Title: TA Kernel-based Algorithm for Multilevel Drawing Web Graphs
Author: X. Huang, W. Lai, D. Zhang and Q. V. N. Maolin Huang
Author Address: xhuang@csu.edu.au
Conference Title: Conference Name: 4th International Conference Computer Graphics, Imaging and Visualization
Year of Conference: 2007
Conference Location: Bangkok, Thailand
Editor: Publisher: IEEE Computer Society
Pages: 454-459
URL: http://www.graphicslink.co.uk/cgiv07/
Keywords: Graph visualization
Filtering
Clustering
Web graph
Abstract: A Web graph refers to the graph that models the hyperlink relations between Web pages in the WWW, where a node represents a URL and an edge indicates a link between two URLs. A Web graph is normally a very huge graph. In the course of users' Web exploration, only part of the Web graph is displayed on the screen each time according to a user's current navigation focus. In this paper, we make use of a fast kernel-based algorithm that is able to cluster large graphs. The algorithm is implemented in an online visualization system of Web graphs. In the system, a Web crawler first generates the Web graph of web sites. The clustering algorithm then reduces the visual complexities of the large graph. In particular, it groups a set of highly connected nodes and their edges into a clustered graph with abstract nodes and edges. The experiments have demonstrated that the employed algorithm is able to cluster graphs
CSU ID: CSU279955

A Kernel-based Algorithm for Multilevel Drawing Web Graphs

Xiaodi Huang[1], Wei Lai[2] ,Di Zhang[3], Maolin Huang[4], Quang Vinh Nguyen[4]

[1]School of Mathematics, Statistics and Computer Science, The University of New England, Australia
[2]School of Information Technology, Swinburne University of Technology, VIC 3122, Australia
[3]Institute of Software, Chinese Academy of Sciences
[4]Faculty of Information Technology, University of Technology, Sydney

xhuang@turing.une.edu.au, wlai@swin.edu.au, zhangdi@mail.rdcps.ac.cn, {maolin, quvnguye}@it.uts.edu.au

***Abstract*—A Web graph refers to the graph that models the hyperlink relations between Web pages in the WWW, where a node represents a URL and an edge indicates a link between two URLs. A Web graph is normally a very huge graph. In the course of users' Web exploration, only part of the Web graph is displayed on the screen each time according to a user's current navigation focus. In this paper, we make use of a fast kernel-based algorithm that is able to cluster large graphs. The algorithm is implemented in an online visualization system of Web graphs. In the system, a Web crawler first generates the Web graph of web sites. The clustering algorithm then reduces the visual complexities of the large graph. In particular, it groups a set of highly connected nodes and their edges into a clustered graph with abstract nodes and edges. The experiments have demonstrated that the employed algorithm is able to cluster graphs.**

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

1.Introduction

THE amount of information available in the Internet has grown explosively. A number of tools are available to assist users to manage and access information on the WWW. 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. 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). This map, however, guides users through only a very limited region of cyberspace, losing the context of the users' 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, 19]. In such a Web graph, a node represents the URL of a Web page and an edge indicates 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. In some cases, this makes a site-mapping view unclear to users.

The primary difficulty for creating an auto-generated sitemap lies in that the number of the links could be quite big, or even huge. The display of all these links will become messy and hard to read. As a consequence, the visualization will become useless. This paper presents a clustering method that reduces visual complexities of the Web graph. This method is applied to an on-line Web visualization system. The system includes the processes of Web crawler, clustering, and visualization. The Web crawler is used to form the Web graph. Clustering the graph reduces the complexities on visualization. The visualization process uses graph drawing algorithms for graph layout. We begin with the description of our system in the following section, and then present the clustering methods in Section 3. Experiments are provided in Section 4, followed by the conclusion in Section 5.

2. The On-Line Web Visualization System

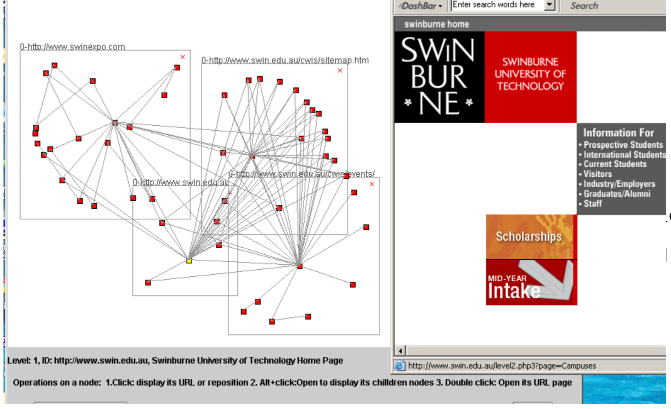
The on-line Web visualization system called the FCG system with clustering graph layout supports a user to use a graph to navigate the cyberspace. The Web graph is a very huge graph as the cyberspace keeps growing. During a user's Web navigation, only a small part of the Web graph is displayed each time.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

Figure 1 shows an interface of the FCG system. Based on the user's choice of a node in a Web graph on the left, the content of the corresponding Web page is shown on the right.

In our system, the user is able to navigate the Web graph by selecting a node. This selected node is called the focused node. The system then smoothly add some new nodes which are closed to the focused node and remove some other nodes which are far away from the focused node with the filtering and clustering processes based on the size of a display window.

The FCG system processes as follows. A Web crawler extracts on-line URLs and their relationships from the WWW. The Web graph is constructed in the form of a text file. This file, together with other required parameters, is inputted into the clustering algorithm. The clustered graphs are layout by a layout algorithm. The user can interact with the system to adjust the parameters related to the clustering, and visualization processes.

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 clustering process, while the third module includes the visualization process. The overall process is detailed in Figure 2.



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

es of the Web crawler [2, 4, 13],
ustment methods [14,20], and the

Fig. 1. Design of the FCG System

Each module runs independently with the given input, producing an output. The dashed line in Figure 2 implies that the Web graph is updated on the basis of the results of the Web crawler. In other words, the new Web pages collected by the Web crawler are reflected immediately in the subsequent 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 is illustrated in Figure 3.

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 set to one then the process will only run once. If the given depth is set to three, then the process will be carried out three times with the immediately previously crawled URLs as new starting points of the exploration.

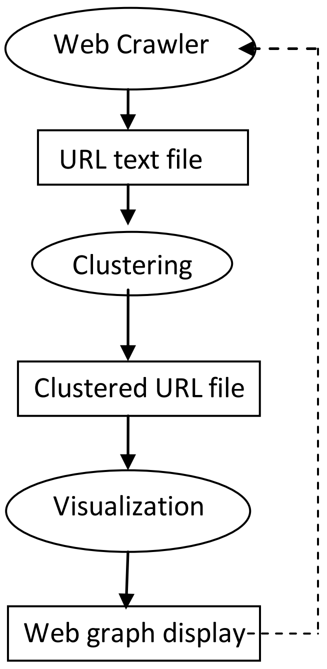


Fig. 2. Design of the FCG System

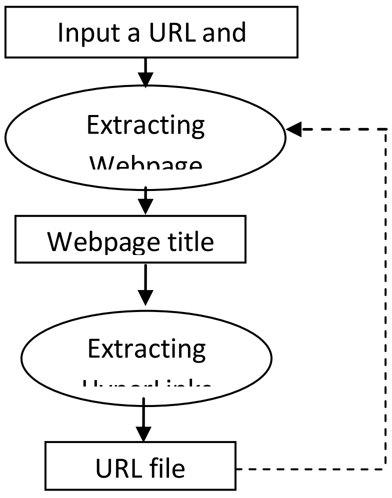


Fig.3. Web crawler process

3. Clustering

In the implementation, 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 4.

At the start of the process, a Web graph is constructed from the URL file. This graph is then represented as an affinity matrix, where each column indicates a node encoded by its connecting edges, and each row represents an edge characterized by its related nodes.

The purpose of clustering a graph is to find relatively highly connected sub-graphs in a graph. Before presenting the algorithm for graph clustering, we introduce a few definitions.

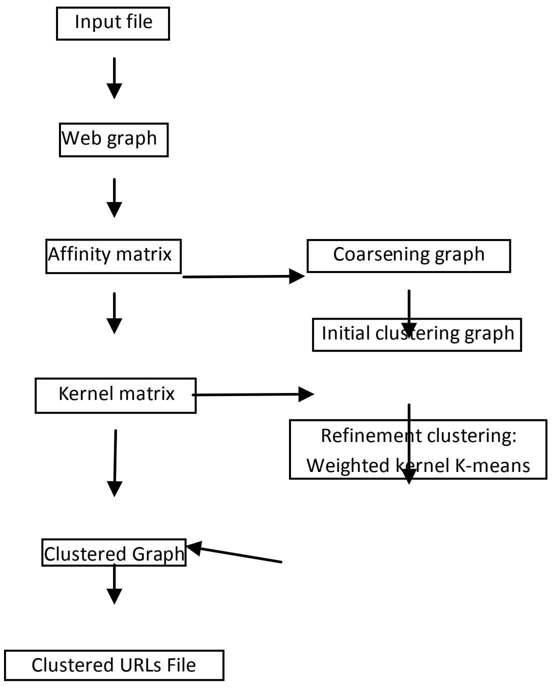


Fig. 4. Clustering process

In the following, given a graph $G = (V, E)$, we have

Definition 1 (affinity matrix A): a $|V|\times|V|$ matrix whose entries 1 if the corresponding node is connected to another, and 0 otherwise.

Definition 2 (coarse graph): An initial graph G_0, is repeatedly transformed into smaller and smaller graphs G_1, G_2, ...,G_l such that $|V_0| > |V_1| > ... > |V_m|$. A set of nodes in G_i forms a supernode in G_{i+1}. The edges between two supernodes in G_{i+1} embody existing edges between any member nodes of a cluster and other nodes not belonging to this cluster. In other words, internal edges among members in a cluster disappear while external edges, between cluster members and other outside nodes in the graph, are contracted into abstract edges.

Definition 3 (kernel function): a nonlinear mapping ϕ from the original (input) space to a higher dimensional feature space. That is, we have $K_{ij} = \kappa(\boldsymbol{a}_i, \boldsymbol{a}_j) = \phi(\boldsymbol{a}_i)\cdot\phi(\boldsymbol{a}_j)$ .Examples of kernel functions are $\kappa(\boldsymbol{a}_i, \boldsymbol{a}_j) = (\boldsymbol{a}_i \cdot \boldsymbol{a}_j + c)^d$ (Polynomial kernel) $\kappa(\boldsymbol{a}_i, \boldsymbol{a}_j) = \exp(-\|\boldsymbol{a}_i - \boldsymbol{a}_j\|^d / 2\sigma^2)$ (Gaussian kernel),

,and $\kappa(\boldsymbol{a}_i, \boldsymbol{a}_j) = \tanh(c(\boldsymbol{a}_i \cdot \boldsymbol{a}_j) + \theta)$ (Sigmoid kernel), where $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ denote vectors. One benefit of the use of the kernel function is able to find clusters that are non-linearly separable in input space, without explicitly mapping into the high dimensional feature space.

Definition 4 (kernel matrix): a matrix induced from data:

$$K = \{\phi(\boldsymbol{a}_i)\cdot\phi(\boldsymbol{a}_j)\}_{i=1,\cdots,|V|;\, j=1,\cdots|V|}$$

Note that the dot products $\phi(\boldsymbol{a}_i)\cdot\phi(\boldsymbol{a}_j)$ are computed using kernel function.

Definition 5 (matrix *D*): a diagonal matrix whose entries are equal to the sum of the corresponding rows of the affinity matrix A That is, $D_{ii} = \sum_{j=1}^{|V|} A_{ij}$.

Definition 6 (normalized cut): One of the objective function for graph clustering:

$$\underset{v_1,\cdots,v_k}{\min\text{imize}} \sum_{c=1}^{k} \frac{\deg(v_c, V \setminus v_C)}{\deg(v_c)}$$

where $\deg(v_c)$ is the sum of the degrees of all nodes in a cluster v_c

Now we are ready to present the algorithm for clustering graph *G*, the objective function of which is normalized cut [8]:

Coarsening Phase

The algorithm for coarsening a given graph is as follows:

```
**Input:** $G= (V, E)$, and abstract level $l$

**Outputs**: a coarse graph $G^l$

For $i$ =1 to $|V|$

    node $i$ = unmarked

end For

Do
```

```
        Randomly pick a node j from V
        If j= marked then go to the While
        If j = unmarked and all nodes in neighbor(j) = marked then
            node j = marked
            go to the While
        end If
        If j = unmarked and a node belonging to neighbor(j) = unmarked then
            $l^* = \arg\max_{l \in neighbor(j)} \frac{1}{\deg(j)} + \frac{1}{\deg(l)}$
            merge nodes j with l*
        end If
    While all nodes =marked
    form abstract nodes using each group of merged nodes, respectively
    aggregate the corresponding edges into abstract edges
    Output the coarse graph
```

Fig. 5. An algorithm for coarsening graphs

In the above algorithm, the neighbor of a node is the nodes which are directly connected to this particular node. deg(x) is the degree of node x. Once all nodes are marked, the coarsening for this level is completed.

Initial Clustering Phase

After coarsening, the graph has a small number of nodes left. The number of those remaining nodes is determined by a threshold. It is normally related to the number of clusters k . We stop coarsening when the graph has less than 20k nodes, for example. In order to speed up the following refinement algorithm, we use the spectral algorithm of Yu and Shi [16], in order to perform an initial partitioning on the coarsest graph obtained in the *Coarsening phase.*

Refinement Phase

The purpose of this phase attempts to locally optimize the objective of normalized cut.

The refinement phase is opposite to the coarsening phase. The refinement algorithm is applied to each coarse graph until the original graph G_0. Since the *initial clustering phase* outputs a good initial clustering at each coarse level, the refinement algorithm often converges very quickly. Thus, this approach is extremely efficient [8]. The refinement algorithm is to employ a weighted kernel k-means algorithm.

Wighted_Kernel_kMeans(K, k, t_{max}, $\{\pi_c^{(0)}\}_{c=1}^{k}$, $\{\pi_c\}_{c=1}^{k}$)

Input: K: kernel matrix, k: number of clusters, *t*

t_{max}: optional maximum number of iterations, $\{\pi_c^{(0)}\}_{c=1}^k$: optional initial clustering

Output: $\{\pi_c\}_{c=1}^k$: final clustering of the nodes

1. If no initial clusters are given, initialize *k* clusters $\pi_1^{(0)},\cdots,\pi_k^{(0)}$ randomly. Set *t* = 0.

2. For each row *i* of *K* and each cluster *c*, compute

$$d(i,\boldsymbol{m}_c) = K_{ii} - \frac{2\sum_{j\in\pi_c^{(t)}} \deg(j) K_{ij}}{\sum_{j\in\pi_c^{(t)}} \deg(j)} + \frac{\sum_{j,l\in\pi_c^{(t)}} \deg(j)\deg(l) K_{jl}}{(\sum_{j\in\pi_c^{(t)}} \deg(j))^2} \quad (1)$$

3. Find $c^*(i) = \arg\min_c d(i,\boldsymbol{m}_c)$, resolving ties arbitrarily. Compute the updated clusters as

$\pi_c^{(t+1)} = \{i : c^*(i) = c\}$

4. If not converged or $t_{max} > t$, set $t = t + 1$

and go to Step 3; Otherwise, stop and output final clusters $\{\pi_c^{(t+1)}\}_{c=1}^k$

Fig. 6. Refinement algorithm using the weighted kernel k-means algorithm

Equation (1) is derived from the weighted kernel k-means [8].

Given a set of vectors $\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_n$, the weighted k-means algorithm seeks to find clusters $\pi_1,\pi_2,\cdots,\pi_k$ that minimize the objective function:

$$D(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\boldsymbol{a}_i\in\pi_c} w_i \,\|\, \phi(\boldsymbol{a}_i) - \boldsymbol{m}_c \,\| \text{ where } \boldsymbol{m}_c = \frac{\sum_{\boldsymbol{a}_i\in\pi_c} w_i \phi(\boldsymbol{a}_i)}{\sum_{\boldsymbol{a}_i\in\pi_c} w_i}.$$

The **m**$_c$ represents the “best” cluster representative since

$$\boldsymbol{m}_c = \arg\min_z \sum_{\boldsymbol{a}_i\in\pi} w_i \,\|\, \boldsymbol{a}_i - \boldsymbol{z} \,\|^2$$

The distances| can be computed using inner products:

$$\|\, \phi(\boldsymbol{a}_i) - \boldsymbol{m}_c \,\|^2 =$$

$$\phi(\boldsymbol{a}_i)\cdot\phi(\boldsymbol{a}_i) - \frac{2\sum_{\boldsymbol{a}_j\in\pi_c} w_i \phi(\boldsymbol{a}_i)\phi(\boldsymbol{a}_j)}{\sum_{\boldsymbol{a}_j\in\pi_c} w_j} + \frac{\sum_{\boldsymbol{a}_j,\boldsymbol{a}_l\in\pi_c} w_j w_l \phi(\boldsymbol{a}_j)\phi(\boldsymbol{a}_l)}{(\sum_{\boldsymbol{a}_j\in\pi_c} w_j)^2} \quad (2)$$

Using the kernel matrix K and the degree as the weight, Equation (2) is rewritten as Equation (1).

The clustering objective in above algorithm is normalized cut.

The corresponding kernel matrix is $K = \sigma D^{-1} + D^{-1} A D^{-1}$, where σ is a chosen real number so that *K* is positive definite. As long as *K* is positive definite, it is theoretically guaranteed that the above algorithm monotonically optimizes the clustering objective of normalized cut.

Fig. 7 describes our algorithm for clustering a graph. The main algorithm begins with constructing the affinity matrix *A* from a given graph *G,* by expressing each node as the column and each edge as the row. A number of coarse graphs is generated by the coarsening phase. We apply a spectral algorithm to these coarse graphs in order to obtain the initial partitioning of the graph. Based on these initial partitioning, a weighted kernel k-means algorithm refines the partitioning, minimizing the objective function by moving nodes from one cluster to another. Finally, a sequence of coarse graphs G^l is laid out by the Spring Embedding algorithm [16]. Overall, the algorithm is summarized below:

```
    Input: $G = (V, E)$
Outputs: Multiple window layouts of clustered subgraphs of $G^l$
$l=0$
Do
    Construct the affinity matrix $A$ of graph $G^l$
    Coarsen $G^l$ using the algorithm in Fig. 5
    Initially Cluster
    Apply the refinement algorithm in Fig. 6 to $D^l$
    Add abstract edges
    Store the clustered graphs to $G^l$
    $l=l+1$
While $l <$ threshold or $|V|=1$
Input the abstract level $l$
Layout $G^l$
```

Fig. 7. An overall algorithm for multilevel clustering graphs.

It is clear that the bottleneck of complexity is the kernel k-means algorithm [8]. If the total number of iterations is τ, then the time complexity of the algorithm is $O(|V|^2(\tau + m))$ where *m* is the dimensionality of the original points. It is $O(|V|z \cdot \tau)$ if a positive definite matrix is given as input, where $|V|z$ is the number of non-zero entries of the matrix ($|V|z = |V|^2$ for a dense kernel matrix).

4. Experiments

In this section, we report two experiments on our approach. The first is a synthetic data for generating a small graph. The second one was generated from our system.

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

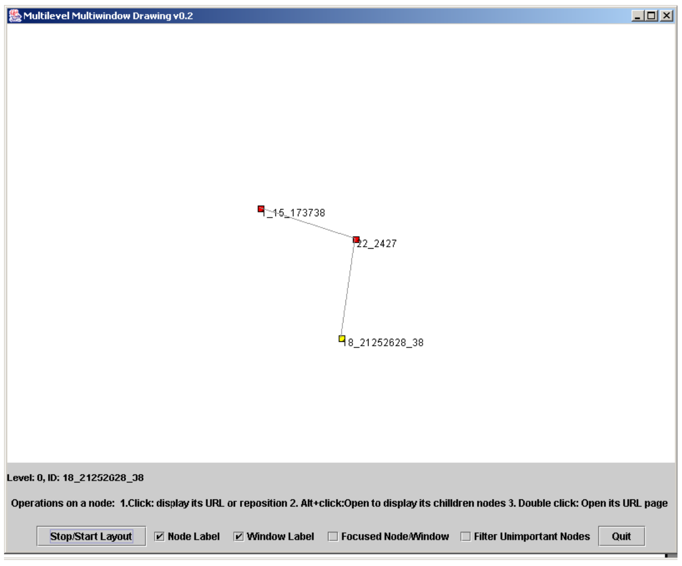
An example of a generated file from a synthetic data set is shown here, which specifies the resulting clusters of a hierarchically clustered graph, where the “ROOT”, 1 and 2 represent the abstract levels of clustering.

```
ROOT; ROOT==0;22_2427--0;1_15_173738

ROOT; ROOT==0;22_2427--0;18_21252628_38

0;1_15_173738==1;1415_173739--1;7_13

0;1_15_173738==1;1415_173739--1;1_6

0;1_15_173738==0;22_2427--1;1415_173739

0;1_15_173738==1;7_13--1;1_6

0;18_21252628_38==1;18_21--1;252628_36

0;18_21252628_38==1;252628_36--0;22_2427

……

1;1_6==2;2--2;17

1;1_6==2;2--1;1415_173739

1;1_6==2;6--1;7_13

1;7_13==2;6--2;7

1;7_13==2;7--2;8

1;7_13==2;7--2;11

1;7_13==2;7--2;12

1;7_13==2;7--2;13

1;7_13==2;8--2;11

1;7_13==2;8--2;9

1;7_13==2;9--2;10

1;7_13==2;10--2;12

……

1;18_21==2;16--2;38

1;18_21==2;17--2;38

1;18_21==2;27--2;17

1;18_21==2;25--2;21

1;18_21==2;21--2;19

1;18_21==2;21--2;18

1;18_21==2;21--2;20

1;18_21==2;20--2;18

1;18_21==2;18--2;19

1;18_21==2;21--2;19

0;22_2427==2;27--2;17

0;22_2427==2;27--2;23

0;22_2427==2;24--2;27
```

```
0;22_2427==2;22--2;27

0;22_2427==2;24--2;26

0;22_2427==2;22--2;24

0;22_2427==2;23--2;24

1;252628_36==2;24--2;26

1;252628_36==2;25--2;26

1;252628_36==2;28--2;26

1;252628_36==2;35--2;34

……
```

Fig.8. the multiple graph clustering result generating from a synthetic data file

A series of coarse graph layouts with different abstract levels are shown below on the basis of the above clustered results.



Level: 0, ID: 18_21252628_38

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

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

Fig.9. The highest abstract level of the graph G^0

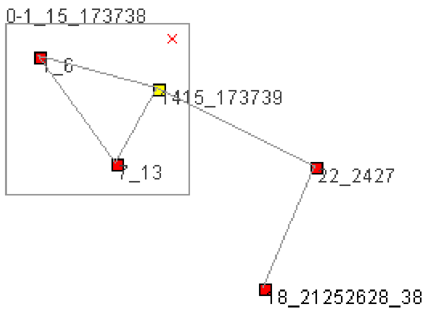


Fig.10. The 0 abstract level of the clustered graph G^0

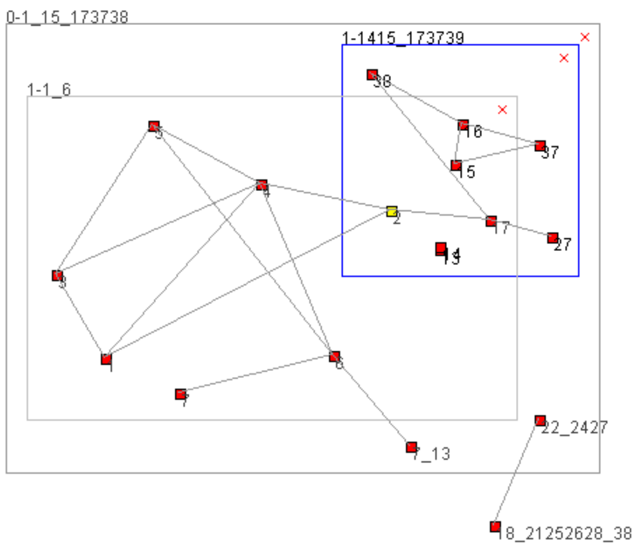


Fig.11. The first abstract level of the clustered graph G^1

In the following, we applied the proposed approach to clustering a web graph. The web graph is generated by using the FCG system. It started with visually navigating http://www.swin.edu.au (Swinburne university Website) with two levels of exploration depth.

The following is part of a file describing the hyperlinks between two Web pages in a Web site(http://www.swin.edu.au), which was generated by the WebCrawler.

```
http://www.swin.edu.au--http://www.swin.edu.au

http://www.swin.edu.au--http://www.swin.edu.au/cwis/sitemap.htm

http://www.swin.edu.au/cwis/sitemap.htm--http://www.swin.edu.au

http://www.swin.edu.au/cwis/sitemap.htm--http://www.swin.edu.au/cwis/sitemap.htm

....

http://www.swin.edu.au/level2.php3?page=The%20University--http://www.swin.edu.au

http://www.swin.edu.au/cwis/sitemap.htm--http://www.swin.edu.au/level2.php3?page=International%20Students

http://www.swin.edu.au/level2.php3?page=International%20Students--http://www.swin.edu.au

……

http://www.swin.edu.au--http://www.swin.edu.au/level2.php3?page=Research

http://www.swin.edu.au--http://www.swin.edu.au/level2.php3?page=Graduates%2FAlumni

http://www.swin.edu.au--http://www.swin.edu.au/cwis/events/

http://www.swin.edu.au/cwis/events/--http://www.swin.edu.au/cwis/events///cwis/events/index.php3

http://www.swin.edu.au/cwis/events/--http://www.swin.edu.au

……
```

Fig.12. The part data file generated from a web site

The following is part of the resulting clusters by using the algorithm, where "ROOT" and 0 represent the abstract level of the clusters.

```
+ROOT;+ROOT==0;http://www.swin.edu.au--0;http://www.swin.edu.au/cwis/events/

+ROOT;+ROOT==0;http://www.swin.edu.au--0;http://www.swin.edu.au/cwis/sitemap.htm

……

0;http://pandoraplus.swin.edu.au/olae/adminforms/forms_display.cfm?type=feedback&cs_code=su94==1;http://pandoraplus.swin.edu.a
u/olae/adminforms/forms_display.cfm?type=feedback&cs_code=su94--0;http://www.swin.edu.au/cwis/events/

0;http://pandoraplus.swin.edu.au/olae/adminforms/forms_display.cfm?type=feedback&cs_code=su94==1;http://pandoraplus.swin.edu.a
u/olae/adminforms/forms_display.cfm?type=feedback&cs_code=su94--0;http://www.swin.edu.au/cwis/sitemap.htm

……
```

Fig.13. The part data file outputting from the clustering algorithm

The following layouts are based on the above results. Although all nodes and edges have been clustered into corresponding clusters using the clustering algorithm, the currently displaying clusters on the screen are depend entirely on the user exploration. If a user chooses to open a higher abstract level of clustered nodes, then its included nodes and their associated edges, that is, the next abstract level graph, will be visible.

5. Conclusion

In this paper, we have presented the algorithms for clustering graphs, along with a system for visualization of Web sites. As opposed to existing approaches that suffer from the limitation of the messy layouts of large graphs, our approach was designed overcome this difficulty in a stepwise and refinement way by using clustering graphs. In particular, we employ a recently proposed algorithm that is able to find good clusters by optimizing the objective function. A prototype called FCG has been implemented to demonstrate the performance of our approach. The future work will include the usability test of this system.

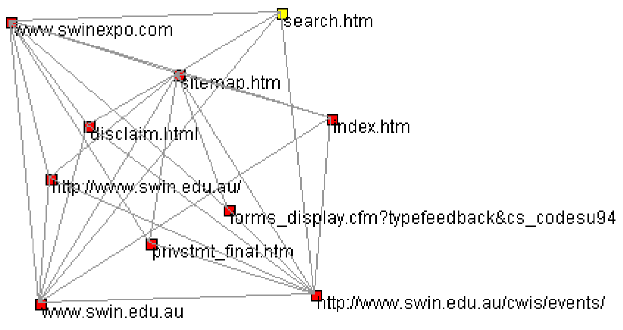


Fig.14. A web graph

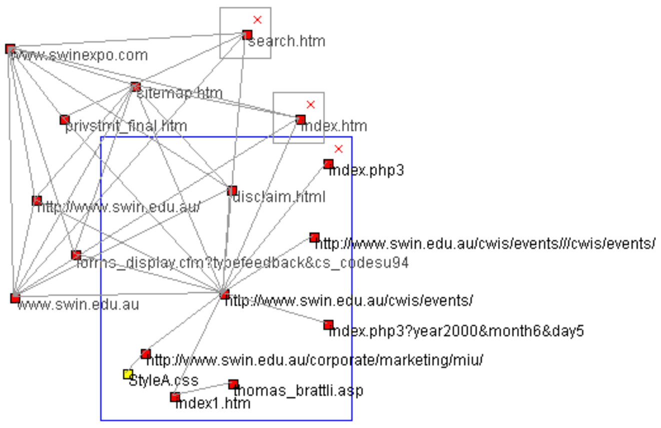

www.swinexpo.com
search.htm
sitemap.htm
privstmt_final.htm
index.htm
index.php3
http://www.swin.edu.au/
disclaim.html
http://www.swin.edu.au/cwis/events///cwis/events/
forms_display.cfm?type=feedback&cs_codesu94
www.swin.edu.au
http://www.swin.edu.au/cwis/events/
index.php3?year2000&month6&day5
http://www.swin.edu.au/corporate/marketing/miu/
StyleA.css
thomas_brattli.asp
index1.htm

Fig.15. A web graph with the user opening the clustered node called "events"

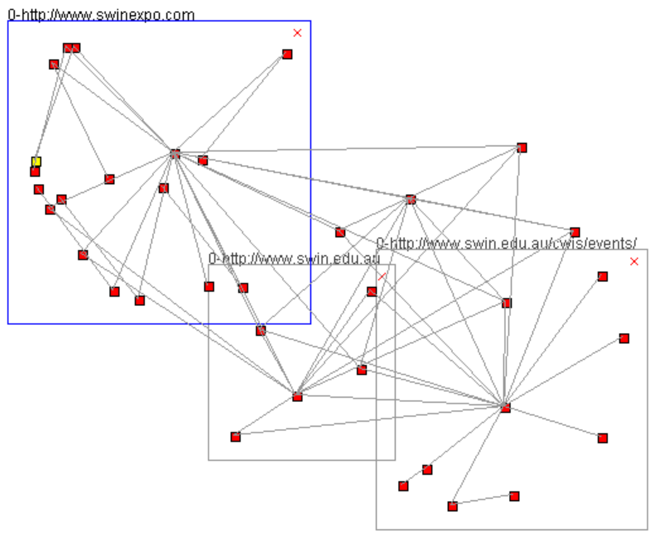

0-http://www.swinexpo.com
0-http://www.swin.edu.au/cwis/events/
0-http://www.swin.edu.au

Fig.16. A web graph with the user interaction

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.

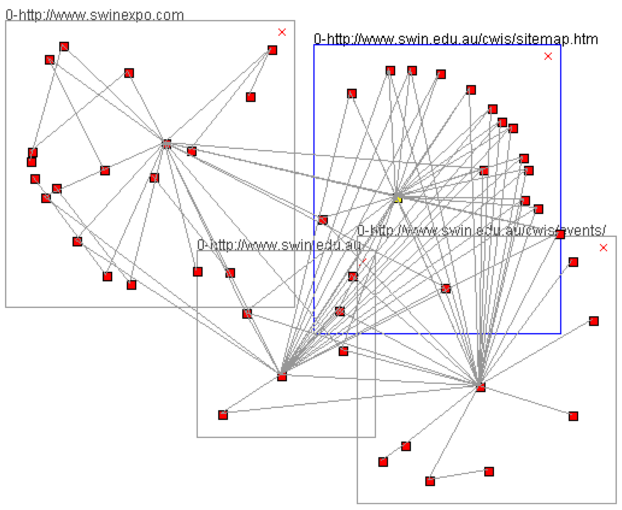

0-http://www.swinexpo.com
0-http://www.swin.edu.au/cwis/sitemap.htm
0-http://www.swin.edu.au/cwis/events/
0-http://www.swin.edu.au

Fig.15. A web graph with more abstract nodes opened

[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] I. Dhillon, Y.Guan, and B. Kulis," A fast kernel-based multilevel algorithm for graph clustering. In Proceedings of the International Conference on KDD, 2005.

[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.

[16] S. X. Yu and J. Shi. Multiclass spectral clustering. In International Conference on Computer Vision, 2003.

[17] B. Scholkopf, A. Smola, and K.-R. M¨uller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Computation, vol. 10, pp. 1299–1319, 1998.

[18] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in Proc.10th ACM KDD Conference, 2004.

[19] X. Huang, W.Lai, Clustering graphs for visualization via node similarities, Journal of Visual Languages and Computing, vol. 17 , pp225-253,2006.

[20] X. Huang,W.Lai, et al., A new algorithm for removing node overlapping, Information Sciences. doi:10.1016/j.ins.2007.02.016, 2007.