AN EFFICIENT WEB MINING ALGORITHM FOR SESSION PATH PATTERNS

*Don-Lin Yang, Shen-Hong Yang, and Ming-Chuan Hong
Department of Information Engineering
Feng Chia University, Taichung, Taiwan 407
*E-mail:dlyang@fcu.edu.tw

ABTRACT

Data mining in the World Wide Web has many practical applications. Using efficient web mining algorithms to discover WWW browsing path patterns can benefit web users as well as administrators. For example, effective information retrieval and better understanding of the user behaviors can help attract more visitors and improve service of E-commerce. However, different definitions of a browsing path result in mining different knowledge. We define a user session as a complete browsing path from the entry page to the last page a user visited. An efficient mining algorithm was proposed to find patterns of frequent session path. We introduce a new measurement of browser's interest in a path as an inclination value. In this paper, we show that our implementation takes less memory space since the algorithm produces fewer candidate transactions than that of others. Scanning the sorted database only once, we can quickly find all large session paths regardless of the support value.

Key words: *Web mining, session path, support, confidence, inclination.*

1. INTRODUCTION

Due to the increasing number of databases and their growing sizes today, data mining technology becomes more important in acquiring knowledge for decision making. One mining application on the World Wide Web is referred to as *web mining* [9][11-12][14-15][18-19]. Web mining has two kinds: w*eb content mining* and w*eb usage mining* [10]. Web usage mining identifies the path patterns when users browse through pages on the web. The information of browsing path patterns helps put advertisements on the web, improve web traffic, analyze users' behavior, and design better web sites [8][16].

Several researchers have studied web usage mining from user sessions [1][2][6]. A user session is a set of pages being referenced by a user during a single visit to the site. User sessions are extracted from web server logs [17]. Path patterns will be found from user sessions. The *transactions* are produced from user sessions as processing units to mine path patterns. Different definitions of transactions have different meanings and results. [1] uses maximum forward reference, and [2] uses reference length of time to separate a user session into transactions. Other methods, such as [6], regard a user session as a transaction directly.

These studies have two similar characteristic: 1) their purpose is to mine the frequent path patterns; 2) algorithms were improved from those mining association rules [3][4][7]. Although the purposes are alike, different methods and algorithms can produce different results from the same web server log. It is hard to point out which one is better. Association rules have two important vocabulary words: *Support* and *Confidence*. Support is an argument that decides whether the candidate is frequent or not. The frequent path patterns are identified by their support values. Confidence is an argument that describes the believable degree of association rules. Those studies only use support arguments but lack confidence. We infer that the cause is because the traditional confidence definition is not suitable for the web structure. The reason will be illustrated in Section 3. The more the confidence is concerned, the more information of browser behaviors is gained.

Web structure will influence the result of path pattern. Because of this special structure, some pages are accessed more frequently. Generally speaking, the home page is accessed most frequently. The pages that are directly linked by the home page also are accessed more frequently. Intuitively, the frequent path patterns also are influenced by web structure. It may confuse us when we analyze browsers' behaviors. We can't recognize if the frequent path pattern is expressed as an interesting browser act or is just caused by web structure.

To solve foregoing problems, we use a new method to mine path patterns. Our method provides both support and confidence arguments. In most algorithms of mining association rules, the support value is used first to find large itemsets. The large itemsets that exceed the confidence value become rules. We swap the order of using those two arguments. First, the confidence argument is used for filtering and then the support argument. We believe this can increase the rules about browsers' behavior and reduce the influence of web structure. To be suitable for web structure, our confidence definition is different from traditional one. In order to be able to distinguish the difference, we use a new definition called *inclination*. Our method emphases the inclination in addition to the support. Since inclination is stressed, the path pattern influenced by web structure will be reduced. We will illustrate it in Section 3. We propose a new algorithm SP to mine path patterns based on the idea of inclination.

The rest of the paper is organized as follows. Data preparation is described in Section 2. The inclination definition is presented in Section 3. Our SP algorithm and its implementation are shown in Section 4. Section 5 discusses the characteristics of path patterns. Section 6 is the conclusion of the paper.

2. DATA PREPARATION

A web log records information when users access web pages. The recorded information includes the IP address, the access time, and the pages accessed. The data of web server logs are in rows. A *user session* expresses a path of how a user browses the web site. Each path may have a different beginning page. Although every web site has a homepage, some users may start browsing from bookmarks or hyperlinks. To mine path patterns, user sessions must be extracted from web server logs. [2] discusses the methods of extracting a user session from the web log.

Data can tie in different algorithms, so some special methods are used in the data preparation phase. A user session may be separated into several *transactions*. For example, a user session of ABCDCEF (each letter of the alphabet represents a web page) can be decomposed into two transactions ABCD and ABCEF based on the "*maximum forward reference*" [1]. Another method of "reference length" [2] separates the session according to the length of time users spend on the pages. A user session can also be regarded as a transaction directly [21]. Each different method of producing a transaction has different algorithms to mine path patterns. Our method treats a user session as a transaction. Unlike other methods, user sessions are sorted to increase the efficiency of our algorithm.

3. INCLINATION DEFINITION

3.1 Web Hyperlink Structure

In a web environment, the probability of each page accessed is not fairly distributed. Page access time relates to its distance from the starting pages (like homepages) of browsers. Browsing transactions are different from traditional transactions. In traditional business transactions, customers may choose any goods arbitrarily. Each choice may influence, but cannot limit, the kind of good to be chosen next. Web browsers start to view a web site from its homepage unless they input the page address directly. To give a simple example, we use a tree structure for the web pages. The pages closer to the top have better access time generally. This problem of imbalance needs to be improved in order to find more useful information.

The confidence cannot be adopted on a web hyperlink structure. Using traditional methods to calculate confidence is not suitable for path patterns. For example, there is a frequent path pattern A✍B. The page A that has less hyperlinks to other pages will have a higher confidence for A✍B. If A is a homepage and has only one hyperlink to page B, the confidence for A✍B should be higher. The reason is that after visiting page A, browsers have no choice to link to other pages except page B. This kind of confidence may throw off the user behavior analysis.

3.2 Our Path Patterns

To overcome the problem of confidence value related to the number of hyperlinks for a path, we propose a different path pattern that Figure 1 illustrates. Figure 1 shows an aggregating tree [20]. It is not the hyperlink structure of a web site. The aggregating tree is constructed by using user sessions. The page of level one is the starting page for user sessions. The page M, for example, was accessed after page C 130 times in the last level. This indicates the number of user sessions containing A✍C✍D✍C✍M.

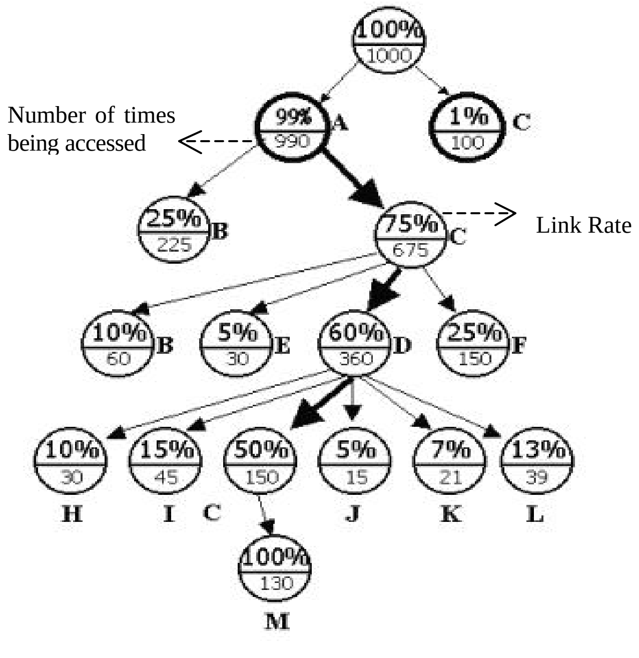


Figure 1. A simple aggregating tree

After browsing page A, one has pages B and C to choose from. Comparing the number of times pages B and C are accessed will show which page is more interesting to users after browsing page A. The example in Figure 1 shows that page C was accessed 675 times which is more than page B's 225 times. This information is useful for analyzing user behaviors. We propose a new term called *inclination* to express the information by using a measurable value.

A page is denoted by S_i, and S_1 is the starting page. Let *L* be a set of user sessions that have the same subset $S_1S_2S_3\ldots S_n$. The set L has *m* kinds of $S_1S_2S_3\ldots S_nS_{n+1}$ (i.e., *m* kinds of S_{n+1}), which are denoted as $S_1S_2\ldots S_n(S_{n+1})_1$, $S_1S_2\ldots S_n(S_{n+1})_2,\ldots$, $S_1S_2\ldots S_n\ (S_{n+1})_m$. For each $S_1S_2\ldots S_n(S_{n+1})_k$ in the set L, the *inclination* of $S_1S_2S_3\ldots S_n \Rightarrow (S_{n+1})_k$ is:

$$N_k / ((\sum_{i=1}^{m} N_i) / \mathrm{m})$$

N_i is the number of times $S_1S_2\ldots S_n(S_{n+1})_i$ is accessed.

We explain the formula with an example. In Figure 1 the inclination of A=>C is:

675/((225+675)/2)=1.5

If users have no special inclination between A=>B and A=>C, the frequencies of A=>B and A=>C should be the same: (225+675)/2=450. The inclination value of 1 means that there is no special inclination between A=>B and A=>C. Here the inclination value of A=>C is 1.5. A value larger than 1 shows that A=>C may be more interesting to users than A=>B.

Our goal is to find the path patterns in which each hyperlink exceeds the specified inclination and support values. As a confidence value, the inclination value can be set arbitrarily. If both A=>C and AC=>D exceed the inclination and support values, they can merge to A=>C=>D. We are interested more in the inclination than the support. The support value confirms that the frequency of the path pattern is large enough. A long path pattern with a high inclination may express an interesting browser behavior.

4. USE SP ALGORITHM TO MINE PATH PATTERNS

4.1 Sorting User Sessions

We use aggregating trees to calculate the inclination value and mine path patterns. Using the sorted user sessions in our algorithm we need to scan the database only once. The sorting is based on the ordering sequence of the *k*-th page in each user session, where k starts from 1. For example, assume we have three user sessions: ADE, BCF, and ABC. Alphabetically, the result of sorting is: (1) ABC (2) ADE (3) BCF. The sorted user sessions can reduce the time to scan the database. Sorting is completed when user sessions are extracted from the web log file at the same time.

4.2 SP Algorithm

4.2.1 Comparing Sorted User Sessions

The SP algorithm reads sorted user sessions sequentially. After the first user session is retrieved, it is stored in a candidate string. The string is compared to the rest of the user sessions one by one. After each comparison, the counter of the candidate string is used to record the support value, as shown in Figure 2. Initially, the counters are set to one. Each counter is incremented after comparison until a match fails. The process repeats until all user sessions are read. We define a user session as $\{S_1,S_2,S_3,\ldots,S_n\}$, and a candidate string as $\{C_1,C_2,C_3\ldots,C_m\}$. Each S_i and C_j represents a page (i=1,2,..,n; j=1,2,..,m). Every C_j has an associated counter to record its occurrences. If the counter of C_3 is 5, it means that $\{C_1C_2C_3\}$ occurred 5 times.

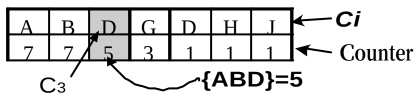


Figure 2. Data structure of a candidate string.

At each comparison step between a user session and a candidate string, S_i is compared with C_j starting from S_1 and C_1. Values i and j are equal at each comparison. Each comparison observes three rules:

1. If C_j equals S_i, the counter of C_j is increased by one.
2. If C_j is not equal to S_i, or S_i is non-existent, then C_j is moved from the string to a stack (Figure 7). S_i becomes C_j if S_i exists. When C_j is not equal to S_i, the other C_j that does not compare is regarded as unequal (i.e., AB$\neq$ AC => ABD$\neq$ ACD). The C_j that has the largest j value is removed first. This simple action does the match process for the stack.
3. If S_i exists but C_j does not, S_i becomes C_j (counter =1).

The removed C_i means that the counter of $C_1C_2..C_i$ is known. For example, the counter of ABC is 11. The next comparing user session is ABDE. This means that the remaining user sessions that are not accessed yet have ABC($C_1C_2C_3$). The reason is because the user session is sorted. The removed C_i (C) will be pushed into the stack. Path patterns that exceed inclination and support values will be found from the stack. When all user sessions are read, all large sessions are found.

Figure 3 shows an example of string comparison. The algorithm for comparing strings is shown in Figure 4.

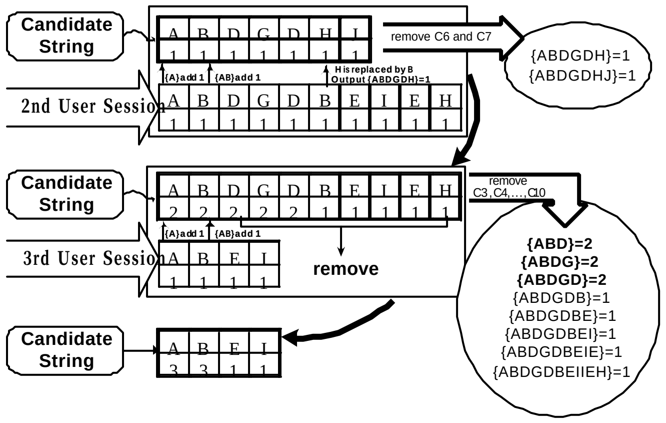


Figure 3. An example of string comparison.

```
candidate = the first user session;
   while(1)
      Read a new user session;
      if   (new user session is NULL) then
          remove all $C_j$ and exit while loop;
    for each $C_j$ and $S_i$
     compare a candidate string with the user session
     /* the rules of comparison are in Section 4.2.1 */
     if ($C_i$ is removed)
         call function Stack ($C_i$, order of $C_i$, counter of $C_i$)
```

Figure 4. Comparing strings

4.2.2 *Finding Path Patterns from the Stack*

In Section 3.2 we described how to calculate inclination. A problem must be overcome. As shown in Figure 1, to calculate the inclination of AC=>B of the aggregating tree, the children node of AC must be known. They are B, E, D, and F nodes. The counter values of ACB, ACE, ACD, and ACF also must be known. A simple solution is to scan the database once to get this information for calculating each inclination. Database scanning requires lots of I/O time. We use a stack to record necessary information and to avoid scanning the database too many times. In fact, we scan the database only once.

The stack records all brother nodes, B, E, D, and F. Before confirming how many brother nodes there are for AC, the brother nodes that are recorded will not pop out. When AC is also moved to a stack, all brother nodes are known. Since the user sessions are sorted, it is impossible to read a user session that is AC* after removing AC. If AC is moved to the stack, the inclinations of ACB, ACE, ACD, and ACF can be calculated. The stack also records each counter value. The path patterns will be filtered out if they pass the inclination and support values that are set by users.

After a successful calculation, B, E, D, and F pop out and C (AC) is pushed in. Then the process continues to read the next user session and compares strings.

We use an example to illustrate our algorithm for the stack. In Section 4.2.1 we move C_i to the stack. The data being pushed into the stack for C_i is: 1) the name of C_i; 2) the order of C_i; 3) the counter of C_i. For example, ABD was removed (C_3 is D). The order of D is three, and the counter of D is 1. Those data will be pushed into the stack. The data structure is shown in Figure 5.

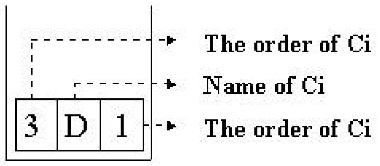


Figure 5. The stack.

We use a simple example to illustrate how path patterns can be found from the stack. A set of sorted user sessions was listed in Figure 6. The inclination value must exceed one, and the support value must exceed two (counter value).

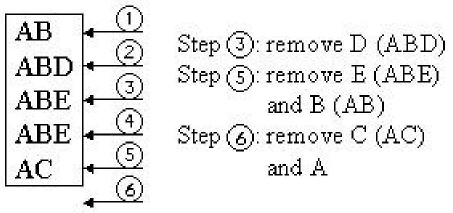

AB
ABD
ABE
ABE
AC
1
2
3
4
5
6
Step 3: remove D (ABD)
Step 5: remove E (ABE) and B (AB)
Step 6: remove C (AC) and A

Figure 6. Example user sessions

First, we read the sorted user sessions sequentially and compare strings. In Step 3 “D” is removed from the string and pushed into the stack (Figure 7). In Step 5 both “B” and “E” are moved to the stack. Since the order of “E” is larger than “B”, “E” is placed in the stack before “B”. This is a very important key for the stack operation. The elements on top of the stack must keep a status that the order of each element cannot be smaller than all the elements which are in the stack. In Figure 5, Step 5, “B” is moved to the stack. The order of “B” is two, and the element of “D” and “E” is three. It means the parent of “D” and “E” is “B”, and “B” has been removed. The children node for AB is known: “D” and “E”. The elements that are ordered smaller than “B” pop out. The elements, which have popped out in the same order, are brother nodes. The inclination value of AB✍D and AB✍E can be calculated: AB✍D is 0.67 (support is two) and AB✍E is 1.33(support is one). Each path is evaluated to become a rule if it passes the support and inclination values. AB✍E is the rule and AB✍D is forsaken. After popping, “B” is pushed back in the stack. In Step 6 all user sessions are read, and all the C_i’s are moved to the stack: C and A. When “A” is pushed into the stack, “B” and “C” pop out to be calculated: A✍B is 1.6, and A✍C is 0.4. Finally, we get that the inclination value of “A” is one.

The algorithm for processing elements on the stack is described in Figure 8.

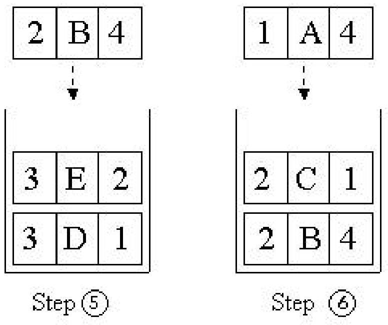


Figure 7. Stack operations

```
Stack ($C_i$, order of $C_i$, counter of $C_i$)
If the order of $C_i$ >= the order of element on the top
    Push $C_i$ into the stack
    top element-> $C_i$
else
  pop out the elements whose order is larger than $C_i$’s
  calculate inclination and output rules
```

Figure 8. The stack process

4.3 Experiments and Discussion

The experiments were performed on an Ultra Enterprise 10000 running the Solaris. The clock speed of the processor is 250Mhz and the memory size is 4G bytes. A simulated Web-page tree based on a real Web site structure was created [1]. Each node represents a page. When a user stays at any one node, the probabilities that the user moves to a child node, backtracks to its parent node, or jumps to some other node in the tree are given just like in a real situation. A large number of user sessions were produced from the tree.

Figure 9 shows the relationship between data size and execution time. The variable S is the average size of the sessions produced from a tree whose average height is ten. We get a linear relationship between the number of user sessions and execution time. Large sessions can show how most users browse among Web pages. To illustrate, we use a large session {ABACDCE} produced according to the tree. The large session shows that after visiting page B, users backtrack to page A and then visit page C. Adding a link from B to C maybe a good way to improve the site structure.

Large sessions can also provide useful information at Internet-based learning sites. Using large sessions to analyze the association between students’ learning behaviors and test scores can really help both teachers and students. If more of user profiles are available, such as sex, age, and education, one can track different levels of users with different large sessions. This information is very valuable to E-commerce services as well.

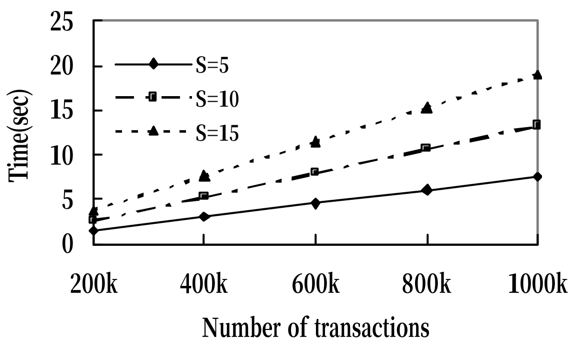
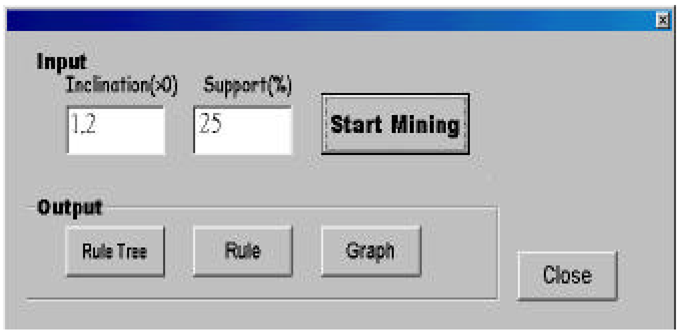


Figure 9. Execution time

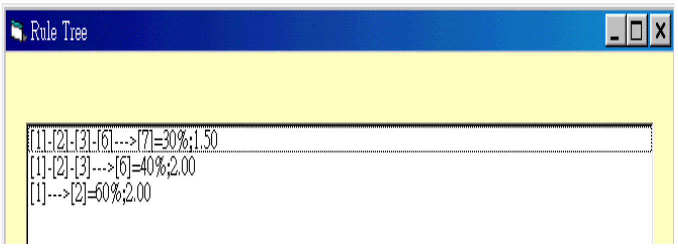
5. VISUALIZATION OF PATH PATTERNS

We designed a prototype that can be used to visualize mining results. Figure 10 shows the prototype. The prototype includes three components: Rule, Rule Tree and Rule Graph. The Rule shows path patterns. Each rule expresses a path with its hyperlinks between nodes, passing inclination and support values. The Rule Tree describes the rule’s position in the pruned aggregating tree. Each sub path where the target node links to leaf nodes of the aggregating tree having no rule is pruned. From the integral rule of distributing over the aggregating tree, we can analyze each rule for browsers more accurately. In order to distinguish the important rules more easily, we adopt the Rule Graph to show the rules.



Inclination(>0): 1.2
Support(%): 25

Figure 10. A prototype of mining path patterns

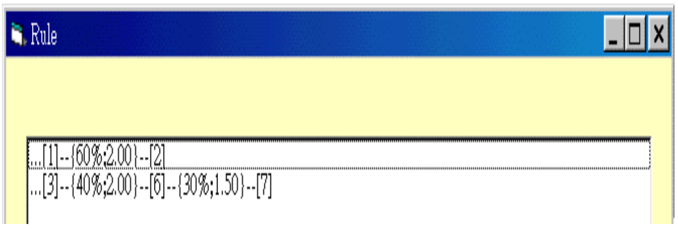


Rule Tree

[1]-[2]-[3]-[6]--->[7]=30%;1.50
[1]-[2]-[3]--->[6]=40%;2.00
[1]--->[2]=60%;2.00

Figure 11. Rule Tree

Figure 11 expresses three rule trees. Each rule tree indicates a link that has larger inclination and support values than we specify. The path of the link is also shown. For example, [1]-[2]-[3]-[6]--->[7]=30%;1.50 means that the [6]--->[7] has 1.50 inclination value and 30% support value. After the browsing path [1]-[2]-[3]-[6], it will show this result.



Rule

...[1]--(60%;2.00)--[2]
...[3]--(40%;2.00)--[6]--(30%;1.50)--[7]

Figure 12. Rules

Figure 12 shows the Rule interface. A rule expresses a path pattern. Each link of a rule has larger inclination and support values than we specify.

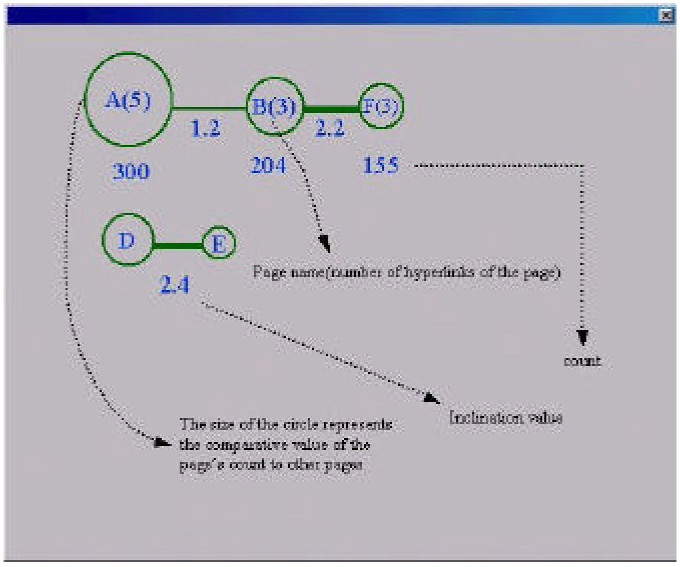

A(5)
1.2
B(3)
2.2
F(3)
300
204
155
D
E
2.4
Page name(number of hyperlinks of the page)
count
Inclination value
The size of the circle represents the comparative value of the page's count to other pages

Figure 13. Graph

Figure 13 shows two rules using the Graph. We use the size of the circle and the width of the line to express different inclination and support values. The thicker line means a larger inclination value. The visualization interface helps us find interesting information from the mined rules [13]

6. CONCLUSION

In this paper, we propose an efficient algorithm for mining session path patterns. We use inclination values with support values to find frequent path patterns. The other methods of mining path patterns, which only concern support values, could lead to controversial results because of web structure. Their methods of mining paths only account for how the web site is browsed by users. They only provide information about what users do. Our method is further concerned about how users select their ways to browse the web site. We can analyze what users want to do. In order to find path patterns efficiently, we propose a new algorithm SP. It scans databases only once and uses a small amount of memory.

Our future works are to continue with further experiments using real user sessions. We will also try to improve the performance and capabilities of SP. We will use SP to mine students' study behaviors on Internet-based learning. Large sessions will be used to find what subjects students like to learn first and what study behaviors occur more frequently. These information can help improve the effectiveness of Internet-based learning.

REFERENCE

[1] M.S. Chen, J.S. Park, and P.S. Yu, "Efficient Data Mining for Path Traversal Patterns," IEEE Tran. on Knowledge and Data Engineering, Vol. 10, No. 2, pp. 209-221, March/April 1998.

[2] B. Mobasher, and J. Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns," Knowledge and Information Systems V1(1), February 1999.

[3] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules," In Proc. of the 20th VLDB Conference, pages 487-499, Santiago, Chile, 1994.

[4] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Database," Proc. ACM SIGMOD, pp. 207-216, May 1993.

[5] J.S. Park, M.S. Chen, and P. S. Yu, "Using A Hash-Based Method with Transaction Trimming for Mining Association Rules," IEEE Trans. On Knowledge and Data Eng., Vol. 9, no. 5, pp. 813-825, Sept./Oct. 1997.

[6] M.S. Chen, J.S. Park, and P. S. Yu, "Data Mining for Path Traversal Patterns in a Web Environment," Proc. of 16th Int'l Conf. on Distributed Computing Systems, 1996.

[7] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," Proc. 21th Int'l Conf. Very large Data Bases, pp. 420-431,

Sept. 1995.

[8] Ellen Spertus, "ParaSite: Mining Structural Information on the Web," 6th Int' l World Wide Web Conference. http://atlanta.cs.nchu.edu.tw/ www/PAPER206.html, Apr., 1997.

[9] N.D. Sweany, T.F. McManus, D.C. Williams, and K.D. Tothero, "The Use of Cognitive and Metacognitive Strategies in a Hypermedia Environment," 1996.

[10] R. Cooley, B. Mobasher, and J. srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web," Proc. of Int' l Conf. on Tools with Artificial Intelligence, pp. 558-567, Newport Beach, CA, 1997.

[11] B. Mobasher, N. Jain, E. Han, and J. Srivastava, "Web mining: Pattern discovery from world wide web transactions," Technical Report TR 96-050, University of Minnesota, Dept. of Computer Science, Minneapolis, 1996.

[12] T.W. Yan, Matthew Jacobsen, G.M. Hector, and Umeshwar Dayal, "From User Access Patterns to Dynamic Hypertext Linking," In Fifth World-Wide Web Conference, 1996.

[13] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, A. I. Verkamo, "Finding Interesting Rules from Large Sets of Discovered Association Rules," The Third International Conference on Information and Knowledge Management, Ed. Nabil R. Adam, Bharat K. Bhargava and Yelena Yesha,, Gaithersburg, Maryland. ACM Press pp. 404-407. Nov.,, 1994.

[14] L. D. Catledge and J.E. Pitkow, "Characterizing Browsing Strategies in the World-Wide Web," Proc. Third WWW Conf., Apr., 1995.

[15] O. R. Zaiane, M. Xin, J. Han, "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs," Proc. Advances in Digital Libraries Conf. (ADL'98), Santa Barbara, CA, pp. 19-29, Apr., 1998.

[16] Hiroyuki Kawano, Toshiharu Hasegawa, "Mondou: Interface with Text Data Mining for Web Search Engine," Proc. 31st Hawaii International Conference on System Sciences, 1998.

[17] A. Luotonen, "The common log file format," http://www.w3.org/pub/WWW/ 1995

[18] Cyrus Shahabi, Amir Zarkesh, Jafar Adibi, and Vishal Shah, "Knowledge Discovery from Users Web-page Navigation," Proceedings of Research Issues in Data Engineering (RIDE) Conference, Apr., 1997.

[19] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava, "Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns," In Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97), Nov., 1997.

[20] Myra Spiliopoulou, Lukas Faulstich, C., and Karsten Winkler, "A Data Miner analyzing the Navigational Behaviour of Web Users," *In* Proc. of the Workshop on Machine Learning in User Modelling of the ACAI'99 Int. Conf., Creta, Greece, July, 1999.

[21] X. Lin, L. Liu, Y. Zhang and X. Zhou, "Efficiently Computing Frequent Tree-Like Topology Patterns in a Web Environment," Proceedings of the 31th international Conference on Technology of Object-Oriented.