An On-Line Web Visualization System with Filtering and Clustering Graph Layout

Wei Lai, Xiaodi Huang, Ronald Wibowo, and Jiro Tanaka

***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. To use it for Web navigation, only a small part of the Web graph is displayed each time according to a user's navigation focus. The graph layout has always been a challenge for visualizing systems. In this paper, we present a visualization system of an online Web graph, together with the methods for clustering and filtering large graphs. In this system, a Web crawler process is used to get on-line information of the Web graph. Filtering and clustering processes reduce the graph complexities on visualization. In particular, the filtering removes those unimportant nodes while the clustering groups a set of highly connected nodes and edges into an abstract node. 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.**

***Index Terms*— Graph visualization, Filtering, Clustering, Web graph**

I. Introduction

The amount of information now available through the World Wide Web (WWW) has grown explosively. An increasing number of tools are available to assist users to manage and access information on the WWW, such as Netscape and Internet Explorer. The key requirement for a Web browser is to show the details for the users' focused information and to facilitate navigation within the whole information hyperspace. It is, however, impossible to display this huge and growing hyperspace for users to get its whole structure in helping navigation. The navigation approach used in most Web browsers is simply from one page to another page. Although current Web browsers can provide bookmarks and history lists in a linear way, they cannot show relationships between the URLs.

Some researchers have proposed "site mapping" methods [3, 12, 15] in an attempt to find an effective way of constructing the structured geometrical map for a Web site (i.e. a local map). However, this map can only guide users through a very limited region of cyberspace, and does not help the 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, 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.

This paper presents an on-line Web visualization system by using filtering and clustering to reduce visual complexities of the Web graph. The system includes the processes of Web crawler, filtering and clustering, and visualization. The Web crawler process is used to get on-line information of the Web graph. Filtering and clustering processes reduce the graph complexities on visualization. The filtering is used to remove those unimportant nodes, and the clustering is used to make a set of nodes and edges (a sub-graph) to an abstract node. The visualization process uses graph drawing algorithms and layout adjustment algorithms for graph layout. We begin with the description of our system in the following section, and then present the filtering and clustering methods in Sections 3 and 4. A case study is provided in Section 5, followed by the conclusion in Section 6.

II. The On-line Web visualization system

The on-line Web visualization system with filtering and clustering graph layout (we call it the FCG system) supports a user to use a graph to navigate the cyberspace. The Web graph is a very huge graph as the cyberspace keeps growing. During the Web navigation, each time only a small part of the Web graph is displayed. We call it a sub-Web graph which is formed based on the user's focus in navigation.

Figure 1 shows the FCG system in action. According to the user's choice of a node in navigation, the relevant Web page is shown up.

The user can navigate the Web graph by selecting a node. This selected node is called the focused node. The system can smoothly add some new nodes which are closed to the focused node and remove some other nodes which are far away from the

Manuscript received September 15 , 2004.

Wei Lai and Ronald Wibowo are with School of Information Technology, Swinburne University of Technology, PO Box 218, Hawthorn, VIC 3122, Australia (wlai@swin.edu.au).

Xiaodi Huang is with Department of Mathematics and Computing, The University of Southern Queensland, Australia.

Jiro Tanaka is with Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan.

focused node with the filtering and clustering processes based on the size of a display window.

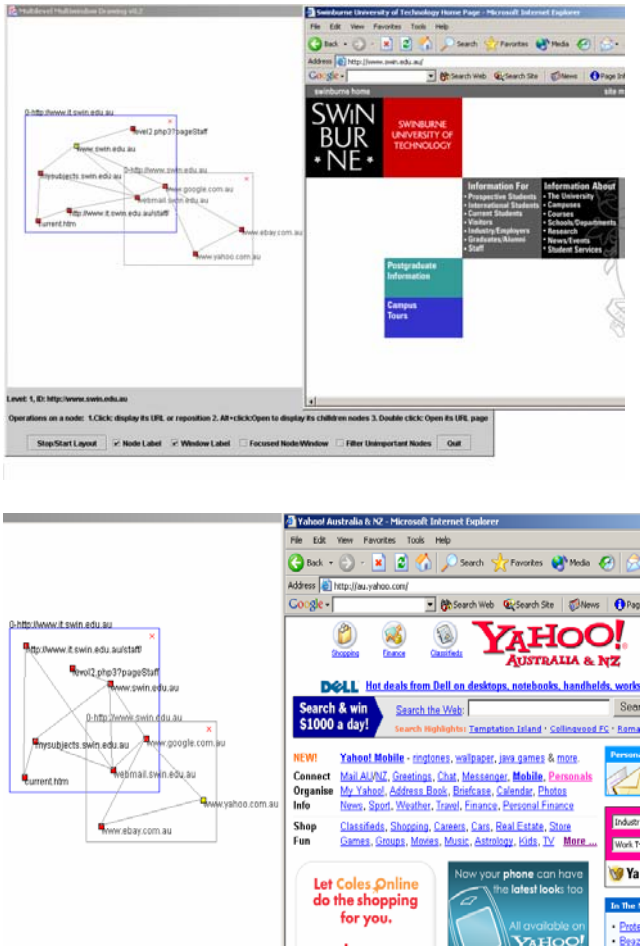


Fig. 1. Design of the FCG System

The design of the FCG system is as follows. A Web crawler extracts on-line URLs relationships from the cyberspace and constructs the Web graph represented in a text file format. This file is processed by the filtering and clustering and then goes to the visualization process for graph layout. The user can interact with the system to adjust the filtering, clustering, and visualization.

The FCG system has three modules. Each module is treated differently and can be implemented individually. The first module, called the Web crawler, is to obtain the hyperlinks among Web sites as mentioned above. The crawler crawls from a given input URL and stop when the defined depth is reached. The Web crawler then saves the URLs list into a text file. The second module is about the filtering/clustering process, while the third module includes the visualization process. The overall process is detailed in Figure 2.

In this system, we integrate the techniques of the Web crawler [2, 4, 13], graph drawing algorithms [1], layout adjustment methods [14], and the filtering and clustering methods.

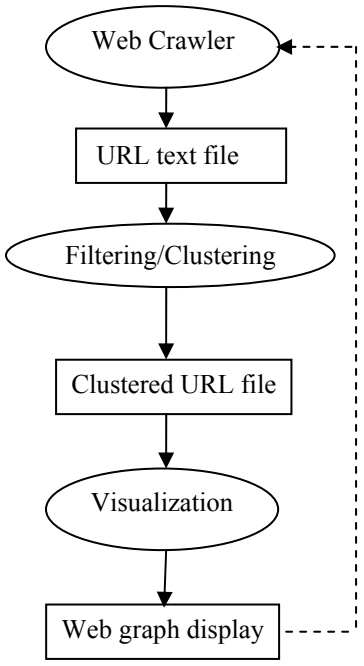


Fig. 2. Design of the FCG System

Each module can run independently with the given input, and it also produces an output. The dashed line in Figure 2 implies that the Web graph is changed on the basis of the results of the Web crawler. In other words, the new Web pages collected by the Web crawler can immediately be reflected in the updated Web graph.

As mentioned before, the Web crawler in Figure 2 is employed to extract the links from a given URL Web site, with a specified depth of exploration. The detailed process of this Web crawler can be illustrated in Figure 3.

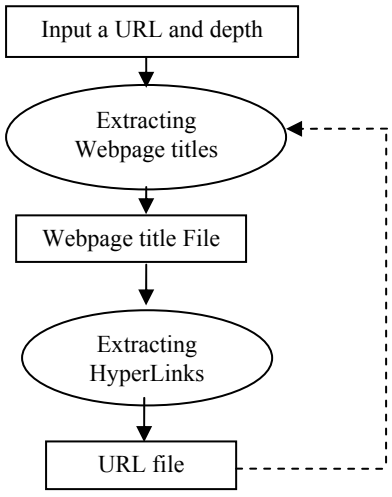


Fig.3. Web crawler process

Note that a dashed line in Figure 3 between “URL file” and

"Extracting Webpage titles" means that the process will continue until reaching the given exploration depth. For example, if the given depth is one then the process will only run once. If the given depth is three, then the process will be carried out three times with the immediately previously crawled URLs as new starting points of the exploration.

III. Filtering

To enhance the readability of the layout, some filter mechanisms are applied to the Web graph. The filter is to reduce the size of the graph by removing weak links, defined as those edges with connected nodes whose degrees are less than a predefined number. The use of the filter makes the Web graph layout easier to read due to reduction of the number of nodes and edges.

Usually, a large graph is automatically generated from an information source. This may unavoidably lead to the creation of "noise" information. For example, the use of a Web crawler program easily extracts some unwanted image files, together with html Web pages, from a Web site when constructing a Web graph. In addition, Filtering can suppress unimportant nodes and their related edges to highlight those important nodes by using an adjustable threshold to control appearances of the nodes.

The purpose of the FCG system is to display a Web graph for users to explore. In the following example of a graph in Figure 5 (this graph is a sub-Web graph), the links to CSS files and to image files are therefore considered to be unimportant links. Figure 6 shows an example by applying the filtering to remove these links.

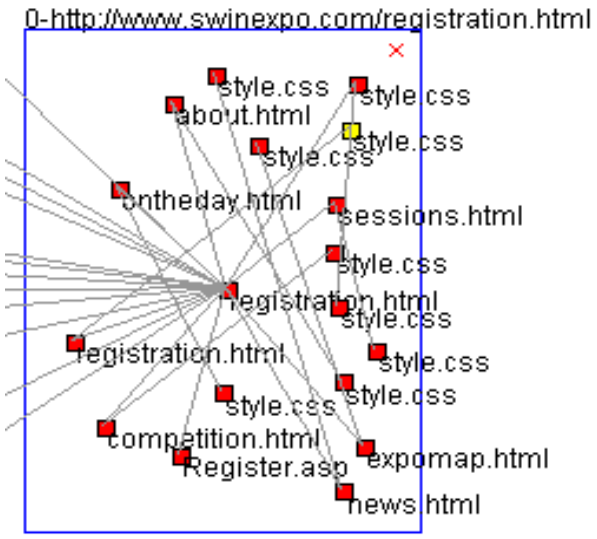


Fig.5. A graph obtained by the Web crawler process

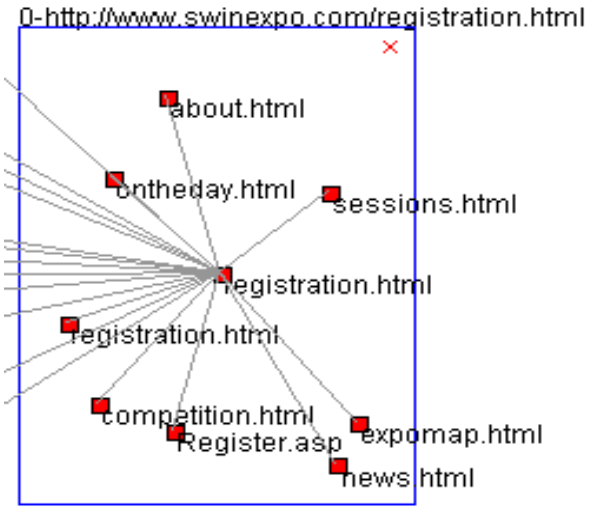


Fig.6. The graph obtained after the filtering

A filtering algorithm [8] is applied directly the Web graph. This algorithm is to calculate the node rank values for every nodes based on their connection degrees with other nodes, geodesic distances with other nodes, and the "intermediary" important role between the various other nodes. The range of the node rank value is between 0 and 1. In practice, with an appropriate threshold, some "noise" nodes or less important nodes are removed.

IV. Clustering

In the implementation, the filtering and the clustering as a whole module starts by accepting an input text file, produced by the Web crawler, and ends with outputting a file containing a list of clustered URLs. The clustering procedure is shown in Figure 7.

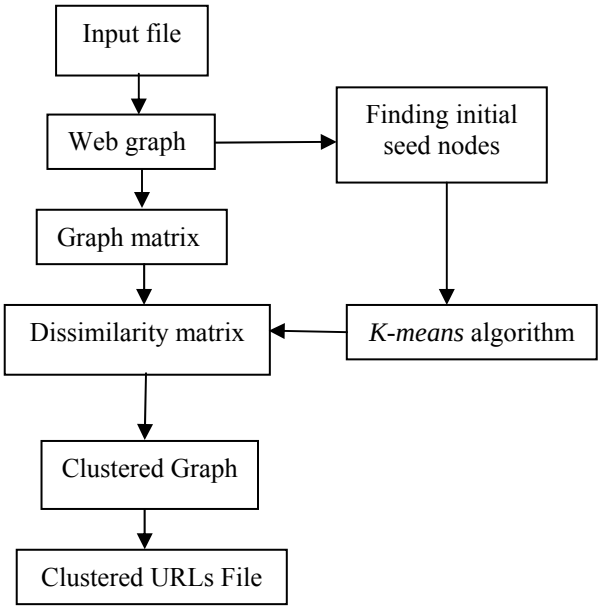


Fig. 7. Clustering process

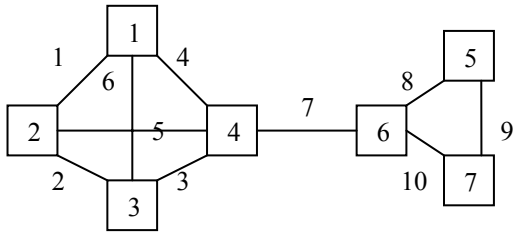
At the start of the process, a Web graph is constructed from the URL file. This graph is then represented as an edge-by-node matrix, where each column indicates a node encoded by its connecting edges, and each row represents an edge characterized by its related nodes. An example is shown in Figure 8. Note that after filtering, the graph is still represented by this kind of matrix.

The purpose of clustering a graph is to find relatively highly connected sub-graphs in a graph. With the matrix as the representation of the graph, a similarity matrix can be derived as to measure the degree of connection between two nodes.

The algorithm for clustering the elements in the matrix [7] for a given graph G can be summarized briefly as follows:

Input: G (V, E)
Output: A clustered graph

- Construct the edge-by-node incident matrix R.
- Calculate similarity values of neighbour nodes pair according to the formula (1).
- Find the shortest paths between non-neighbour node pairs by using the Dijkstra's algorithm.
- Construct the node similarity matrix according to the formula (2)
- Apply the k-means algorithm to the dissimilarity matrix constructed by the formula (3)



$$\begin{array}{c} Node \;\; 1\;\;2\;\;3\;\;4\;\;5\;\;6\;\;7 \\ R = \begin{bmatrix} 1&1&0&0&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&0&1&1&0&0&0 \\ 1&0&0&1&0&0&0 \\ 0&1&0&1&0&0&0 \\ 1&0&1&0&0&0&0 \\ 0&0&0&1&0&1&0 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&1&0&1 \\ 0&0&0&0&0&1&1 \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \quad Edge$$

Fig. 8. An example of a graph and its edge by node matrix

$$sim(r_i, r_j) = \frac{r_i^T r_j}{r_i^T r_i + r_j^T r_j - r_i^T r_j}$$

$$sim(r_i, r_j) = \frac{(R^T e_i)(e_j^T R)}{(R^T e_i)(e_i^T R) + (R^T e_j)(e_j^T R) - (R^T e_i)(e_j^T R)} \quad (1)$$

where i and $j = 1, \ldots, n$, and e_i (or e_j) denotes the i-th (or j-th) canonical vector of dimension e, i.e., $e = (1,1,\ldots,1)^T$.

This expression is used to calculate the similarity values between two neighbour nodes. In the case of non-neighbour node, the following formula is employed instead.

$$sim(r_i, r_j) = \begin{cases} \max_{P'} \{ \prod_{(r_m, r_n) \in P} sim(r_m, r_n) \} & if \; P \neq \phi \\ 0 & if \; P = \phi \end{cases} \quad (2)$$

where P is a set of pairs of nodes in the shortest path between nodes i and j, namely $P = \{(i,m), \cdots, (k, j)\}$. Such several possible shortest paths consist of a set P'

For example, the symmetric similarity matrix of Figure 8 is shown in Figure 9. The formula (2) is in fact a metric of measure the degree of two nodes in a graph, which satisfies fundamental properties: non-negativity, symmetry and ranging within [0, 1].

$$S = \begin{bmatrix} 1.000 & & & & & & \\ 0.200 & 1.000 & & & & & \\ 0.200 & 0.200 & 1.000 & & & & \\ 0.167 & 0.167 & 0.167 & 1.000 & & & \\ 0.007 & 0.007 & 0.007 & 0.042 & 1.000 & & \\ 0.028 & 0.028 & 0.028 & 0.167 & 0.253 & 1.000 & \\ 0.007 & 0.007 & 0.007 & 0.042 & 0.333 & 0.250 & 1.000 \end{bmatrix}$$

Figure 9: The node similarity matrix of the graph in Figure 8.

From similarity matrix S, we can easily obtain the dissimilarity matrix D by the following formula (3), which will be used as an input of the k-means algorithm.

$$D = [1]_{n \times n} - S \quad (3)$$

The clustering process produces a text file containing many lines, with each line describing the links between two URLs, and between two clustered groups.

If we apply the clustering process to a graph several times, the multi-level abstraction of clustered graphs will be generated. Figure 10 shows the result of applying the clustering to the graph in Figure 8.

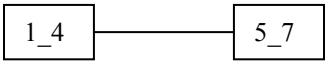


Fig. 10. An example of applying the clustering

The time requirements for calculation of the similarity matrix is O($e + n \log n$) while the running time for the k-means is O(rkn), where e is the number of edges, n is the number of nodes r is the number of iteration, and k is the number of desired cluster. Thus the total time complexities will be O ($e + n \log n + rkn$).

V. A CASE STUDY

For the purposes of this case study, we restrict our interest to visually navigate http://www.swin.edu.au (Swinburne university Website) with two levels of exploration depth using the FCG system. Specifically, our goal is to evaluate the FCG system in producing visualizations that can be used properly. We investigate the drawing, representation of a Web graph produced by the FCG system.

That is, we focus on three issues:

- The visualization of Swinburne university Website as a Web graph.
- Discussion of the drawings and representations, in terms of their ability to navigate the Web graph.
- The measurement of performance of the FCG.

A. The Experiments

To investigate the FCG system in producing Web graph visualization, we tested the FCG system to view Swinburne Website with two levels of exploration depth.

The results of this case study are presented in the following two sections. First, we present a picture gallery of different layouts produced by the FCG system. Second, we present the discussion and performance measures of the layouts shown in the picture gallery.

B. Picture Gallery

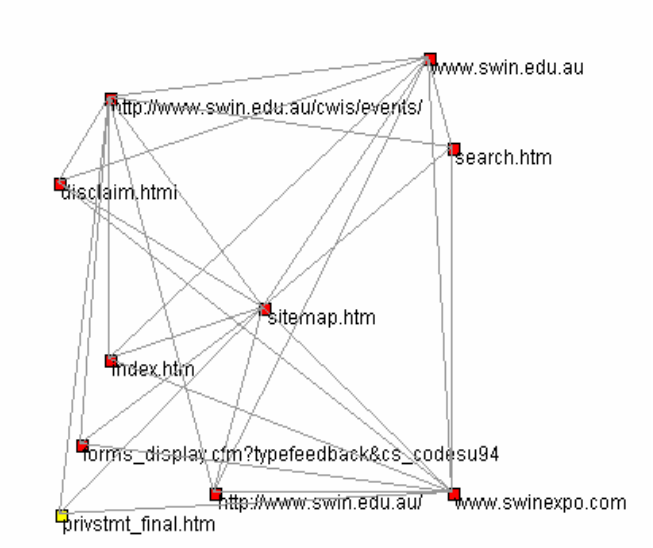


Fig.11. The Swinburne Web site using FCG

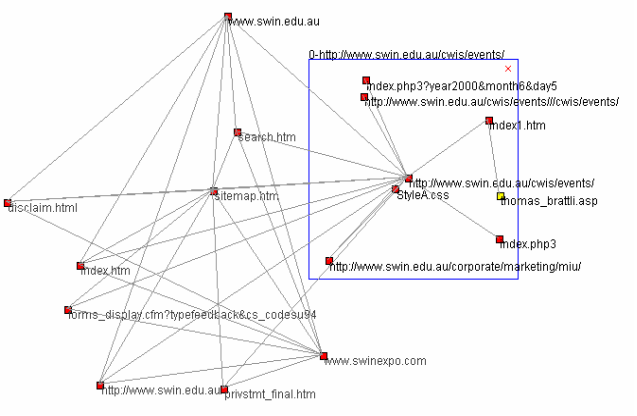


Fig.12. Expanded node labeled as http://www.swin.edu.au/cwis/events/

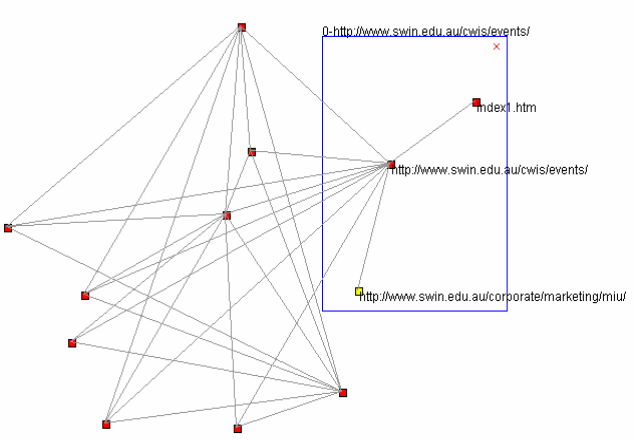


Fig. 13. Filter applied to the expanded node

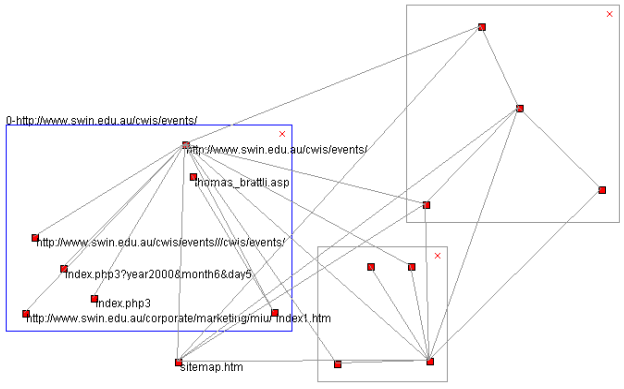


Fig.14. Expanded nodes

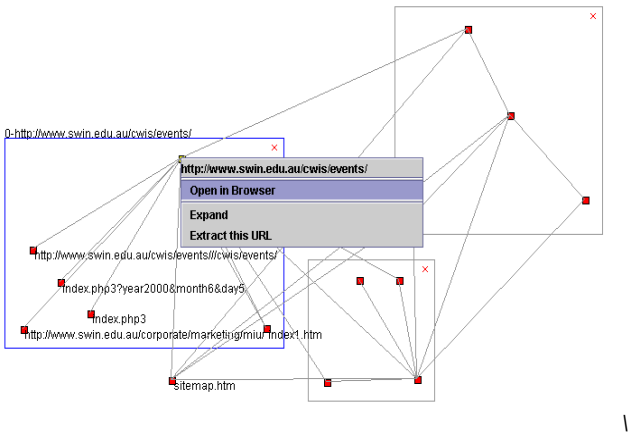


Fig.15. Node navigation menu

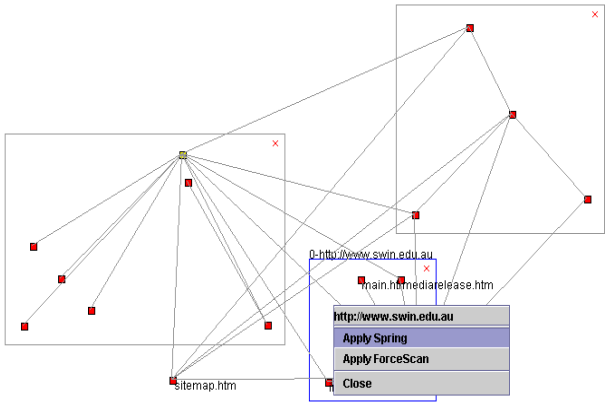


Fig.16. Navigation menu for expanded node

C. *Discussion*

Figure 11 shows the visualization result generated by FCG for the Website of Swinburne University with two levels of exploration depth. In Figure 11 graph drawing algorithms are applied to the Web graph, which assign the positions for all nodes to ensure that there are no overlapping nodes in the graph.

The running time taken by FCG to render the Web graph in Figure 11 is 1.3 seconds in Pentium 4 computer with 1.8GHz clock and 512 RAM. The drawing process using modified spring algorithm [5], took with the spring animation off. The modified spring algorithm only applied in fifty iterations, so if the animation to position the node is turned on, it took half second per iteration (the system timer in FCG to call the spring algorithm is per half second). The force-scan algorithm [10], used to remove the overlapping nodes, took 0.17 seconds to perform on ten nodes.

The Web crawler and clustering/filtering process took longer time than the layout process. The reason for this is that it depends on the Internet speed. The clustering process depends heavily on the number of URLs, taking 2.4 seconds to cluster 178 URLs in this case.

The ten nodes in the Web graph in Figure 11 can be shown in more detail as shown in Figure 12 by expanding the nodes. In Figure 12 the node http://www.swin.edu.au/cwis/events/ is expanded from the corresponding node in Figure 11, and graph drawing algorithms are applied on the expanded node window. Without animation, it took less then 1 second to apply the modified spring algorithm to the expanded node, and 0.22 seconds to apply the force-scan algorithm to the expanded node and root level nodes. The node expanded creates a new window, so it is easy to see the nodes inside the expanded node. When the window is closed, the Web graph in Figure 12 is returned back to Web graph in Figure 11.

Figure 13 shows the resulting graph when applied filtering to the Web graph in Figure 12. The weak links and unimportant link filters have been removed. The node labels in Figure 13 are visible only for the active window, in which the expanded node resides.

An example of the Web graph with more than one expanded node is illustrated in Figure 14 Note that the filtering rules have been applied to the graphs shown in Figures 13 and 14. In the presence of more than one expanded nodes, the Web graph tends to grow too large to be fitted in the screen. When this problem occurs, expanded nodes that are not focused and positioned outside will be closed.

The FCG system provides the navigation menu for a focus node, as shown in Figure 15. There are three menus: first, “Open in Browser”, which opens the URL page of the corresponding node using the system default browser; second “Expand” , which expands the node into the detailed nodes, similar to the node http://www.swin.edu.au/cwis/events/ in Figure 12, and the last menu called “Extract this URL” , which extracts all other URLs connected to the current URL in order to expand the Web graph. When the latter menu is clicked, the entire visualization process described before will be performed again. For the updated graph two windows will be generated, along with the first window displaying the original graph layout, and the second showing the newly extracted URLs.

Figure 16 shows the navigation menu that is available for the expanded node when the focused expanded node is right-clicked. There are three menus items. The first menu item displays the focused expanded node name (a URL). The second menu item, “Apply Spring”, applies the modified spring algorithm to lay out the expanded node that includes all nodes inside it. The other nodes will, however, not be affected. The second menu item, “Apply force-scan”, enforces the force scan algorithm to adjust the layout of the expanded node and its parent nodes. Although there are three expanded nodes in Figure 16, for example, the force-scan algorithm is restricted to the nodes inside the window titled http://www.swin.edu.au, and their root level nodes, if the “Apply force-scan” menu item is chosen for the expanded node. The last menu item “close” closes the expanded node.

VI. CONCLUSION

In this paper, we have presented a system for visualization of Web sites, along with the algorithm and approach for clustering and filtering graphs. As opposed to existing approaches that suffer from the limitation of the messy layouts of large graphs, our approach was designed to overcome this difficulty in a stepwise and refinement way by using clustering and filtering graphs. A prototype called FCG has been implemented to demonstrate the performance of our approaches with a case study. The future work will include the usability test of this system.

REFERENCES

[1] G. D. Battista, P. Eades, R. Tamassia, and T. Tollis, *Graph drawing: algorithms for the visualization of graphs,* Prentice Hall, 1999.

[2] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.

[3] Y. Chen and E. Koutsofios, "WebCiao: a Website visualisation and tracking system," In *Proceedings of WebNet 97 Conference*, 1997.

[4] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," In *Proceedings of the Seventh International World Wide Web Conference*, pages 161–172, April 1998.

[5] P. Eades, "A heuristic for graph drawing," *Congressus Numerantium*, 42:149-1601984.

[6] M. Huang, P. Eades, and J. Wang, "On-line animated visualization of huge graphs using a modified spring algorithm," *Journal of Visual Languages and Computing*, vol. 9, no.6, pp. 623-645,1998

[7] X. Huang and W. Lai, "Automatic abstraction of graphs based on node similarity for graph visualization," In *Proceedings of The Fifteenth International Conference on Software Engineering and Knowledge Engineering*, pp167-173, San Francisco Bay, July 2003.

[8] X. Huang and W. Lai, "NodeRank: a new structure based approach to information filtering," In *Proceedings of the International Conference on Internet Computing*, pp167-173, Las Vegas, USA. 2003.

[9] W. Lai, M. Huang, Y. Zhang, and M. Toleman, "Web graph displays by defining visible and invisible subsets," In *Proceedings of AusWeb99 - the Fifth Australian Web Conference,* pp. 207-218, Ballina, NSW, April 1999.

[10] W. Lai, M. Huang, and J. Tanaka, "Fitting Web graphs in a display area with no overlaps for Web navigation," In *Proceedings of the International Conference on Internet Computing*, pp. 601-607, June, 2002.

[11] W. Lai and P. Eades, "Removing edge-node intersections in drawings of graphs," *Information Processing Letters*, vol.81, pp105-110, 2002.

[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] R. C. Miller and K. Bharat, "SPHINX: A framework for creating personal, site-specific Web crawlers," In *Proceedings of the Seventh International World Wide Web Conference*, pp.119–130, April 1998.

[14] 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

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