University of Alberta

FootPath: Web Topology and Navigation Visualization

by

Lisheng Sun 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2004

Your file Votre référence
ISBN: 0-612-95862-0
Our file Notre référence
ISBN: 0-612-95862-0

Acknowledgements

Thanks to my family, for their selfless love and support. Thanks to my supervisors: Dr. Osmar R. Zaïane and Dr. Randy Goebel, for their guidance. Thanks to Jiyang Chen, Jia Li, Tong Zeng and all my colleagues.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As one of the most important inventions in the 20th century, the World Wide Web (WWW) has changed human's life styles. People buy and sell things, make reservations, read news, and even get diplomas on the Internet. Recently, both the number and size of websites have dramatically increased. A normal website contains thousands of webpages, and some big websites contain millions of webpages. Cross-referencing and navigational behaviour make a website's structure difficult to analyze. To better organize websites and provide better service to visitors, optimizing a website's structure and understanding visitors' visiting patterns become crucial to website administrators.

While web visitors are assisted by many navigation recommendation systems ([6, 12, 22, 54, 56]) and other information retrieval tools, web site administrators and decision makers still lack effective visualization tools to help them discover and explain interesting navigational patterns.

Our system aims to visualize a website's structure and visitors' visiting patterns. The linking information of a website is used to generate a website's basic structure. Web access logs, which captures incoming visitors' activity, is used to generate new information that reflects visitors' important visiting patterns. We treat new generated information as either a usage layer or a data mining result layer, depending on the technique we use to generate information, and we put different kinds of layers on top of the website structure. Our system can improve a websites organization and

indirectly help visitors to get the information they need quickly.

1.2 Contribution

In this dissertation, we introduce the WEBKVDS: Web Knowledge Visualization and Discovery System. WEBKVDS has two parts: FootPath and Web Graph Algebra. FootPath has been developed to visualize a given website's structure and visualize the visitors' visiting patterns and data mining results on the original website's structure. Web Graph Algebra is a series of operations which are developed to manipulate a website's structure or information layers. FootPath is the basis of Web Graph Algebra. We concentrate only on FootPath; and Web Graph Algebra is discussed in [15].

Our contributions are described as follows:

- Disktree is widely-used in website structure visualization. Breadth First Search is a straightforward method to convert a graph to a disktree. Unlike other website structure visualization systems, we use a new "Usage Based" method to generate a website's structure from the original linking information. The experiments show that the new method can simplify the visualization and make the result more readable.

- We put a website's visiting pattern and data mining results on the website's structure in the disktree representation. We use both a vertex's colour and size and a edgee's colour and thickness to indicate one of four kinds of information. By visualizing information in this way, the user can get an overview of the website's structure as well as the other four kinds of information at the same time.

- When we visualize usage information on top of the website's structure, a node's size and a link's thickness may change according to the information they reflect. To fit this change and eliminate occlusion, dynamic layout is introduced. Users can control the thresholds and choose only the webpages or links that they are interested in, and the visualization result changes dynamically.

- If we take the webpages visiting sequence as a transaction, the association rule (one important branch of Data Mining) technique can be integrated into our system, and the result of association rules can be shown as a layer on top of the website's structure.

1.3 Thesis Structure

In Chapter Two, we review the background and related work on website visualization systems. In Chapter Three, we present the structure and display issues that are involved in our system. In Chapter Four, we discuss the four information layers and one data mining layer, and the revised layout algorithm. In Chapter Five, we introduce system architecture, functions, and algebra operators. In Chapter Six, we sum up and present our conclusions.

Chapter 2

Related Work

Visualization is the process of transforming data, information, and knowledge from abstract information structures into a visual form, in order to make use of human being's natural perceptual capabilities [23]. Researchers show that people are better at perceptual recognition than cognitive understanding [21]. Visualization is probably the most effective and straightforward method to help to understand complex problems.

In this chapter, we introduce some background knowledge about data visualization, data mining technique and website's structure and pattern visualization.

2.1 Data Visualization

By presenting the data in various forms and interactions, visualization helps people understand structure, features, patterns, trends, anomalies and relationships in original data. Visualization techniques have been used by human being for a long time. From the earth map to weather forecast, from DNA structure to industrial design, visualization plays an important role. Information visualization became an independent branch from scientific visualization during the last 15 years [23].

Data visualization techniques are applied to database and information retrieval systems [48] at the beginning. The basic visualization techniques include [?]:

- Point with position, colour, and size (See Figure 2.1)

  In Figure 2.1, points are put on a 3-D space. The colour of points represent a property of the point.

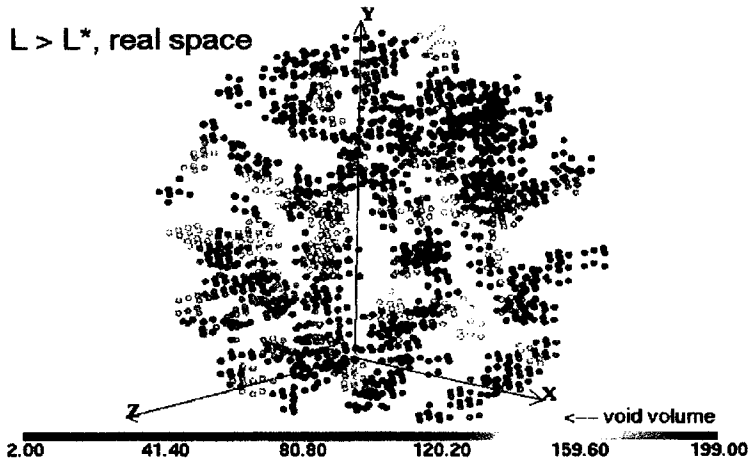

Figure 2.1: Point's position and density [34]

- Line with location, length, width, colour, and style (See Figure 2.2)

  In Figure 2.2, length and colour of lines are used to visualize properties of line object.

- Polygon with location, shape, size, orientation, colour, texture, translucency. (See Figure 2.3)

  In Figure 2.3, a series of polygons are used to visualize the distribution.

- Image with pattern, range of values, colourmap, size and scale (See Figure 2.4)

  In Figure 2.4, colour represents a property's value. The area with the same colour means this area has the similar value. This visualization technique is widely used in forecast and geographical field.

- Image with 2D or 3D scalar fields (See Figure 2.5)

  In Figure 2.5, information is visualized in 2D and 3D dimension. Each dimension represents one property.

- Point plots with position coordinates (See Figure 2.6)

- Wire frames with connection components (edges)

- Isosurface with contours or surface extracted from a field

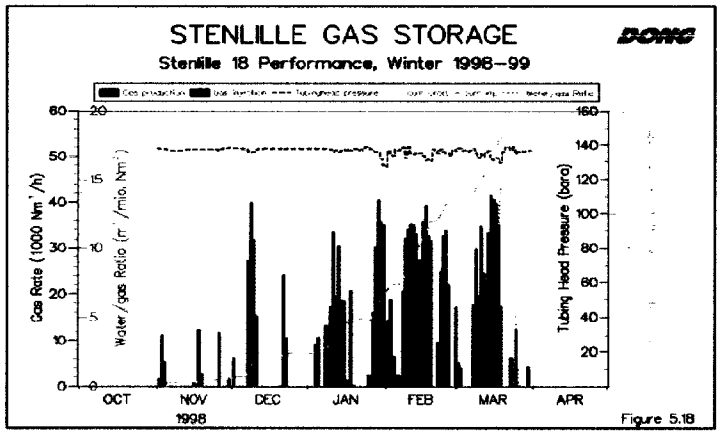

Figure 2.2: Line with location, length, width, colour, and style [32]

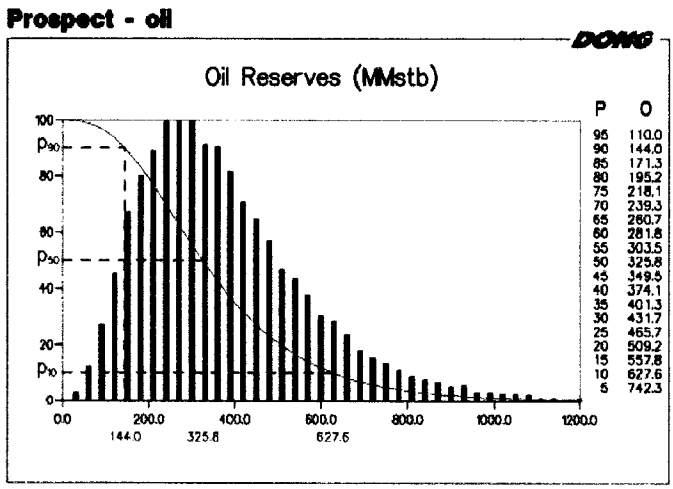

Figure 2.3: Polygon with location, size [32]

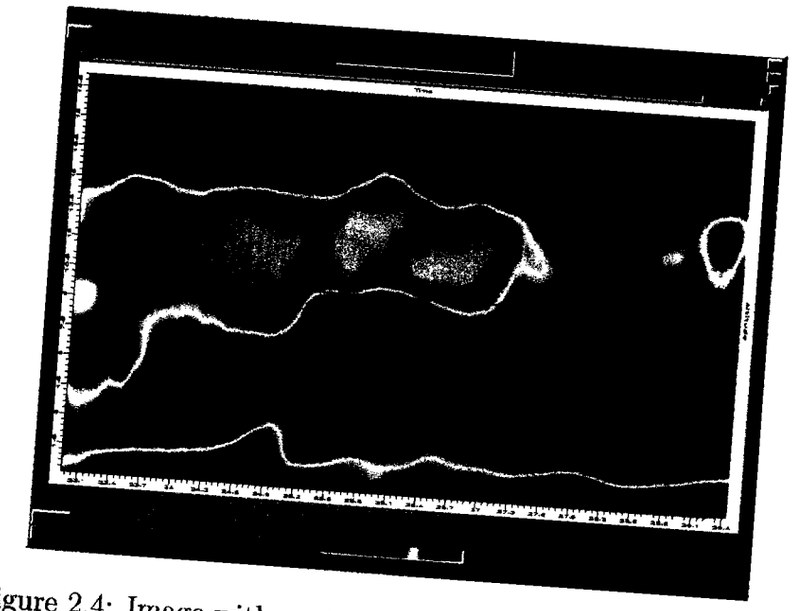

Figure 2.4: Image with pattern, range of values, colourmap [34]

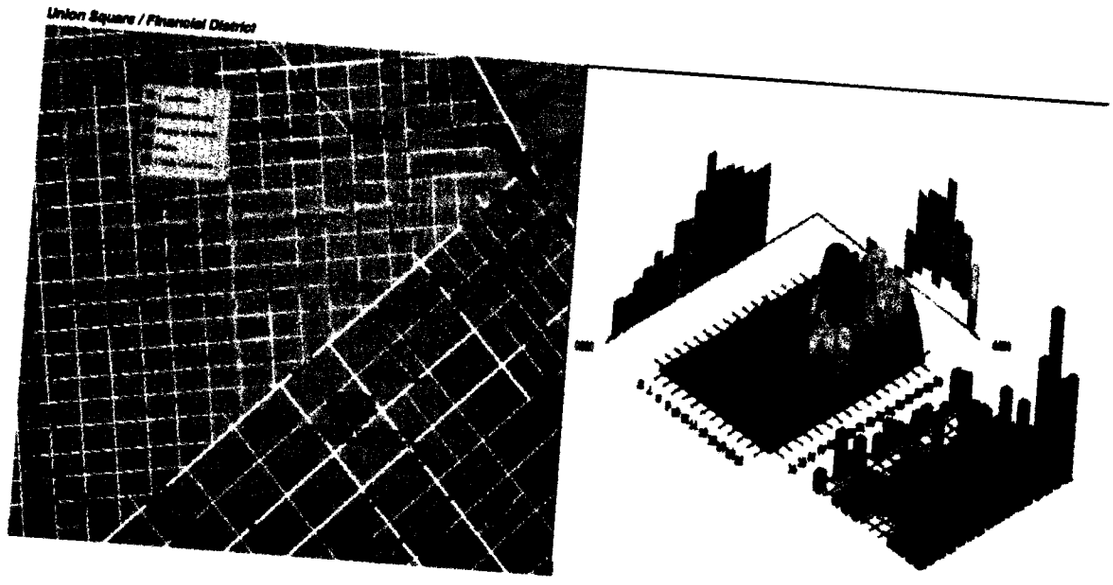

Figure 2.5: 2D and 3D scalar fields [35, 33]

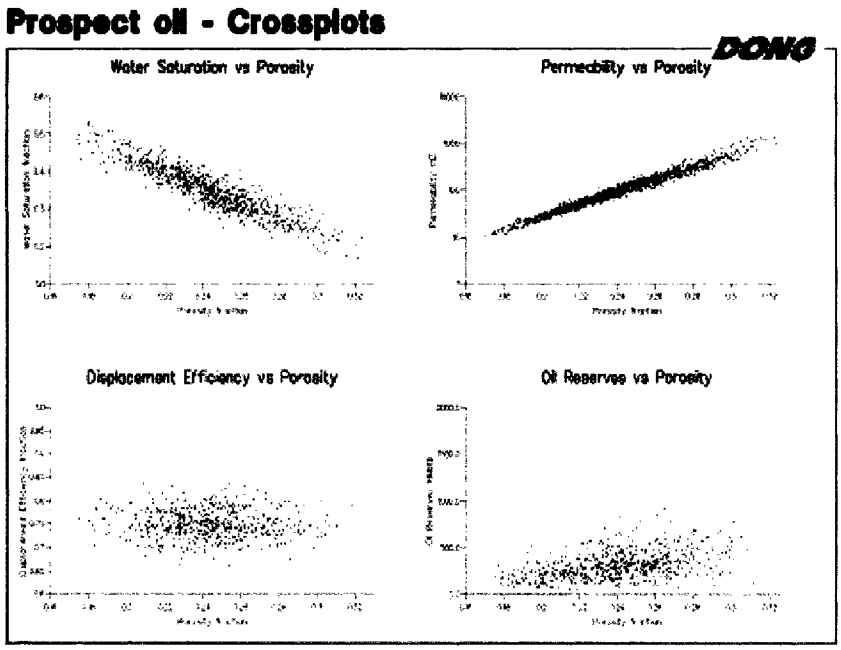

Figure 2.6: Point plots with position coordinates [32]

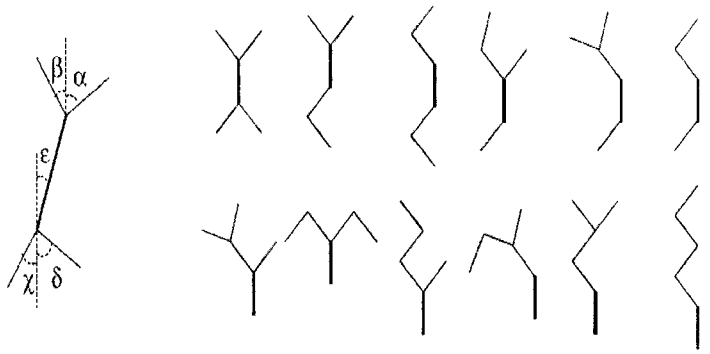

Figure 2.7: Stick figure example [51]

- Rubber sheet with 3D surface computed from 2D scalars
- Streamline flow, such as the path of a particle through a dynaic vector filed
- Ribbons and tube, extensions to the previous two flow visualizations
- Icons and glyphs, such as arrows, spheres, boxes, stars, stick figures, and faces (See Figure 2.7)

  [51] present the stick figure icon visualization technique. In Figure 2.7, each stick figure represent a object with five properties. These five properties are indicated by the five angles which form the stick figure.
- Parallel coordinates taking a multivariate point and mapping it a polyline across N vertical axes
- Dimensional stacking taking multivariate points and mapping them to space recursively partitioned (See Figure 2.8)
- Multiple objects of like type (streamlines, slices, glyphs) differentiated by style, position, or colour
- Fisheye view (See Figure 2.9)

  In Figure 2.9, the central part is enlarged and the other part is minimized.

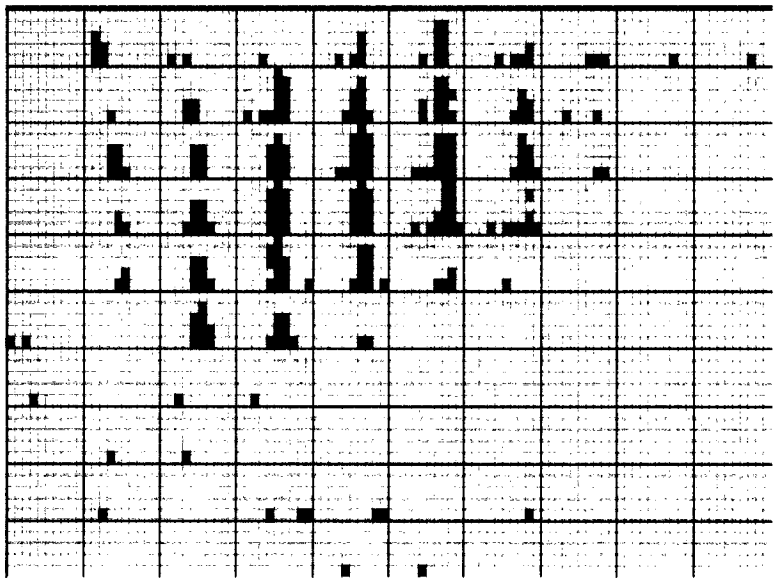

Figure 2.8: Dimensional Stacking example [35]

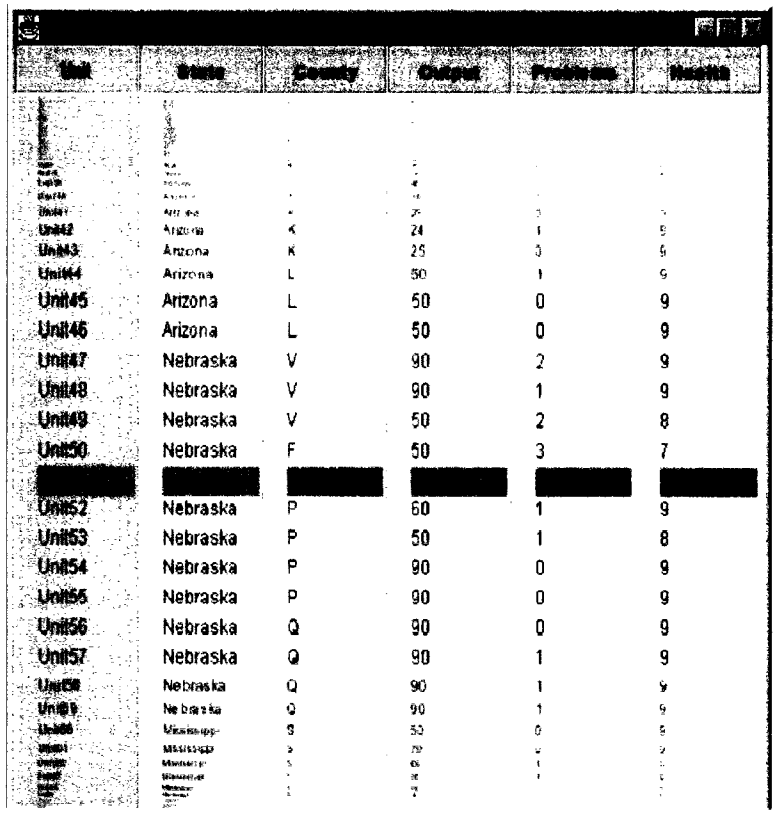

Figure 2.9: Fisheye example [32]

2.2 Data Mining

Data mining, the process of extracting hidden knowledge from large volumes of raw data, has attracted a lot of attention in the information industry recent years. Data mining tools describe existing patterns and predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions.

Data mining techniques can be classified into five major categories: association rule analysis, classification, clustering analysis, outlier analysis, time sequence analysis. We introduce these five techniques in the following subsections.

2.2.1 Association Rule Analysis

Association rule mining searches for interesting relationships among items in a data set[27]. It is a descriptive approach to explore data. A well-known example of association rule is "Market Basket Analysis". Market Basket Analysis can be described as examining a long list of transactions in order to determine which items are most frequently purchased together. The probability that a transaction contains a particular product is called *support*. The conditional probability of an transaction containing a product X also contains a product Y, is called the $confidence$. Both "support" and "confidence" are important parameters in association rule mining.

An "Market Basket" association rule example is: (Buy(A) indicates the customer buys A)

$$Buy(milk) \wedge Buy(bacon) \Longrightarrow Buy(bread)$$

$$[support = 5\%, confidence = 60\%]$$

In this example, the association rule indicates that of all the customers under study, 5% ($support$) buy together milk, bacon, and bread. There is 60% probability ($confidence$) that a customer who buys milk and bacon will purchase bread as well.

When the rules are generated out of the data, they can be used either for better understanding the business problems that the data reflects or for performing actual predictions against some predefined prediction target [10]. For example, a store manager can use the association rules mined from store's transaction data to decide whether discontinuing a product will have any effect on other products.

In the context of web mining, a transaction is simply a session of page visits done together. An item represents a visited page and the association rule associates a set of visited pages with another set of visited pages in the same session.

Apriori [2, 3] is a well-known association rule mining algorithm. A set of product items is called an itemset. Apriori algorithm first finds all itemsets that occur at least as frequently as a user-determined minimum support value. It uses an iterative approach where k-itemsets are used to find (k+1) itemsets. After finding all frequent itemsets, apriori generates strong association rules which must satisfy the minimum support and minimum confidence value.

The efficiency of Apriori algorithm is determined by three factors: the way candidate sets are generated, the data structure that is used and the implementation details. In [11], the author introduce a fast Apriori algorithm which concentrates on the implementation. It shows that some algorithms, which seem to be quite promising, may turn out to be ineffective on the implementation level.

FP-growth method [28] tries to mine the frequent itemsets without generating candidate sets. It transforms the problem of finding frequent patterns to finding shorter ones recursively.

For more details about association rule mining, please refer to [31].

2.2.2 Classification

"Classification is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown" [27, 19, 43]. Training data is used to train the model and after the model is accurate enough, we can use it to classify new unknown data. The model derived can be represented in classification rules, decision trees, mathematical formulae or neural networks.

A classification model example represented by classification rules is below: (Buy(A) indicates the customer buys A)

IF $Buy(milk)$ THEN $Buy(bread)$

IF $NotBuy(milk)$ AND $Buy(bacon)$ THEN $Buy(bread)$

OTHERWISE $NotBuy(bread)$

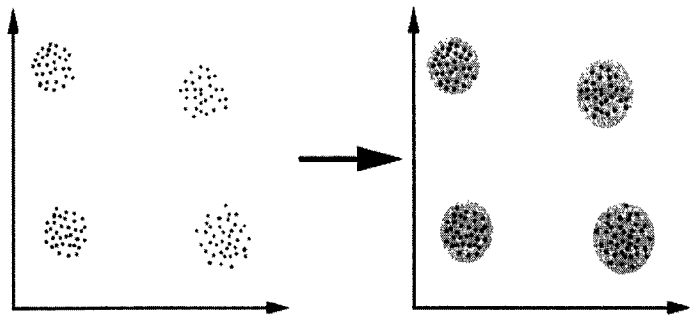

Figure 2.10: Clustering Example

The above classification rules can also be described by a decision tree [38], or neural networks [29]. After we build the model we can use this model to classify or predict new data.

The ideas of association rule mining can be applied to classification [40]. ARCS [39] finds the association rules with a form of $A_1 \wedge A_2 \Longrightarrow A_c$. A_c is assigned a class label for an attribute and A_1, A_2 are tests on attribute ranges. For more details about classification, please refer to [5].

2.2.3 Clustering

A cluster is a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. Clustering analyzes data without a known class label (unsupervised learning). The objects are clustered based on the principle of "maximizing the intraclass similarity and minimizing the interclass similarity" ([27], page 25).

Figure 2.10 shows that the points are divided into four clusters based on the 2-D distance measure.

K-means [41] is a well-known partitioning algorithm for clustering analysis. It is based on the Euclidean distance. Hierarchical clustering methods group data objects into a tree of clusters either bottom-up or top-down. BIRCH [61], CURE [25], CHAMELEON [37] are typical hierarchical clustering algorithms. Density-based clus-

tering methods DBSCAN [20], CLIQUE [1], are developed to find clusters with arbitrary shape.

For more details about clustering analysis, please refer to [59, 36].

2.2.4 Outlier Analysis

Outlier is the data item whose value falls outside the bounds enclosing most of the other data values in the sample set. Outliers may indicate anomalous data and should be examined carefully. Outlier analysis methods include statistical-based methods, distance-based methods and deviation-based methods [27]. Outlier analysis is important in some real applications, for example, credit card fraud detection, or other.

2.2.5 Time Sequence Analysis

Time sequence data consists of sequences of values changing with time [27]. Time sequence data is widely used in stock market, scientific experiments and medical treatments. By applying data mining techniques on time sequence database, we can generate the patterns of the time sequence data.

For more details, please refer to [55].

2.3 Visual Data Mining

Since data mining extracts knowledge from a database that the user did not already know about, it is not reasonable to take the output of data mining system directly and translate it into an actionable solution to a business problem. Researchers try to use natural graphical model to represent data mining result and allow users to discuss and explain the logic behind the model.

2.3.1 Association Rule Visualization

There are a few attempts to visualize association rules, but the most relevant to our work is the visualization proposed in DBMiner [26]. DBMiner visualizes association rules in two ways: plane form (Figure 2.11) and rule graph (Figure 2.12). In plane form, each bar, which locates in the intersection of two axes, represents an association

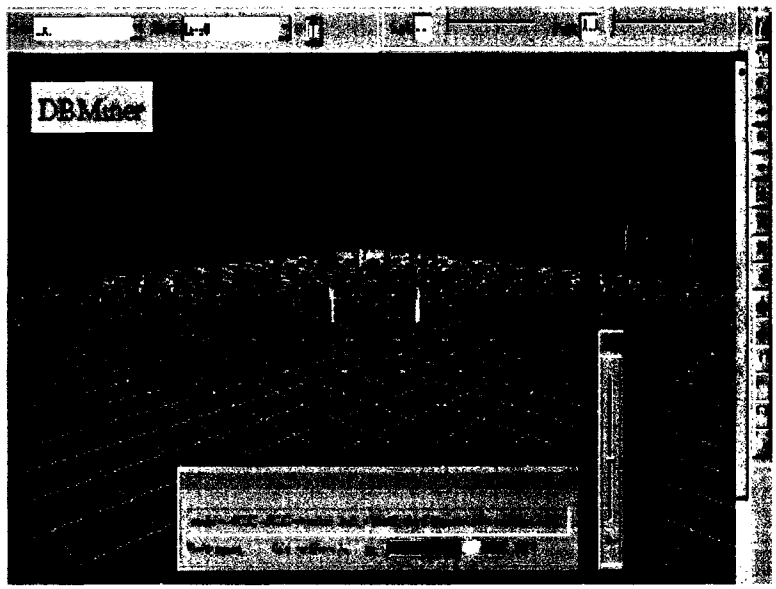

Figure 2.11: Association Rule Visualization [26]

rule. The labels on two axes represent the rule head and the rule body correspondingly. The height of a bar represents support and the colour of a bar represents confidence. Rule graph consists of a set of nodes and arrows. A circular node represents a frequent data item and the volume of the ball represents the support of the item. An arrow between two nodes represents the rule. The confidence of the rule is represented by the size of the arrow head. This idea is later borrowed in our own system to visualize associations between webpages visited together in a website.

2.3.2 Classification Visualization

The authors of [4] introduce a fully interactive method based on a multidimensional visualization technique and appropriate interaction capabilities (See Figure 2.13). In [4], a pixel-oriented visualization technique is introduced to map each attribute value of each data object to one coloured pixel and represent the values belonging to different attributes in separate subwindows.

Many other visualization technologies have been devised but also with the invention of interactively mine for the classes. In our system we intend to represent classes by colour. Each class member rendered with a class predefined colour.

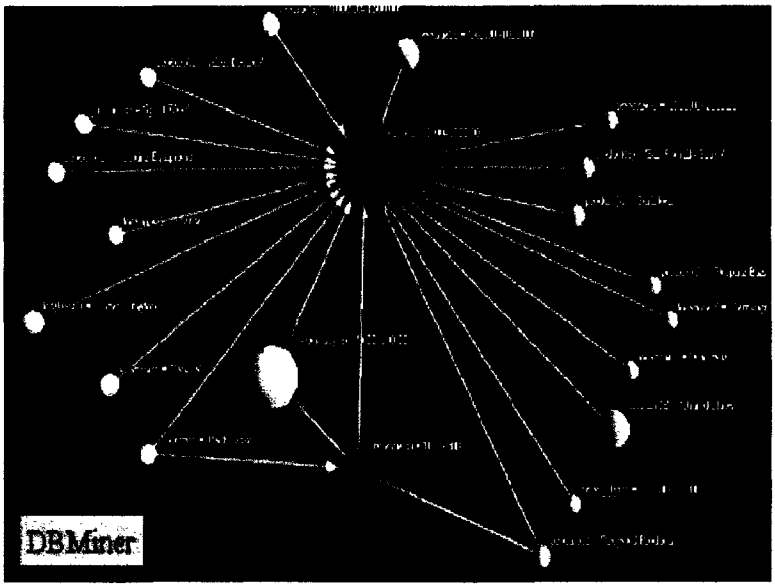

Figure 2.12: Association Rule Visualization [26]

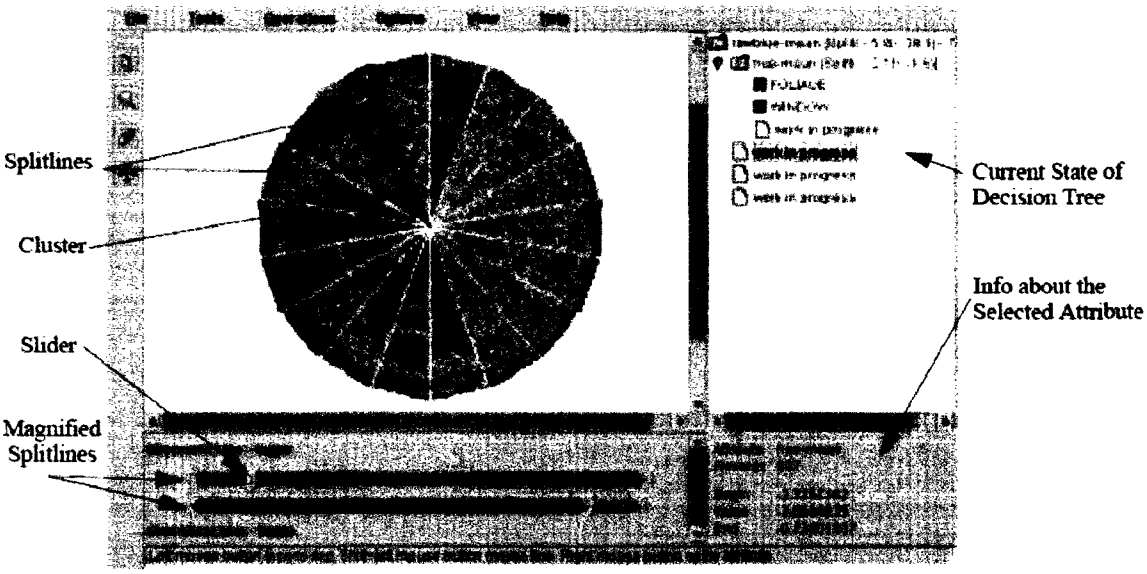

Figure 2.13: Classification Visualization [4]

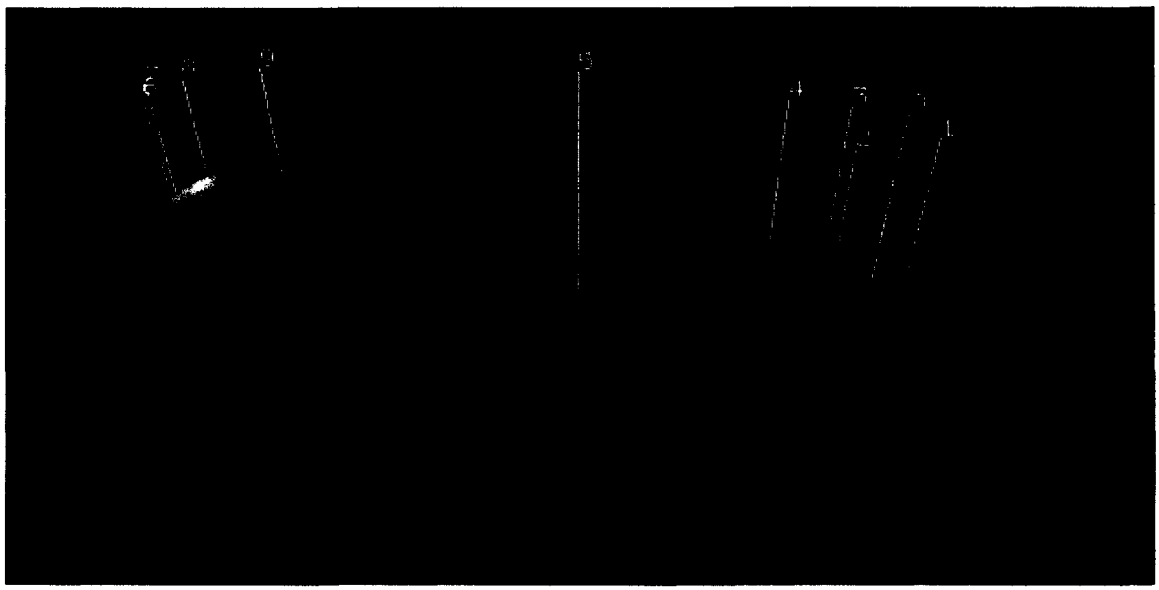

Figure 2.14: Clustering Visualization [42]

2.3.3 Clustering Visualization

Normally the clustering result is visualized by colouring each cluster with one unique colour [42] is. In Figure 2.14 the mountain visualization displays ten clusters in two major groups. Here each cluster indicates the area that has a similar height. In our system clusters, like classes, are visualized using different colours.

2.3.4 Time Sequences/Patterns Visualization

A pattern is a frequent or widespread incidence and always important to learn and study. Learning different patterns can help discovering and understanding the new trends. The authors of [60] use a hierarchical tree to show the history of user's visiting sequence. Each page that visited is indicated by a thumbnail picture and all the pictures are connected in the same order and sequence that the user visits the corresponding webpages. See Figure 2.15 for detail.

Zoom In/Out functions are provided in [60] to help navigating both overall and detail information. The system only concentrates on single user traversal. As a result, the visualizing result is always simple. The drawback of this method is that it can not discover the patterns which are composed by many users' traversal information.

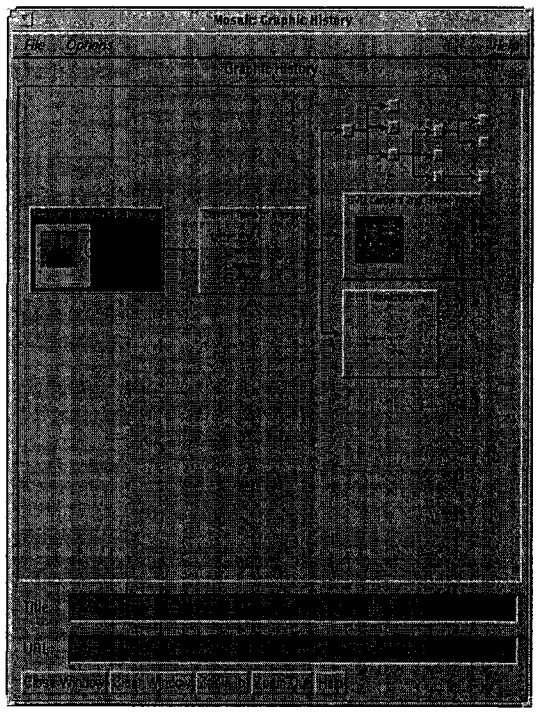

Figure 2.15: Mosaic system demo [60]

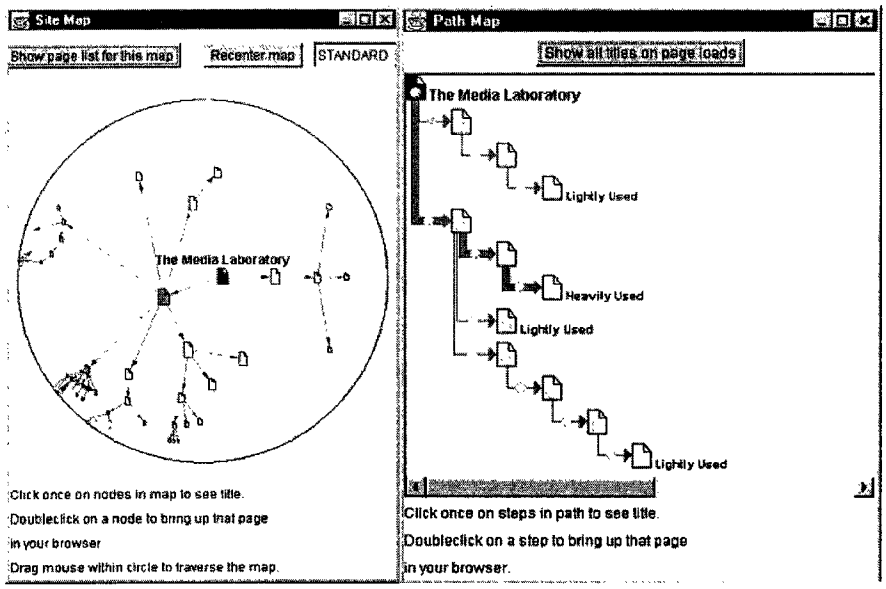

Figure 2.16: Footprint result [57]

The technique introduced in [57, 44] does not only concentrate on single user's experience, it focuses more on the history-based data. A hyperbolic tree technique is used to visualize the well-used hyperlinks and webpages. By collecting historical data as a reference and visualize them, the system can help future users navigating the complex website. See Figure 2.16.

Similar to [57], the technique proposed in [8, 7, 9] concentrates on all visitors' visiting histories. The system tries to visualize website's visiting patterns by dividing all webpages into different levels. "Concept hierarchies" and "interval-based coarsening" are used to aggregate webpages. Applying the techniques mentioned above, the system constructs a high level description of customer behaviors on a commercial website. Figure 2.17 is a result with a specific setting. It shows how customers visit this commercial website and which webpage and hyperlink are frequently visited.

Besides the above mentioned technique, the author in [13] uses clustering technique to visualize website navigation patterns.

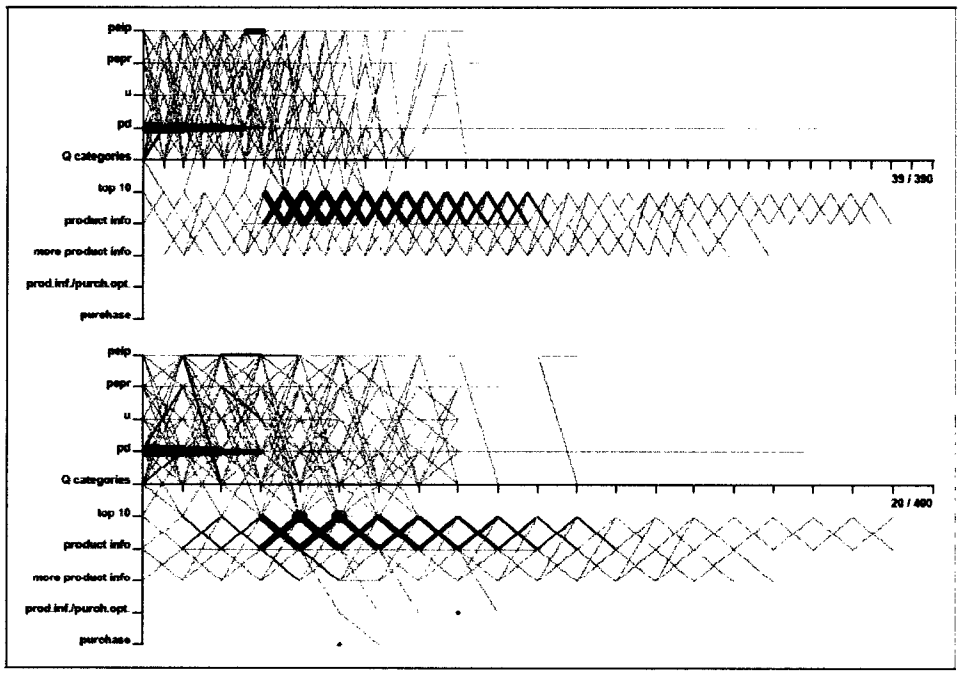

Figure 2.17: Customer's visiting sequence [8]

2.4 Web Visualization

If we map a website to a graph: a webpage is mapped to a node and a hyperlink is mapped to a edge, the result is a complicated directed graph. Researchers have proposed different visualization methods to visualize websites. These methods can be divided into four major categories: tree/disktree visualization, 3-D conetree visualization, net visualization and 3D hyperbolic visualization.

2.4.1 Tree/Disktree Visualization

Techniques proposed in [50], [58], [17] are in this category.

Disktree is a tree-like structure. The difference is that in disktree, the root is put at the center and its descendants are put around it, with 360 degree to expand.

To use a tree/disktree to visualize the website structure, a key step is to convert the complicated directed graph to a undirected tree. BFS (Breadth first search) and DFS (Depth first search) are the two methods that are often used. When use breadth first search to travel from the index page, the links which are not visited during

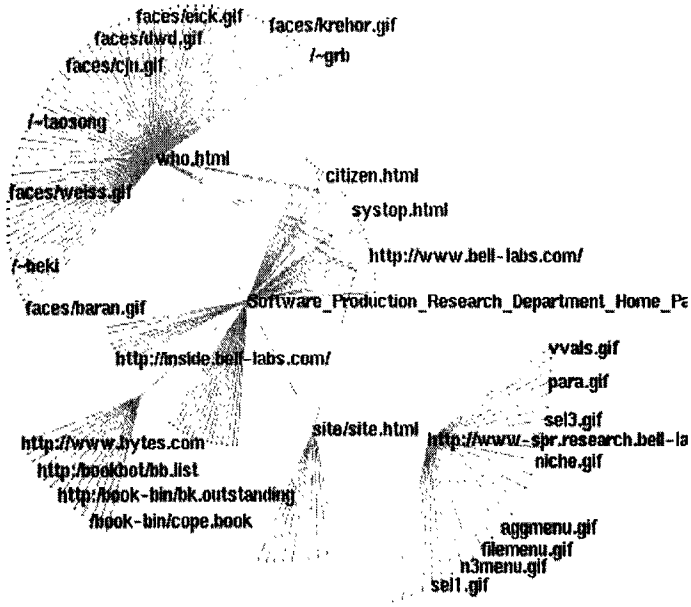

Figure 2.18: Disktree visualization example [58]

breadth first search are removed. In Chapter Three, we will talk about BFS and DFS in detail. Figure 2.18 and 2.19 are two example of tree/disktree visualization.

2.4.2 3D conetree Visualization

In [53], the authors introduce a 3D conetree visualization to visualize hierarchical information structures. To maximize effective use of available space, the hierarchical structure is presented in 3D:

In Figure 2.20, the top of the hierarchy is placed on the top of the space, and is the peak of a cone with its children placed along the base evenly. The nodes in the next layer are put below the first, with their children in cones.

A drawback of the conetree visualization is that this method "break down" once the hierarchy to be displayed exceeds roughly 1000 nodes. In [14], the author augments cone trees by some techniques: filtering, zooming, rotation, fish-eye zooming and enhance the usefulness of cone tree visualization by eliminating cluster.

Figure 2.21 is a top view of the revised cone tree algorithm. Experiments show that the system can successfully rendered up to 5000 nodes [14].

Figure 2.19: Disktree visualization example(for four different month data) [17]

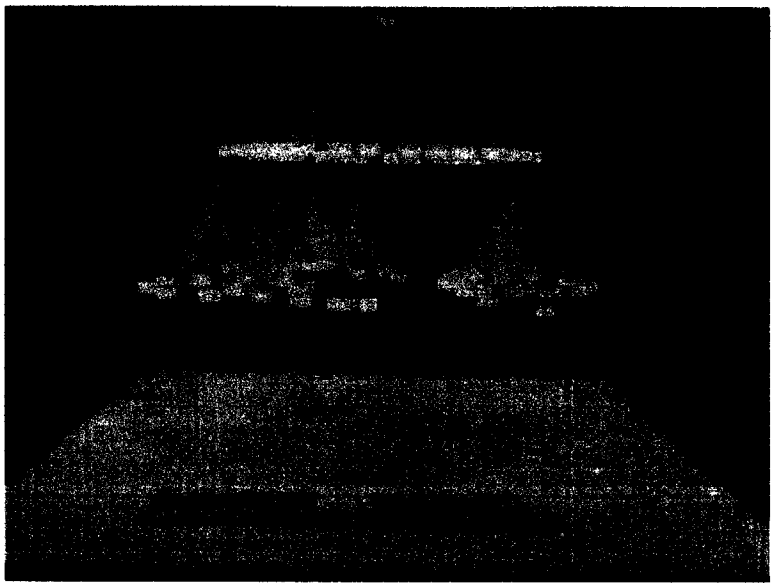

Figure 2.20: 3-D cone tree example [53]

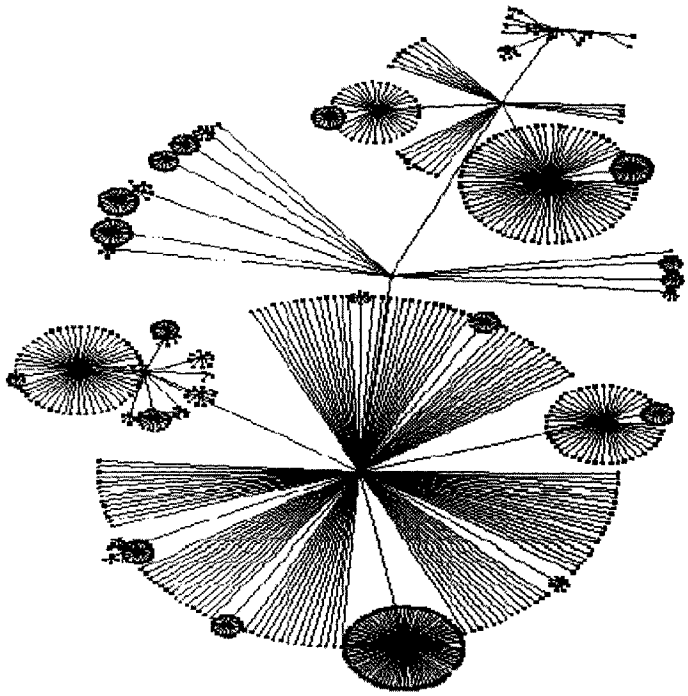

Figure 2.21: Top view of revised cone tree visualization result [14]

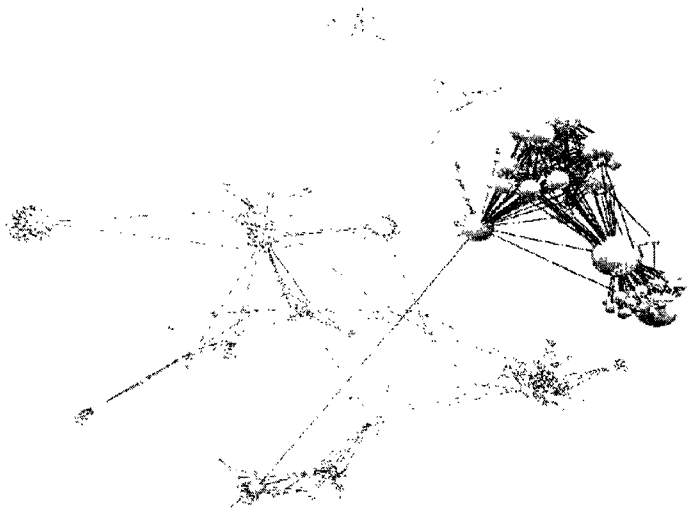

Figure 2.22: Net visualization [30]

2.4.3 Net Visualization

Though hierarchical tree structure prevails in website visualization, tree structure is only an approximation of website's structure. The author of [30] uses a net structure to visualize website's structure, which does not lose any information. It uses a 3-dimensional representation and users can get more detail information (Figure 2.22).

2.4.4 3D Hyperbolic Visualization

H3Viewer system [47] visualizes the website structure as a graph in 3D hyperbolic space. The advantage of the system is that it can handle graphs two orders of magnitude larger than other systems. (See Figure 2.23)

3D conetree visualization, net visualization and 3D hyperbolic visualization have a common drawback: occlusion. 3D methods can provide user high dimensional visualization result, but using 2 dimensional screen to display 3 dimensional pictures may cause occlusions. With large website and large number of webpages, the occlusions may mess the visualization result and the users may lose some important details.

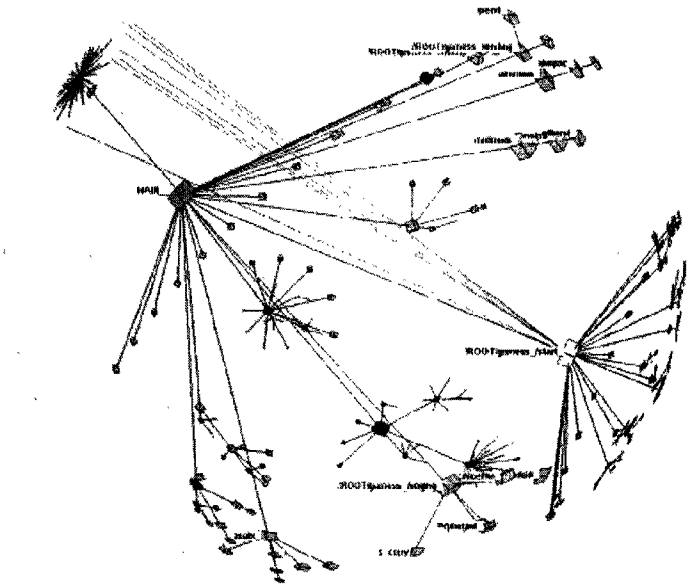

Figure 2.23: 3D hyperbolic visualization [47]

Chapter 3

Displaying Web Structure

3.1 Website Structure Visualization

As previously mentioned, our approach for exploratory data analysis is basically to visualize a website's structure and then put the usage or data mining information on the original website structure as layers. We can see that the website's structural visualization is the fundamental part of our visualization system.

At the same time, the layout algorithm is another important issue which we need to consider carefully. Due to the large number of pages in a website and screen space constraints, we try to find the best algorithm that can use the screen space most efficiently. In the following sections, we will describe FootPath, the visualization component of WEBKVDS, using data from the Department of Computing Science at University of Alberta's website and discuss the system design issues.

3.1.1 Tree Layout vs. Disktree Layout

Two natural ways to map a website's structure are Tree layout and Disktree layout.

Tree structure is defined as a hierarchical organization in which a given node is considered to be an ancestor of all the lower level nodes to which the given node is connected. Tree layout was first used in visualizing file systems. The author of [18] describes a tree layout algorithm to map a website's structure to a screen. The advantage of the tree layout is that it is simple and easy to understand, while the disadvantage is obvious, too: it only works well to display very simple graphs, and it does not use screen space efficiently; tree layout expands only to one direction and

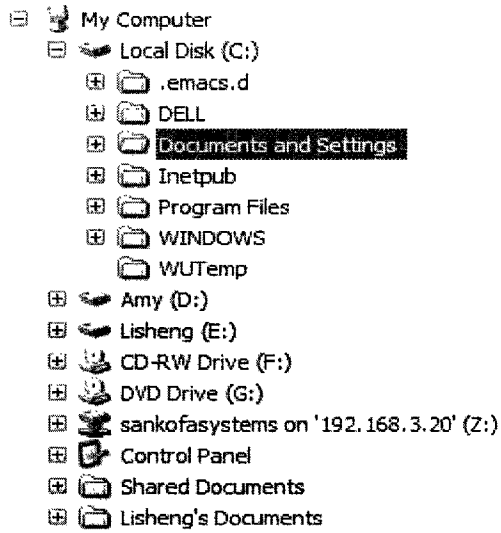

Figure 3.1: Tree visualization of file system

wastes spaces in the other directions (See Figure 3.1).

Disktree layout [50, 58, 17] is a revised version of tree layout. It puts the root to the center and expands around the center (See Figure 3.2). Disktree layout is more suitable for visualizing a website's structure than tree layout, for it can display more in the same space than a tree display. Generally speaking, a disktree layout requires 360 degrees to expand, while the tree layout needs less than 180 degree to expand.

3.1.2 Disktree Representation of Web Topology

We use a website's linking information, which we can easily obtain from a webpage's HTML code, to visualize a website's structure. A graph can be generated after reading all linking information. Here, a node indicates a webpage and a edge indicates the hyperlink from one webpage to another. Unfortunately, the visualization result using a graph is distorted by the serious cross-linking between webpages. So the disktree is introduced to visualize the website's structure instead of the original graph.

In previous research, a tree was used to represent hierarchical information. To use space most efficiently, visualization researchers manipulate the original tree to a disktree, by putting the primary node at the center with each successive descendant

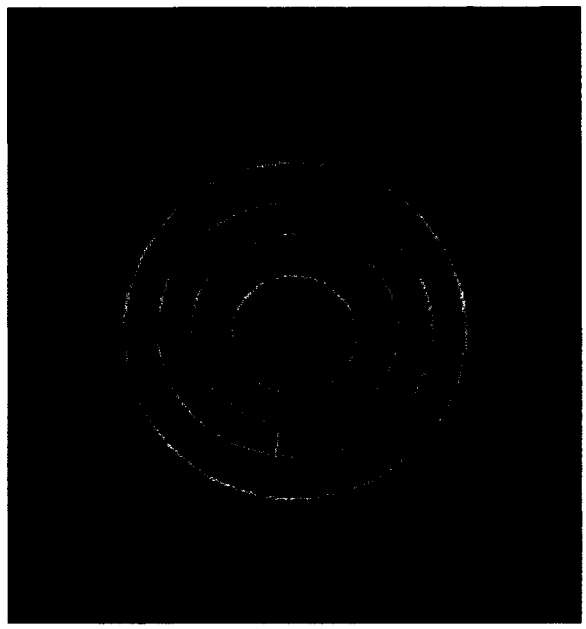

Figure 3.2: Manipulated disktree [17]

falling on concentric rings spanning out from the center. On each ring, a node is allocated space according to how many leaf nodes fall under it (See Figure 3.2).

The original website structure is a complicated graph. To visualize the structure clearly, we use a disktree instead of the original graph. In the process of converting a graph to a disktree, two points need to be considered. First, we must determine which node in the graph should be choose as the disktree's root. It is put in the center. Website's index page is unique and important in many ways, so we usually take the index page as the root of the disktree. Second, knowing the root node, we must decide how to convert a graph to a tree. In another words, we must determine which edges we should keep and which we should delete.

3.1.3 BFS and DFS

The choice of how to convert a graph to a tree is made depending upon the algorithm to traverse the graph. In most previous systems which aim to visualize websites' structures, the BFS (breadth first search) method is the most often used technique to convert a graph to a tree/disktree. This algorithm travels the graph from the index page using breadth first search and removes the edges which are never used in the

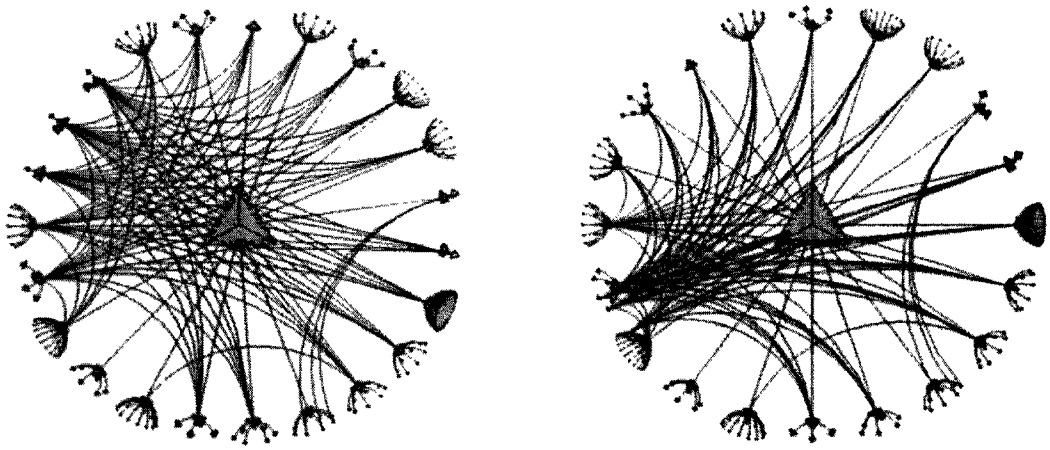

Figure 3.3: BFS V.S. DFS result [49]

process of traversal. Like BFS, the DFS (depth first search) method traverses the graph from the index page using a depth first search and removes the edges which are never used in the process of traversal. The BFS method is straightforward and easy to implement, but it can not be guaranteed that the BFS method is better than the other methods. The DFS method tries to visit as deeply as possible and as a result, the branches of the root, which are visited first, are likely to contain more nodes than the branches which are visited later. In [49], the author compares both the BFS and the DFS representation and comes to the conclusion that the choice of traversal order has a major impact on the comprehensibility of the visualization result. The study concludes that BFS method can provide a more balanced, clear result (See Figure 3.3).

3.1.4 Usage Based Method

Simplicity and clarity are important to visualization result. Adding information layers on the original graph requires more space for each webpage or hyperlink object. This is because the layers are represented by the size of nodes or thickness of edges which obviously needs physical space to be rendered. For this reason, we try to optimize

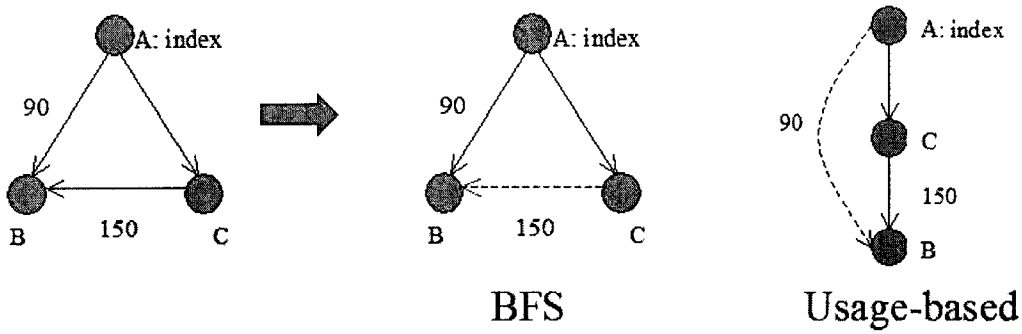

Figure 3.4: Simple Usage Based method example

the space between nodes in the graph and at the same time eliminate occlusion. The BFS method intends to put more nodes in the lower levels, and, consequently, the visualization result is crowded in the low levels. In our system, we introduce a "Usage Based" method to convert a graph to a disktree. In the new method, we consider both a website's linking information and hyperlinks' usage information to decide which edges should be kept and which should be removed. Here, link usage means how many times visitors visit a link in a specific period. Normally, each node is connected from more than one parent links, and only one parent link can be kept in order to construct a disktree.

In the new method, if more than one link is connected to a node, we only keep the link which has the largest visiting number to the node (See Figure 3.4).

This "Usage Based" method still uses BFS to traverse the graph from index page. The difference between the Usage Based method and the BFS method is that when visiting link connecting node A to node B, Usage Based method checks whether this link is the largest usage link which connect to B. If yes, this link is kept, otherwise, the link is removed. This new method renders all nodes like with the BFS method but typically results in a graph that is deeper than the one generated by the BFS method.

In the process of the Usage Based method, we need to consider the "Usage Loop" case. "Usage Loop" is a set of webpages, in which a loop is formed by the hyperlinks between these webpages. In other words, there is a cycle in the most used hyperlinks

formed by these pages. Webpages in a usage loop are connected and all the links are those that connect to another node with the largest usage to that node. For example, node A has a link to node B and this link is the largest usage link to B. At the same time node B has a link to node A, which is the largest usage link to A. In this case, both node A and node B will be excluded from the result disktree, which is not acceptable. The above example is a two usage loop case, chances are that the usage loop has a length more than two. To eliminate usage loops, our usage based method detects if there is a usage loop involved. A simple way to detect "Usage Loop" is to traverse the graph from this node only following the largest usage link. If the largest usage links can form a loop, then a "Usage Loop" has been detected. To make the process efficient, we consider up to ten levels. We try to break the usage loop by keeping the not-largest-usage link in the tree. For example, if a link L connects from node A to node B, and node B is involved in a usage loop, though this link may not be the largest usage link to B, we still keep this link to eliminate usage loop.

Table 3.1 is the pseudocode of Usage Based method.

3.1.5 BFS method vs. Usage Based Method

We use one month's weblogs of Department of Computing Science at University of Alberta's website to do a comparison of the BFS method and the usage based method visualization results. The results show that the "Usage Based" method can simplify the visualization, which means more space is allocated between neighboring nodes. This space is needed for adding extra layer information and we will talk about it in detail next chapter. Like the BFS or the DFS method, the usage based method does not delete nodes, instead, it moves the nodes, which show on lower level in BFS or DFS method, to the upper level to simplify the visualization. Figure 3.5 is a comparison of BFS and usage based method visualization results with four levels.

Graph A of Figure 3.5 is the BFS result and B is the usage based method result, in both cases rendering only four levels. Note that B has less nodes than A. All nodes from A are rendered in Figure 3.6 with "Usage Based" method in six levels.

The minimum spanning tree of a weighted graph with n nodes, is a set of $n-1$ edges of minimum total weight which form a spanning tree of the graph. "Usage

Algorithm Usage Based Search
Input: A graph's orthogonal representation
Output: A tree
Procedure Usage Based Search
1. Vex-Queue $Q \leftarrow rootnode$
2. set root visited
3. **while** Q is not empty
4. $n \leftarrow front(Q)$
5. pop(Q)
6. **while** $n \neq NULL$
7. **IF**((n is not the largest usage link)**AND**(!Detect_Usage_Loop(G, n, 1)))
8. **OR**(n is not visited)
9. set n undisplayed
10. **ELSE** $Q \leftarrow n$, set n visited
11. n = n's next sibling

Procedure Detect_Usage_Loop(G, A, level)
Input: G, a graph's orthogonal representation, a node A, and the current loop level
Output: **True**, if there is a Usage Loop involved, **False** if not
1. Add A to list
2. **IF** $level >= 10$, **RETURN False** /*level 10 is arbitrarily chosen*/
3. **FOREACH** A's child i
4. **IF** link(A to i) is the highest usage of all links to i
5. **IF** i is in the list, **RETURN True**
6. **ELSE**
7. Add A to list
8. $level \leftarrow level + 1$
9. Detect_Usage_Loop(G, i, level)
10. **RETURN False**

Table 3.1: Usage Base Search Algorithm

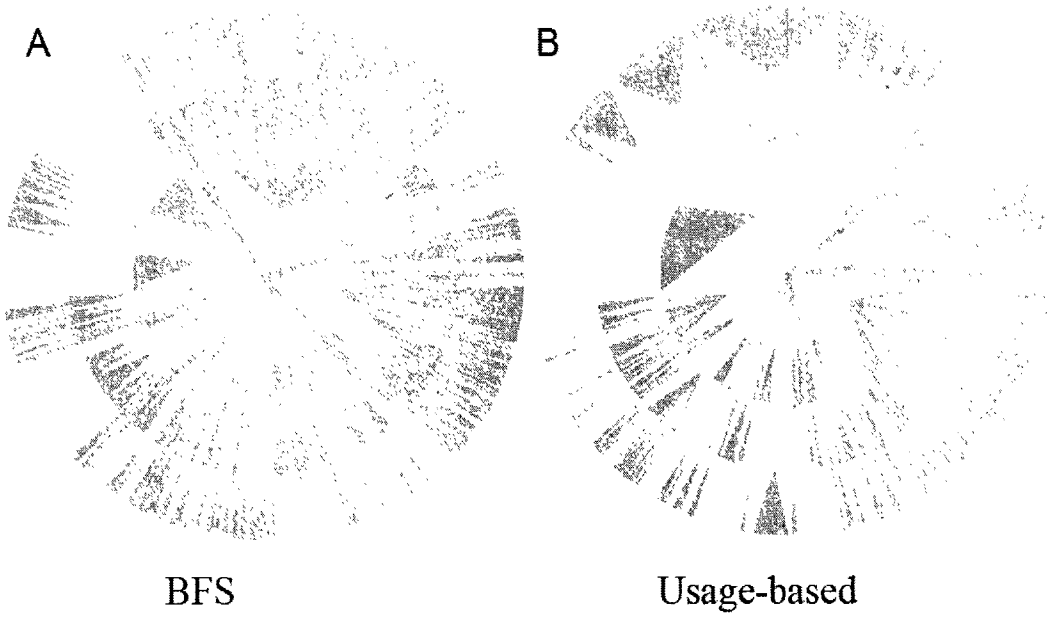

Figure 3.5: BFS vs. Usage Based result (four levels)

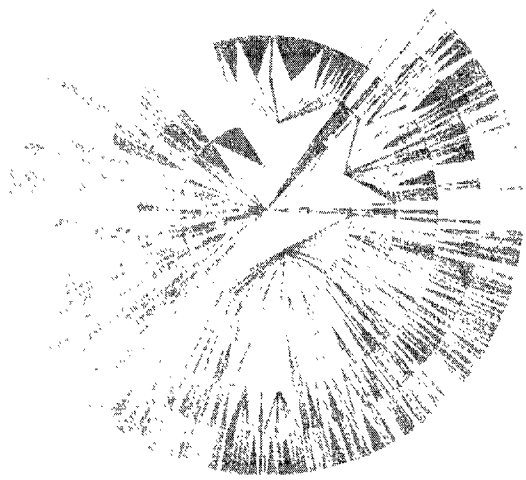

Figure 3.6: Usage Based method (six levels)

Based" method is similar as the minimum spanning tree [24] algorithm for both of them consider the weight of the edges. The difference between "Usage Based" method and minimum spanning tree algorithm is that minimum spanning tree algorithm does not care about the structure of the result tree. Moreover, minimum spanning tree algorithm focuses on a local optimum while we are trying to optimize our graph globally.

3.2 Layout Algorithm

In this section, we introduce the layout algorithm. We try to find a method which can display a graph containing large amount of nodes and edges and eliminate occlusion.

The visualization using this algorithm is constituted with nodes and edges. A node indicates a webpage and a edge indicates a hyperlink connecting two webpages. The layout algorithm maps the website's structure to a disktree, with the most important webpage: index page as the root of the disktree.

In the next chapter, we talk about the layers, which are used to put extra information on the original graph. We do not consider layers now and the graph we consider at this stage is called "Bare Graph", which means the original structure graph of the website.

Assume the current number of levels is l, the first level node (index page) is put to the center of the screen, the second level nodes are put around the center with a distance $d_2 = \frac{r}{l-1}$, where r is the vertical distance from the center to the edge of the screen. The ith level nodes have a distance of $d_i = \frac{r \times (i-1)}{l-1}$ from the center.

In [18], the author proposes a tree layout algorithm which can adjust the tree dynamically. The main idea of this algorithm is keeping all the leaf nodes with the same horizontal distance. This algorithm can not be used directly in our system because the width of the tree may increase permanently with the growing number of the leaf nodes and we do not have enough space to draw all the nodes horizontally within a constant size screen.

To draw the "Bare Graph", we use the same algorithm that the author of [50] uses. The main idea of this algorithm is to put all the leaf nodes around the center

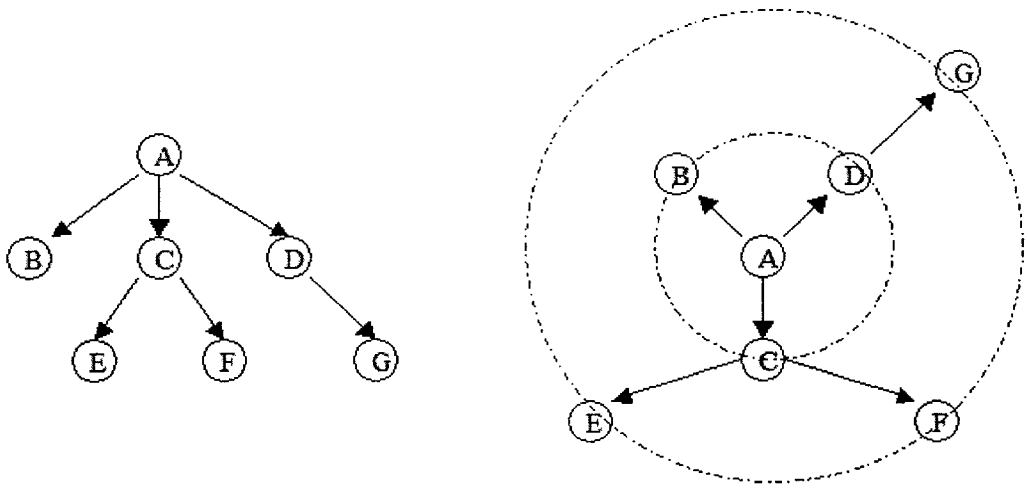

Figure 3.7: Layout algorithm simple example

and each leaf node is assigned a degree of $\frac{360}{LeafNode\#}$. After assigning the position of leaf nodes, the other nodes' position can be assigned by putting them at the middle of their children and one level lower than their children. Figure 3.7 is a simple example:

In Figure 3.7, we have a simple website's tree structure on the left hand side, we try to use disktree to display this tree structure. First, we count the number of leaf nodes: four (node B, E, F, G). We divide 360 degrees by four and node B, E, F, G are put around the center with a $\frac{360}{5-1} = 90$ interval degrees with neighboring nodes. The other nodes C, D's position can be decided by their children: C is put at the middle of E and F, D is put with the same degree as G for G has only one parent node. Our layout algorithm does not change nodes' level information, but only allocates degree to each node.

Table 3.2 is the pseudocode of layout algorithm for BareGraph:

BareGraph Layout Algorithm
Input: A graph's orthogonal representation
Output: A tree with degree and level information, ready to visualize
Procedure BareGraph Layout
1. post-order-travel(disktree), count(leaf nodes#)
2. The degree between any two neighboring leaf nodes = $\frac{360}{n}$
3. post-order-travel(disktree) again, allocate the degree to each leaf node
4. Compute leaf nodes parent's degree: parent's level = child's level - 1,
5. position = middle of all childrens

Table 3.2: Layout Algorithm

Chapter 4

Layers and WebGraph

In this chapter, we discuss layers in detail and the definition of webgraph. In last chapter, we use disktree representation and "Usage Based" method to visualize the website structure. So far, what we are considering is the "Bare Graph", the original graph mapped from the website structure. The information we extract from weblogs by statistics method is called usage information. The information we extract from weblogs by data mining techniques is called data mining information. We try to visualize these information as layers on top of the "Bare Graph".

4.1 Usage Layer

Web servers maintain weblog files which list every request made to the server. By analyzing weblogs, we can get a good idea of where visitors are coming from, how often they visit, and how they navigate through a website. Using cookies enables webmasters to distinguish individual users and log detailed information about how individual users are accessing a site.

The typical format of web log item is:

remotehost rfc931 authuser [date] "request" status bytes

remotehost represents "remote hostname", **rfc931** is the "remote logname of the user", **authuser** shows the "username as which the user has authenticated himself", **[date]** is the "Date and time of the request", **"request"** shows the request line exactly as it came from the client, **status** represents the HTTP status code returned to the client, **bytes** is the content-length of the document transferred. Here is a real

web log file item:

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

Our system has a module to analyze the weblogs and generate the information that we are interested in, for example, in a period of time, how many times visitors visit a specific webpage, what's the average time the visitors view a webpage, how many times visitors follow a hyperlink connecting one webpage to another and what's the probability that a visitor will traverse this hyperlink.

4.1.1 Node's Visit_Number Layer

We first talk about the node's "Visit_Number Layer". The preprocessed weblogs can be used to generate the information of how many times visitors visit a webpage in a period of time. The result is computed simply by scanning the weblog items one by one and counting the number of appearance of that webpage.

To visualize a webpage's visiting info, we use the size of the node to indicate it and display it as "Node's Visit_Number" layer. The size of the node corresponds the number of visitor to this webpage. Node size's scale is also provided to the user for further comparison.

Figure 4.1 is an example of "Node's Visit_Number Layer" on the original "bare graph".

4.1.2 Node's View_Time Layer

Another information we are interested in is the visitors' average view time of a webpage. We use the colour of the node to indicate the average view time of that webpage. Every colour indicate a range of viweing times. Colour's scale is also available.

Figure 4.2 is an example of both "Node's Visit_Number Layer" and "Node's View_Time Layer" displaying on the original "bare graph".

4.1.3 Link's Link_Usage Layer

Like "Node's Visit_Number" layer, we introduce "Link's Link_Usage" layer to visualize hyperlink's visiting usage, that is how many people visit this hyperlink connecting

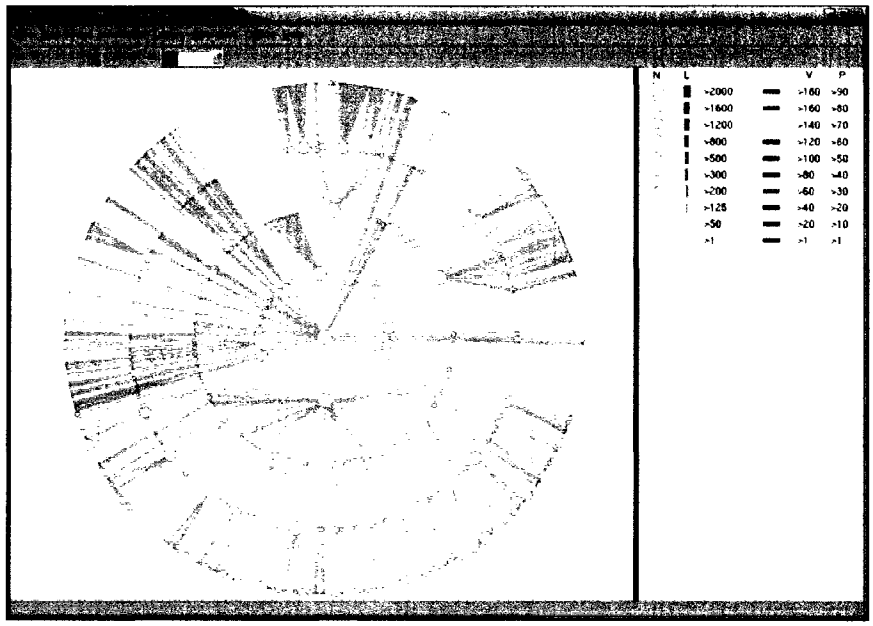

Figure 4.1: Node's Visit_Number Layer

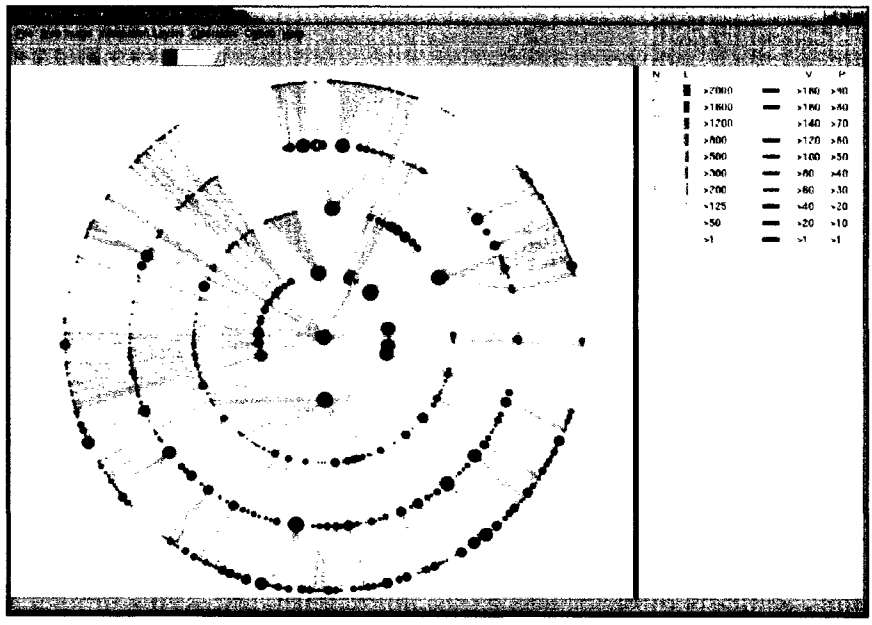

Figure 4.2: Node's Visit_Number and View_Time Layer

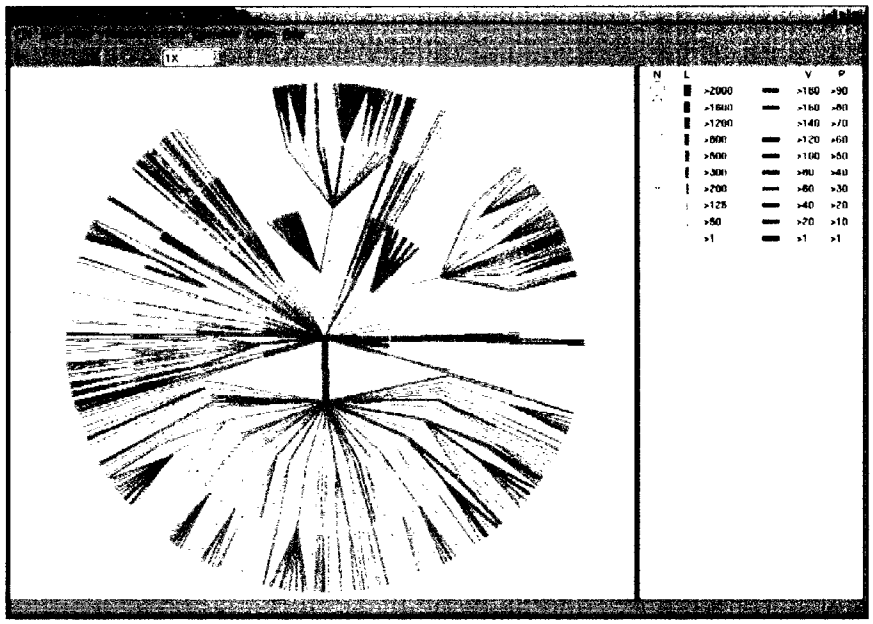

Figure 4.3: Link's Link_Usage Layer

one webpage to another in a period of time. The thickness of the link corresponds to the usage of the corresponding hyperlink.

Figure 4.3 is an example of "Link's Link_Usage Layer" displaying on the original "bare graph".

4.1.4 Link's Prob_Usage Layer

We use the colour of the link to indicate the probability that a visitor will traverse the corresponding hyperlink, and visualize it as "Link's Prob_Usage Layer" on original "bare graph".

Figure 4.4 is an example of "Link's Link_Usage" layer and "Link's Prob_Usage" layer displaying on the original "bare graph".

4.1.5 Interpretation of Layers

One advantage of using the concept of "layer" is that we can visualize many different kinds of information together. We can put different kinds of layers on the original "bare graph" together.

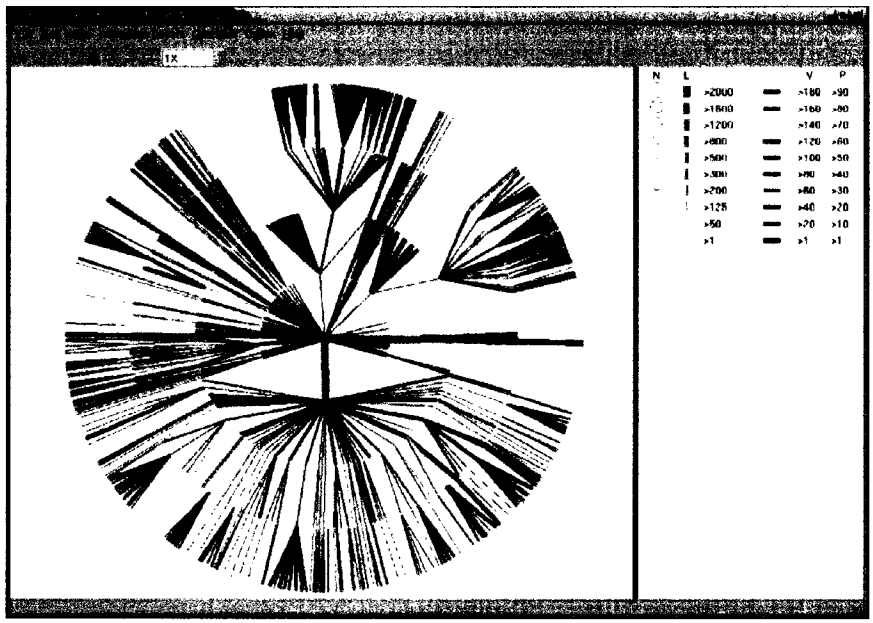

Figure 4.4: Link's Link_Usage and Prob_Usage Layer

Figure 4.5 is an example of mapping all four layers we discuss above on the original "bare graph".

In addition to the four layers we discuss above, we can generate more new layers to visualize new information. For example, the size of a page, the type of a page, etc. could be represented using texture or shape of the node, or other visual means.

How to interpret the visualization result we get? An example can be found from Figure 4.5: both node A and B experienced a large amount of visiting, but the link between A and B is thin, which indicate few people follow this link travelling from A to B. From the above analysis, we can suggest extracting node B and its children as a separate website.

4.2 Data Mining Layer

We introduce four information layers in the last section. Now we introduce one data mining layer: "Association Rule" layer.

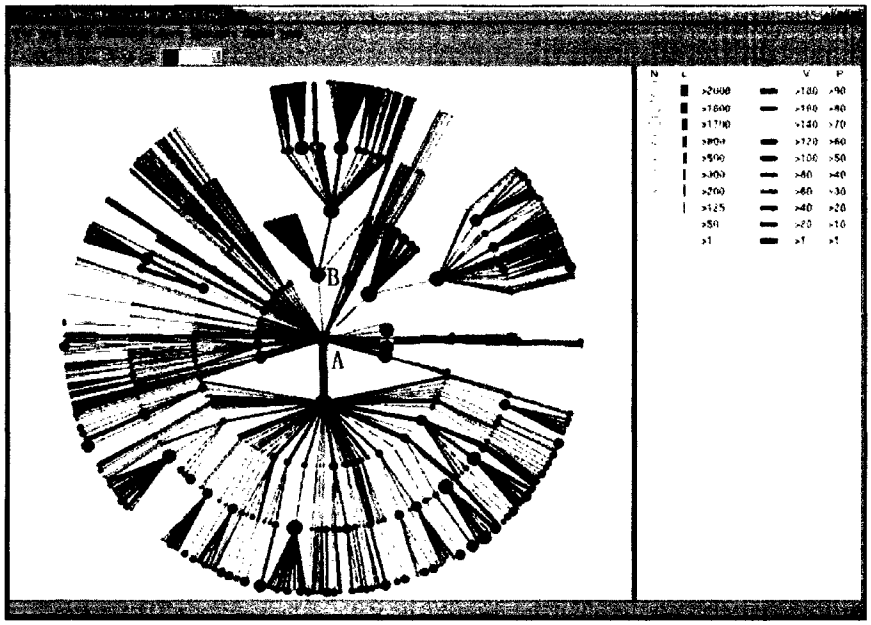

Figure 4.5: Four Layer Together

4.2.1 Association Rule Layer

Association rule is a important branch of data mining and in chapter two, we review the background of association rule analysis. Association rule mining can be adapted to our system by taking visitor's visiting sequence (a sequence of webpages) as transaction. The visiting sequence can be easily generated from the server weblog files after identifying user and session. After the user set the "support" and "confidence" thresholds, our system generate the association rules and visualize the association rules result on the original graph. Figure 4.6 is a simple example:

In Figure 4.6, we use a blue line with red triangle on it to indicate one association rule. The thickness of the blue line corresponds to the association rule's "support" and the size of the red triangle corresponds to the "confidence".

4.3 Dynamic Layout Algorithm

We discuss the layout algorithm for "bare graph" in the last chapter. Now, when different kinds of information layers or data mining layer are put on to the original bare graph, we need to revise the layout algorithm to eliminate occlusion and fit the

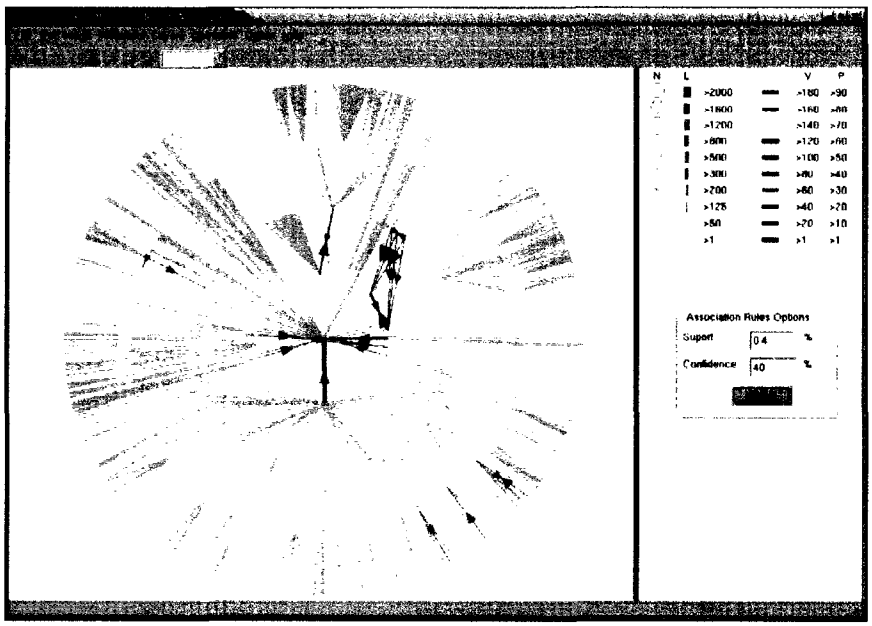

Figure 4.6: Association Rule Layer

new visualization result. This is due to the increase or decrease in the node sizes or edge thicknesses.

As the layout algorithm for bare graph, the visualization in this algorithm is constituted with nodes and edges. A node indicates a webpage and an edge indicates a hyperlink connecting two webpages. Different from other website visualization system [50, 58, 17, 53], our new dynamic layout algorithm considers both the number of nodes and the size of the nodes. The size of the node corresponds to how many times the webpage has been visited in a period of time. The new algorithm allocate more space to nodes which are visited frequently and less space to the nodes which are seldom visited. The thickness of the line represents how many times the user follow the corresponding hyperlink to traverse from one webpage to another.

In our algorithm, we consider both the idea from [45], the node's size and edge's thickness. We first divide 360 degrees of a circle by all the leaf nodes while considering their size. Any two neighboring leaf nodes have the same degree between them. Suppose there are n leaf nodes, and their radiuses are $r_1, r_2, r_3, \ldots, r_n$ respectively, $t_1, t_2, \ldots, t_n$ are the half thickness of edges which connecting them, d_i is the distance from leaf node r_i to center, the degree between any two neighboring leaf nodes is:

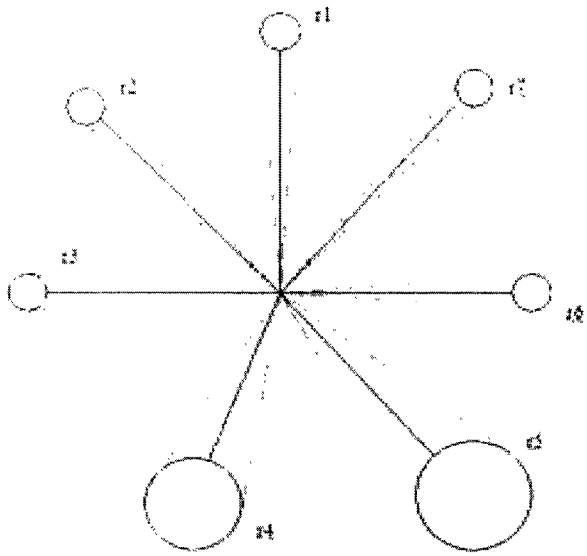

Figure 4.7: Dynamic Layout

$\frac{360-\sum_{i=1}^{n} 2\times atan(\frac{max(r_i,t_i)}{d_i})}{n}$. See Figure 4.7.

By the above formula, we can decide the degree of all leaf nodes considering their size and corresponding edges' thickness. After determining the position of every leaf node, we can calculate the position of their parents by putting their parent to the middle position of all its children, while one level lower than its children.

Table 4.1 is the pseudocode of the dynamic layout algorithm.

Most of the websites these days contain huge number of webpages. With a good layout method, the space can be used efficiently. In our system, all the leaf nodes share the 360 degree and the parent is in between of all it's children.

The website analysts may not always choose all the webpages and links. They can use threshold or operators(we will discuss operators later) to choose only the data that they interest. For example, they may only choose the hyperlinks which have been visited more than 2000 times in one month, or only choose the webpages on which the visitors spend more than 20 seconds. After user changes thresholds, the visualization result changed dynamically, keep using the screen space efficiently. The dynamic layout algorithm and our system framework could help animate users'

WebGraph Dynamic Layout Algorithm
Input: A graph's orthogonal representation
Output: A tree with degree and level information, ready to visualize
Procedure WebGraph Dynamic Layout
1. post-order-travel(disktree), count(leaf nodes number n)
2. &count(degree that occupied by leaf nodes)
3. The degree between any two neighboring leaf nodes =
4. $\frac{360-\sum_{i=1}^{n} 2\times atan(\frac{MAX(r_i,\ t_i)}{d_i})}{n}$
5. post-order-travel(disktree) again, allocate the degree to each leaf node
6. Compute leaf nodes parent's degree: parent's level = child's level - 1,
7. position = middle of all childrens

Table 4.1: Dynamic Layout Algorithm

traversal. The users' traversal affects visualization result by increasing or decreasing nodes' size, edges' thickness and changing node and edge's colour. The dynamic layout algorithm can adopt the visualization result to the changes. Redrawing the graph dynamically with its changes generates an animation showing how the traversal is happening.

Chapter 5

System and Functions

In this chapter, we concentrate more on system's architecture and different functions. These functions are either fundamentals of the system or the extra functionalities which can help navigating the visualization results. Web Graph Algebra is another part of our system, we do a short introduction. For detail of Web Graph Algebra, please refer to [15].

5.1 Architecture

Figure 5.1 shows the system architecture. The whole system works in the following steps:

- By reading the website's linking information, the system builds a "Link Info" file. The "Link Info" file contains the information like there is a link connecting page A to page B.
- By pre-processing, the system generates session and usage data. These data are the key to the visualizing system and will be used as the basis of layers.
- Different layout algorithms generate different results. We provide different algorithms and user can use them to generate different results and compare them.
- With "Link Info", session and usage data and layout algorithms, we can render original disktree representation.

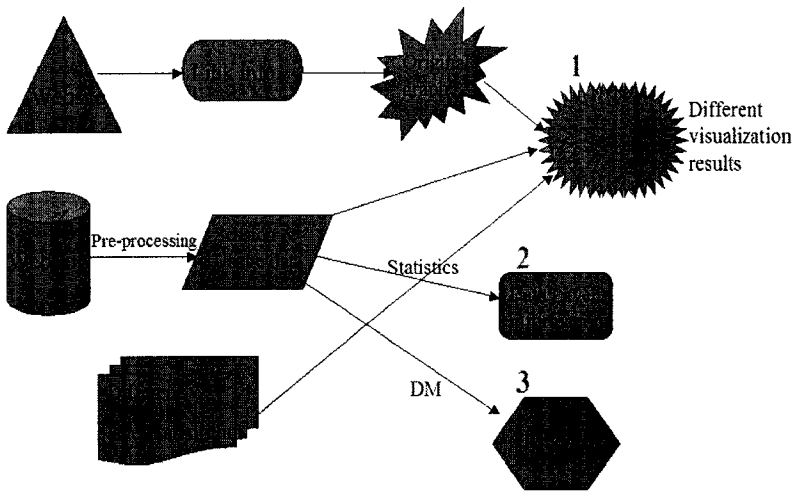

Figure 5.1: System Architecture

- By different simplification methods, like Usage Based method, we get the original visualization results. We call it "Bare Graph" (1).
- From session and usage data, by using statistical methods, we get different data layer information (2).
- By applying data mining technique on session and usage data, we get pattern layer information (3).
- At last, we put either data layer information (2) or patter layer information (3) or both on top of the "Bare Graph" (1).

5.2 Weblog Preprocessing

The system provides many functions which can help user easily and efficiently exploring the visualization result.

The first and the most preliminary task is to process the original weblog and generate the information we need [52]. In section 4.1, we introduce the weblog. Weblog contains much information that we don't need. "Weblog Preprocessing" is the step to pre-process the weblog, discard the information we don't need, leave the

information that are valuable to us, format these useful information in a file for later use.

A typical weblog item contains the following information: IP address or domain name from which the request is, user id and password, request command, status field, transfer volume, referrer, user browser information, user operating system information, cookie information. Some parts of the weblog are irrelevant to our visualizing system, for example: user browser information, user operating system information.

Besides the information we can extract directly from weblog, there are some kinds of information we have to discover by our own. For example, the number of how many times a webpage is visited can be calculated directly from weblog by counting the time of appearances of that webpage. While to answer the question like how many times a hyperlink is visited, we have to make some changes from the weblogs. Two important tasks involved are recognizing the user and recognizing the session. Recognizing user means that we need to recognize which requests are from the same user. If some weblog items has the same user id, or they come from the same IP address, we can assume that they are from the same user. Session means a time period that a user continues visiting a website. Recognizing session means recognizing the weblog items which supposed to be visited continually by the same user in a short period of time. If a user visits a website once, and then after 12 hours, he visits again, these two visiting activity are not in the same session.

After the process of recognizing user and recognizing session, we generate a file which indicate in each session, which page user has visited. This file can be used to answer the question like how many times a hyperlink is visited, or if a user visit this page, which pages else he seems like to visit.

Another task of pre-processing is to get linking information of a website, which will be used to generate website's structure. It can simply be done by filtering the webpage's HTML code.

After the pre-processing, we keep the result file for later use, and call them .pwl (preprocessed web log) file.

5.3 Change Background Colour

The background colour can be chosen by user. The default colour is "White" and the other two colours: "Gray" and "Black" are also available.

5.4 Choose BFS Method or Usage Based Method

After user load the pre-processed web log, the system is ready to render either BFS or usage based visualization result. It's up to user to choose the method he wants and user can compare these two methods' results.

5.5 Information Layers

As mentioned in Chapter Four, we generate four kinds of usage information, two for webpage, two for hyperlink. Each information can be visualized as a layer added to original "Bare Graph". Webpage's two layers include page's visit_number layer (indicated by size of the node) and view_time layer (indicate by colour of the node). Hyperlink's two layers include link's visit_number layer (indicated by colour of the link) and probability_usage layer (indicated by thickness of the link). User can choose which information layer is shown. Since these four layers are visualized by different properties, they can be shown together or one by one.

5.6 Association Rule Pattern Layer

In Chapter Four, we mention the association rule pattern layer visualization. When this function is chosen, user is asked to choose a appropriate "support" and a "confidence", which are two parameters to control association rules (See Figure 5.2). Applying the association rule mining algorithm on the pre-processed file, we can generate different association rule. We call them association rule layer. Then we visualize the association rule layer on top of the bare graph. We use a directed line with a arrow goes from one webpage to another, the thickness of the line indicates the support, and the size of the arrow indicates the confidence.

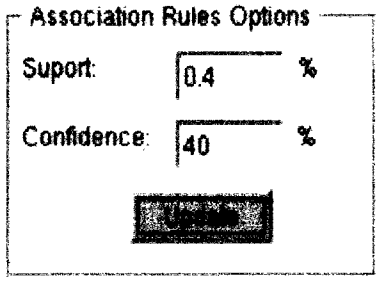

Figure 5.2: Choose "support" and "confidence"

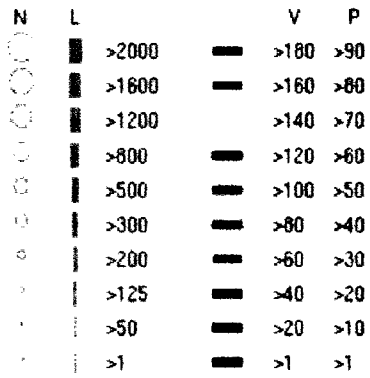

Figure 5.3: Scale

5.7 Scale Indication

On the right hand side of the screen, we put four information layers' scale information. By reading that, user can compare the data on the graph with the scaler.

On left hand side of Figure 5.3 are the scales for page's visit_number layer (with a "N" above) and link's visit_number layer (with a "L" on top). Right hand side are the colour scales for page's view_time layer (with a "V" on top) and link's probability_usage layer (with a "P" on top).

5.8 Rotation

Another useful function is "Rotation". After user renders the web graph and puts different layer on it, user can press the specific "hot key" to rotate the graph, either clockwise or counter-clockwise. When user renders two different web graph using

different weblog file, rotation can be very helpful to compare these results for the user can adjust the position and degree that he prefers.

5.9 Change Start Page and Level Depth

By default, the system visualizes the website beginning from the website's index page, and choose a depth of 4 levels. This may not always work if the user likes to concentrate on other webpage and with other depth level.

We provide user the dialog to choose the webpage to start with and choose the depth of levels. The user also can simply click a node in graph, and the system will re-generate web graph begining from that webpage.

5.10 Zoom In/Out

After user chooses the graph to render and put some layers we have discuss above or choose a large number of levels to be shown, normally the graph becomes messy and difficult to recognize. Zoom In/Out function can help perfectly. User can choose to Zoom In/Out up to 4 times the original graph.

Figure 5.4 is an example of 3 times zoom in. The graph in the main window shows part of the original graph with 3 times zoom in. On the right bottom of the screen, there is a small window containing a small black area indicating which part is showing now without loss of overview. By dragging the small black area in the small window or moving the scroll bar around the small window, user can change the part to be viewed.

5.11 Show Grid

When user chooses large depth of level, the screen is divided by the number of levels. Therefore, the distance between two neighboring levels is shorten significantly and it is difficult for user to identify which level the nodes belong to. "Show Grid" function is used to help user identifying which nodes are in the same level. By our layout algorithm, all the nodes that are in the same level are put around the center with the

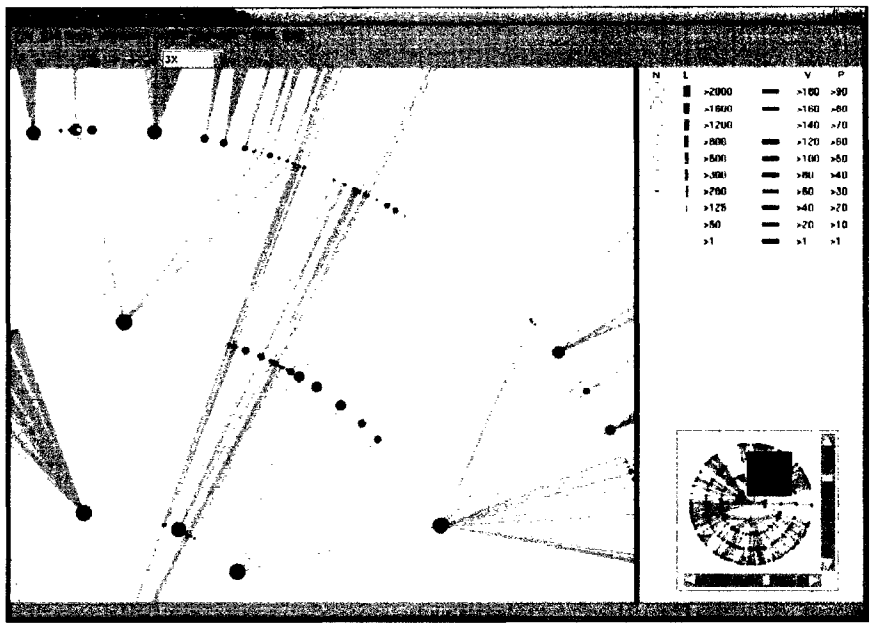

Figure 5.4: Zoom In/Out Function

same distance, so the grid in our system is a circle drawn by dots. User can disable showing the grid at any time as well. Figure 5.5 is an example of showing grid.

5.12 Show Information

In the above section, we discuss the methods that we use to help visualizing. But only the graphical elements are not enough in most of circumstances. When a user finds a interesting node or edge, he wants to know not only the colour, the thickness, the size, but also which webpage this node indicates, which hyperlink this edge indicates and so on. To provide more information to the user, the system provides the "Show Information" function. After user turns on "Show Information" option, when he moves the mouse on top of a node or an edge, the corresponding information will appear on the status bar, which locatess at the bottom of the screen.

When the user moves mouse to a node, information of level of node, the URL corresponding this node, visiting number and view time are shown on the bottom. When user moves mouse to an edge, information of level of this edge belongs to, which two webpages' URLs this hyperlink connects, visiting number and probability

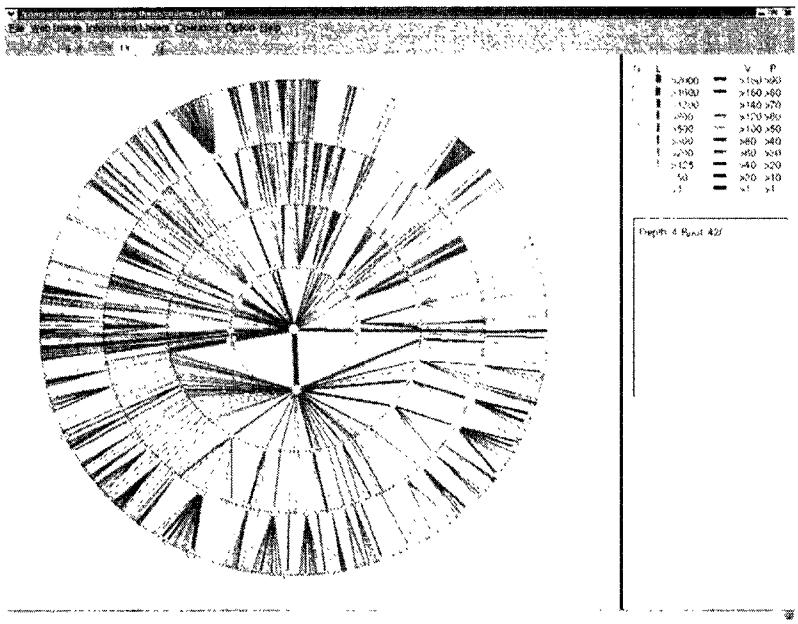

Figure 5.5: Show Grid Function

of visiting are shown. Figure 5.6 and 5.7 are examples of showing information of node and edge separately.

5.13 Web Graph Algebra

WEBKVDS system contains two parts: FootPath, which we discuss above, and Web Graph Algebra, which is introduced to operate on web graph objects to highlight interesting characteristics of web data, and consequently helps discovering new patterns and interesting hidden facts.

Variables in Web Graph Algebra are web graphs. A combination of unary and binary operators can form expressions and equations. Unary operators strip, add or subtract from given information layers, or extract sub-parts of the underlying web structure. Binary operators combine two web graphs by evaluating arithmetic expressions on the common information layers as well as the nodes and edges constituting the web images.

We take one operator: "ADD" as an example:

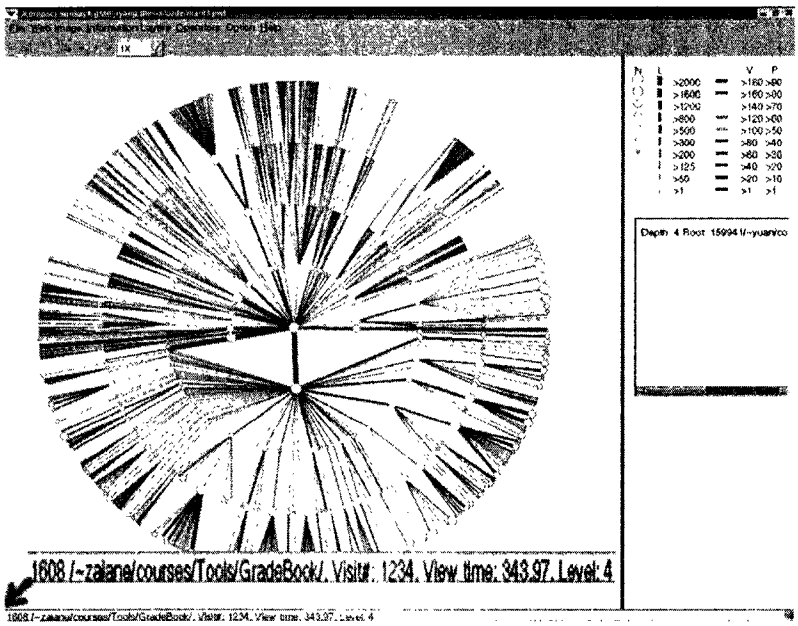

Figure 5.6: Show Node's Information

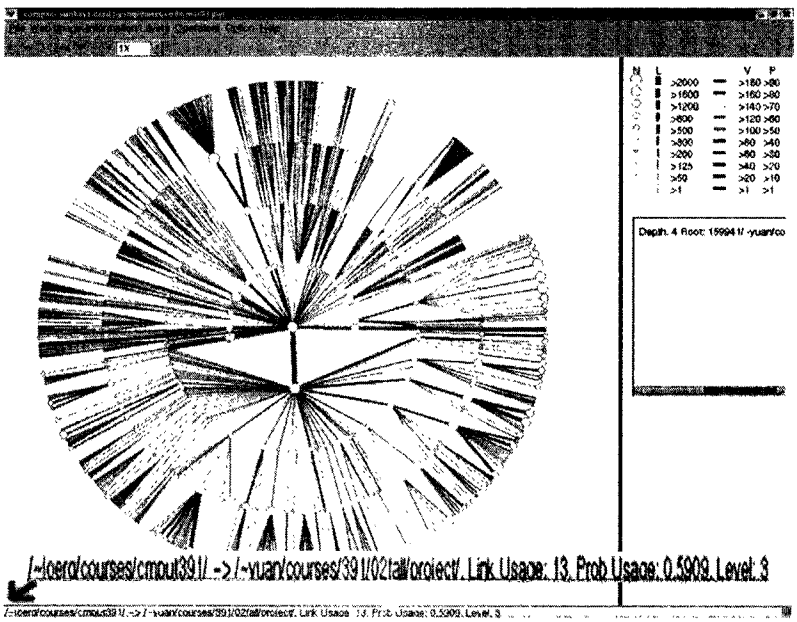

Figure 5.7: Show Link's Information

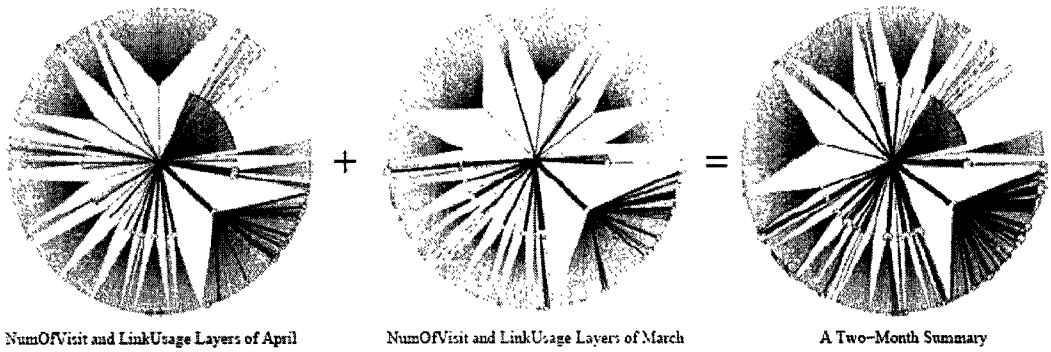

Figure 5.8: Binary Operator "ADD"

The binary operator ADD selects the objects that exist in both graph G1 and graph G2, and transposes them into graph G with the sum of their respective content from the available layers in both graphs. The sums are done at each individual information layer.

Figure 5.8 shows the ADD operation to web graphs of two months with 2 layers, NumOfVisit and LinkUsage layers. We get two months' data by adding two separate month's data.

For detail about algebra and operators, please refer [15].

Chapter 6

Conclution

In this dissertation, FootPath, the visualization part of WEBKVDS system has been presented, visualization design issues have been discussed and experimentation results have been made and compared.

WEBKVDS system is built in an attempt to help website administrators and information publishers understand visitors' visiting patterns and consequently improve website's structure. FootPath visualizes the website structure and topology, and visualizes both apparent and implicit useful data. The Web Graph Algebra manipulates the web graphs for ad-hoc interactive web mining. FootPath is the basis for Web Graph Algebra as it allows the rendering of variables and results of the web algebra expressions.

We propose a new "Usage Based" method to visualize website structure to disktree representation. We present the idea of layering data and patterns in distinct layers on top of a disktree representation of website's structure, allowing the display of information in context which is more suited for the interpretation of discovered patterns. Dynamic layout algorithm can generate a clearer and more friendly result, especially in our scenario: visualize multi-information together. Association rule mining technique is also integrated to our system to help website administrators discover implicit knowledge.

Appendix A: Data Structure

The Major data structure used in the WEBKVDS is a "Orthogonal List". "Adjacency Lists", "Adjacency Matrix", and "Orthogonal Lists" are three widely used graph's data structure.

In "Adjacency Lists" structure, the edges are stored as lists of connections between nodes. In "Adjacency Matrix" structure, the presence of edges between nodes is indicated by an entry in a matrix. In "Orthogonal Lists", each node is represented by a node structure , each edge is represented by a edge structure (see Figure 6.1).

In node's structure, "data" is the node's ID or name. "first in" is the link linking to a structure of edge, which links to this node. "first out" is the link linking to a structure of edge, which links out from this node. Like node's structure, in edge's structure, "Tailvex" and "Headvex" are the ID or name of the tail node and head node of a edge. "hlink" and "tlink" link to the edge which has the same "head" or "tail". Figure 6.3 is a example of graph's "Orthogonal Lists" representation.

In our system, we use graph's "Orthogonal Lists" representation as the data structure for identifying all forwarding and backward links in "Orthogonal Lists" is efficient.

When we generate information layers, our data structure should be expanded to hold other information. We can easily add a field to node structure or edge structure for each additional property. In our system, "bool visit" is to indicate whether this node or edge has been visited, "long layer_Info" contains edge's link_usage or node's visit number, "float prob_Usage" contains edge's prob_Usage or node's view_time, "bool display_Info" indicates whether display it or not, "float degree", "int level".(See Figure 6.2)

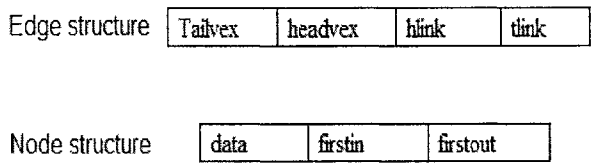

Figure 6.1: Node and Edge structure in Orthogonal Lists

tailvex	headvex	hlink	tlink	visit	layer_info	prob_usage	display_info	degree	level

Figure 6.2: Node and Edge structure in WEBKVDS

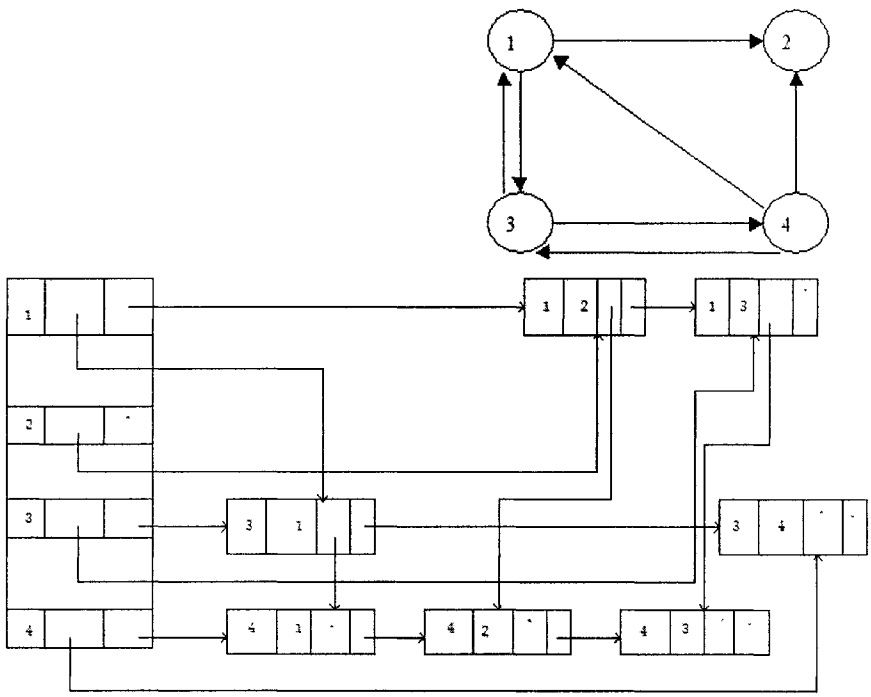

Figure 6.3: Orthogonal Lists

Bibliography

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. pages 94–105, 1998.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[4] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–396, 1999.

[5] M.-L. Antonie. *Categorizing Digital Documents by Associating Content Features.* Master Thesis in University of Alberta, 2002.

[6] M. Balabanovic. An adaptive web page recommendation service. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 378–385, New York, 5–8, 1997. ACM Press.

[7] B. Berendt. Understanding web usage at different levels of abstraction: coarsening and visualising sequences. In *Workshop.*

[8] B. Berendt. Web usage mining, site semantics, and the support of navigation. In *R. Kohavi, B. Masand, M. Spiliopoulou, and J. Srivastava, editors, Working Notes of the Workshop Web Minig for E-Commerce - Challenges and Opportunities. 6th ACM SIGKDD International Conference of Knowledge Discovery and Data Mining*, Boston, MA, August 2000.

[9] B. Berendt. Detail and context in web usage mining: Coarsening and visualizing sequences. In *R. Kohavi, B. Masand, M. Spiliopoulou, and J. Srivastava, editors, WEBKDD 2001- Mining Web Log Data Across All Customer Touch Points*, pages 1–24, Springer Verlag, 2002.

[10] A. Berson, S. Smith, and K. Thearling. *Building Data Mining Applications for CRM.* McGraw-Hill, 1999.

[11] F. Bodon. A fast apriori implementation. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[12] R. Burke. Hybrid recommender systems: Survey and experiments. In *User Modeling and User-Adapted Interaction*, 2002.

[13] I. V. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *Knowledge Discovery and Data Mining*, pages 280–284, 2000.

[14] J. Carriere and R. Kazman. Interacting with huge hierarchies: Beyond cone trees. In *Proc. of Information Visualization*, 1995.

[15] J. Chen. *Web Graph Algebra for Interactive Ad-hoc Mining.* Master Thesis in University of Alberta, 2004.

[16] J. Chen, L. Sun, O. R. Zaïane, and R. Goebel. Visualizing and discovering web navigational patterns. In *Seventh ACM SIGMOD International Workshop on the Web and Databases (WebDB 2004)*, June 2004.

[17] E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S. K. Card. Visualizing the evolution of web ecologies. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'98*, 1998.

[18] P. Domel. Webmap - a graphical hypertext navigation tool. In *Advance Proceedings of Mosaic and the Web: the Second International WWW Conference '94*, pages 785–798, 1994.

[19] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons Inc, 1973.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, page 226C231, 1996.

[21] U. Fayyad, G. G. Grinstein, and A. Wierse. *Information Visualization in Data Mining and Knowledge Discovery.* Morgan Kaufmann, 2001.

[22] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Intelligent User Interfaces*, pages 106–112, 2000.

[23] N. Gershon, S. G.Eick, and S. Card. Information visualization. In *ACM SIGGRAPH Course notes 8*, August 1996.

[24] R. L. Graham and P. Hell. *On the History of the Minimum Spanning Tree Problem.* History Comput, 1985.

[25] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 73–84. ACM Press, 1998.

[26] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. R. Zaïane. DBMiner: A system for mining knowledge in large relational databases. In *Proc. 1996 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'96)*, pages 250–255, Portland, Oregon, 1996.

[27] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, 2000.

[28] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM-SIGMOD*, 2000.

[29] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, 1994.

[30] Hendley, N.S.Drew, A.M.Wood, and R.Beale. Narcissus: Visualizing information. In *Proceedings of IEEE Information Visualization Symposium, Los Alamitos*, pages 90–97, Oct 1995.

[31] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[32] http://www.avs.com/solutions/oil_gas/dong.html.

[33] http://www.avs.com/white papers/infoviz.html.

[34] http://www.ghcc.msfc.nasa.gov/globeflt14.html.

[35] http://www.sfgate.com/maps/.

[36] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[37] G. Karyapis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. In *IEEE Computer, Special Issue on Data Analysis and Mining*, 1999.

[38] R. Kohavi. The power of decision tables. In N. Lavrac and S. Wrobel, editors, *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 914, pages 174–189, Berlin, Heidelberg, New York, 1995. Springer Verlag.

[39] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In *Proceedings of the Thirteenth International Conference on Data Engineering*, pages 220–231, 1997.

[40] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[41] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability*, pages 281–297, Berlin, Heidelberg, New York, 1967. Berkeley, CA: University of California Press.

[42] B. Matt. gcluto – an interactive clustering, visualization, and analysis system.

[43] R. Michalski. *Machine Learning : An Artificial Intelligence Approach.* Morgan Kaufmann, 1983.

[44] N. Minar and J. Donath. Visualizing the crowds at a web site. In *Proceedings of CHI99.*, 1999.

[45] S. Mukherijea and Y. Hara. Focus+context views of world-wide web nodes. In *In Processding of the eighth ACM conference on Hypertext*, pages 187–196, 1997.

[46] S. Mukherjea and J. Foley. Visualizing the world-wide web with the navigational view builder. *Computer Networks and ISDN Systems*, 27(6):1075–1087, 1995.

[47] T. Munzner. Drawing large graphs with h3viewer and site manager. In *In Proceedings of Graph Drawing*, pages 384–393, Montreal, Canada, 1998.

[48] T. Munzner. *Interactive Visualization of Large Graphs and Networks.* Ph.D Thesis, Stanford University, 2000.

[49] T. Munzner and P. Burchard. Visualizing the structure of the world wide web web in 3rd hyperbolic space. In *ACM VRML Conf.*, page 33C38, 1995.

[50] Y. Niu, T. Zheng, J. Chen, and R. Goebel. Webkiv: Visualizing structure and navigation for web mining applications. In *Proceedings IEEE WIC, Halifax, Canada*, Oct 13-17 2003.

[51] R. M. Pickett and G. G. Grinstein. Iconographics displays for visualizing multidimensional data. In *In Proceedings IEEE Conference on Systems, Man, and Cybernetics*, page 514C519, May 1988.

[52] J. Pitkow and K. Bharat. Webviz: A tool for world wide web access log analysis. In *Proceedings of First International World-Wide Web Conference*, pages 271–277, 1994.

[53] G. Robertson, J. Mackinlays, and S. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *In Proc. of ACM SIGCHI conference on Human Factors in Computing Systems*, pages 189–194, 1991.

[54] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167.

[55] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[56] A. Wexelblat. History-based tools for navigation. In *HICSS*, 1999.

[57] A. Wexelblat and P. Maes. Footprints: History-rich tools for information foraging. In *CHI'99*, pages 270–277, May 1999.

[58] G. J. Wills. Nicheworks - interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics 8*, pages 190–212, 1999.

[59] O. R. Zaïane, A. Foss, C.-H. Lee, and W. Wang. On data clustering analysis: Scalability, constraints and validation. In *Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*. ACM Press, 2002.

[60] E. Z.Ayers and J. T.Stasko. Using graphic history in browsing the world wide web. In *Fourth International World Wide Web Conference*, pages 11–14, Boston, December 1995.

[61] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.

[62] T. Zheng, Y. Niu, and R. Goebel. Webframe: In pursuit of computationally and cognitively efficient web mining. In *PAKDD*, pages 264–275, 2002.