Fitting Web Graphs in a Display Area With No Overlaps for Web Navigation

Wei Lai

School of Information Technology
Swinburne University of Technology
PO Box 218 Hawthorn, Vic 3122, Australia

Maolin Huang

Faculty of Information Technology
University of Technology, Sydney
Sydney, Australia

Jiro Tanaka

Institute of Information Sciences and Electronics
University of Tsukuba
,Tsukuba, Ibaraki 305-8573, Japan

Abstract - Most Web browsers, including Netscape and Microsoft Internet Explorer, cannot give users a visual "map" to guide their Web journey. An approach of using a graph for Web navigation has been proposed. That is to look at the whole of cyberspace as one huge graph. To explore this huge graph, it is critical to find an effective method of tracking a sequence of subsets (Web sub-graphs) of the huge graph based on the user's focus. Any on-line Web sub-graph should fit in the display window. To enhance the display of a Web sub-graph, there should not be any overlaps between node images in the Web sub-graph. This paper introduces an approach that ensures any on-line Web sub-graph has no overlapping node images by letting the user, or the system itself, define visible and invisible parts of the Web graph. It is a challenge task to let the system automatically display a web graph has no overlaps and also ensures the graph can fit in the display area.

Keywords: Web browser, Web graph, Web sub-graph, Visual map, Navigation.

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.

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, 14]. 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.

Huang et al's proposed on-line exploratory visualisation approach [8] which provides a major departure from traditional site-mapping methods. It does not pre-define the geometrical structure of a specific Web site (a part of cyberspace); instead it incrementally calculates and maintains a small visualisation of a subset of cyberspace on-line, corresponding to the change of the user's focus. That is, it automatically displays a sequence of web sub-graphs with smooth animation following the user's orientation. This feature enables the user to logically explore the entire cyberspace without requiring the whole structure of the cyberspace to be known.

However, Huang et al's approach [8] uses the FIFO (first in and first out) rule to animate web sub-graphs, which cannot support the user to define a web sub-graph. Also, Huang et al's approach cannot ensure its web sub-graph layout has no overlapping node images. This paper introduces an approach for web graph displays that can overcome these drawbacks.

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.

There are two major features of our Web graph displays introduced in this paper. One is that the user can interact with the Web graph to let a node's sub-graph be visible or invisible. The other is that overlapping node images / sub-graphs are detected and defined as visible or invisible, automatically, based on the user's selection.

2. Examples of Web Graph Displays

The best demonstration of our Web graph display system is through using some examples. Figure 1 shows an on-line Web sub-graph. Our system can ensure that any on-line Web sub-graph has no overlapping node images and fits in the window. We provide three kinds of modes for the user's intersection. In the mode *LayoutAdjust*, the user can adjust a Web sub-graph layout. For example, If the user clicks a node in the Web sub-graph, the node's sub-graph is changed from invisible to visible or from visible to invisible. Figure 2 shows the results after the user clicks the nodes - Phone, Fax and Teaching in the Web sub-graph shown in Figure 1. If the user clicks these nodes again, their sub-graphs would become invisible (i.e. they would disappear as shown in Figure 1). In this way, the user can define a node's sub-graph as visible or invisible by direct manipulation.

When a node's sub-graph becomes visible, our system checks whether there will be overlaps between any part of the current display and this sub-graph. If so, those parts overlapping the sub-graph become invisible automatically. For instance, in Figure 3, after the user clicks the node with label Research, the sub-graph of this node appears and the sub-graph of the node Teaching automatically disappears.

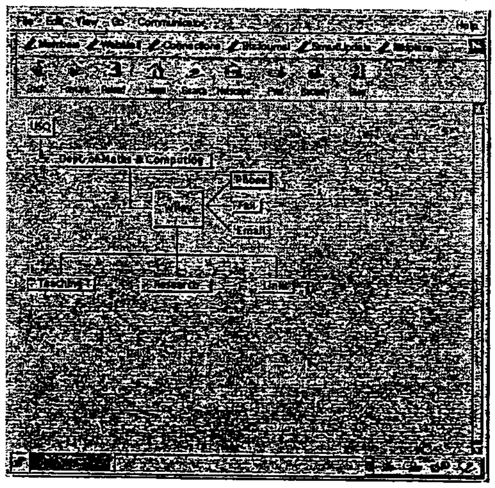


Figure 1: A Web sub-graph display

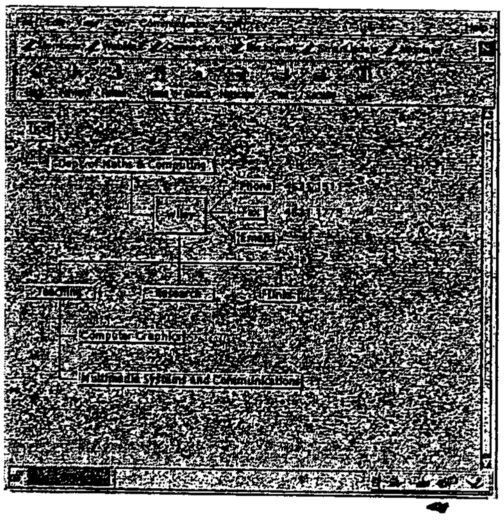


Figure 2: Some sub-graph become visible after the user's interaction

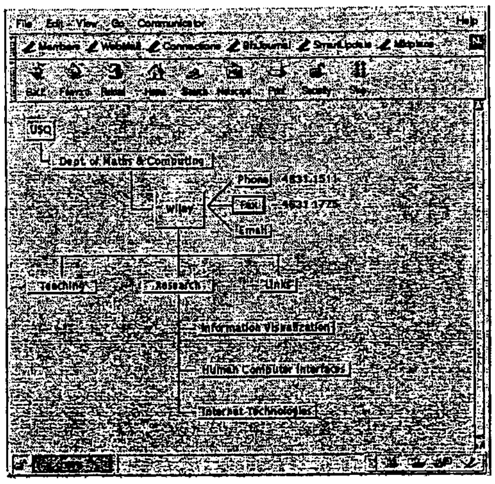

USQ
Dept. of Maths & Computing
Wiley
Phone
Fax
Email
Teaching
Research
Links
Information Visualization
Human Computer Interfaces
Internet Technologies

Figure 3: A sub-graph becoming visible makes another one invisible

A node in the Web graph is linked to a URL. For example, the node with label Computer Graphics is linked to the Web sit of the unit 66333 - Computer Graphics. The user can switch into the *ShowPage* mode by clicking the middle mouse button. Our system can support the display of a detailed Web page corresponding to a node in a Web sub-graph (after the ShowPage model is set up). Figure 4 shows the result of the user's selecting the node with label Computer Graphics in the ShowPage mode.

The third interaction mode is the *Navigation* mode which can be selected by clicking the right mouse button. In this mode, the user can change the focused node to get another Web sub-graph. Suppose that the user's current focused node is Wiley, after the user clicks the node Links and then the node CNN under the node Link in the navigation model, we can get the Web sub-graph corresponding to the user's new current focused node CNN. This Web sub-graph is shown in the window on the left in Figure 5. A Web sub-graph keeps track of the user's navigation. That is, it includes two nodes - Wiley and Links for indicating the previous two steps of navigation. The other nodes linking to the node with label CNN are formed in this way: our system analysed the source HTML file of the CNN Web site and extracted the URLs in this file to form those nodes. In this way, we could test that the system can navigate from one Web site to another one.

The user can switch among the three interaction modes. For example, after selecting the ShowPage mode, if the user clicks the node CNN, the Web page of CNN appears (see Figure 5). The user can also use the LayoutAdjust mode by clicking the left mouse button.

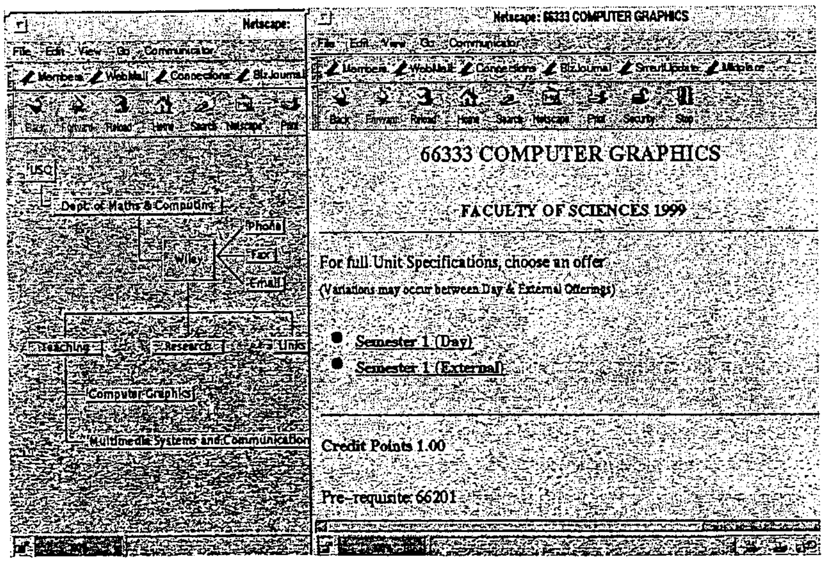
Our on-line Web sub-graph is formed dynamically based on the user's focus. When the user changes the focus (i.e. click a node in the Navigation mode), a new Web sub-graph is formed by dropping old nodes and adding new nodes. This is similar to driving a car: new views arrive in the front and old views vanish in the back.

3. Automatic graph layout techniques

This section introduces the automatic graph layout techniques used in our system 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


Netscape:
USQ
Dept. of Maths & Computing
Wiley
Phone
Fax
Email
Teaching
Research
Units
Computer Graphics
Multimedia Systems and Communication
Netscape: 66333 COMPUTER GRAPHICS

66333 COMPUTER GRAPHICS

FACULTY OF SCIENCES 1999

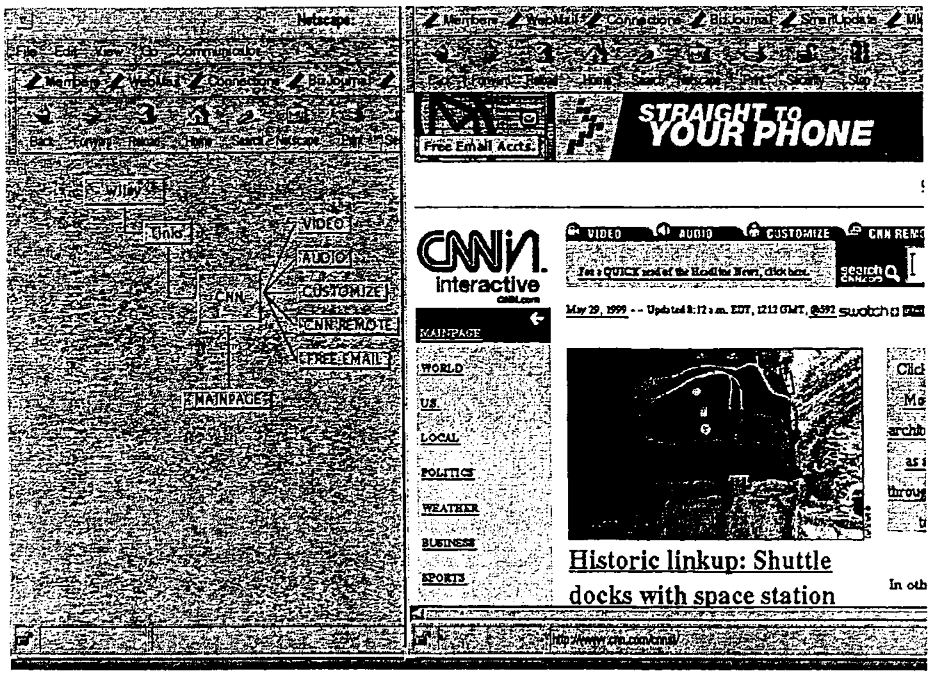
For full Unit Specifications, choose an offer
(Variations may occur between Day & External Offerings)

- Semester 1 (Day)
- Semester 1 (External)

Credit Points 1.00

Pre-requisite: 66201

Figure 4: A Web page corresponding to a node is shown up



Historic linkup: Shuttle docks with space station

Figure 5: Another Web site and its Web graph

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, 15] 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 visualisation 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]. 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]. 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] for removing overlaps of node images and edge-node intersections.

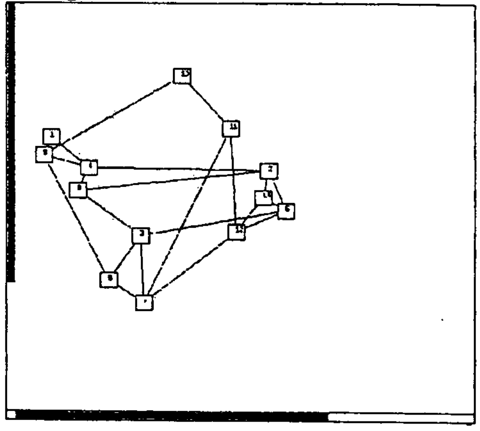


Figure 6: An abstract graph layout

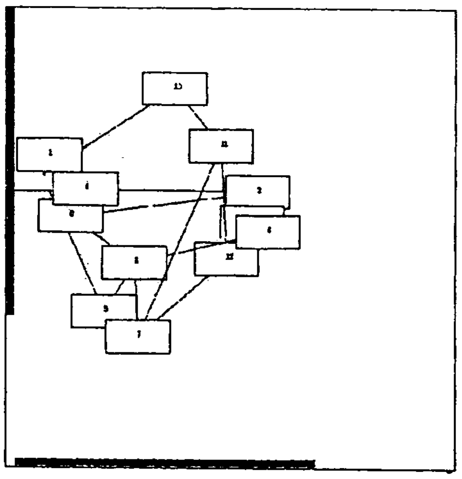


Figure 7: A practical graph

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:

1. Use the techniques (Lai & Eades, 2002) to remove overlapping nodes and edge-node intersections.
2. 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).

3. 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).

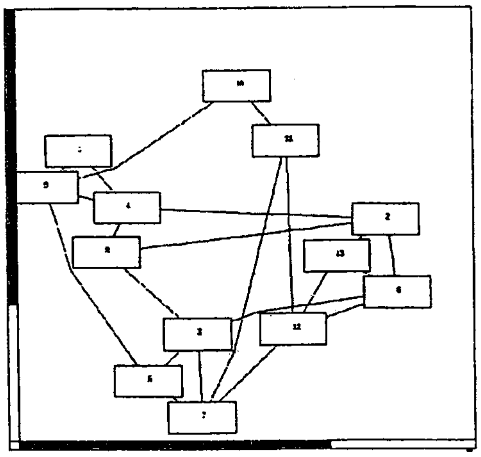


Figure 8: Layout adjustment

We used Java as the major software development tool for the implementation of our system. 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 the system 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 the system. Purchase has presented a method for testing presentation and usability of graph layouts (Purchase 1998). 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.

References

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

[2] Chen, Y. and Koutsofios, E. (1997). WebCiao: A Website visualisation and tracking system. *Proceedings of WebNet 97 Conference.*

[3] Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, 42:149-160.

[4] Eades, P. and Lai, W. (1990). Visual interface design for relational systems. *Proceedings of the 5th Australian Software Engineering Conference*, Sydney, pages 259-263.

[5] Eades, P. and Lai, W. (1991). Algorithms for disjoint node images. *Australian Computer Science Communications*, Vol. 14, No. 1, pages 253-265.

[6] Eades, P., Lai, W., Misue, K. and Sugiyama, K. (1991). Preserving the mental map of a diagram. *Proceedings of COMPUGRAPHICS 91*, pages 34-43.

[7] Harel, D. (1988). On visual formalisms. *Communications of the ACM*, 31(5):514-530.

[8] Mao Lin Huang, Peter Eades and JunHu Wang, On-line Animated Visualization of Huage Graphs using A Modifies Spring Algorithm, Journal of Visual Language and Computing, 1998,9,623-645

[9] Kamada, K. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7-15.

[10] Lai, W. and Eades, P. (1995). CIGRAPHS: A new graph model. *Australian Computer Science Communications*, 17(1):262–270.

[11] Lai, W. and Eades, P. (2002). Removing edge-node intersections in drawings of graphs. *Information Processing Letters*, 81 (2002): 105-110.

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

[13] Misue, K., Eades, P., Lai, W. and Sugiyama, K. (1995). Layout adjustment and the mental map. Journal of Visual Languages and Computing, (6): 183-210.

[14] Pilgrim, C. and Leung, Y. (1996). Applying bifocal displays to enhance WWW navigation. Proceedings of the Second Australian World Wide Web Conference.

[15] Sugiyama, K. and Misue, K. (1991). Visualisation of structural information: Automatic drawing of compound digraphs. IEEE Transactions on Systems, Man and Cybernetics, 21(4):876-892.