SWING: A System for Visualizing Web Graphs

[1]Wei Lai, [2]Xiaodi Huang, [3*]Mao Lin Huang
[1] *Faculty of Information and Communication Technologies, Swinburne University of Technology*, PO Box 218, Howthorn, VIC 3122, Australia, *wlai@swin.edu.au*
[2] *School of Computing and Mathematics, Charles Sturt University, Albury, NSW 2640, Australia, xhuang@csu.edu.au*
[3]* *Department of Computer Systems, University of Technology, Sydney, Australia, maolin@it.uts.edu.au*
doi:10.4156/aiss.vol3. issue2.11

Abstract

A Web graph refers to the graph that is used to represent relationships between Web pages in cyberspace, where a node represents a URL and an edge indicates a link between two URLs. A Web graph is a very huge graph as growing with cyberspace. This paper presents a pipeline for extracting web information from cyberspace to a web graph and layout techniques for making the web graph more readable. As the size of computer screen is limited, only a small part of the Web graph can be displayed. Several layout techniques should be adapted and combined effectively for web graph visualization. The visualization process incorporates graph drawing algorithms, layout adjustment methods, as well as filtering and clustering methods in order to decide which part of the Web graph should be displayed and how to display it based on the user's focus in navigation.

Keywords*: Graph Layout, Information Visualization, Graph*

1. Introduction

Most Web browsers, such as Netscape and Microsoft Explorer, cannot provide a contextual overview required for global orientation; instead they can only give a set of URL lists. As in other areas [19,20,21], visualization plays an important role in solving this problem .

A graph is more suitable for the World Wide Web (WWW) navigation. Nodes in a graph can be used to represent URLs and edges between nodes represent links between URLs. We look at the whole cyberspace of the WWW as one graph - a huge and dynamic growing graph. However, it is impossible to display this huge graph on the computer screen.

Most current research interests towards to use "site mapping" methods [2, 12, 16]. That is, they try to find an effective way of constructing a structured geometrical map for one web site (a local map). This can only guide the user through a very limited region of cyberspace, and does not help users in their overall journey through the cyberspace.

Other attempts use a graph for the WWW navigation. The whole cyberspace of the WWW is regarded as a Web graph [6, 8, 9, 10]. In the Web graph, a node represents a Web page's URL and an edge represents a link between two URLs. This approach is placed an emphasis on navigation, but ignores achievement of a better local view for the site mapping. The graph layout by this approach shows all possible hyperlinks and makes the layout look so messy. This makes a site-mapping view sometimes unclear to users.

The primary difficulty for creating an auto-generated sitemap lies in that the number of the links can be quite big, or even huge. The presentation of these links will become messy and hard to read, so that the visualization will become useless.

To help the web navigation using graphs, we should provide clear web graph displays so that the user can easily understand the relationships shown in a web graph. This requires that interaction facilities should be provided to the user for defining and adjusting a Web sub-graph. Automatic web sub-graph displays based on the user's current focus should fit in the display window and should have no any overlaps. As the computer screen's size is limited, we should display only those information based on the user's interest. This requires we should effectively extract those information by filtering unnecessary data, and display the relevant web graph effectively using clustering and layout adjustment.

2. Examples of Web graph display

The best demonstration of our Web graph display system called SWING is through using some examples such as part of the Web site of Swinburne University of Technology.

Figures 1, 2, 3 show examples of a Web graph and its Web site respectively. The SWING has the following salient features:

- It potentially supports the incremental exploration of Web sites
- The unimportant nodes and their associated edges can be filtered to improve the readability if needed
- The horizontal and vertical exploration models are provided, which enable users to navigate the Web sites both broadly and deeply

The interface of SWING is shown in Figures 1, 2, 3, with the Web graph on the left, and the display of a Web page on the right. Except for some abstract nodes, each node in the Web graph is linked to a URL. For example, the nodes with labels www.swin.edu.au and www.cnn.com are directly linked to the corresponding Web page contents respectively, as shown in the Figure 3.

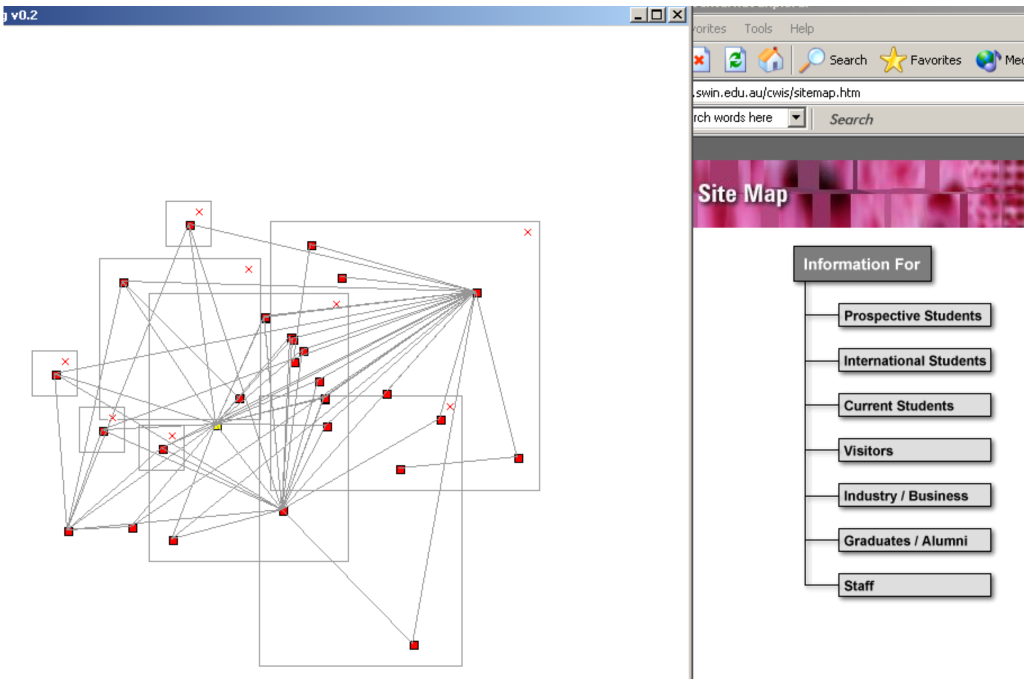


Figure 1. A Web page corresponding to a node is shown up

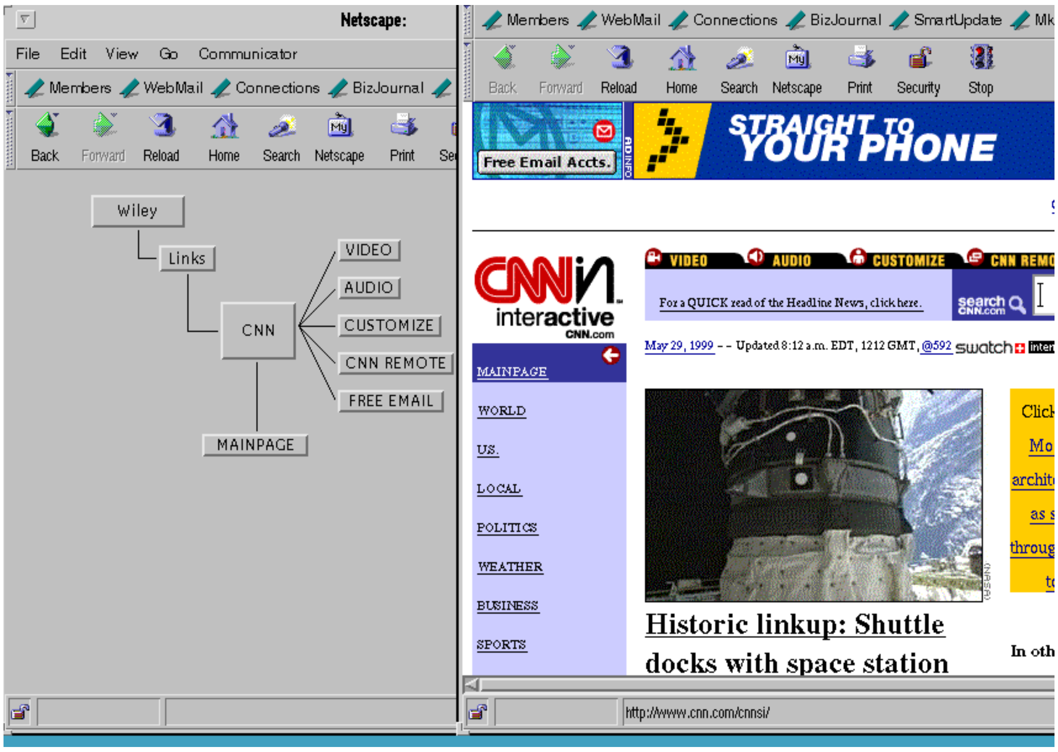
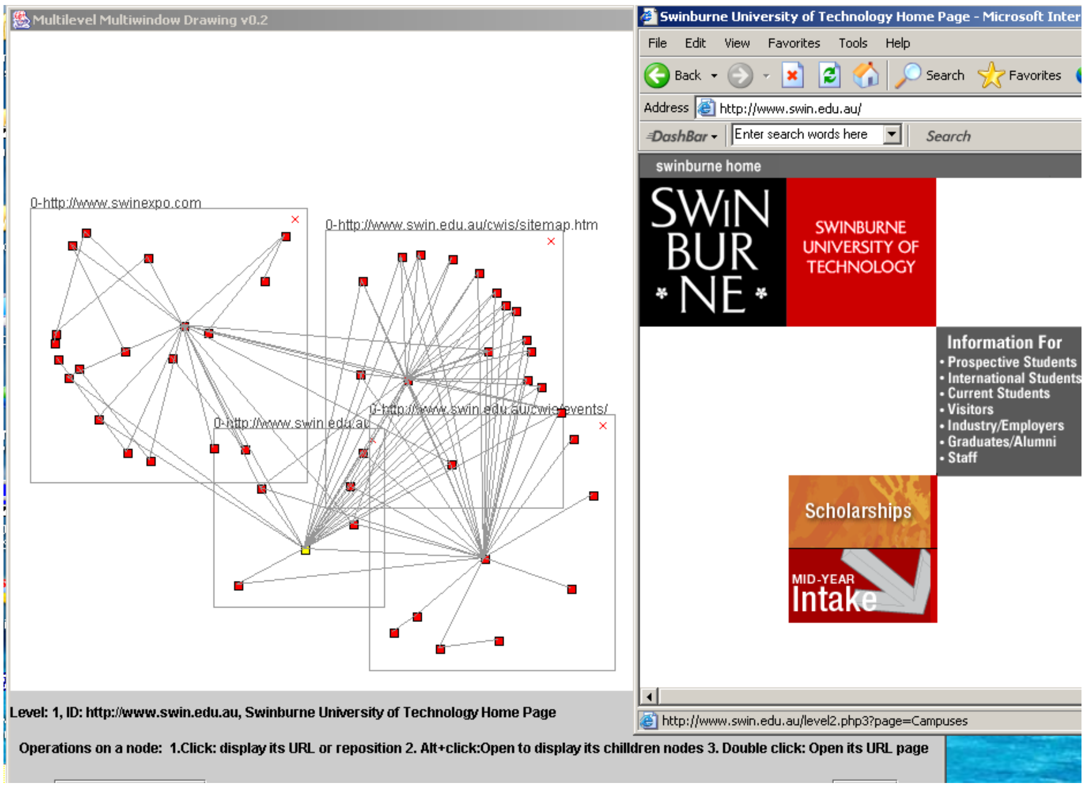

Netscape:
Wiley
Links
CNN
VIDEO
AUDIO
CUSTOMIZE
CNN REMOTE
FREE EMAIL
MAINPAGE
Historic linkup: Shuttle docks with space station
http://www.cnn.com/cnnsi/

Figure 2. Another Web site and its Web graph


Multilevel Multiwindow Drawing v0.2
0-http://www.swinexpo.com
0-http://www.swin.edu.au/cwis/sitemap.htm
0-http://www.swin.edu.au
0-http://www.swin.edu.au/cwis/events/
Swinburne University of Technology Home Page - Microsoft Inter
http://www.swin.edu.au/
swinburne home
SWINBURNE UNIVERSITY OF TECHNOLOGY
Information For
Prospective Students
International Students
Current Students
Visitors
Industry/Employers
Graduates/Alumni
Staff
Scholarships
MID-YEAR Intake
http://www.swin.edu.au/level2.php3?page=Campuses

Level: 1, ID: http://www.swin.edu.au, Swinburne University of Technology Home Page

Operations on a node: 1.Click: display its URL or reposition 2. Alt+click:Open to display its chilldren nodes 3. Double click: Open its URL page

Figure 3. Navigation of a Web site

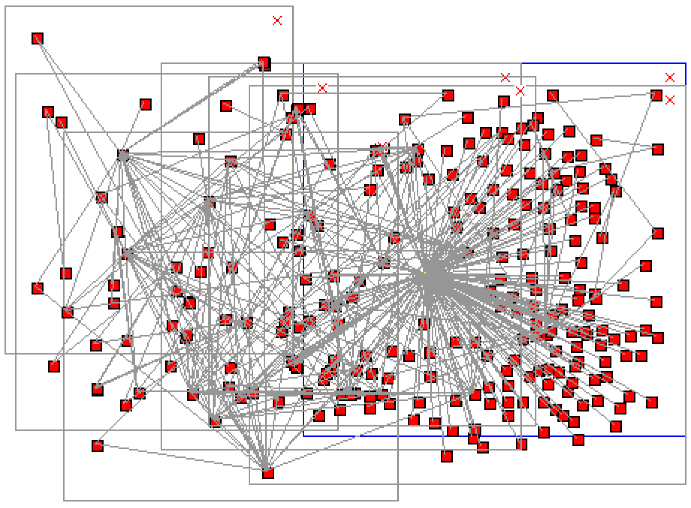
The process of the example shown in Figure 3 includes the following steps.

2.1. Data Extraction

This example was extracted from the Web site of Swinburne University of Technology by a WebCrawler. The software accepts a starting URL address as well as the exploration depth as its input, and then analyzes the hyperlinks among Web pages. A URL text file, which contains all the extracted URL addresses of two hyperlinked Web pages, is eventually produced for the following investigation.

2.2. Filtering

A filtering algorithm is applied directly to the above URL text file, or to a Web graph generated from the URL text file, where a node presents a Web page, and an edge indicates a hyperlink between two Web pages. By specifying a threshold, we remove some "noise" nodes, or less important nodes. Figure 4 is an example.


(a) A Web graph before filtering

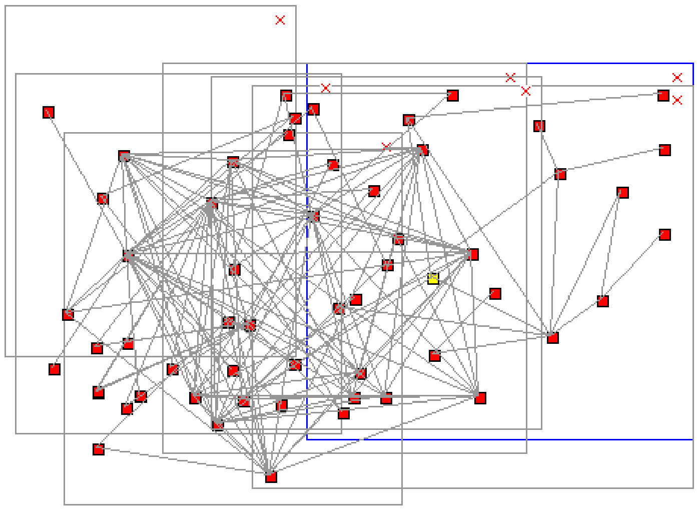
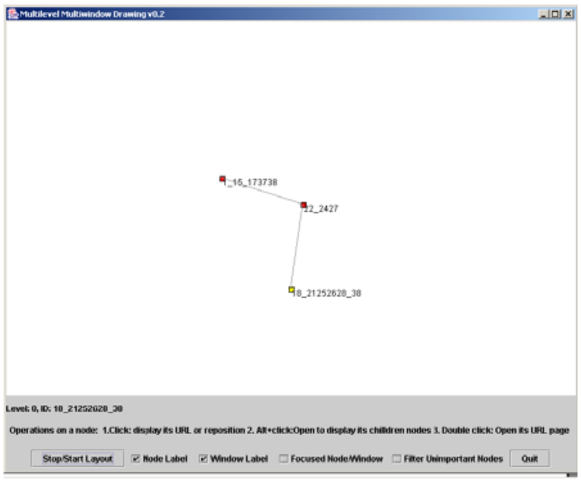

(b) The Web graph after filtering

Figure 4. An example of filtering

2.3. Clustering

After filtering is applied to the filtered Web graph, for simplicity, the highest abstract level of the clustered Web graph contains only two nodes as shown in Figure 5 (a). With two expanded abstract nodes from Figure 5 (a), Figure 5 (b) details all their children nodes and corresponding edges in the second abstract level of the Web graph.

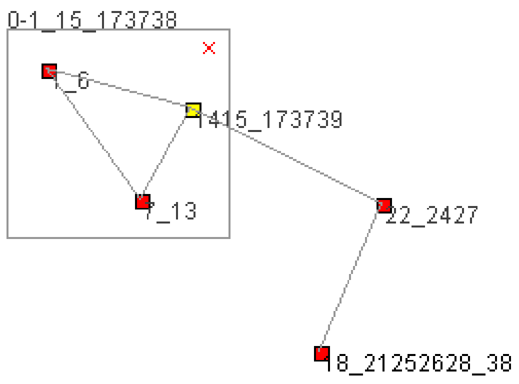


Level: 0, ID: 18_21252628_38

Operations on a node: 1.Click: display its URL or reposition 2. Alt+click:Open to display its children nodes 3. Double click: Open its URL page

☑ Node Label ☑ Window Label ☐ Focused Node/Window ☐ Filter Unimportant Nodes

(a) The first abstract level of a Web site



(b) The second abstract level of the Web site

Figure 5. A clustered Web graph

2.4. Layout Adjustment

To make sure that a web graph is displayed within the computer screen and also graph nodes have no overlaps, layout adjustment techniques are needed. They are described with graph layout techniques in the following section.

3. Effective web graph layout

This section introduces the effective graph layout techniques used in SWING for ensuring a web graph layout fits in the display window and has no overlaps.

The most difficult editing function for a Web graph is layout - assigning a position for each node and a curve for each edge. The assignment must be chosen to make the resulting picture easy to understand and easy to remember. A good layout can be like a picture - worth a thousand words; a poor layout can confuse or mislead the user. This problem is called the graph drawing problem – how to create a nice layout, automatically. Automatic layout can release the user from the time-consuming and detail-intensive chore of generating a readable diagram. However, most existing systems that incorporate diagrams, such as CASE tools, do not support automatic layout; the layout decisions in these systems have to be made by the user by using the mouse and the screen to replace the pen and paper.

Most classical graph drawing algorithms [1] produce aesthetically pleasing abstract graph layouts. These algorithms can be applied to draw practical graphs as long as the sizes of nodes take very little space. This is because such algorithms were often originally designed for abstract graphs where nodes take up little or no space.

However, in applications, the images of nodes are circles, boxes, diamonds and similar shapes, and may contain a considerable amount of text and graphics. In some systems [4, 7, 10, 16,18] nodes are used to represent sub-graphs, and may be quite unpredictable in size and shape. Applying such algorithms to practical graphs may result in overlapping nodes and/or edge-node intersections. Algorithms which exemplify this problem can be found in [3, 9] - they produce diagrams which are symmetric and well spread out, and have great potential for use in visualization of network structures. However nodes of nontrivial size in a diagram produced by these algorithms tend to overlap.

We are interested in the problem of how to display diagrams, that is, how to lay out practical graphs in applications. The term abstract graph layout refers to layout techniques for abstract graphs where nodes are negligible in size. The term practical graph layout refers to layout techniques for practical graphs where nodes vary in shape and size.

Our approach is to make use of existing classical graph drawing algorithms. That is, to apply a classical graph drawing algorithm to a practical graph. Then we need to develop some post-processes to avoid overlaps of node images and edge-node intersections by rearranging the graph layout [5, 11, 17]. The techniques for adjusting a graph layout should preserve the mental map of the original graph [6, 13].

The critical part of our approach is to remove overlapping nodes. We use the techniques for removing overlaps of node images and edge-node intersections [5, 11,17]. We have experimented with these techniques using many sets of overlapping nodes and found that it is quite effective. An example of using these techniques is shown below. Figure 6 shows a graph layout generated by an abstract graph layout algorithm (the "spring" algorithm [3]). Figure 7 shows the result of replacing the nodes with rectangles, which gives not only the overlapping nodes but edge-node intersections. Figure 8 is the result of applying the force-scan algorithm [11,17] for removing overlaps of node images and edge-node intersections.

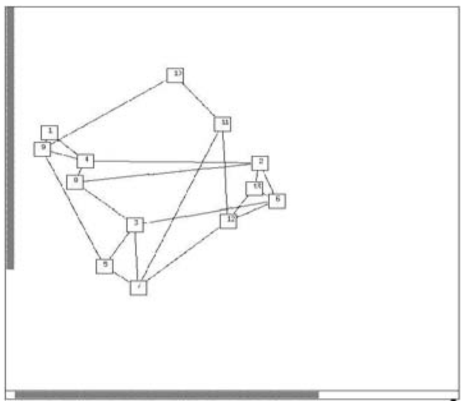


Figure 6. An abstract graph layout

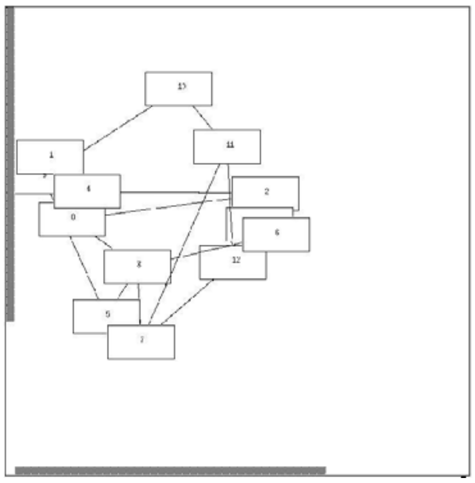


Figure 7. A practical graph

Although these techniques can not only make nodes in a diagram disjoint but also as compact as possible, it cannot guarantee the size of a diagram fits in the display window. We also need to solve this problem. To this aim, our layout adjustment includes the following three parts:

- Use the techniques in [11,17] to remove overlapping nodes and edge-node intersections.
- If the size of the diagram exceeds the viewing area, find the minimum size diagram by changing sub-graph layout (e.g. change from h-tree to tip-over).
- If the diagram still exceeds the viewing area, let some sub-graphs become invisible (in the order of those which overlap the user's currently selected sub-tree).

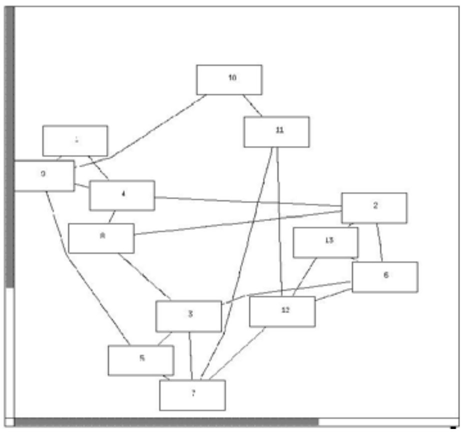


Figure 8. Layout adjustment

We used Java as the major software development tool for the implementation of SWING. A prototype of the Web graph user interface for WWW navigation has been developed.

4. Conclusion and future work

This paper introduces a new Web graph display system. The major feature of SWING is to provide visible subsets of the Web graph for WWW navigation. Our Web sub-graph display technique creates an automatic layout that does not exceed the viewing area and has no overlapping nodes.

Recent feedback from the users is that they would like to combine our Web graph interface and a current Web browser (such as Netscape) together for Web navigation. It seems that they do not like to use the Web graph interface alone for navigation.

We will continue to investigate layout techniques that will enhance potential usability of SWING. Purchase has presented a method for testing presentation and usability of graph layouts [14]. We can adopt this method to evaluate the performance of our Web graph layout.

Regardless of reviewing the work of other researchers, we need to conduct usability studies of end-users to see whether they prefer this kind of interface for WWW navigation over more traditional styles.

5. References

[1] G. D. Battista, P. Eades, R. Tamassia and T. Tollis, Graph drawing: algorithms for the visualization of graphs, Prentice Hall, 1998

[2] Y. Chen and E. Koutsofios, “WebCiao: A Website visualisation and tracking system”, In Proceedings of WebNet Conference, pp.66-75,1997.

[3] P. Eades, “A heuristic for graph drawing”, Congressus Numerantium, vol.42, pp.149-160, 1984.

[4] P. Eades and W. Lai, “Visual interface design for relational systems”, In Proceedings of the 5th Australian Software Engineering Conference, Sydney, pp.259-263, 1990.

[5] P. Eades and W. Lai, “Algorithms for disjoint node images”, Australian Computer Science Communications, Vol. 14, No. 1, pp. 253-265, 1991.

[6] P. Eades, W. Lai, K. Misue and K. Sugiyama, “Preserving the mental map of a diagram”, In Proceedings of COMPUGRAPHICS, pp. 34-43, 1991.

[7] D. Harel, “On visual formalisms”, Communications of the ACM, vol.31, no.5, pp.514-530, 1988.

[8] M. L. Huang, P.Eades and J.H.Wang, “On-line animated visualization of huge graphs using a modified Spring Algorithm, Journal of Visual Language and Computing, No.9,pp.623-645,1998.

[9] K. Kamada and S. Kawai, S. “An algorithm for drawing general undirected graphs”, Information Processing Letters, vo.31, no.1, pp.7-15, 1989.

[10] W. Lai and P. Eades, “CIGRAPHS: A new graph model”, Australian Computer Science Communications, vol.17, no.1, pp.262—270, 1995.

[11] W. Lai and P. Eades, “Removing edge-node intersections in drawings of graphs. Information Processing Letters, vol.81 no.2002, pp. 105-110.

[12] Y. S. Maarek and I. Z. B. Shaul, “WebCutter: A system for dynamic and tailorable site mapping”, In Proceedings of the Sixth International World Wide Web Conference, pp.713-722, 1997.

[13] K. Misue, P. Eades, W. Lai and K. Sugiyama, “Layout adjustment and the mental map. Journal of Visual Languages and Computing, No.6, pp. 183- 210, 1995.

[14] H. Purchase, “Performance of layout algorithms: comprehension, not computation, Journal of Visual Language and Computing, no.9, pp. 647-657, 1998.

[15] C. Pilgrim and Y. Leung, “Applying bifocal displays to enhance WWW navigation”, In Proceedings of the Second Australian World Wide Web Conference, 1996.

[16] K. Sugiyama and K. Misue, K “Visualisation of structural information: Automatic drawing of compound digraphs”, IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no.4, pp.876-892, 1991.

[17] X. Huang, W. Lai, A. S. M. Sajeev, Junbin Gao,” A new algorithm for removing node overlapping in graph visualization”, Inf. Sci. vol.177, no.14, pp. 2821-2844, 2007.

[18] X. Huang, W. Lai,” Clustering graphs for visualization via node similarities”, J. Vis. Lang. Comput. vol.17, no.3, pp. 225-253, 2006.

[19] D. Shifei, Q. Xu Li, Z. Xiangwei, J. Fengxiang, "A Clustering Algorithm Based on Information Visualization ", JDCTA, vol. 5, vo. 1, pp. 26 -31, 2011.

[20] N. G. Nik Daud, W. A. Wan Adnan, N. L. Md Noor, "Information Visualization Techniques and Decision Style: The Effects in Decision Support Environments", JDCTA, vol. 2, no. 2, pp. 20 -24, 2008.

[21] J. Liu, J. Sun, Z. Q Xu, L. Gao, "Security Web Release and Visualization of Remote Sensing Images ", JDCTA, vol. 4, no. 3, pp. 146 -153, 2010.