1 On Analyzing Web Log Data: A Parallel Sequence Mining Algorithm

Ayhan Demiriz

Information Technology, Verizon Inc.,

919 Hidden Ridge, Irving, TX 75038

E-mail:ayhan.demiriz@verizon.com

ABSTRACT

Activities at enterprise-class web sites, as well as other web sites, are usually recorded via web logs. Collected logs consist of records from many click streams, which are defined as collections of hits (requests) from a specific user during a specific session. Using web logs is the most common way of collecting click stream data at this time. Thus data warehouses are built based on the crucial data extracted from web logs. This article proposes a parallel sequence mining algorithm, webSPADE, to analyze the click streams found in site web logs. In this process, raw web logs are first cleaned and inserted into a data warehouse. The click streams are then mined by webSPADE, the proposed algorithm, which uses one full scan and several partial scans of the data. An innovative web-based front-end is used for visualizing and querying the sequence mining results. By utilizing relational database technology, this analysis technique enables the analysis of very large amounts of data in a short amount of time.

1.1 INTRODUCTION

Many traditional companies see the enormous opportunities in using e-commerce sites as ways to reach customers outside the traditional business channels. Simply running an e-commerce site will not improve customer satisfaction and retention, however. While a user-friendly e-commerce site may attract new customers and strengthen relationships with existing customers, a poorly designed or implemented site may drive away potential customers and damage relationships with current customers. Because of this, businesses should approach e-commerce sites as new markets, where companies must open and maintain the lines of communication with new, and existing, customers.

Activities at e-commerce sites as well as other web sites are usually recorded via web logs. This is the most common technology used at this time to collect information regarding site activities. From an analysis perspective, collected data should reflect user sessions properly. Assuming that user sessions in web logs are constructed by appropriate technology, we must first clean the web logs to remove redundant information. Parsing the cleaned web logs and inserting the data into a repository (data warehouse or relational database) is the next step in the analysis process. Data stored in a repository can easily be used for descriptive statistics (frequency analysis) with proven database technologies to create excellent summary reports. However, when it comes to analyzing the sequences, even with well defined process flows, the number of nested queries required to follow the processes step by step within a relational database framework make the analysis prohibitively expensive. This expense, combined with the fact that simple database queries are unlikely to discover hidden sequences and relationships within the data, make it important to use an effective sequence mining algorithm to analyze the data contained within the web logs.

A click stream is defined as the ordered sequence of pages visited (or requested) by a customer/user (both registered and anonymous) in a session. The first challenge in web log data preprocessing is to define a session and a customer. This is the main step in the preprocessing phase of web usage mining. This step is dependent on the way the web log data is collected: server level collection, client level collection, or proxy level collection. The choice of collection is dependent on the technology. Technical difficulties might prevent a valid analysis of the web log data depending on the choice of collection. Throughout this paper we assume that by using an appropriate technology, such as cookies, we can distinguish the sessions and the customers properly.

The simplest analysis is to get the first level statistics based on the web log data. For example, the most visited page, the longest path, average transaction time, and the proportion of registered customers to the rest are some of the most common reports and are easily generated by off-the-shelf reporting tools such as Analog [1] for web log data. While these types of simplistic reports (traffic analysis) will help business managers understand the "business at a glance," additional value is found thorough analysis of the web log data using Web Mining techniques.

We propose a parallel sequence mining algorithm, webSPADE, based on [2, 3]. There are several major differences in this paper compared to earlier work [2, 3]. Differences and improvements can be summarized as follows:

- webSPADE is a Wintel-based parallel implementation.
- It only requires one full scan of the data compared to three full scans in previous algorithms.
- Temporal joins are strictly used in webSPADE rather than the existence of the non-temporal joins in the original algorithm.
- The design of webSPADE achieves data and task parallelism simultaneously.
- The current system has been in production since Mid-October of 2001 without any major problem.
- Click stream data is analyzed daily and sequences are stored in a relational database.
- A user-friendly front-end is used to visualize and mine stored sequences for a user-determined time range and support level.

- By using a front-end application, it is possible to analyze click stream data from a very large time period (e.g. whole year) in a short time by aggregating stored sequences.

The paper is organized as follows: We briefly discuss web mining and sequence mining in Section 1.2. We propose an integrated solution for click stream analysis in Section 1.3. The solution has four components: The web log data parser, the data warehouse, the sequence mining algorithm webSPADE, and the front-end (user interface) used for displaying and analyzing the mining results. The first two components, the log parser and data warehouse, are mentioned very briefly in this paper, but further discussion of these two components is outside this paper's scope. An analysis of web log data from Verizon.com during the months of January and August of 2002 is given in Section 1.4 for illustration purposes. Computational times are also reported in Section 1.4. The paper wraps up with a conclusion and information about our future work.

1.2 RELATED WORK

When customers visit either commercial or non-commercial web sites and click on links, their actions speak to web site owners/managers. How clearly one understands the customers' behavior depends on whether or not one performs accurate click stream analysis. In general, click stream analysis is regarded as an activity under the web mining umbrella. Web Mining can be defined in short as the application of data mining algorithms to web related data. There are three components of web mining:

- Content Mining.
- Structure Mining.
- Web Usage Mining.

Content mining is the analysis of the information (data) such as text and graphics that are presented in the web pages. One example would be the classification of homepages. One can identify course homepages on the web by distinguishing course homepages from the rest. Clustering is also used to segment pages. Structure mining is used to understand the topology of a web site specifically the inter-page relations hidden in tree-like structures of web sites. In this paper, we specifically explore web usage mining. Web usage mining consists of three phases: Preprocessing, pattern discovery and pattern analysis [4].

In the next sub-section we will describe some of the pattern discovery techniques.

1.2.1 Common Methods in Pattern Discovery

This section describes some of the analytical methods used in web usage mining. As we mentioned earlier, basic level statistical analysis is the most common way of extracting knowledge from web log data. Having descriptive statistics such as the most frequently requested pages, average access time, and the most common error codes might help to improve web traffic, system performance, and security. Statistical analysis is used to monitor the site and help IT professionals evaluate its efficiency and functioning. From an operational perspective, its merits are very important, but from a business point of view they are very limited.

Clustering is also used for extracting knowledge from the web log data. It is very useful, especially when the customer data is merged with the demographic data to generate the customer segmentations. In addition to customer segmentation, some web personalization methods also use the clustering approach. Classification can also be used to categorize customers based on the properties extracted from the web log data and related demographic information. To classify such data, the task must be defined very carefully and the required categorization should be monitored carefully. The difficulty, in classification methods such as decision trees, is to prepare proper training set(s) prior to extracting rules.

Another useful way of pattern discovery is dependency modelling. The aim here is to model user behavior during the various stages of navigation. This approach uses the probabilistic learning techniques, such as Hidden Markov Models and Bayesian Belief Networks, during the learning process. An example for such an approach is the visualization of user navigation patterns by using a Markov model based on clustering [5]. Since a click stream is an ordered sequence, sequence mining plays an important role in knowledge extraction from the web log data. In the following section, we will describe the sequence mining and its predecessor, association mining, used to discover frequent sequences for both temporal and non-temporal data.

1.2.2 Sequence Mining

Sequence mining is an extension of association mining that only finds non-temporal patterns. Association mining is used to find the related pages that are accessed together in the click stream context. Association mining was originally used to perform Market Basket Analysis to determine which items were purchased together. The goal of Market Basket Analysis is to find the related items that are purchased together most frequently. In this context association rules identify pages which are accessed together above a threshold level (support.) Many algorithms have been successfully developed by various researchers for sequence analysis. One of the earliest is Apriori [6]. The Apriori algorithm can reveal the relationships between pages that are not directly connected (that is pages are not linked through hypertext links.) This is desirable, because it generates hidden rules along with the known relations (frequent hypertext links.) Businesses can benefit greatly from the knowledge extracted from these rules. As we mentioned earlier, simple statistical analysis has limited value for generating business rules.

The Apriori algorithm makes several passes over the data to find frequent items (pages in our context.) In the kth pass, it finds all the item sets having k items. Each pass consists of two phases: candidate generation and support counting. The item sets from $(k-1)$th pass are used to generate the candidate list in kth pass. Then the data is scanned in the support counting phase to find the support of the items that are in the candidate list. The items in the candidate list that satisfy the minimum support are kept for the next pass. The algorithm continues until no item set supports the minimum threshold.

Although many variations of the Apriori algorithm have been developed, they still require multiple passes over the data. This is of questionable utility, especially in the case of very large datasets. To speed up the process SQL variations of Apriori algorithm have been also developed e.g. in [7, 8]. Special SQL operators were proposed for such implementations. A SQL based language also was used in [9].

One of the drawbacks of the Apriori-like algorithms is generation of redundant rules. Depending on the support level, association mining may generate an excessive number of rules. On the other hand, since the Apriori algorithm does not take into consideration the order which items are selected (i.e. pages are viewed), some of the rules found are invalid or misleading for click stream analysis. The last phase of web mining, pattern analysis, is designed to filter out these unnecessary rules.

The problem of mining sequential patterns was first studied in [10]. Authors in [10] also introduced three different algorithms for solving this problem. Based on the study in this paper, it was shown that the *AprioriAll* algorithm performed the best compared with other two approaches. An advanced algorithm, - compared to the ones in [10] - GSP, was later proposed in [11]. It was shown that GSP can perform 20 times faster than *AprioriAll*. GSP is an iterative algorithm and requires kth database scan to find the candidate list of the frequent sequences of length k i.e. sequences with k itemsets. An itemset is a collection of items. GSP uses special data structures such as Hash Trees to count frequent sequences efficiently. The terms maximum gap, minimum gap, and sliding time window constraints were also introduced in [11]. Basically, minimum, or maximum, gap constraint requires that the minimum, or maximum, number of events (e.g. page views) should occur between two different items in that particular frequent sequence. Time window constraint simply limits the mining process to find the frequent sequences from a certain time period.

Mining for *frequent episodes*, which are simply frequent sequences in a single long input-sequence, was proposed in [12]. However, we focus on the problem of finding frequent sequences across the multiple transactions (input-sequences.) Authors of [12] extended their framework in [13] to discover *generalized episodes*, which allows one to express arbitrary unary conditions on individual sequence events, or binary conditions on event pairs. The MEDD and MSDD algorithms [14] discover patterns in multiple event sequences; they explore the rule space directly instead of the sequence space. PrefixSpan is one of the most recent algorithms for sequence mining, it uses repeated database projections to mine frequent sequences [15].

Essentially, sequence mining can be considered as association mining over a temporal database. Association mining deals with intra-event patterns (called itemsets), but sequence mining addresses inter-event patterns (sequences.) Since association and sequence mining have many common points, algorithms like *AprioriAll*, GSP, etc., utilize some of the ideas initially proposed for the discovery of association rules.

A recent work, SPAM, in [16] introduces a new approach by using a vertical bitmap representation. SPAM is a depth-first search algorithm with two different pruning steps: S-step and I-step. SPAM is compared with SPADE and PrefixSpan in [16]. Experiments on synthetic data shows that it outperforms both algorithms for especially large datasets. But it should be noted that a serial version of SPADE was used for those experiments [1]. It is also mentioned in [16] that space requirements of SPAM are higher than those of SPADE. It is indeed the same analogy with sparse and full matrix representations of the problems in mathematical sciences. Somewhat SPAM uses the full matrix representation as bitmap data, in contrast, SPADE uses the sparse matrix representation. It is a well known fact that sparse format is always efficient in terms of both space and CPU requirements for many mathematical models.

We chose to modify SPADE [2, 3], since it is a very efficient algorithm for general sequence mining. Unlike previous approaches, which make multiple database scans and use complex hash-tree structures that tend to have sub-optimal locality, SPADE partitions the original problem into smaller sub-problems using equivalence classes on frequent sequences. Not only can each equivalence class be solved independently, but it is also very likely that it can be processed in the main-memory. So SPADE usually makes only three database scans - one for frequent 1-sequences, another for frequent 2-sequences, and one more for generating all other frequent sequences. If the support of 2-sequences is available then only one scan is required. SPADE uses only simple temporal join operations, and is thus ideally suited for direct integration with a DBMS.

An efficient click stream analysis requires both a robust parser for the web log data and an efficient sequence mining algorithm to analyze the cleaned data. In the next

[1]Through personal communication with M. J. Zaki

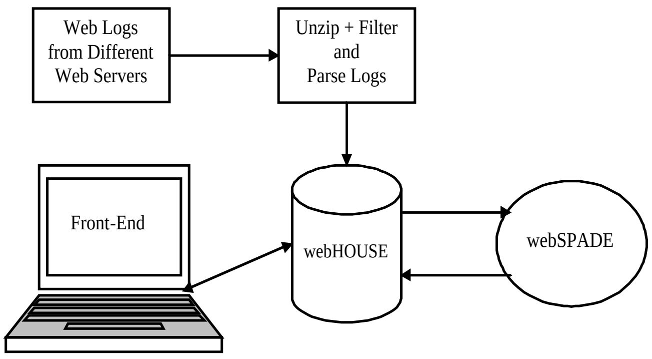


Fig. 1.1 System Architecture

section, we propose an integrated solution to analyze the web log data. First web log data is parsed efficiently and stored in a data warehouse. We then run the webSPADE sequence mining algorithm, a special application of SPADE [2, 3], to generate the rules. Since click streams are ordered by time, rules found by webSPADE have temporal relations e.g. $A \rightarrow B$ means that if A happens then later B happens i.e. if page A is visited then later page B is also visited.

1.3 CLICK STREAM ANALYSIS: AN INTEGRATED SOLUTION

Click stream analysis plays an important role in the decision-making process of an e-commerce site. The knowledge obtained from such analysis can be deployed in restructuring the web site, personalizing the homepages and approaching the customer in a better way (i.e., enhanced Customer Relationship Management.)

We propose an integrated solution for performing click stream analysis in this section. A simplified system architecture is shown in Figure 1.1. The parser feeds the data into webHOUSE, a data warehouse. The sequence mining algorithm, webSPADE, reads daily data from webHOUSE and inserts the daily sequences into webHOUSE. The front-end is used to query the webHOUSE to analyze sequences. In this section, we first explain parsing the web log data and creating the database. Then we explain the modifications to the sequence mining algorithm SPADE introduced in [2]. Finally, we explain the usage of front-end in Section 1.3.4.

1.3.1 Parsing The Web Log Data

Logs recorded by web servers provide the data for the analysis. The web-servers record each page request and related information such as specifications of the requestor's browser type, IP address, user name, time of request, action and page requested, protocol, etc. Different web-servers generate different types of web logs and some are capable of keeping more than one type of log, e.g. Microsoft Internet Information Server. The two most common log formats are NCSA and W3C-extended. Even though there are other types of logs, some of them do not provide sufficient information for analysis. Therefore, our web log processor only supports NCSA and W3C-extended format. The W3C-extended format allows the option of only capturing the desired fields and thus we expect the web-server using the W3C-extended format to be configured to capture at least the required information fields.

The web log processor is designed to be intelligent and does not require the user to know the type and specifications of the web log. It only requires the user to provide the location of the log file. It automatically determines the type of the log and whether it contains the required fields. The current parser runs in a parallel mode and is extremely efficient. On average it takes four hours to parse and insert the cleaned data into the data warehouse for daily hits at Verizon.com. This cleaned data consists of both anonymous and registered requests and contains more than three and a half million hit records each day. Note that we do not include all the hits to Verizon.com and it should be noted that certain pages on Verizon.com use cookies to keep track of the user sessions and the application process flows. This means that session information and some other operational data is also kept in the data warehouse. When Verizon.com was launched in October 2001, the early designs of many applications required registration for most of the activities. Thus non-persistent cookies were used to determine the user sessions. Certainly persistent cookies are potentially more useful than non-persistent ones due to their ability to track anonymous users over several different sessions. Other technologies such as messaging services e.g. Microsoft Message Queue Server (MSMQ) and MQSeries by IBM can be used at server site to monitor the user session. Obviously this requires application level integration.

A sophisticated data warehouse is used for storing daily request data and related information. Due to the proprietary nature of the application, we will only mention a very early version of data warehouse in the next sub-section. In practice, we only need three data fields -session id, time stamp and page id- for click stream analysis.

1.3.2 webHOUSE: A Web Log Data Warehouse

Since there is a huge amount of data to be stored and analyzed, a sophisticated database system is needed, which not only prevents data redundancy but also provides readily available analysis. Considering the constraints and available tools, Microsoft SQL Server is utilized as the database engine. A star schema such as in Figure 1.2 is used in this data warehouse. It should be noted that the star schema given in Figure 1.2 is a very early version of the current data warehouse. It is only mentioned here to illustrate the system. In very early versions of the data warehouse, we used IP address as session id for simplicity. OLAP reporting tools are also available with Microsoft SQL Server.

Utilizing the star schema depicted in Figure 1.2 reduces the storage requirements to approximately 5% of the raw data size. For example, if a web-page contains a mixture of 9 image and script files, the web server will record it as 10 hits: 1 for the page itself and 9 for the images and script files. If only the page hit is kept and the rest are discarded, the log would be reduced to 1/10th of its size. Not only do we reduce the storage size, but also bring the ability to run both static and ad hoc queries to the web log data by creating the database. The fact table is the main table in the star schema. Several OLAP reports are also created or updated when this table is updated. In the next sub-section we explain the webSPADE algorithm used in this paper to perform the sequence mining.

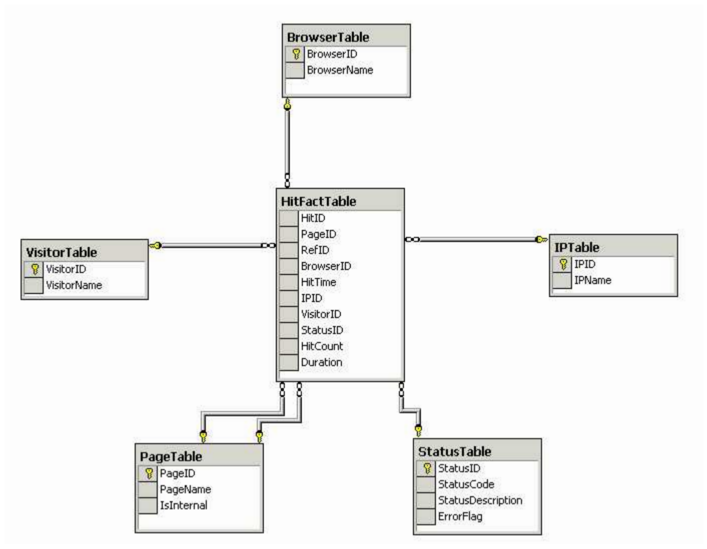


Fig. 1.2 Star Schema of the Web Log Data Warehouse

1.3.3 webSPADE: A Parallel Sequence Mining Algorithm

The nature of the sequence mining problem makes massive computation unavoidable. This situation is an ideal application for parallel programming. In this section we present a parallel sequence mining algorithm that differs from previous work [3, 17, 18] in many ways. Parallel sequence mining algorithms are generally derived from sequential ones by introducing load balancing schemes for multiple processors and distributed memory. Since the data warehouse is built on MS SQL Server in our implementation, webSPADE has been developed in the Wintel environment, simplifying the parallelization of the serial programs by letting the operating system to handle the load balancing.

The core point of our implementation is to modify the SPADE algorithm for the purpose of performing click stream analysis. The details of the original SPADE algorithm can be found in [2]. SPADE is a generic algorithm used for performing sequence mining for both time-dependent and time-independent data. The main advantage of using this algorithm is the use of join operations instead of scanning all the data to count certain item sets. The original SPADE algorithm proposed in [2] requires three full scans of the data. The advantage of this algorithm compared to the Apriori algorithm is the ability to use the time information. Our implementation is even better since it requires only one full data scan which is performed as the data is retrieved from the database. Note that both SPADE and webSPADE may require a large number of scans on intermediate partial data.

The reason behind using the join operation in SPADE algorithm is that, as mentioned in Corollary 1 of [2], any sequence X can be derived by the joining of its first two lexicographical subsequences. For example in order to find support for the rule $A \rightarrow B \rightarrow C$, it is sufficient to join the sets of the rules $A \rightarrow B$ and $A \rightarrow C$. It is obvious that the rule $A \rightarrow C \rightarrow B$ is also obtained by joining the same two subsequences. This is the most crucial step of the SPADE algorithm that gives it superiority over the Apriori algorithm. In this case, A is called as an "equivalence class".

Before going into details of webSPADE, we want to briefly discuss the differences between webSPADE and SPADE introduced in [2]. Since click streams are strictly ordered by time and no two items (page views for the same session) occur at the same time, we only use temporal joins in contrast to the existence of the non-temporal joins in the original algorithm. Another difference is that support counting is not performed at a session level i.e. if $A \rightarrow B$ occurs twice in a given session, support of the sequence (rule) increments by two. In the original algorithm, it would count only one instead. We consider repeating subsequences in a given session as significant for click stream analysis. Therefore, our algorithm is designed to handle repetition within a session, though a counter-argument can be made regarding the misleading nature of the repeating sequences. For example: Assume that there are several independent hits on page A and B and all page B hits occur after page A hits. In this case, the support of the sequence $A \rightarrow B$ will be counted as many as the lowest number of hits on pages A and B. However, in reality, user goes from page A to B only once. This situation might cause the sequence mining algorithm to find unnecessary or illogical sequences. Our analysis shows that this situation does not exist within the click stream data and the resulting sequences of webSPADE reflect the true frequencies after confirming with SQL queries. Sequences make sense from the perspective of both business and real traffic flow. Nevertheless, taking the repeating sequences into consideration is important to understand the true traffic load.

Table 1.1 Sample Set for Item A

SID	Time
1	10
1	15
2	12

As we mentioned earlier, due to the design of our implementation, the algorithm requires only one full scan of the database. Since we do not have vertical to horizontal database recovery in this implementation, we skip the second and the third scan in the original implementation, creating a significant performance improvement. The main data structure is the map of multisets in our implementation. A sample set is depicted in Table 1.1. SID stands for session id. This gives us the ability to join tables efficiently and reduce the number of full scans.

In terms of parallelization of the serial programs, the memory type of the computer system is an important factor to consider when designing the algorithm. Algorithms proposed in [3, 17, 18] are designed to run on distributed shared-memory due to the selection of parallel computer systems (SGI Origin 2000 and IBM SP2.) Since webSPADE runs on a single machine with multiple CPUs, it is designed to utilize a single memory. Indeed, this particular server has 8 CPUs at 700 MHz clock speed with 8GB of total memory. The system used in [3] is composed of 12 processors SGI Origin 2000 with 195 MHz R10000 MIPS processors and 2GB main memory. Algorithms discussed in [17, 18] are run on an IBM SP cluster that consists of 79 four-processor and 3 two-processor machines with a total of 391 GB of memory. These machines have 222 MHz Power3 processors. When we compare the three different systems, a Wintel based system is the simplest and certainly the cheapest.

Sequence mining can be considered as an irregular tree search algorithm [3], with each node corresponding to an equivalence class. According to argument in [3], parallelization of the sequence mining can be achieved either by data or task parallelism. In data parallelism, processors work on distinct partitions of the database but process the global tree structure concurrently. Task parallelism, on the other hand, requires each processor to have a separate copy of the database and run on different branches of the global tree. One static and two dynamic load balancing schemes are examined in [3] as part of task parallelism. Experiments in [3] show that task parallelism is more favorable than data parallelism, with the best task parallelism approach using recursive dynamic load balancing [3].

A parallel version of the tree projection algorithm is proposed in [17]. Each node in the projection tree corresponds to k-itemset. This is indeed similar to the equivalence class representation of SPADE algorithm. The projection tree grows in a breadth-first manner. Data and task parallelism are compared again in [17]. Bipartite graph partitioning and bin packing are used as task parallelism approaches. These are not considered as dynamic load balancing schemes. Similar to the results in [3], task parallelism results are more favorable in terms of work load and computation time. An extension to [17] is introduced in [18] by implementing a dynamic load balancing scheme. The dynamic load balancing scheme in [18] performs similar to or better than the static load balancing schemes used in [17].

Algorithm 1.3.1 (webSPADE).

```
Given min_support and database $\mathcal{D}$
$\mathcal{F}_1 = \{$Frequent items or 1-sequences $\}$
$\mathcal{F}_2 = \{$Frequent 2-sequences (item-pairs) $\}$
$Enumerate - Freq - Seq(\mathcal{F}_2)$;
```

With the design of webSPADE, data and task parallelism are achieved simultaneously. Since webSPADE is developed in a Wintel-based environment, it has been coded as a multi-threaded program. A high level pseudo-code of the algorithm is given in Algorithm 1.3.1. Finding frequent sequences is a recursive depth-first search step to reduce the required memory for searching new sequences. In this step, all the item-pairs with the same equivalence class in the item-pair set are joined pairwise to form the next item-pair set. This recursive step is repeated for this new item-pair set until no two-item exists in the following item-pair set which has minimum support. The current rules are printed at the proper places within the recursive search. A thread is created for each recursive branch in the search space. Required data is passed to the child thread and immediately deleted from the parent thread. Thus while task parallelism is done during depth-first search, data parallelism is also achieved by passing the required data to the child threads. Load balancing is left to the operating system (Windows 2000), which reduces the coding efforts drastically. In our early implementation [19], there was no limitation on the number of threads in our application. They were created as needed and were monitored by the operating system. However, webSPADE is modified to limit the number of child threads at five (can be changed as a parameter) for each individual thread to

1. stabilize the program for extremely large data (occasionally),
2. stabilize the program for dense data (see Section 1.3.4),
3. prevent the unexpected outcomes of the very long sessions - especially for the test users.

Frequent items (pages) ($\mathcal{F}_1$) are found during the creation of the required data structures (data retrieval.) This step is linear and does not require parallelization. Data is directly imported from the data warehouse for a specified time window by running a SQL query. Finding frequent 2-sequences ($\mathcal{F}_2$) is also parallelized in our implementation. This step requires exhaustive search by two nested for-loops. Since sequence mining may involve with the pages that are not linked via hypertext links and backward movement is also considered in our problem setup, exhaustive search is unavoidable. For each step in the main for-loop, a thread is created to find $\mathcal{F}_2$. `Enumerate-Freq-Seq` is the main function in webSPADE and each time this function is called, except in the main process, a new thread is also created. Due to the nature of the exhaustive search involved, finding $\mathcal{F}_2$ is the most CPU intense step in webSPADE.

Potentially, hundreds of threads are created during the run time of webSPADE. The details of `Enumerate-Freq-Seq` function is given in Algorithm 1.3.2. The term σ represents the support count of the corresponding data structure. T is the set of item-pairs to be passed to the next branch. There are two join operations as mentioned above to find sequences - for example to find both $A \to B \to C$ and $A \to C \to B$ sequences. To illustrate the algorithm well, we use two different steps in 1.3.2, but in our application these two join operations are done in a single pass. $\mathcal{L}$ represents the intermediate item-pair sets after join operations. Sequences that have frequency below the support level are removed from further analysis. $\mathcal{F}$ again corresponds to the frequent sequences to be printed later. This part of webSPADE does not differ much from the original SPADE algorithm and it is suggested that interested readers see [2] for further details.

Algorithm 1.3.2 (Enumerate-Freq-Seq).

```
Given Set S
for all item-pairs $A_i \in S$ do:
 { print the sequence
  $T_i = \emptyset$;
  if antecedent (equivalence class) changed then
   {if $size(T_i) > 1$ then
      $Enumerate - Freq - Seq(T_i)$;
    if $size(T_i) = 1$ then
      print the sequence }
  while $A_j$ and $A_i$ have same antecedent do:
   {$R = A_i \bigvee A_j$;
    $\mathcal{L}(R) = \mathcal{L}(A_i) \cap \mathcal{L}(A_j)$; Join operation
    if $\sigma(R) \geq min\_support$ then
     {$T_i = T_i \bigcup \{R\}$;
      $\mathcal{F}_{|R|} = \mathcal{F}_{|R|} \bigcup \{R\}$; }
    $R = A_j \bigvee A_i$;
    $\mathcal{L}(R) = \mathcal{L}(A_j) \cap \mathcal{L}(A_i)$; Join operation
   if $\sigma(R) \geq min\_support$ then
    {$T_i = T_i \bigcup \{R\}$;
     $\mathcal{F}_{|R|} = \mathcal{F}_{|R|} \bigcup \{R\}$; }
   }
 }
```

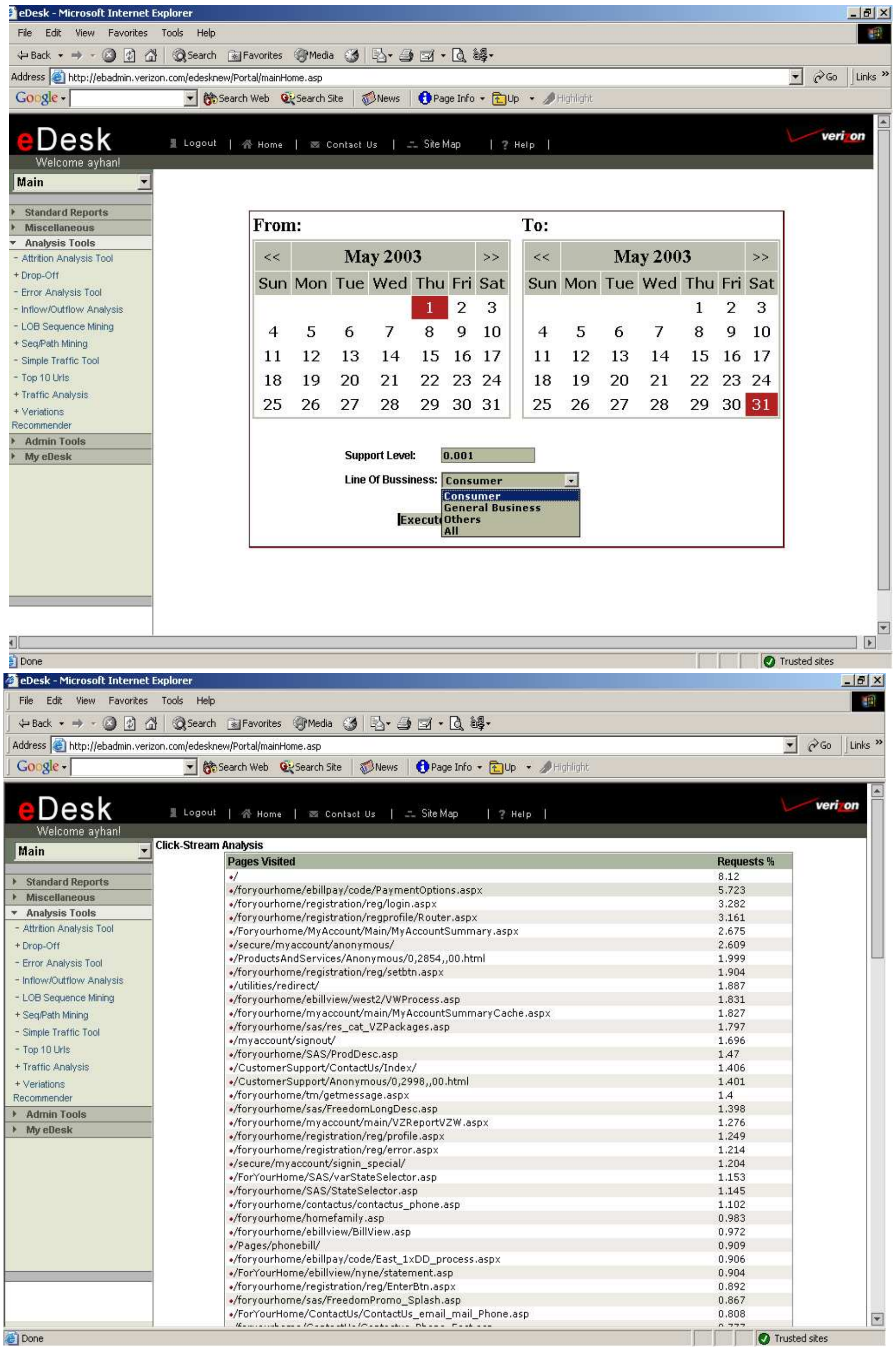
The join operator in Algorithm 1.3.2 first finds identical Session Identifiers (SID), then compares the time stamps between two hits with the same SID. As mentioned above, the basic data structure in our implementation consists of a set of SID and Time stamp pairs (see Table 1.1.) When the two sets are joined line by line (e.g. $A \rightarrow B$ and $A \rightarrow C$), the join operator first verifies that the SIDs are equal. If this condition is met, and the time stamp of $A \rightarrow B$ is less than the time stamp for $A \rightarrow C$, a new line is inserted into the set of $A \rightarrow B \rightarrow C$. Note that the time stamp for this new line is equivalent to the time stamp from $A \rightarrow C$. Once a specific SID and Time pair is used in a join operation to generate a new line in the sequence, that pair is removed from the set. In other words, the join operation creates distinct instances. If the resulting set satisfies the minimum support limit, a new sequence is added to the list of frequent sequences. It should also be noted that the join operation is used heavily in finding $\mathcal{F}_2$.

One important benefit of sequence mining is to automatically find all the possible sequences, because it is not humanly possible to consider all the possible sequence combinations. Therefore it is not possible to write SQL queries for every single possible combination, yet it takes quite some time to run such queries for even very small set of items (pages.) There are commercially available data warehouses, such as the one found in Microsoft Commerce Server 2000, which allow users to query the related databases but it is practically impossible to come up with all the combinations of SQL queries to replace sequence mining. Moreover, such strategy will only be suitable for a static web site. For the commercial web sites that change almost constantly in terms of both content and structure, that is quite unacceptable and cumbersome.

The strength of our application comes from utilizing database technology to store and aggregate the sequence mining results and present them to the end user in a very short time. While running sequence mining on-the-fly for large datasets is generally regarded as practically impossible, in this paper we introduce an unprecedented analysis technique in the next sub-section to do just that. Our methodology can be considered as a new way to perform a mine and explore approach, i.e. mine the partial data now and then aggregate the results to mine larger data later.

1.3.4 Scaling up Sequence Mining Algorithms: Application of webSPADE

We present a very innovative way of scaling up any sequence mining algorithm by utilizing webSPADE and the relational database technology together. This allows users to analyze very large datasets spanning a large time window e.g. a year. A user friendly front-end visualizes the sequences in a very effective way and enables users to choose three different parameters: support levels, time window(start and end date of analysis) and line of business (LOB) to query the sequences. Similar visual pattern analysis techniques are introduced in [20, 21] in the context of the “mine and explore" paradigm using episode and association mining. Rules are found at a



From: May 2003, 1 — To: May 2003, 31

Support Level: 0.001

Line Of Bussiness: Consumer

Consumer
General Business
Others
All

Execute

Click-Stream Analysis

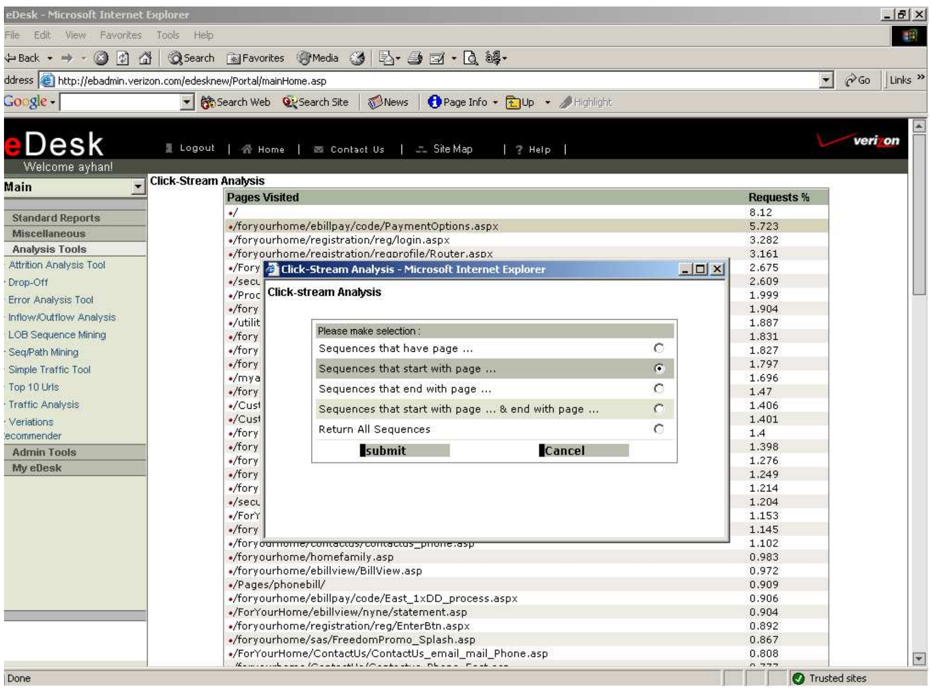
Pages Visited	Requests %
•/	8.12
•/foryourhome/ebillpay/code/PaymentOptions.aspx	5.723
•/foryourhome/registration/reg/login.aspx	3.282
•/foryourhome/registration/regprofile/Router.aspx	3.161
•/Foryourhome/MyAccount/Main/MyAccountSummary.aspx	2.675
•/secure/myaccount/anonymous/	2.609
•/ProductsAndServices/Anonymous/0,2854,,00.html	1.999
•/foryourhome/registration/reg/setbtn.aspx	1.904
•/utilities/redirect/	1.887
•/foryourhome/ebillview/west2/VWProcess.asp	1.831
•/foryourhome/myaccount/main/MyAccountSummaryCache.aspx	1.827
•/foryourhome/sas/res_cat_VZPackages.asp	1.797
•/myaccount/signout/	1.696
•/foryourhome/SAS/ProdDesc.asp	1.47
•/CustomerSupport/ContactUs/Index/	1.406
•/CustomerSupport/Anonymous/0,2998,,00.html	1.401
•/foryourhome/tm/getmessage.aspx	1.4
•/foryourhome/sas/FreedomLongDesc.asp	1.398
•/foryourhome/myaccount/main/VZReportVZW.aspx	1.276
•/foryourhome/registration/reg/profile.aspx	1.249
•/foryourhome/registration/reg/error.aspx	1.214
•/secure/myaccount/signin_special/	1.204
•/ForYourHome/SAS/varStateSelector.asp	1.153
•/foryourhome/SAS/StateSelector.asp	1.145
•/foryourhome/contactus/contactus_phone.asp	1.102
•/foryourhome/homefamily.asp	0.983
•/foryourhome/ebillview/BillView.asp	0.972
•/Pages/phonebill/	0.909
•/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.906
•/ForYourHome/ebillview/nyne/statement.asp	0.904
•/foryourhome/registration/reg/EnterBtn.aspx	0.892
•/foryourhome/sas/FreedomPromo_Splash.asp	0.867
•/ForYourHome/ContactUs/ContactUs_email_mail_Phone.asp	0.808

Fig. 1.3 (A)- Parameter Selection Screen, (B)- Frequent Pages

predetermined support and confidence level, then a user-interface is used to query the rule base to narrow down the selection of important rules.

Our approach is very simple. webSPADE runs daily with a predetermined support level to find sequences based on different lines of business; Support level is set to 0.1% for General Business, Consumer and Others (the rest) in our application (see Figure 1.3(A)) . Daily sequences are then stored in a relational table. By design, webSPADE limits the length of sequences and, in this particular application, the maximum length of sequences is set to ten. Depending on the application domain, this parameter can be chosen accordingly. Since important processes in e-commerce sites, such as registration, should be kept as brief as possible, we believe that ten-level sequences are adequate for click stream analysis. Thus the relational table is composed of ten fields to determine the sequences, analysis date, line of business, frequency of sequence and total hits on analysis date. Since web log data is collected daily, we use a day as our atomic time unit. Different atomic time units may be used on other domains. For example, we may store financial sequence data hourly.

As in the case of association mining, sequence mining might result in an excessive number of sequences. Finding potentially valuable sequences among numerous repetitive sequences might become impossible. Our front-end enables users to see and query the sequences in a user-friendly manner. Stored sequences can be used to analyze click streams between user specified dates. As seen from Figure 1.3(A), users can specify any support level and dates for a given line of business allowing a great degree of flexibility. **It should be noted that webSPADE is run daily and results are stored; there is no on-the-fly computation when the user selects parameters using the parameter selection screen.** In fact, the stored results are aggregated and presented to the user. We can also aggregate the data for all the LOBs as in the case of "**All**" option in Figure 1.3(A). Aggregating the results of micro mining activities is the major difference from the previous work introduced in [20, 21]. Because mining algorithm in [20, 21] is run on all the available data (macro mining) and then resulting patterns are analyzed and searched by a user-interface. On the other hand, our methodology requires mining efforts on a portion of data (for one day.) Resulting patterns are then aggregated and presented to the user for visual analysis and pattern



Click-Stream Analysis

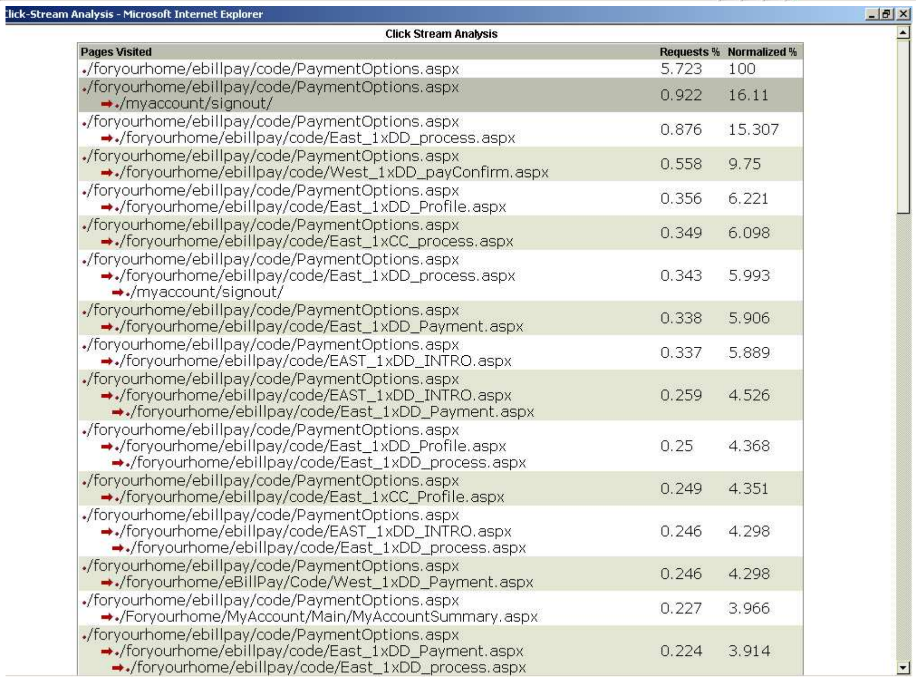
Pages Visited	Requests %
•/	8.12
•/foryourhome/ebillpay/code/PaymentOptions.aspx	5.723
•/foryourhome/registration/reg/login.aspx	3.282
•/foryourhome/registration/regprofile/Router.aspx	3.161
•/Fory	2.675
•/secu	2.609
•/Proc	1.999
•/fory	1.904
•/utilit	1.887
•/fory	1.831
•/fory	1.827
•/fory	1.797
•/mya	1.696
•/fory	1.47
•/Cust	1.406
•/Cust	1.401
•/fory	1.4
•/fory	1.398
•/fory	1.276
•/fory	1.249
•/fory	1.214
•/secu	1.204
•/ForY	1.153
•/fory	1.145
•/foryourhome/contactus/contactus_phone.asp	1.102
•/foryourhome/homefamily.asp	0.983
•/foryourhome/ebillview/BillView.asp	0.972
•/Pages/phonebill/	0.909
•/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.906
•/ForYourHome/ebillview/nyne/statement.asp	0.904
•/foryourhome/registration/reg/EnterBtn.aspx	0.892
•/foryourhome/sas/FreedomPromo_Splash.asp	0.867
•/ForYourHome/ContactUs/ContactUs_email_mail_Phone.asp	0.808

Click-Stream Analysis - Microsoft Internet Explorer

Click-stream Analysis

Please make selection :
- Sequences that have page ... ○
- Sequences that start with page ... [✓ ●]
- Sequences that end with page ... ○
- Sequences that start with page ... & end with page ... ○
- Return All Sequences ○

submit   Cancel



Click Stream Analysis

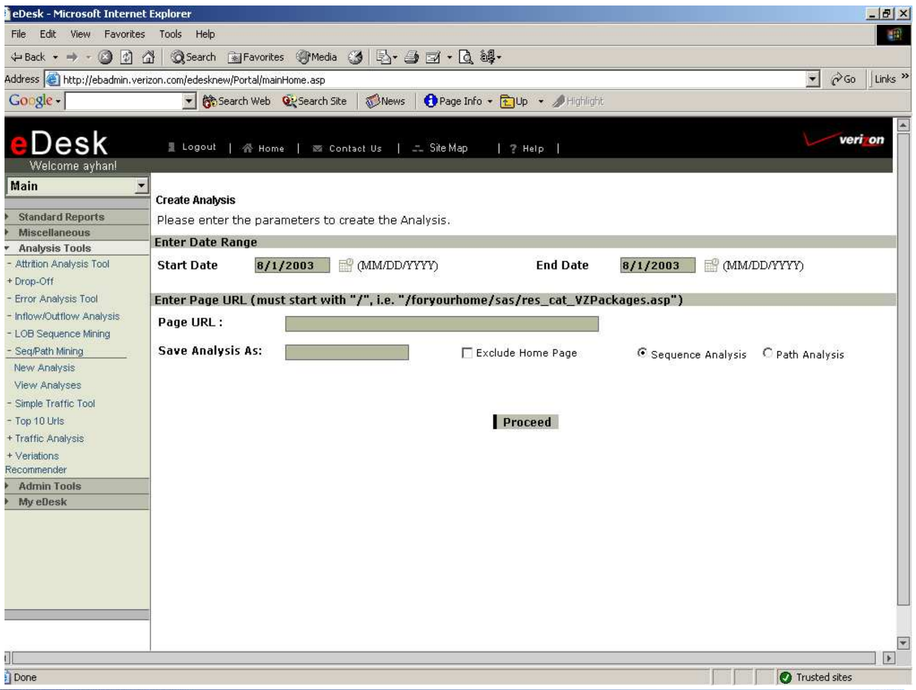
Pages Visited	Requests %	Normalized %
•/foryourhome/ebillpay/code/PaymentOptions.aspx	5.723	100
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/myaccount/signout/	0.922	16.11
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.876	15.307
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/West_1xDD_payConfirm.aspx	0.558	9.75
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_Profile.aspx	0.356	6.221
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xCC_process.aspx	0.349	6.098
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_process.aspx → •/myaccount/signout/	0.343	5.993
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_Payment.aspx	0.338	5.906
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/EAST_1xDD_INTRO.aspx	0.337	5.889
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/EAST_1xDD_INTRO.aspx → •/foryourhome/ebillpay/code/East_1xDD_Payment.aspx	0.259	4.526
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_Profile.aspx → •/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.25	4.368
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xCC_Profile.aspx	0.249	4.351
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/EAST_1xDD_INTRO.aspx → •/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.246	4.298
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/eBillPay/Code/West_1xDD_Payment.aspx	0.246	4.298
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/Foryourhome/MyAccount/Main/MyAccountSummary.aspx	0.227	3.966
•/foryourhome/ebillpay/code/PaymentOptions.aspx → •/foryourhome/ebillpay/code/East_1xDD_Payment.aspx → •/foryourhome/ebillpay/code/East_1xDD_process.aspx	0.224	3.914

Fig. 1.4 (A)- Analysis Selection, (B)- Sequences

search. Moreover, our approach scales up the underlying mining algorithm and visualizes the results simultaneously. This is a significant improvement in terms of computation efficiency.

After selecting the parameters, frequent pages are shown to the user as seen in Figure 1.3(B). Further analysis can be deployed by clicking on any page name in the frequent page list (see Figure 1.3(B).) Once clicked, a pop-up window appears as seen in Figure 1.4(A). There are five options to choose on this screen. Users can select to see sequences that contain a selected page, start with a selected page, or end with a selected page. Users can list all the sequences as well. There is another option to list all the sequences starting with the selected page and ending with another page. Each option corresponds to a different stored procedure in the database. Having results stored in a relational table gives tremendous flexibility to report and query the results. Once an option is chosen, the resulting sequences are listed in the browser window as seen in Figure 1.4(B).

The user interface introduced above is designed to analyze the click stream data at LOB level. In certain cases, application owners might be interested to know what else our customers do when they visit a particular page. In other words, we might want to analyze all the click stream data from those sessions that a particular page is visited. webSPADE can easily be used for this purpose. All we need to do is provide the click stream data to webSPADE from those sessions. Thus resulting analysis can be called "*Page Based Sequence Mining*." Since sequence mining might take some time to run and cause unpredictable CPU load, we designed a front-end for batch processing this type of analyses (Figure 1.5(A)). In the first screen, the user determines the date range (multiple-day selections are allowed) and the particular page that he/she asks for analysis. The user also gives a name for that particular analysis to be saved for a later time to view. Since the user is interested only in analyzing the data mainly around a particular page, the presence of the homepage might skew the analysis. Thus the user has a choice to exclude the homepage from the analysis. Given these parameters, algorithm runs at 0.1% fixed support level. Considering the multiple users (analysts) case, the batch process only allows one instance of the algorithm to run at any given time.



Create Analysis

Please enter the parameters to create the Analysis.

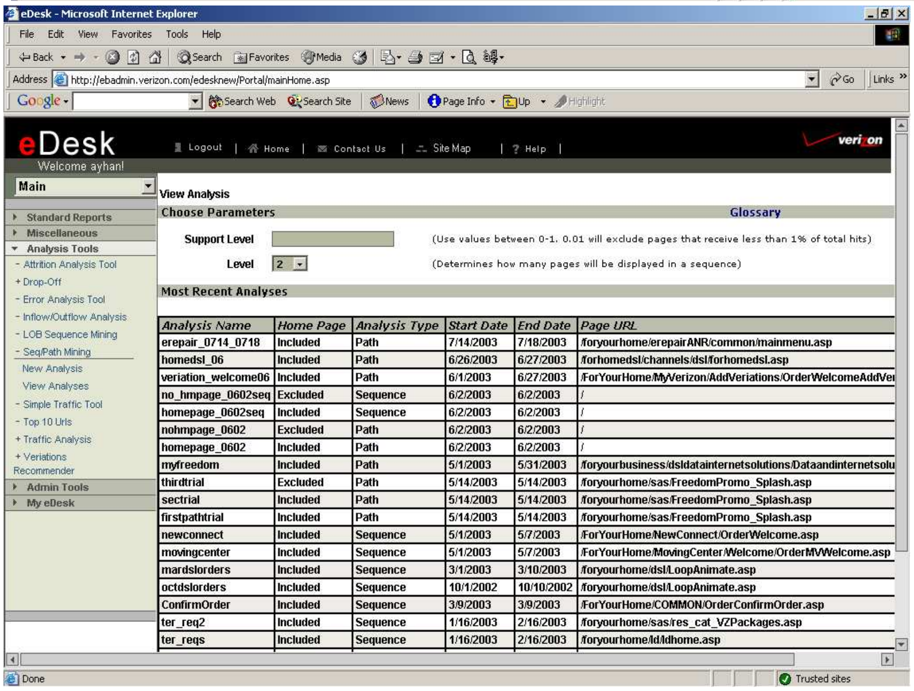
Enter Date Range

Start Date 8/1/2003 (MM/DD/YYYY) End Date 8/1/2003 (MM/DD/YYYY)

Enter Page URL (must start with "/", i.e. "/foryourhome/sas/res_cat_VZPackages.asp")

Page URL :

Save Analysis As: ☐ Exclude Home Page [✓ Sequence Analysis] Path Analysis



View Analysis

Choose Parameters

Support Level (Use values between 0-1. 0.01 will exclude pages that receive less than 1% of total hits)

Level 2 (Determines how many pages will be displayed in a sequence)

Most Recent Analyses

Analysis Name	*Home Page*	*Analysis Type*	*Start Date*	*End Date*	*Page URL*
erepair_0714_0718	Included	Path	7/14/2003	7/18/2003	/foryourhome/erepairANR/common/mainmenu.asp
homedsl_06	Included	Path	6/26/2003	6/27/2003	/forhomedsl/channels/dsl/forhomedsl.asp
veriation_welcome06	Included	Path	6/1/2003	6/27/2003	/ForYourHome/MyVerizon/AddVeriations/OrderWelcomeAddVe
no_hmpage_0602seq	Excluded	Sequence	6/2/2003	6/2/2003	/
homepage_0602seq	Included	Sequence	6/2/2003	6/2/2003	/
nohmpage_0602	Excluded	Path	6/2/2003	6/2/2003	/
homepage_0602	Included	Path	6/2/2003	6/2/2003	/
myfreedom	Included	Path	5/1/2003	5/31/2003	/foryourbusiness/dsldatainternetsolutions/Dataandinternetsolu
thirdtrial	Excluded	Path	5/14/2003	5/14/2003	/foryourhome/sas/FreedomPromo_Splash.asp
sectrial	Included	Path	5/14/2003	5/14/2003	/foryourhome/sas/FreedomPromo_Splash.asp
firstpathtrial	Included	Path	5/14/2003	5/14/2003	/foryourhome/sas/FreedomPromo_Splash.asp
newconnect	Included	Sequence	5/1/2003	5/7/2003	/ForYourHome/NewConnect/OrderWelcome.asp
movingcenter	Included	Sequence	5/1/2003	5/7/2003	/ForYourHome/MovingCenter/Welcome/OrderMVWelcome.asp
mardslorders	Included	Sequence	3/1/2003	3/10/2003	/foryourhome/dsl/LoopAnimate.asp
octdslorders	Included	Sequence	10/1/2002	10/10/2002	/foryourhome/dsl/LoopAnimate.asp
ConfirmOrder	Included	Sequence	3/9/2003	3/9/2003	/ForYourHome/COMMON/OrderConfirmOrder.asp
ter_req2	Included	Sequence	1/16/2003	2/16/2003	/foryourhome/sas/res_cat_VZPackages.asp
ter_reqs	Included	Sequence	1/16/2003	2/16/2003	/foryourhome/ld/ldhome.asp

Fig. 1.5 (A)- Create Analysis , (B)- View Analyses

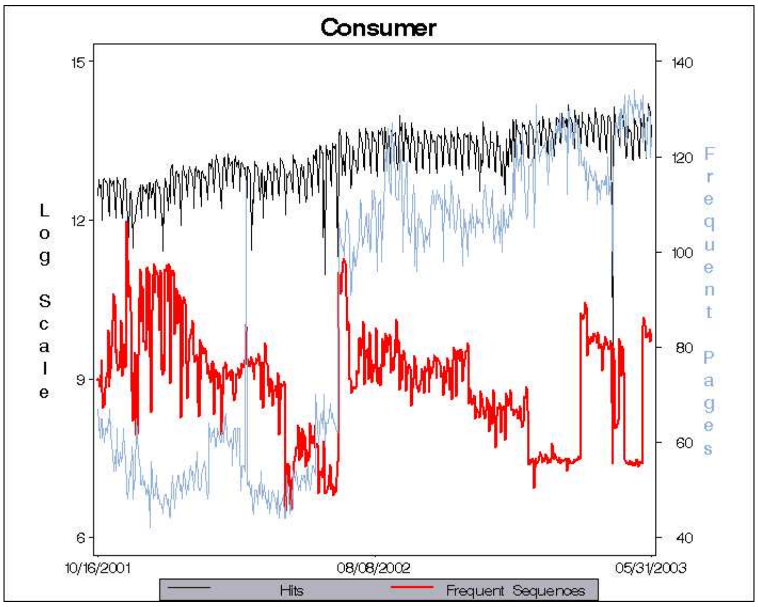


Fig. 1.6 Daily Sequence Statistics: Consumer

To view saved analyses at later times, a second page is designed (Figure 1.5(B)) as part of the page based sequence mining front-end. Users can only see their own analyses in this particular page. With two more parameters, a user can choose to view results at higher support levels than 0.1% and up to the length of their choice (`Level`.) Again, the algorithm finds up to length ten sequences by default. But user can select any length between 2 and 10 by changing the `Level`. From this point, the user interface is used as in Figure 1.4 to present the frequent sequences at the specified support level with a constraint up to the length of `Level`.

Commercial web sites are composed of several applications e.g. Bill View, Bill Pay and Registration. The reason behind the page based sequence mining is to give the ability to application owners to analyze click stream data specifically in the neighborhood of a page that they select. Usually the first page in any given application gets the highest hits and the last page (e.g., Thank-you and Confirmation pages) gets the least hits. This is so called a funnel type shape in terms of traffic. So, if page based analysis is done for the first page, the resulting click stream data will be very sparse. However, if the same analysis is done for the last page, the resulting click stream data will be very dense, since sessions are conditioned to include the successful visits to the last page. In this case, support level will not be able to prune the rule space and number of rules will be very high. For instance, the sequence mining algorithm could end up finding 800K sequences for merely 40K hits of click stream data. Thus controlling the number of child threads is almost mandatory to stabilize the program as mentioned in Section 1.3.3.

Note that analyzing thousands of frequent sequences may not be very informative due to repeating similar sequences. Hence path analysis will be very useful in this type of situation to analyze the traffic well. As seen from Figure 1.5(A), the user also has the choice of performing path analysis. The algorithm behind the path analysis is similar to webSPADE in nature but also significantly different in details. It is out of scope of this paper to explain the path analysis algorithm.

webSPADE and the web-based front-end, depicted in Figures 1.3 and 1.4, can be used for other time dependent sequential data. To illustrate the practical usage of sequence mining and to assess the performance of webSPADE we present some graphs and analyses in the next section.

1.4 PERFORMANCE AND A SAMPLE BUSINESS ANALYSIS

webSPADE has been used for analyzing web log data since mid-October of 2001. We can now analyze virtually all the frequent sequences since the inception day by using the front-end reporting tool mentioned in the previous section. As of August, 2003, there are approximately 1.2B cleaned hits in the data warehouse (webHOUSE.) Roughly, 66% of them are sessionized. Based on these data, webSPADE has found approximately 10.5M frequent sequences.

Figures 1.6, 1.7, 1.8 depicts daily hits, number of sequences and number of frequent pages from Consumer, General Business and Others (the rest.) These graphs have two axes each. The number of total hits and frequent sequences are depicted on log scaled left axes. The black lines represent the total hits, the red (thicker) lines

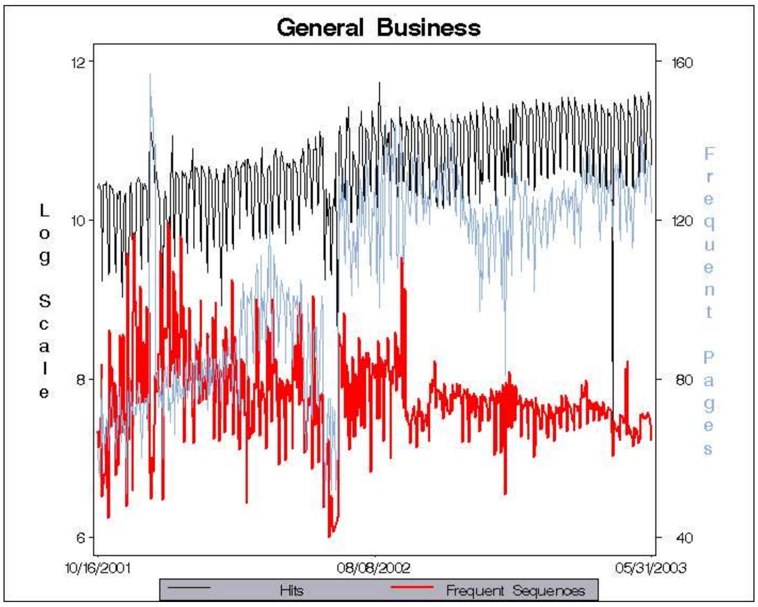


Fig. 1.7 Daily Sequence Statistics: General Business

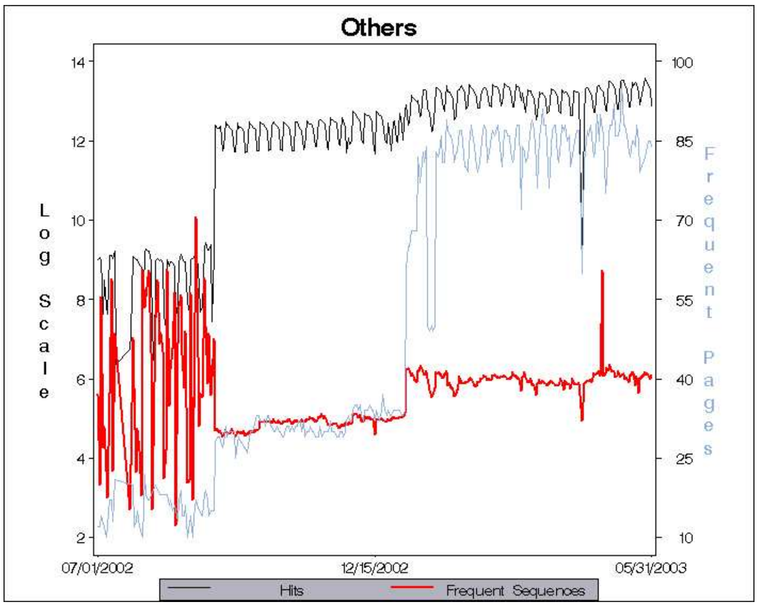


Fig. 1.8 Daily Sequence Statistics: Others

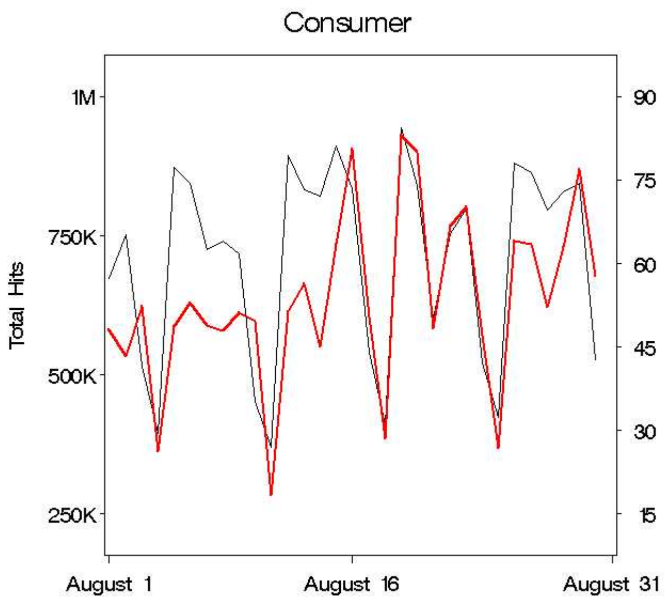
represent the frequent sequences. The number of frequent pages is depicted on the right axes with a very light blue line.

Sessionization has not been perfect since the beginning, due to the some organizational challenges. Thus we see some changes in the number of frequent pages. We can also draw some conclusion on the introduction of the .NET technologies. As seen from Figure 1.6, although the number of hits increases steadily, the number of frequent sequences fluctuates. During the introduction of the .NET technology, we had some problems in sessionization. The major problem was the transition between ASP and .NET ASP pages in terms of passing the cookie information. The other observation is that it is now possible to combine several URLs into single one with .NET implementation. Thus hits will be logged repeatedly from the same URL in the web logs. This is a pitfall for the sequence mining algorithms when the click stream data are collected via web logs. Another observation is that click stream data is highly cyclical (see Figure 1.8 with weekly cycles.)

To illustrate the usage of the webSPADE algorithm, we consider the data from January and August 2002 separately. As we mentioned above certain pages on Verizon.com are tagged to collect session information and our analysis covers only these pages. Pages from two different lines of businesses (Consumer and General Business) are analyzed separately in this section.

Figure 1.9 shows the runtimes from August 2002 data. Times are reported in minutes. The red (thicker) lines represent runtimes. Note that webSPADE runtimes are almost linearly correlated with the number of total hits. Although webSPADE is not run on a dedicated production server, the daily jobs are scheduled for the part of the day that the server usage is the lowest. Thus the example runtimes of webSPADE can be considered close to the reality. Note that runtimes also include both database access time and insertion time of sequences into the relational table.

The operational value of sequence mining is undeniable. For example, it is easy to monitor the traffic to understand whether a web page is functioning well or not. More specifically, certain pages might experience heavy traffic but the following pages may experience very low traffic. Sequence mining can easily catch such patterns. Some design problems might cause such patterns (e.g., a misplaced next button at the



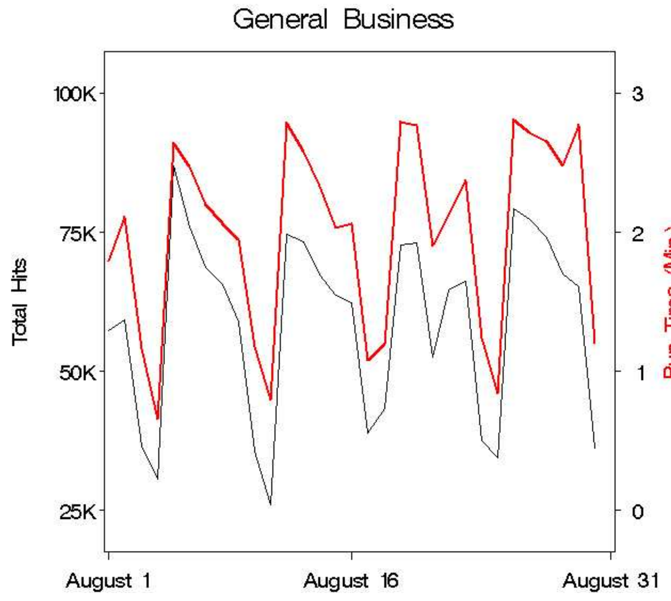


Fig. 1.9 (A) Consumer Hits and Runtimes, (B) General Business Hits and Runtimes

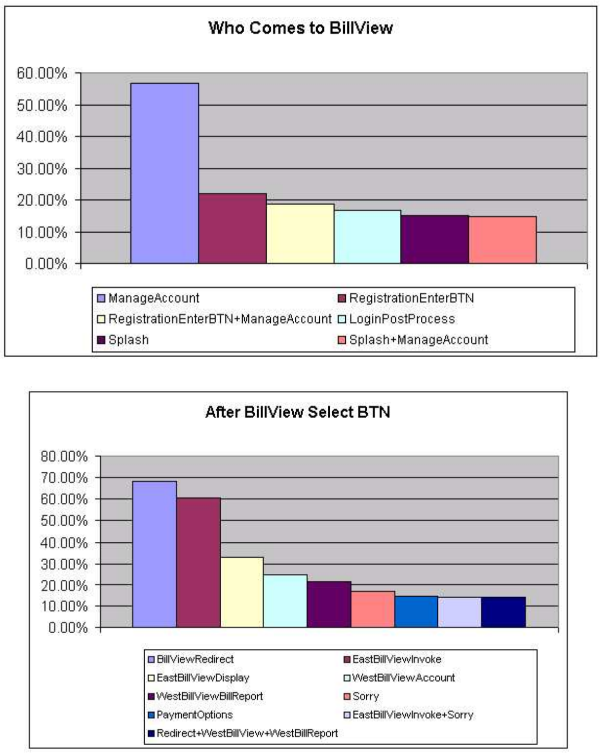


Fig. 1.10 Analysis on Bill View Page

bottom of the page; many people may not be able to see the next button because of smaller monitors.) Such changes are requested in the light of sequences mining findings.

Sequence mining can also be used to find out who comes to a certain page and where they go after that page. An analysis of a General Business page (Bill View) is depicted in Figure 1.10. Such analysis can be done easily with SQL queries at a micro level but it is not possible to cover all the relations at a macro level (whole web site.) Note that bar charts in Figure 1.10 do not correspond to probability distributions. So the probabilities do not sum up to one.

The analysis such as given in Figure 1.10 can result in very important insights. For example, although bill view and bill pay processes are independent from each other in general business pages, a significant portion of customers first view their bill and then pay it in the same session. It is also found that the bill view process sometimes fails to show bills due to the database access timeout. So, if we can increase the reliability of the bill view process, some of our customers will pay their bills in the same session. This is a very simple conclusion but it might take some time to reach with plain SQL analysis. Large scale sequence mining enables us to come to a conclusion on this matter more rapidly.

1.5 FUTURE WORK AND CONCLUSION

We successfully applied a parallel sequence mining algorithm to perform click stream analysis. webSPADE requires only one full scan of the database, but several partial scans of the database. Data and task parallelism are easily achieved using multi-threaded programming. Load balancing is left to the operating system, but the number of child threads is limited. Since the early design of the algorithm, there have been many improvements to increase the efficiency of the algorithm. In a way, the algorithm has been very stable and robust for a long time. An innovative analysis technique to scale up the sequence mining algorithm is also used for value-added business analysis.

The current implementation can be adapted to other domains as well. One immediate usage of the click stream analysis is to predict future pages through which a user may navigate. This has a potential application in proxy server management and, obviously web personalization also depends on such prediction.

Market basket analysis (MBA) is the main application domain for association mining in which rules are extracted based on purchased items in customer transactions. Rule and sequence mining algorithms such as Apriori and SPADE have superiority over other approaches such as clustering and dependency modelling. One extension to MBA is to couple its results with predictive modelling. Rules found by association mining can be used in creating new indicator variables to perform predictive modelling [22]. Such predictive models will help to categorize or segment customers based on their purchase (or navigation) patterns. In this manner, click stream analysis can be utilized to improve predictive abilities of product recommender systems by better segmenting the customers based on their navigational patterns.

Acknowledgments

I would like to thank to my colleagues who work in the webHOUSE project nearly two years of time span. Some of them are no longer with Verizon though. This project has come to the life under my former boss Walid Nouh's patient and challenging management style. I am also very much debted to Dr. Mohammed J. Zaki for his helpful comments and directions on improving this paper and work since the very beginning.

REFERENCES

1. S. Turner, "Analog." http://www.analog.cx, 2000.
2. M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning Journal*, vol. 42, pp. 31–60, Jan/Feb 2001. Special issue on Unsupervised Learning (D. Fisher, editor.).
3. M. J. Zaki, "Parallel sequence mining on shared-memory machines," *Journal of Parallel and Distributed Computing,*, vol. 61, pp. 401–426, March 2001. Special issue on High Performance Data Mining (V. Kumar, S. Ranka and V. Singh, editors.).
4. J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, "Web usage mining: Discovery and applications of usage patterns from web data," *SIGKDD Explorations*, vol. 1, pp. 12–23, Jan 2000.
5. I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Visualization of navigation patterns on a web site using model based clustering," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (Boston, MA), pp. 280–284, 2000.

6. R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in *Proceedings of the 20th VLDB Conference,Santiago, Chile*, pp. 487–499, 1994.

7. K. Rajamani, A. Cox, B. Iyer, and A. Chadha, “Efficient mining for association rules with relational database systems,” in *Proceedings of the International Database Engineering and Applications Symposium (IDEAS’99)*, 1999.

8. S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications,” in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 343–354, ACM, 1998.

9. M. Spiliopoulou, C. Pohle, and L. C. Faulstich, “Improving the effectiveness of a web site with web usage mining,” in *Advances in Web Usage Mining and User Profiling: Proceedings of the WEBKDD’99 Workshop, LNAI 1836* (B. Masand and M. Spiliopoulou, eds.), pp. 139–159, Springer Verlag, July 2000.

10. R. Agrawal and R. Srikant, “Mining sequential patterns,” in *11th Intl. Conf. on Data Engg.*, 1995.

11. R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements,” in *5th Intl. Conf. Extending Database Technology*, Mar. 1996.

12. H. Mannila, H. Toivonen, and I. Verkamo, “Discovering frequent episodes in sequences,” in *1st Intl. Conf. Knowledge Discovery and Data Mining*, 1995.

13. H. Mannila and H. Toivonen, “Discovering generalized episodes using minimal occurences,” in *2nd Intl. Conf. Knowledge Discovery and Data Mining*, 1996.

14. T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen, “A family of algorithms for finding temporal structure in data,” in *6th Intl. Workshop on AI and Statistics*, Mar. 1997.

15. J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, “Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth,” in *Proceedings*

of International Conference on Data Engineering (ICDE'01), 2001. Heidelberg, Germany.

16. J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential pattern mining using a bitmap representation," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (D. Hand, D. Keim, and R. Ng, eds.), ACM Press, 2002. Edmonton, CANADA.

17. V. Guralnik, N. Garg, and G. Karypis, "Parallel tree projection algorithm for sequence mining," in *Proceedings of Seventh European Conference on Parallel Computing (Euro-Par)* (R. Sakellariou, J. Keane, J. Gurd, and L. Freeman, eds.), pp. 310–320, Springer, 2001.

18. V. Guralnik and G. Karypis, "Dynamic load balancing algorithms for sequence mining," Tech. Rep. 00-056, Department of Computer Science, University of Minnesota, 2001.

19. A. Demiriz, "webSPADE: A parallel sequence mining algorithm to analyze web log data," in *Proceedings of The Second IEEE International Conference on Data Mining (ICDM 2002)*, pp. 755–758, IEEE Computer Society, 2002. Maebashi City, Japan.

20. M. Klemettinen, H. Mannila, and H. Toivonen, "Interactive exploration of discovered knowledge: A methodology for interaction, and usability studies," Tech. Rep. C-1996-3, Department of Computer Science, University of Helsinki, 1996.

21. M. Klemettinen, H. Mannila, and H. Toivonen, "Interactive exploration of interesting patterns in the telecommunication network alarm sequence analyzer tasa," *Information and Software Technology*, vol. 41, pp. 557–567, 1999.

22. S. Gayle, "The marriage of market analysis to predictive modeling," tech. rep., SAS Institute Inc., Cary, NC, 2000.