et al. [25] proposed a technique that employs Doubly Linked Tree to mine Web Sequential patterns. The Web access data available is constructed in the form of doubly linked tree. This tree keeps the critical mining related information in compressed format based on the frequent event count. It is shown that for low support threshold and for large database Doubly Linked Tree mining performance is better than conventional schemes such as a priori-all and GSP. However, Doubly Linked Tree does not work well in a distributed environment.

Jha et al. [26] proposed a Frequent Sequential Traversal Pattern Mining based on dynamic Weights constraint of Web access sequences (FSTPMW) to find the information gain of sequential patterns in session databases. The weight constraints are added into the sequential traversal pattern to control the number of sequential patterns that can be generated in addition to the minimum threshold. FSTPMW is efficient and scalable in mining sequential traversal patterns. However, it should be noted that FSTPMW does not consider levels of support along with the weights of sequential traversal patterns.

Wang et al. [27] proposed a Web personalization system that uses sequential access pattern mining based on CS-mine algorithm. The access patterns are stored in a compact tree structure called Pattern tree, which is then used for matching and generating Web links for recommendations. Pattern tree has shown to achieve good performance with accurate predictability.

Saxena et al. [28] integrated mining and analysis by proposing the One Pass Frequent Episode discovery (FED) algorithm. In this approach, significant intervals for each website are computed first and these intervals are used for detecting frequent patterns (Episodes). Analysis is then performed to find frequent patterns which can be used to forecast the users' behaviour. The FED algorithm is very efficient as it finds out patterns within one cycle of execution itself.

Oikonomopoulou et al. [29] proposed a prediction schema based on Markov Model that extracts sequential patterns from Web logs using website topology. Full coverage is achieved by the schema while maintaining accuracy of the prediction. Since Markov Models are infamous for their precision, the proposed prediction schema fails to deploy a more complex categorization method for each sequential pattern.

Rajimol et al. [30] proposed First Occurrence List Maximum (FOLMax-mine) for mining maximal Web access patterns based on FOL-Mine. It is a top-down method that uses the concept of first occurrence to reduce search space and improve the performance. This is achieved by finding out the Maximal Frequent Path in the patterns generated from the Web logs.

## 3.3 System Architecture

### *3.3.1 Problem Definition*

Given a Web log database $W$ and the set of pages $P = \{p_i: 1 \leq i \leq n\}$ in a Web server, a $session$ is a subset of $P$, denoted by $(p_1, p_2, \ldots p_k)$, where $p_i \in P$, $i \in \{1, \ldots, k\}$. Here, the parentheses are omitted for a session with one page only. A *Web access sequence* $q$ is a list of sessions, denoted by $< q_1 q_2 \ldots q_m >$, where $q_i$ is a session and $q_i \subseteq P$, $i \in \{1, \ldots, m\}$. The number of sessions in $q$ is called the length of $q$.

Given two access sequences $x =< x_1 x_2 \ldots x_j >$ and $y =< y_1 y_2 \ldots y_k >$, $x$ is said to be a $subsequence$ of $y$ and $y$ a $supersequence$ of $x$ if $k \geq j$ and there exists integers $1 \leq i_1 < i_2 < \cdots < i_j \leq k$, such that $x_1 \subseteq y_{i1}, x_2 \subseteq y_{i2}, \ldots, x_j \subseteq y_{ij}$. Here, $x$ is also contained in $y$ which is denoted by $x \alpha y$. A *session database* is a set of tuples $< ip, q >$, where $ip$ is the users IP address which is used as a sequence identifier and $q$ is the users access sequence. A tuple $< ip, q >$ is said to contain an access sequence $\alpha$ if $\alpha q$.

The support of a subsequence $\alpha$, denoted by Support $(\alpha)$, is the number of sequences for which $\alpha$ is a subsequence. A subsequence $\alpha$ is said to be a Web sequential access pattern, when its support is greater than user-specified minimum support $(MinSup)$, i.e. $Support(\alpha) \geq MinSup$. Given a session database and Min-Sup, the objective is to extract Web sequential access patterns using Bidirectional Pattern Growth algorithm, analyse them using Cyclic Model Analysis to determine the periodicity and cyclic behaviour of the 2-sequence patterns and employ them as prefetching rules. *Assumptions*: During the preprocessing operation of the Web logs, it is assumed to be sufficient to consider only those URLs whose domains contain text and images only. Web sequential patterns pertaining to multimedia Web pages need not be mined and hence are filtered out during preprocessing of the raw Web logs.

**Basic Definitions**

**Web Prefetching**: Web Prefetching is the process of fetching Web pages from the Web server before they are actually requested by the users. Periodicity [31]: The periodicity $P$ of the 2-sequence $< p_i p_j >$ is defined as the time period $t$ after which $p_j$ shall be accessed periodically after $p_i$ has been accessed. Example: The periodicity

$P$ between a pair of Web pages $(p_i, p_j)$ is 10 ms indicates that the page $p_j$ is accessed periodically with period 10 ms after page $p_i$ has been accessed.

**Tendency [31]**: The tendency between a pair of Web pages $(p_i, p_j)$ is defined as the line in the trend graph that determines whether the cyclic behaviour of the 2-sequence $< p_i p_j >$ is increasing or decreasing.

**Cyclic Behaviour [31]**: The cyclic behaviour $C$ of the 2-sequence $< p_i p_j >$ is defined as the time that gives the stopping criteria for accessing page $p_j$ after $p_i$ has been accessed. It is calculated using Periodicity and Tendency. The periodicity is increased by adding the value to itself each time the page is accessed. The page will not be accessed after periodicity has reached the limit of cyclic behaviour, i.e. $P \leq C$. Example: The periodicity $P$ between a pair of Web pages $(p_i, p_j)$ is 10 ms, the trend line is decreasing and the cyclic behaviour $C = 50$ ms indicates that accessing of page $p_j$ repeats for every 10 ms after accessing the page $p_i$. This ends when the time $t$ reaches 50 ms.

The system architecture consists of the following components, (i) Web logs, (ii) Preprocessing Engine, (iii) Encoder, (iv) Sequential Pattern Miner, (v) Pattern Analyser and (vi) Prefetching Rules is as shown in Fig. 3.1.

**Web Logs**: A Web log is a large database stored in Web servers that contain details of the transactions of Web users. There are many fields in the Web log database, which conform to either Common Log Format (CLF) or the Extended Common Log Format (ECLF) as shown in Figs. 3.2 and 3.3, respectively. The fields specified by this format are IP address of the destination page, destination URL, code which denote the packet number, protocol which specify the type of network protocol used (e.g. TCP), method which specifies the type of HTTP method used (e.g. GET, POST),

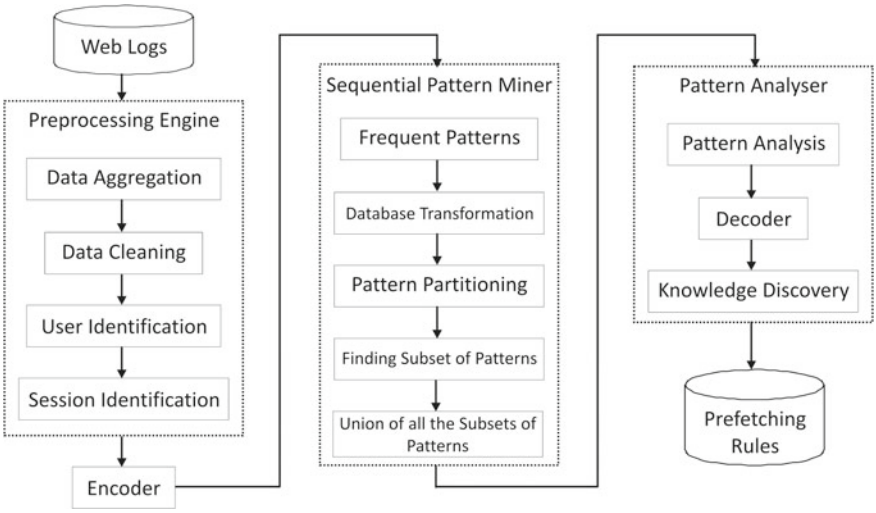


**Fig. 3.1** System architecture

<IP_addr><base_url><date>
<method><file><protocol>
<code><bytes>

**Fig. 3.2** Common log format (CLF)

<IP_addr><base_url><date>
<method><file><protocol>
<code><bytes><referrer>

**Fig. 3.3** Extended common log format (ECLF)

**Fig. 3.4** Access log from the Web server

Web-proxy,debug,packet1307775248.816 363 30.0.1.2
TCP_MISS/200 960 GET
http://www.facebook.com/ajax/typehead/search.php?-
DIRECT/66.220.146.32 application/x-javascript in 11-Jun
12:25;6.76 from 30.0.7.254

web-proxy, debug, packet 1307775249.609 586
30.0.0.223 TCP_MISS/200 397 POST
http://channeltvunetworks.com/all-DIRECT/
38.103.62.170 text/html in 11-Jun 12:25:7.69
from 30.0.7.254

type which specifies the file type, date which gives the date and time when the page was accessed, referrer which gives the IP address of the client which requested the page, and finally the size of the page in bytes. An extract from a Web log is as shown in Fig. 3.4. Web logs are very useful for predicting the behaviour of the clients and hence different mining techniques can be used to find and extract interesting browsing patterns of the users.

**Preprocessing Engine**: This component aggregates the Web logs data from different sources and produces session database as the result using the following steps:

*Data Aggregation*: The Web logs from different sources with different formats are extracted and integrated into a single database so that they pertain to a single format with no redundancy.

*Data Cleaning*: In this step the aggregated data is cleansed, i.e. useless records such as URLs containing images, multimedia, scripts and entries corresponding to crawlers are removed and only human-initiated entries (i.e. URLs ending with HTM, HTML, XHTML, PHP and JSP) are retained. Only a few of these fields are important for the mining process and hence after extracting or collecting only the important fields such as the User IP Addresses (referrers), URLs and date and the type of file in the URL (whether text, image or script), the rest can be ignored.

*User Identification*: The identity of a user is not a prerequisite for Web usage mining. However, it is necessary to differentiate users to identify sessions. The cleaned database is grouped by different IP addresses and sorted by Date and Time for each IP for identifying different users. As the database is used to store this information,

simple queries can be used to achieve this operation. By doing so, we can identify individual users and also find out the total number of users.

*Session Identification*: Session Identification is the process of partitioning the user activity records of each user into sessions in order to reconstruct the actual sequence of actions done by each user. A session is a package of activities that consists of a user's navigation history. These sessions are then aggregated to create a session database. The user activity records are divided into sessions by assigning unique identifiers for each session. Each user is assigned a separate session and also a separate session is assigned for the same user if the user exceeds a certain threshold of time (e.g. 15 min).

**Encoder**: Encoding is the process where each URL in the preprocessed database is assigned a unique identifier which is a non-negative integer. This is done as it would be too cumbersome to mention the URLs by their domain names in the session database. For this purpose, a table consisting of the list of URLs can be maintained where they are mapped to their respective identifiers. User Identification, Session Identification and Encoding of URLs are all shown in the example in Fig. 3.5. After encoding, the session database, which consists of users and their sessions, is generated and an example is shown in Fig. 3.6.

**Pattern Miner**: Pattern Mining is used to find hidden patterns from a large database when a minimum threshold of occurrence (MinSup) has been specified. The Pattern Miner utilizes UDDAG that grow patterns bidirectionally along both ends of detected patterns and allows faster pattern growth with fewer levels of recursion. UDDAG eliminates unnecessary candidates and supports efficient pruning of invalid candidates. This represents a promising approach for applications involving searching in large spaces like Web Sequential Pattern Mining.

**Pattern Analyser**: Pattern Analysis is the process to study and conduct the analysis of the results obtained from the Web sequential access patterns derived by the miner. The analyser utilizes CMA to find out the periodicity and cyclic behaviour of all the 2-sequence patterns mined. Decoding is the process where the identifiers are

| IP | DATE_TIME | URL | URL_ID | SESSION_ID |
|---|---|---|---|---|
| 30.0.7.231 | 12-Jun-11 12:36:18 AM | http://dnl-02.geo.kaspersky.com/diffs/t | 253 | 2 |
| 30.0.7.231 | 12-Jun-11 12:54:54 AM | http://myinfo.any-request-allowed.com | 480 | 3 |
| 30.0.7.231 | 12-Jun-11 12:56:51 AM | http://myinfo.any-request-allowed.com | 480 | 3 |
| 30.0.7.231 | 13-Jun-11 12:16:08 AM | http://myinfo.any-request-allowed.com | 480 | 4 |
| 30.0.7.231 | 13-Jun-11 12:37:37 AM | http://myinfo.any-request-allowed.com | 480 | 5 |
| 30.0.0.62 | 11-Jun-11 12:41:54 AM | http://www.orkut.com/Glogin? | 1071 | 6 |
| 30.0.0.62 | 11-Jun-11 12:41:54 AM | http://www.phoenixads.co.in/delivery/ | 1103 | 6 |
| 30.0.0.62 | 11-Jun-11 12:41:59 AM | http://channel.tvunetworks.com/list/all | 185 | 6 |
| 30.0.0.62 | 11-Jun-11 12:48:45 AM | http://indiarailinfo.com/main/Getbanner | 397 | 6 |
| 30.0.0.62 | 11-Jun-11 12:50:07 AM | http://clients1.google.co.in/generate_2 | 194 | 6 |
| 30.0.1.3 | 11-Jun-11 12:00:53 AM | http://metric.ind.rediff.com/www.rediff | 465 | 7 |
| 30.0.1.3 | 11-Jun-11 12:00:55 AM | http://www.erail.in/parter/GetStation | 909 | 7 |
| 30.0.1.3 | 11-Jun-11 12:00:57 AM | http://ad.doubleclick.net/adi/N6404.272 | 88 | 7 |

**Fig. 3.5** Preprocessed data

| IP | ACCESS_SEQ |
|---|---|
| 30.0.5.108 | < 1119 324 566 67 942 >,<187 187 98 194>,< 878 878 187 56 443 242 449 942 718 > |
| 20.0.0.98 | < 67 826 185 185 293 1059 1192 990 946 1218 189 1006 960 98 1219 316 324 926 1215 67 962 98> |
| 30.0.0.114 | < 597 939 653 > |
| 30.0.0.142 | < 324 641 649 185 983 185 722 53 806 185 951 189 324 950 951 740 676 720 1185 797 721 797 > |
| 30.0.1.19 | < 324 > |
| 20.0.0.88 | <739 739 274 272 277 279 850 654 561 434 627 324 194 658 1241 277 275 > |
| 30.0.0.151 | < 295 294 87 55 21 >,< 1074 1074 946 > |
| 30.0.0.153 | < 388 940 800 > |
| 30.0.0.111 | < 324 893 194 575 > |
| 40.0.1.35 | < 187 243 1004 185 185 919 388 185 397 71 395 1003 > |
| 30.0.1.17 | < 315 741 1242 > |
| 30.0.4.189 | < 1103 898 > |
| 30.0.0.50 | < 238 98 > |
| 30.0.0.110 | < 563 196 874 717 779 103 566 > |

**Fig. 3.6** Session database

replaced with their corresponding URLs. In the Knowledge Discovery process, the analysed data is transformed into Web prefetching rules and sent to the Prefetching Rule Depository so that they can be used to prefetch and cache Web pages and reduce the round-trip delay experienced by the users.

**Prefetching Rule Depository**: This component is a large database consisting of prefetching rules pertaining to the requested Web pages. After the pattern analysis, each 2-sequence pattern having periodicity and cyclic behaviour are stored here in the form of prefetching rules and are triggered when the first page in the sequence is accessed by a user.

## 3.4 BGCAP Algorithm

Given a raw Web log database $W$, we first perform the preprocessing and generate the session database $SD$ as shown in Table 3.1. The session database consists of a set of tuples, with each tuple consisting of the user (IP) and the access sequence of that user. From Table 3.1, the user with IP address 1.0.1.2 has accessed the Web pages $A$, $B$, $C$ and $D$ with the sequence $< A(A, C)BD >$, where $\{A, B, C, D, E, F\}$is the set of unique Web pages accessed by different users. Here $(A, C)$ in the above sequence denotes that in a single session, the two Web pages $A$ and $C$ were visited by the user in that order, otherwise the user is assumed to visit a single page per session if the parentheses () are not specified.

Web sequential patterns are mined from the session database $SD$ using updated version of UDDAG [1]. This mining technique is basically a divide-and-conquer approach, which tries to construct the patterns simultaneously along both directions of a Directed Acyclic Graph (DAG). The approach consists of three main steps: (i) Database Transformation (ii) Pattern Partitioning and (iii) Finding subsets of Patterns.

Database Transformation is used to remove infrequent pages, i.e. those pages that do not have $MinSup = 2$. From Table 3.1, the frequent items with $MinSup \geq 2$ are

**Table 3.1** Example sessions database

| User (IP) | Sequences |
|---|---|
| 1.0.1.2 | $< A(A, C)B\ D >$ |
| 1.0.1.3 | $< A\ D(E, F) >$ |
| 1.0.1.4 | $< (B, D)C\ F >$ |
| 1.0.1.5 | $< (C, E)(A, B, C, D) >$ |
| 1.0.1.6 | $< A\ B\ C\ D\ E >$ |

**Table 3.2** After database transformation

| User (IP) | Sequences |
|---|---|
| p | < 1 2 3 6 > |
| q | < 1 6(7, 8) > |
| r | < 4 5 8 > |
| s | < (5, 7)(1, 3, 5, 6) > |
| t | < 1 3 5 6 7 > |

$(A)$, $(B)$, $(C)$, $(D)$, $(E)$, $(F)$, $(A, C)$, $(B, D)$. Since we require only these patterns, the remaining can be eliminated by substituting these patterns with non-negative integers like $(A) - 1$, $(A, C) - 2$, $(B) - 3$, $(B, D) - 4$, $(C) - 5$, $(D) - 6$, $(E) - 7$, $(F) - 8$. For the simplicity of representation, we assign each IP a unique identifier as well. The transformed database is as shown in Table 3.2.

Next, the patterns have to be partitioned into projected databases for each pattern, denoted by ${}^{n}$D, where $n$ represents the frequent item. By partitioning, we select only that tuples in which $n$ is present, called projected database for item $n$ and ignore the rest. So, Table 3.2 is partitioned into $m$ projected databases as shown in Table 3.3.

The third and final step, finding subsets of patterns, is not as straightforward as the previous steps. If $WP$ is the set of all sequential patterns mined from $W$ with $MinSup = 2$, then let $WP_1$, $WP_2$, ..., $WP_8$ be the subsets of patterns, i.e., the pattern mined from ${}^{1}$D, ${}^{2}$D, ..., ${}^{8}$D respectively. If the condition $|{}^{n}D| \geq 2$ is not met then such projected databases can be ignored. Here, the projected databases ${}^{2}$D and ${}^{4}$D can be ignored as they contain only one tuple each and hence do not contribute frequent patterns. So we now have to find $WP_1$, $WP_3$, $WP_5$, $WP_6$, $WP_7$ and $WP_8$.

Let us consider ${}^{6}$D. As seen in Table 3.3, the projected database for 6 contains four tuples with IPs $p, q, s$ and $t$. We have to find out the sequential patterns in ${}^{6}$D by recursively partitioning the projected database into prefix and suffix databases until no frequent patterns are found. In the first round of partitioning, we split the projected database into Prefix (Pre (${}^{6}$D)) and Suffix (Suf (${}^{6}$D)) subsets, each containing the tuples with the prefix and suffix sequences, respectively, pertaining to 6 as shown in Table 3.4.

Again, the frequent items, i.e. the patterns in Pre (${}^{6}$D) are $< 1 >$, $< 3 >$ and $< 5 >$, and the pattern in Suf (${}^{6}$D) is $< 7 >$. So, Pre (${}^{6}$D) is further split into $PP_1$, $PP_3$ and $PP_5$, and Suf (${}^{6}$D) is split as $PS_7$ and this process continues recursively

**Table 3.3** Projected databases

| ${}^1D$ | ${}^2D$ |
|---|---|
| p:< 1 2 3 6 > | p:< 1 2 3 6 > |
| q:< 1 6(7, 8) > | |
| s:< (5, 7)(1, 3, 5, 6) > | |
| t: < 1 3 5 6 7 > | |
| ${}^3D$ | ${}^4D$ |
| p: < 1 2 3 6 > | r: < 4 5 8 > |
| s: < (5, 7)(1, 3, 5, 6) > | |
| t: < 1 3 5 6 7 > | |
| ${}^5D$ | ${}^6D$ |
| r: < 4 5 8 > | p:< 1 2 3 6 > |
| s: < (5, 7)(1, 3, 5, 6) > | q:< 1 6(7, 8) > |
| t: < 1 3 5 6 7 > | s: < (5, 7)(1, 3, 5, 6) > |
| | t: < 1 3 5 6 7 > |
| ${}^7D$ | ${}^8D$ |
| q:< 1 6(7, 8) > | q:< 1 6(7, 8) > |
| s:< (5, 7)(1, 3, 5, 6) > | r: < 4 5 8 > |
| t: < 1 3 5 6 7 > | |

**Table 3.4** Prefixes and suffixes for ${}^6$D

| Sequence sets | Frequent patterns |
|---|---|
| Pre (${}^6$D):{< 1 2 3 >, < 1 >, < (5, 7) >, < 1 3 5 >, } | < 1 >, < 3 >, < 5 > |
| Suf (${}^6$D):{< (7, 8) >, < 7 > } | < 7 > |

until no frequent patterns are available. The prefix and suffix databases pertaining to $PP_1$, $PP_3$, $PP_5$ and $PS_7$ are as shown in Table 3.5.

From Table 3.5, it can be seen that only two patterns $< 3 >$ and $< 1 >$ are frequent and no other frequent patterns exist in projected database for $< 3 >$ and $< 1 >$. Hence, the partitioning stops and the DAG is constructed as shown in Fig. 3.7 to find all the patterns of ${}^6$D.

In Fig. 3.7, $< 6 >$ is the root node of the DAG and it contains occurrence set $\{p\ q\ s\ t\}$ to show that the pattern $< 6 >$ occurs in the tuples with IPs $p$, $q$, $s$ and $t$. In a DAG, the Up-children and Down-children denote the prefixes and suffixes of the root node, respectively. Here, the root node $< 6 >$ has three Up-children and one Down-child. Pre (${}^6$D) yielded $< 1 >$, $< 3 >$ and $< 5 >$ and hence these are prefixes of $< 6 >$ and $< 1\ 6 >$, $< 3\ 6 >$ and $< 5\ 6 >$ are frequent patterns as they occur in tuples with IPs $\{p\ q\ t\}$, $\{p\ t\}$ and $\{s\ t\}$, respectively. Suf (${}^6$D) yielded $< 7 >$ and is a suffix of $< 6 >$ and hence $< 6\ 7 >$ is a frequent pattern as it occurs in tuples with IPs $\{q\ t\}$. From Suf ($PP_1$), suffix of $< 1 >$ is $< 3 >$ and from Pre ($PP_3$), the prefix

**Table 3.5** Prefixes and suffixes of prefix and suffix databases of $^6$D

| Sequence sets | Frequent patterns |
|---|---|
| Pre ($PP_1$):{ <> } | – |
| Suf ($PP_1$):{< (2, 3) >, < (3, 5) > } | < 3 > |
| Pre ($PP_3$):{ < 1 2 >,< 1 > } | < 1 > |
| Suf ($PP_3$):{ < 5 > } | – |
| Pre ($PP_5$):{ < 1 3 > } | – |
| Suf ($PP_5$):{ <> } | – |
| Pre ($PP_7$):{ <> } | – |
| Suf ($PP_7$):{<> } | – |

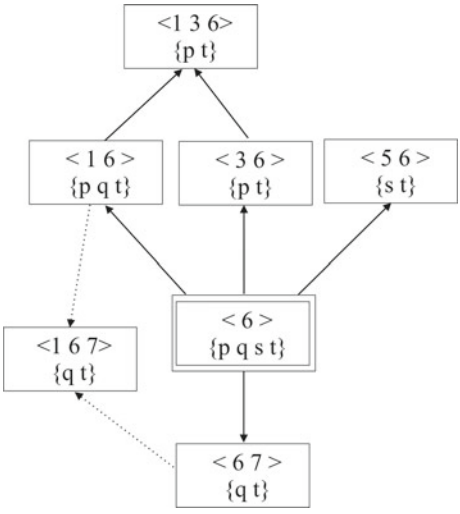


**Fig. 3.7** The example DAG

of < 3 > is < 1 >, therefore, < 1 3 6 >, the upmost child node, is a frequent pattern as it occurs in tuples with IPs $\{p\ t\}$. The patterns in an Up-child and Down-child of a DAG can be combined to form a new node only if the number of tuples of the occurrence sets of the Up-child and Down-child is greater than or equal to $MinSup$. So, the node < 1 6 7 > has one Up-parent < 1 6 > and one Down-parent < 6 7 > and as it is a frequent pattern in $\{q\ t\}$, < 167 > is a valid sequential pattern.

Therefore, the complete set of patterns in $^6$D is, $WP_6 = \{< 6 >, < 1\ 6 >, < 3\ 6 >, < 5\ 6 >, < 6\ 7 >, < 1\ 3\ 6 >, < 1\ 6\ 7 >\}$. Similarly, $WP_1 = \{< 1 >, < 1\ 3 >, < 1\ 6 >, < 1\ 3\ 6 >, < 1\ 6\ 7 >\}$, $WP_3 = \{< 3 >, < 1\ 3 >, < 3\ 6 >, < 1\,3\,6 >\}$, $WP_5 = \{< 5 >, < 5\,6 >\}$, $WP_7 = \{< 7 >, < 1\,7 >, < 6\,7 >, < 1\,6\,7 >\}$ and $WP_8 = \{< 8 >\}$. The complete set of patterns in the session database is $WP = \{< 1 >, < 3 >, < 5 >, < 6 >, < 7 >, < 8 >, < 1\,3 >, < 1\,6 >, < 1\,7 >, < 3\,6 >, < 5\,6 >, < 6\,7 >, < 1\,3\,6 >, < 1\,6\,7 >\}$. Maximal forward references save memory and can be used to generate all the above sequential patterns for $WP$

and so $WP$ (max) is $\{< 8 >, < 5\ 6 >, < 1\ 36 >, < 1\ 6\ 7 >\}$. As this algorithm focuses on deriving prefetching rules, we use $WP$ (max) to generate all 2-sequence Web access patterns. Set of all 2-sequence patterns are $\{< 1\ 3 >, < 1\ 6 >, < 3\ 6 >, < 1\ 7 >, < 6\ 7 >, < 5\ 6 >\}$. These 2-sequence patterns are analysed using Cyclic Model Analysis (CMA) [2] to find out the periodicity and cyclic behaviour of these sequences in the sequence database $D$. After analysis, the set of prefetching rules PR are derived from the periodicity and cyclic behaviour and stored in the server.

**Algorithm 3.1:** BGCAP Algorithm

```
Purpose: To find Periodicity and Tendency of sequential patterns.
Input  : W (Web Log Database)
Output : PR (Set of Prefetching Rules)
1  begin
2      F = Cleaning(W)
3      SD = SessionIdentifier(F,THRESHOLD)
4      WP = Bidirection Pattern GrowthP(SD,minsup)
5      PR = Pattern Analysis(WP)
6  Cleaning(W)
7  begin
8      for each l ∈ W do do
9          if (URL in l contains (js, css)) then
10             ignore
11         else
12             insert l into F
13 SessionIdentifier(F,THRESHOLD)
14 Purpose: To identify Sessions
15 Input: F- Cleaned database and sorted Web log entries according to IP address
16             and time, Threshold time limit for each session (L_i.ip IP
17              address at record L_i and L_i.t Date/Time entry at record L_i )
18 Output : F- with Sessions
19 begin
20     Vsession_id = 0
21     for each l ∈ f do do
22         if (l_i.ip != l_{i+1}.ip) then
23             session_id++
24             l_i.sid = session_id
25             else if (l_i.ip == l_{i+1}.ip and (l_{i+1}.t - l_i.t) > threshold) then
26                 session_id++
27                 l_i.sid = session_id
28             else
29                 l_i.sid = session_id
```

For example, if the 2-sequence pattern $< 3\ 6 >$ occurs frequently in $D$ with a Periodicity of 10 s, that means the user accesses $< 6 >$ 10 s after he has accessed $< 3 >$. If this behaviour repeats itself and then stops after 80 s, then it is said to be

the threshold of Cyclic Behaviour of $< 3\ 6 >$ after which this pattern will not repeat again. Using this knowledge, the Web page $< 6 >$ can be prefetched for the user before the 10th s and stored in the cache for future references and hence reduce his perceived latency. The Web page can be deleted from the cache when the Cyclic Behaviour of 80 s has been reached. The algorithm summarizes the above processes and is shown in Algorithm 3.1.

## 3.5 Experiments

The algorithm BGCAP has been implemented using Java language using NetBeans 6.9.1 platform on MSNBC dataset in a Pentium Dual-Core processor environment, with a 2 GB Memory and 100 GB HDD. The MSNBC Web log data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of *msn.com* for 989818 users. Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. The page requests served *via* caching mechanism are not recorded in the server logs and hence, not present in the data.

Experiments are pursued to compare the efficiency of BGCAP and TD-Mine. BGCAP demonstrated satisfactory scale-up properties with respect to various parameters such as the total number of Web access sequences, the total number of pages, the average lengths of sequences. The following comparisons show that BGCAP outperforms TD-Mine in quite a significant margin and has better scalability than TD-Mine.

### *3.5.1 Data Size Versus Run Time*

The data size is the number of transactions of the input session database and it is a significant factor that affects the performance of BGCAP. This is demonstrated in Fig. 3.8, where we show how different data sizes have different execution times. The higher the number of transactions, the more time it would take to process the data and generate patterns. However, as seen in Fig. 3.8, TD-Mines execution time is moderately higher than that of BGCAP. This is because of the Bidirectional pattern growth approach adopted by BGCAP that reduces the run time by 5–10% than that of TD-Mine. However, as the data size increases to the order of about a million transactions, the run time of both the approaches will tend to be the same since BGCAP must recursively generate prefixes and suffixes for all the frequent items mined.

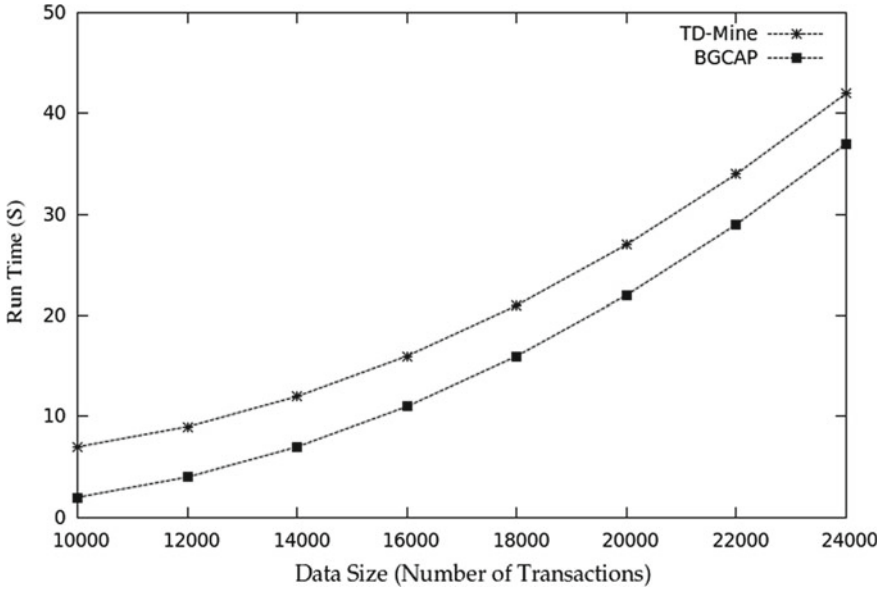


**Fig. 3.8** Data size versus run time

### *3.5.2 Threshold Versus Run Time*

The minimum support threshold ($minSup$) is another important factor that affects the performance of BGCAP. This is demonstrated in Fig. 3.9 where we show how different support thresholds (for a fixed 2000 transactions) have different execution times. As the data size is fixed, the mining only depends on the $minSup$ values. As the $minSup$ value increases, the execution time gradually decreases as it would be a smaller number of patterns that are to be mined. In Fig. 3.9, the graph shows that TD-Mine takes more time to generate patterns compared to BGCAP and they both tend towards the same run time as $minSup$ increases. It is observed that the approach adopted by BGCAP reduces the run time by 10–15% than that of TD-Mine.

### *3.5.3 Threshold Versus Number of Patterns*

Next, we mine Web sequential patterns for different thresholds for a fixed data size of 2000 transactions. Figure 3.10 shows how many patterns can be mined for different support thresholds using BGCAP and TD-Mine. We can see that the number of patterns mined using BGCAP is significantly higher (5–15%) than that of TD-Mine as shown in Fig. 3.10. This is because of the Bidirectional pattern growth approach adopted by BGCAP that is more scalable and accurate than TD-Mine. However, as

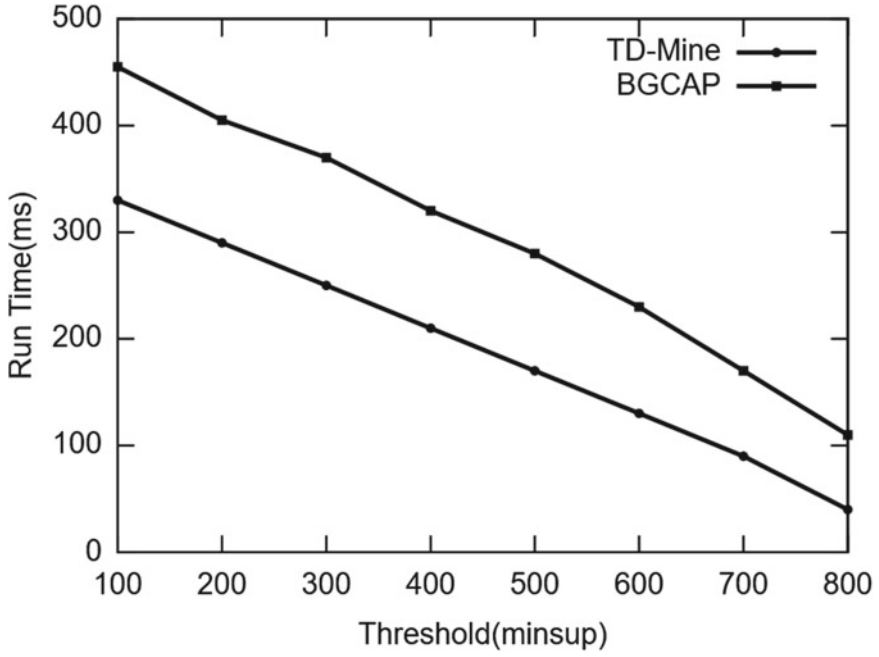


**Fig. 3.9** Threshold versus run time

the minSup increases to the order of 1000 frequent items, the number of patterns generated using both these approaches will tend to be the same as such patterns rarely occur for such large values of threshold.

After generating the Web 2-sequence patterns, BGCAP analyses them using CMA to derive the prefetching rules in terms of periodicity and cyclic behaviour. Consider the same example in Sect. 3.4, where we derived the 2-sequence patterns: {< 1 3 >, < 1 6 >, < 3 6 >, < 1 7 >, < 6 7 >, < 5 6 >}. After analysis, we get the following information as shown in Table 3.6.

From Table 3.6, for the 2-sequence pattern < 1 3 >, the Periodicity is 9 s which means Web page (3) will be accessed by a user 9 s after Web page (1) has been accessed and so it can be prefetched from the server and stored in the client system before that users actually requests the page (3). As the prefetched Web page has already travelled from the server, the users perceived delay is insignificant as the page (3) is simply fetched from the client systems memory itself when the user requests the prefetched Web page. The threshold of cyclic behaviour for < 1 3 > is 57 s which means that this behaviour stops after 57 s, i.e. the user does not request page (3) after page (1) has been accessed, therefore implying that (3) need not be prefetched after the cyclic behaviour has been reached and hence it can be removed from the clients memory. Thus, it reduces network traffic and saves resources by not prefetching Web pages that shall never be requested by the users. Similarly, the prefetching rules can be generated for $n$-sequence Web patterns.

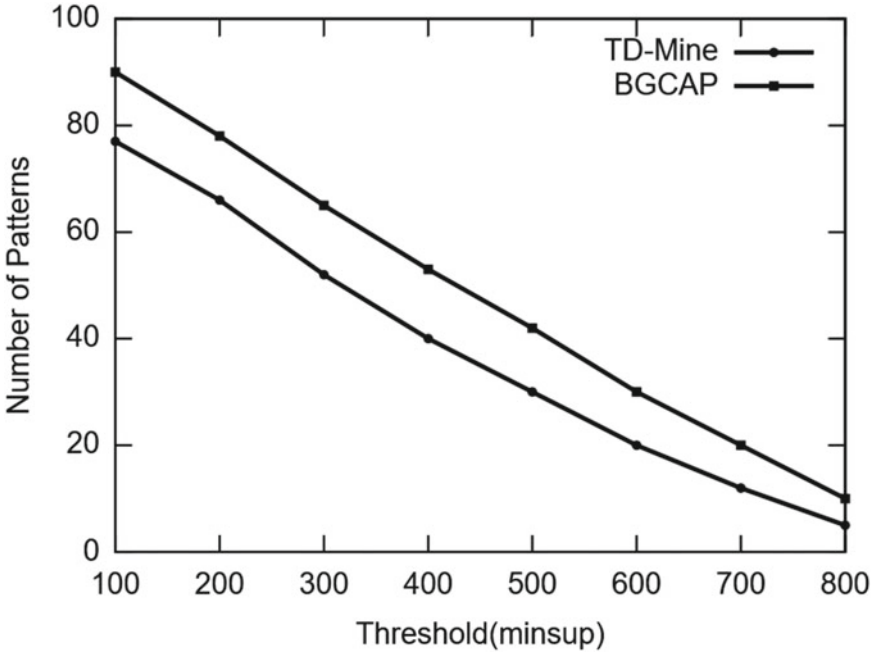


**Fig. 3.10** Threshold versus number of patterns

**Table 3.6** Example of prefetching rules

| 2-sequence pattern | Periodicity (s) | Cyclic behaviour (s) |
|---|---|---|
| < 1 3 > | 9 | 57 |
| < 1 6 > | 5 | 93 |
| < 3 6 > | 7 | 134 |
| < 1 7 > | 3 | 68 |
| < 6 7 > | 8 | 74 |
| < 5 6 > | 4 | 101 |

## 3.6 Summary

The proposed mechanism BGCAP mines Web sequential patterns using UDDAG and analyses them using CMA to generate Web Prefetching rules. As UDDAG is based on Bidirectional pattern growth, BGCAP performs only (log n+1) levels of recursion for mining n Web sequential patterns. Prefetching rules generated based on Periodicity and Cyclic Behaviour of 2-sequence Web patterns is very accurate and the said rules hold good only until the threshold of cyclic behaviour has been reached, thus helping to implement a dynamic necessity-based prefetching strategy. Further, our experimental results show that prefetching rules generated using BGCAP

is 5–10% faster for different data sizes and 10–15% faster for a fixed data size than TD-Mine. Also, BGCAP generates about 5–15% more prefetching rules than TD-Mine. The Web Prefetching rules generated from CMA are used for Caching and Prefecthing Web pages [31].

## References


1. C. Jinlin, An updown directed acyclic graph approach for sequential pattern mining. IEEE Trans. Knowl. Data Eng. **22**(7), 913–928 (2010)
2. D-A. Chiang, C-T. Wang, S-P. Chen, C-C. Chen, The cyclic model analysis on sequential patterns. IEEE Trans. Knowl. Data Eng. **21**(11), 1617–1628 (2009)
3. Y. Hirate, H. Yamana, Generalized sequential pattern mining with item intervals. J. Comput. **1**(3), 51–60 (2006)
4. M.J. Zaki, Spade: an efficient algorithm for mining frequent sequences. Machine Learning, vol. 42, pp. 31–60 (2001)
5. Z. Yang, M. Kitsuregawa, LAPIN-SPAM: an improved algorithm for mining sequential pattern, in *IEEE International Conference on Data Engineering Workshops*, pp. 1222–1226 (2005)
6. Z. Yang, Y. Wang, M. Kitsuregawa, LAPIN: effective sequential pattern mining algorithms by last position induction for dense databases, in *International Conference on Database Systems for Advanced Applications*, pp. 1020–1023 (2007)
7. J. Wang, J. Han, C. Li, Frequent closed sequence mining without candidate maintenance. IEEE Trans. Knowl. Data Eng. **19**(8), 1042–1056 (2007)
8. K-Y. Huang, C-H. Chang, J-H. Tung, C-T. Ho.: COBRA: closed sequential pattern mining using bi-phase reduction approach, in *International Conference on Data Warehousing and Knowledge Discovery*, pp. 280–291 (2006)
9. J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M-C. Hsu, Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans. Knowl. Data Eng. **16**(11), 1424–1440 (2004)
10. X. Cheng, H. Liu, Personalized services research based on web data mining technology, in *IEEE International Symposium on Computational Intelligence and Design*, pp. 177–180 (2009)
11. F. Lumban Gaol, Exploring the pattern of habits of users using web log sequential pattern, in *IEEE International Conference on Advances in Computing, Control and Telecommunication Technologies*, pp. 161–163 (2010)
12. J. Pei, J. Han, B. Mortazavi-asl, H. Zhu, Mining access patterns efficiently from web logs, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining Current Issues and New Applications*, pp. 396–407 (2000)
13. T. Xiaoqiu, Y. Min, Z. Jianke, Mining maximal frequent access sequences based on improved WAP-tree, in *IEEE International Conference on Intelligent Systems Design and Applications*, pp. 616–620 (2006)
14. S. Yang, J. Guo, Y. Zhu, An efficient algorithm for web access pattern mining, in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 726–729 (2007)
15. L. Liu, J. Liu, Mining web log sequential patterns with layer coded breadth-first linked WAP-tree, in *IEEE International Conference on Information Science and Management Engineering*, pp. 28–31 (2010)
16. V. Mohan, S. Vijayalakshmi, S. Suresh Raja, Mining constraint-based multidimensional frequent sequential pattern in web logs. Eur. J. Sci. Res. **36**(3), 480–490 (2009)
17. H-Y. Wu, J-J. Zhu, X-Y. Zhang, the explore of the web-based learning environment based on web sequential pattern mining, in *IEEE International Conference on Computational Intelligence and Software Engineering*, pp. 1–6 (2009)

18. B. Verma, K. Gupta, S. Panchal, R. Nigam, Single level algorithm: an improved approach for extracting user navigational patterns to technology, in *International Conference on Computer and Communication Technology*, pp. 436–441 (2010)
19. O. Nasraoui, M. Soliman, E. Saka, A. Badia, R. Germain, A web usage mining framework for mining evolving user profiles in dynamic web sites. IEEE Trans. Knowl. Data Eng. **20**(2), 202–215 (2008)
20. A. Pitman, M. Zanker, Insights from applying sequential pattern mining to E-commerce click stream data, in *IEEE International Conference on Data Mining Workshops*, pp. 967–975 (2010)
21. F. Masseglia, M. Teisseire, Pascal poncelet.: real time web usage mining with a distributed navigation analysis, in *International Workshop on Research Issues in Data Engineering*, pp. 169–174 (2002)
22. B. Zhou, S.C. Hui, K. Chang, An intelligent recommender system using sequential web access patterns, in *IEEE Conference on Cybernetics and Intelligent Systems*, pp. 393–398 (2004)
23. S-J. Yen, Y-S. Lee, M-C. Hsieh, An efficient incremental algorithm for mining web traversal patterns, in *IEEE International Conference on e-Business Engineering*, pp. 274–281 (2005)
24. Z. Zhang, X. Qian, Y. Zhao, Galois lattice for web sequential patterns mining, in *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pp. 102–106 (2008)
25. S. Jain, R.K. Jain, R.S. Kasana, Efficient web log mining using doubly linked tree. Int. J. Comput. Sci. Inf. Sec. **3**(1), 1–5 (2009)
26. D.K. Jha, A. Rajput, M. Singh, A. Tomar, An efficient model for information gain of sequential pattern from web logs based on dynamic weight constraint, in *IEEE International Conference on Computer Information Systems and Industrial Management Applications*, pp. 518–523 (2010)
27. X. Wang, Y. Bai, Y. Li, An information retrieval method based on sequential access patterns, in *IEEE Asia-Pacific Conference on Wearable Computing Systems*, pp. 247–250 (2010)
28. K. Saxena, R. Shukla, Significant interval and frequent pattern discovery in web log data. IJCSI Int. J. Comput. Sci. Issues **7**(3), 29–36 (2010)
29. D. Oikonomopoulou, M. Rigou, S. Sirmakessis, A. Tsakalidis, Full-web prediction based on web usage mining and site topology, in *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 716–719 (2004)
30. A. Rajimol, G. Raju, Mining maximal web access patterns- a new approach. Int. J. Mach. Intell. **3**(4), 346–348 (2011)
31. K.C. Srikantaiah, N. Krishnakumar, K.R. Venugopal, L.M. Patnaik, Web caching and prefetching with cyclic model analysis of web object sequences. Int. J. Knowl. Web Intell. **5**(1), 76–103 (2014)