ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.8, August 2008, pp.1933–1946
DOI: 10.3724/SP.J.1001.2008.01933
© 2008 by *Journal of Software*. All rights reserved.

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

用多层次聚类法完成的大规模关系图的可视化*

黄茂林+，NGUYEN Quang Vinh

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

Large Graph Visualization by Hierarchical Clustering

HUANG Mao-Lin+, NGUYEN Quang Vinh

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

+ Corresponding author: E-mail: maolin@it.uts.edu.au

Huang ML, Nguyen QV. Large graph visualization by hierarchical clustering. ***Journal of Software*****, 2008,19(8): 1933–1946.** http://www.jos.org.cn/1000-9825/19/1933.htm

Abstract: This paper proposes a new technique for visualizing large graphs of several ten thousands of vertices and edges. To achieve a graph abstraction, a hierarchical clustered graph is extracted from a general large graph based on the community structures discovered in the graph. An enclosure geometrical partitioning algorithm is then applied to achieving the space optimization. For graph drawing, it uses a combination of spring-embbeder and circular drawing algorithms that archives the goal of optimization of display space and aesthetical niceness. The paper also discusses an interaction mechanism accompanied with the layout solution. The interaction not only allows users to navigate hierarchically through the entire clustered graph, but also provides a way to navigate multiple clusters concurrently. Animation is also implemented to preserve user mental maps during the interaction.

Key words: graph drawing; information visualization; view navigation; interaction; clustered graph

摘 要: 提出了一种新的大规模图形可视化技术.它可显示含有几万个接点和边的大规模关系图.为了完成对图形的抽象化,一个多层次的聚类图形从原始的大规模关系图中抽取了出来.这种抽取是建立在大规模关系图的内在结构基础上来完成的.一种递规封入式的几何划分算法被应用来完成对几何空间的优化,在具体的制图技术上,使用了一种用力導向布局算法和环形制图法相结合的新方法,从而完成了对显示空间的优化和美学上的优化.同时也讨论了相关的人机交互技术,所采用的人机交互算法不仅能让使用者从上到下层次式地浏览整个聚类图形,同时也能提供多层次聚类图形的并行浏览.动画技术也同时被运用,以保护使用者的精神图不被打乱.

关键词: 图形绘制;信息可视化;场景游览;人机交互;聚类图形
中图法分类号: TP391　　　文献标识码: A

1 Introduction

Graph visualization has been widely used in human-computer interaction. A graph commonly includes a node set and an edge set to represent entities and relationships between entities respectively. Graphs generated in

* Supported by the Australia Research Council under Discovery Research under Grant No.DP0665463
Received 2008-01-22; Accepted 2008-04-18

real-world applications could be very large with thousands or perhaps millions of nodes, such as citation and collaboration networks and the World Wide Web (WWW). As the result of rapid increasing of the size in networks, the large scale visualization has become one of the hottest topics in Information Visualization. The question about how to comprehensively display large graphs on the screen becomes the key issue in graph visualization. However, the display of large graphs can decrease significantly the performance of a visualization technique which normally performs well on small or medium size of datasets. Large graph visualization usually suffers from poor running time and the limitation of display space. In addition, the issue of "view-ability" and usability also arises because it will be almost impossible to discern between nodes and edges when a dataset of thousands of items are displayed[1].

It seems that classical graph models with a simple node-link diagram tend to be inadequate for large scale visualization with several thousands of items. The lack of formal hierarchical structures in real world applications could limit the conveying and perception of the complicated information. Figure 1 shows an example of the graph visualization of a WWW site which illustrates two typical major problems:

- Too many nodes (pages) to be displayed and the layout of such a large geometrical area could not be fitted in one single screen
- The layout of the graph has inefficient utilization of display space with many unused areas in the display.

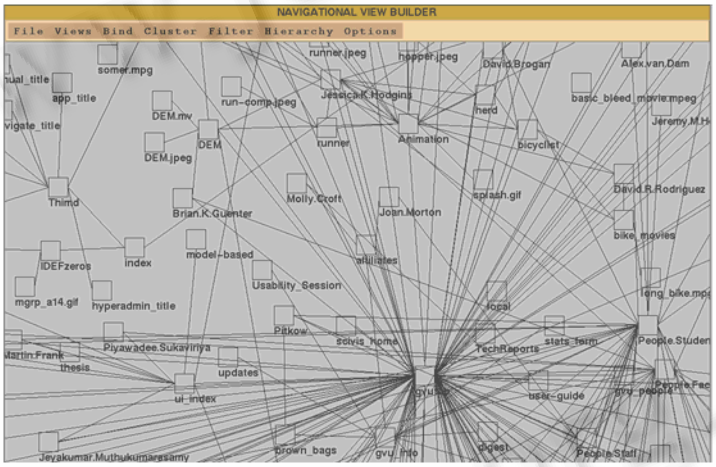


Fig.1 An example of a large graph visualization using the classic virtual-page technique

To address the first problem, a well established new graph model to accommodate with the visualization of large graphs is required. We believe that one way to deal with the display of large graphs is to provide users with a certain degree of Graph Abstract. That is to filter out some details of the graph drawing which is assumed at a time the viewer is not interested, while the overall structure of the graph drawing is maintained for navigation.

Among several available graph visualization approaches, we believe that the use of clustered graph is a better option for graph abstraction. Therefore, a good visualization system for very large graphs should be a combination of three components including graph drawing, graph clustering and interaction[2,3]. Visualization of clustered graphs such as the ones in Refs.[4,5] is one of an excellent approaches to deal with large graphs through the graph abstraction. A clustered graph can be extracted from a general graphs by partitioning recursively the graph into a hierarchy of sub- graphs, so that it simplifies the complex structure of the large graph for easy interpretation, perception and navigation of large information spaces.

To solve the second problem mentioned above, we need to optimize layout algorithms to maximize the utilization of display screen by allowing more nodes to be displayed. The research from Ware[6] shows that more information can be displayed on very high-resolution and large screen, but it does not necessarily provide very much more information into the brain. This is because the conventional monitor covers only 5–10% of visual field in the normal condition, but it uses as much as 50% of brain pixels[7]. The study also shows that the uniquely stimulated brain pixels peak at the width of a normal monitor view, and it is effective (but not critical) to increase the number of pixels for the normal desktop to reach the limit of the brain pixels. Therefore, investigation of optimized visual abstraction (clustering) techniques that could provide viewers with more comprehensive views of the large graphs becomes important.

2 Related Work

Large graphs visualization has recently received a lot of attention from researchers in both information visualization and graph drawing communities. Although some newly available techniques such as techniques found at[3,8–12], are quite capable of visualizing large graphs of thousands to hundred thousands of nodes and edges, visualization of large graphs, is still one of the open topics in information visualization.

Harel and Koren[8] described a technique to draw a graph that used high-dimensional embedding and then projected it onto a 2D plane. Although this technique is very fast and is capable of exhibiting graph in various dimensions with some good navigational ability, it is more suitable for visualizing mesh-graphs rather than tree-like graphs or clustered graphs. One of the good approaches for handling large graphs is to use multi-scale visualization[11,13,14]. This approach typically applies a force-directed algorithm to draw large graphs using multi-scale scheme in which they try to beautify the coarsest-scale representation. Techniques in this approach aim to improve the processing speed while maintaining the graph niceness. A good visualization of large graphs can also be archived by using multilevel techniques[9,15]. In short, these techniques improve the visual appearance of the visualization by defining different levels for a structure so that they can present the graphs using an optimal algorithm at each level.

Although the above techniques are quite capable of visualizing large graphs, the space-efficiency is not considered in the visualization which could limit the amount of information to be visualized on the screen at a time. Fekete, *et al*.[16]. presented a space-efficient visualization of graph using a modification of the well-known Tree-Maps[17]. Technically, the authors used Tree-Maps to display the tree structure of graph and used explicit link curves to present the other links. This technique was optimal in term of using display space and it is quite useful for visualizing structures that the underlying trees have some meaning. However, it did not perform well in general graphs and clustered graphs because the link curves might cause unnatural look of the graphs. Some preliminary works have been carried out and from which two tree visualization techniques *Space-Optimized Tree* (or *SO-Tree*)[20] and *EncCon Tree*[19] have been developed that can quickly display large trees with maximized utilization of display space. However, these solutions are only suitable for trees (hierarchical structures).

This paper proposes a new technique for visualizing very large general graphs. Our proposed technique is very similar to the framework of Tulip[3] which consists of three components: *graph clustering*, *graph layout* and *interaction*. We first use a new clustering algorithm to partition the complete graph into abstract clusters for achieving the view abstraction; that aims to reduce the visual complexity of the graph layout, and enhance the comprehension and understanding of the graph.

The clustered graph is then visualized using a new space-efficient layout technique which is a combination of different layout algorithms. This geometrical optimization of graph layout allows more data items (and clusters) to be displayed within a limited screen resolution. Our visualization provides viewers with not only an abstract view of the entire graph but also an interaction technique for the navigation of large graphs. The navigation method allows users to browse hierarchically through the clustered graph and navigate across a number of selected clusters. All the interactions are accommodated with animation to preserve user mental maps during the navigation.

3 The Architecture of the Visualization

Our model for visualizing large graphs includes several processes which are illustrated at Fig.2. There are two major phases involved in this model including the *clustering analysis* and the *user interface* phases. The two phases operate independently.

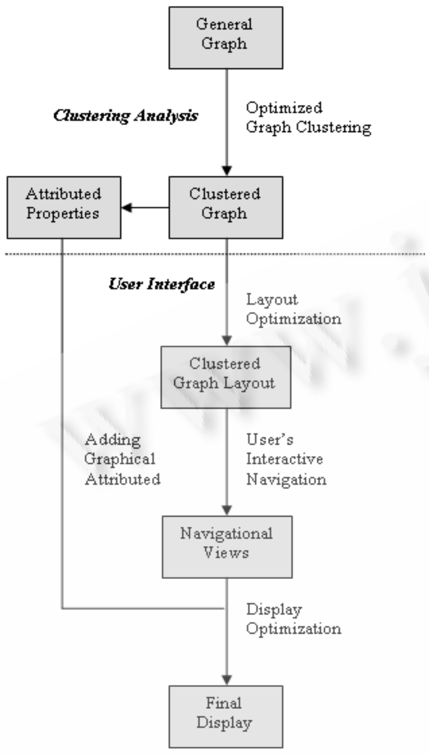


Fig.2 A architectural model for visualizing large graphs

The *clustering analysis* phase is responsible for analyzing a large graph and partitioning it into a clustered graph based on discovered internal communities. In short, the clustering algorithm recursively divides the graphs into smaller sub-graphs based on the density of connection within and between subgroups. Although this process does not require a very fast algorithm and it can operate independently, the computational cost of clustering algorithms should be controlled with running time of $O(n^2)$ or better on a sparse graph to ensure its capability of handling hundreds thousands of items within a few hours using a ordinary personal computer. The clustering process also extracts attributed properties for nodes, edges and relations between sub-graphs.

The *user interface* phase is responsible for the visualization and navigation of the clustered graph, including layout optimization, interactive viewing and display optimization. A combination of different layout algorithms is employed which aims to optimize the geometrical space and so that the large graph can be drawn at a normal screen size.

During the navigation of clustered graphs, we allow users to interactively adjust the views to reach an optimal representation of the graph; from where they can obtain the best understanding of the data and structures. This visualization is involved with real time human-computer interaction. Therefore, very fast graph layout and navigation algorithms is required for handling hundred thousands of items within minutes or seconds using a personal computer with limited display space and computational power.

The final display is created through the view navigation and graphical properties. We use rich graphic attributes to assist viewers to quickly identify the domain specific properties associated with data items. We next describe briefly of the clustering and the technical detail of our visualization technique.

4 Graph Clustering

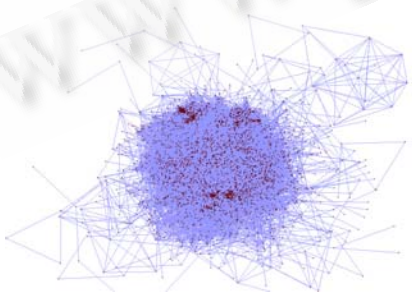
We use a graph clustering method which can quickly discover the community structure embedded in a large graph and divide the graph into densely connected sub-graphs. The graph clustering algorithm partitions the graphs into smaller sub-graphs based on the density of connection within and between subgroups. Although this process does not require a very fast algorithm, the computational cost of clustering algorithms should be controlled with the worst-case running time $O(n^2)$ or faster on a sparse graph to ensure its capability of handling hundreds thousands of items within a few hours using a ordinary personal computer.

Research in graph clustering for large datasets has recently received a lot of effort from researchers. However, the discovery of fast and effective clustering algorithms for handling large graphs is still a big challenge. In fact, the finding of an exact solution for graph clustering is still believed to be an NP-complete problem. Kernighan and Lin[23] and Newman and Girvan[24] have presented their heuristic techniques that can produce quite good solutions for graph clustering. However, their techniques are very slow with the worst–case running time $O(m^2n)$ or $O(n^3)$ on sparse graphs. This makes it almost impossible to partition a graph with more than a few thousands of elements. Newman[18] later proposed a new fast algorithm for detecting community structure in networks. This method runs in worst-case time $O((m+n)n)$ or $O(n^2)$ on a sparse graph. The algorithm is very fast and can handle graphs with hundred thousands of elements. However, its clustering results are not very consistent, especially a poor balance between clusters.

Although the quality or usefulness of an embedded graph drawing algorithm is highly dependent on its application domain, aesthetics is still one of the most important quality factors in graph drawing or graph visualization in which the readability of a graph is measured or justified. The aesthetical criteria for graph drawing can be found from a book on Graph Drawing by Di Battista, *et al*.[25] and the revised version from Ware, *et al*.[26]. Among the aesthetical criteria, the even distribution of vertices and the maximization of display symmetry of a graph structure are two important factors to ensure the quality of graph visualization. These criteria are closely related to the quality of the balance of clusters produced by a clustering algorithm. In order words, we believe that a good clustering algorithm should achieve the goals of balanced clustering in which in each level of the hierarchy the size of the clusters should be about the same. This property helps associated graph drawing method to provide good visualization in term of readability.

This paper used a graph clustering method[27] which can quickly discover the community structure embedded in large graphs and divide the graph into densely connected sub-graphs. The proposed algorithm can not only run fast in time $O((m+n)n)$, but also achieve a consistent partitioning result in which a graph is divided into a set of clusters of the similar size. Although our objective, i.e. keeping clusters balance, is similar to Duncan, *et al*.[28], this clustering algorithm is more general rather than just by using a binary space partition (BSP) clustering.

The balance in size of clusters provides users with a clearer view of the clustered graph and thus it makes it easier to visualize and navigate large graphs. Our balanced clustering technique creates a layout optimization at both global and local levels of the display through the use of enclosure+connection layout technique. This allows more visual items to be displayed within limited screen resolutions and with comprehensive views. The combination between our clustering method and a space-efficient layout technique would enable the visualization of very large general graphs with several thousands of elements. Figure 3 shows an example of a clustering output using the balanced clustering algorithm on a large and highly connected graph.


(a) The original graph with high connections

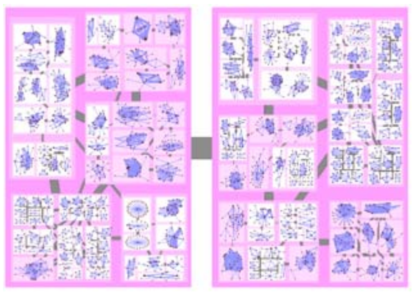

(b) The output clustered graph

Fig.3 An example of a clustering

5 Graph Visualization

We use a new space-efficient visualization technique similarly to *EncCon*[19] to optimize the geometrical space for visualizing large clustered graphs with several thousands of nodes and edges. This technique consists of two components, the space-efficient layout and the interactive navigation. The layout of clustered graph is generated by using a combination of an extended fast enclosure partitioning algorithm, called *Clenccon*, and a number of traditional graph drawing algorithms, including a *spring-embedder* algorithm[21], a circular drawing, and simple layout algorithm, to archive the objectives of space-efficiency, aesthetical niceness and fast computation. Although some existing techniques can use any layout algorithm at each clusters, such as using a *spring-force* algorithm[9,11], in our belief, the choice of a space-efficient enclosure-partitioning algorithm at high levels will be more efficient because it provides more space for displaying information at the limited display.

The *Clenccon* layout algorithm is only applied to those non-leaf sub-graphs in which the space utilization and computational cost are crucial. In other cases, the other layout algorithms, including *spring-embedder* and *circular drawing*, is applied to the calculation of the position for those leaf sub-graphs, which contain a small number of nodes in which the space utilization issue becomes less important and, therefore, the aesthetic niceness and flexibility issues need to be further considered. The use of a particular layout algorithm depends on the nature of the leaf sub-graphs. Our system also displays a high-level node-link diagram to present the overall clustering structure explicitly (see examples at Figs. 5 and 6).

Our *Clenccon* layout algorithm inherits essentially the advantage of space-filling techniques[17,19] that utilize display space by using area division for the partitioning of sub-trees and nodes. Note that the issue of space utilization becomes significantly important when visualizing large graphs with thousands or even hundred thousands of nodes and edges because of the limitation of screen pixels. It is similar to *EncCon*[19] that uses a rectangular division method for recursively positioning the nodes hierarchically. This property aims to provide users with a more straightforward way to perceive the visualization and ensures the efficient use of display space. However, our new technique is applied for clustered graphs rather than simple tree structures. Therefore, the algorithm takes the connectivity property between sibling nodes into its partitioning process. We now describe the technical detail of our layout and navigation algorithms.

5.1 Layout algorithm

Our layout algorithm is responsible for positioning of all nodes in a given clustered graph in a two-dimensional geometrical space, including a vertex subset $\{v_1,v_2,\ldots,v_n\}$ in V and a cluster subset $\{v'_1,v'_2,\ldots,v'_n\}$ in V'. A clustered graph $C=\{G,T\}$ is derived by a general graph $G=\{V,E\}$ and a cluster tree T whose leaves are in V. Each cluster $v'_i = C'$ is a sub clustered graph, contains a subset of V given by the leaves of the sub-tree T' rooted at r'. The root r' of the sub-tree T' is also called a super-node. The super nodes are not displayed in our visualization but they are used for partitioning process of calculating the local region for sub clustered graph. For the partitioning of clustered graph C, We define a virtual tree consisting of a set of super-nodes for area division. We define a super-node $r(v'_i)$ *for each cluster* $v'_{i.}$ Further description of the clustered graph can be found at[5].

The layout algorithm is a combination of two algorithms: 1) *Clenccon* - a fast area division algorithm and 2) graph drawing algorithms, including a *spring-embedder* algorithm, a *circular drawing* and a simple algorithm to lay out a very small number of nodes. Each cluster v'_i is bounded by a rectangular local region $R(v'_i)$ centered at super-node $r(v'_i)$ and the drawing of the corresponding sub-clustered-graph $G(v'_i)$ is restricted to be inside the geometrical area of $R(v'_i)$. Therefore, the local region $R(v'_i)$ of cluster v'_i is the sum of the rectangular areas assigned to its children. The position of the super-node $r(v'_i)$ of v'_i is at the centre of the rectangle defined by $R(v'_i)$. The position of leaf nodes is defined by either the *spring embedder*, the *circular drawing* or the simple layout algorithms.

We first assign the entire rectangular display area as the local region to the clustered graph *C*. We then recursively partition the local regions for every sub-clusters until all the clusters are reached.

We assign a weight $w(v')$ to each vertex v' for the calculation of the local region $R(v')$ of the vertex. Although the weight of each vertex can be associated with its property, all the leaf vertices in our experiments have the same weight. Suppose that the rectangular local region $R(v')$ for cluster v is drawn, we then need to calculate the local regions $\{R(v'_{l+1}),R(v'_{l+2}),\dots,R(v'_{l+k})\}$ for its sub-clusters $\{v'_{l+1},v'_{l+2},\dots,v'_{l+k}\}$. The partitioning ensures that the area of each rectangle $R(v'_{l+i})$ is proportional to the weight $w(v'_{l+i})$ of the cluster v'_{l+i}. The calculation of $w(v')$ of a cluster v' is done recursively from leaves of the cluster tree to the root of the cluster tree. The calculation is done by the following formula:

$$w(v)=w_o+S\sum_{i=1}^{k}w(v_{l+i}),$$

where w_o is the internal weight of cluster v'. Although the internal weight of a cluster can be defined by the cluster's attributed property, we define the internal weight of all clusters to be 1 for all experiments. S is a constant ($0<S<1$), and $w(v'_{l+i})$ is the weight assigned to the *ith* child of cluster v'. The constant S determines the size difference of local regions of all clusters based on the number of descendants of those vertices.

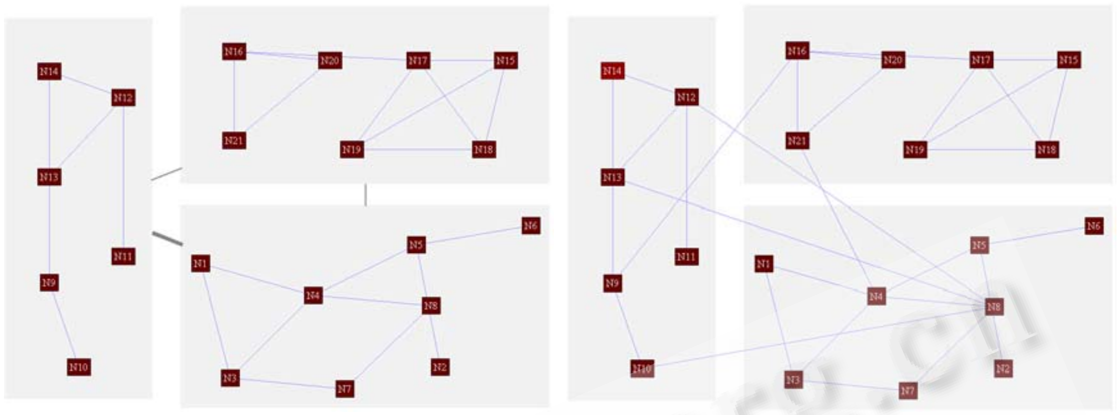
The process of recursive partitioning $R(v')$ into sub-regions $\{R(v'_{l+1}),R(v'_{l+2}),\dots,R(v'_{l+k})\}$ for all its children clusters $\{v'_{l+1},v'_{l+2},\dots,v'_{l+k}\}$ is illustrated as the procedure below:

```
procedure partitioning (Node N)
{
    if all child-nodes of N are leaf-nodes then
    {
        lay out the child-nodes using a graph algorithm;
        scale the layout to fit with rectangular local region;
    }
    else
    {
        lay out the child-nodes using Clenccon algorithm;
        for each non-leaf child-node of N
        {
            partitioning(child-node);
        }
    }
}

procedure Clenccon-layout (Node [] Nodes)
{
    group linked-nodes into subgroups;
    sort the subgroups based on connection and size;
    lay out subgroups using EncCon algorithm;
    for each subgroup
    {
        lay out nodes in subgroup using EncCon algorithm;
    }
}

procedure childnode-layout (Node [] Nodes)
{
    if number of child-nodes < $K1$ then
    {
        lay out the child-nodes using simple algorithm;
    }
    else if number of child-nodes > $K1$ and
          no. edges / no. child-nodes > $K2$ then
    {
          lay out the child-nodes using circular drawing;
    }
    else
    {
        lay out the child-nodes using Spring algorithm;
    }
}
```

where $K1=6$ indicates the number of nodes that is suitable for each algorithm. If there are just a few number of nodes, i.e. less than 10 nodes, a simple node location algorithm can perform well. $K2=5$ indicates the ratio of the number of edges over the number of nodes. Because the force directed algorithms do not usually perform well for a graph whose the number of edges is much larger than number of nodes, we use the circular drawing in this situation.

The detail description of the area partitioning *EncCon* can be found at[19]. Although there are several improved force-directed layout algorithm, the traditional *Spring Embedder*[21] layout algorithm is chosen in our implementation. This is because the algorithm is simple, easy to implement, flexible and perform well in general for a small number of nodes in which a more complicated algorithm is not necessary. The circular drawing is a simple technique but is very effective to show the pattern of relationship for a graph whose the number of edges is much larger than number of nodes. Technically, we place all nodes equally on a circle where relational nodes are located close together so that the pattern of connections can be display more clearly (see Figs.5 and 6).

Figure 4(a) shows an example of partitioning and drawing a small clustered graph using our algorithm. We can see that the algorithm uses the *Clenccon* algorithm to layout three cluster nodes and their inter-relationships to ensure the efficient utilization of space and uses the *spring-embedder* algorithm to draw the sub-graphs within each cluster to achieve the aesthetic niceness and flexibility. Note that the inter-relationships among clusters here are represented by using abstract links. Figure 4(b) shows the same example of the partitioning, but the inter-relationships among clusters are represented by using the original structure of relationships. Figure 5 to Figure 7 illustrate the visualizations of our layout algorithm on various very large datasets. These pictures show clearly the structure of clustered graphs, in which sub-graphs are efficiently partitioned and drawn inside their local regions. All of these pictures use abstract links to represent the inter-relational structures among clusters.



(a) Use the abstract links among clusters (b) Use the original structure of relationships

Fig.4 An example of partitioning and drawing a small clustered graph using our algorithm

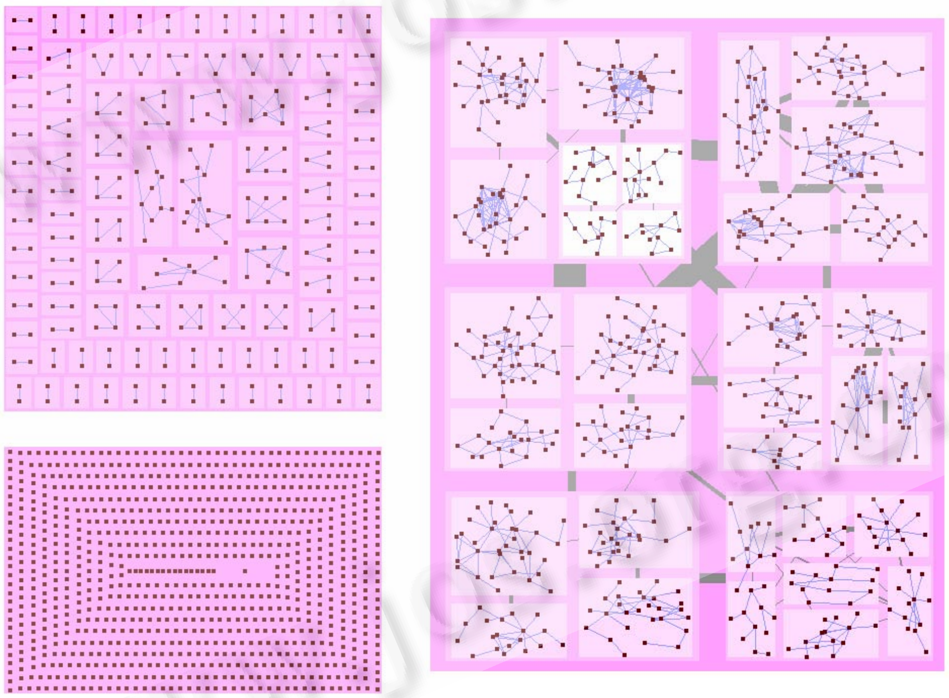


Fig.5 An example of clustering on a large citation dataset with over 2 000 vertices and 4 200 edges of the social network papers

5.2 Navigational views

There is no pure visualization technique that could assist data retrieval without providing users with an associate navigation mechanism in graphic user interface design. In our user interface component, during the navigation we enable users to interactively adjust the views to reach an optimized representation of the graph; from which users can obtain the best understanding of the data and its relational structures they are interesting in.

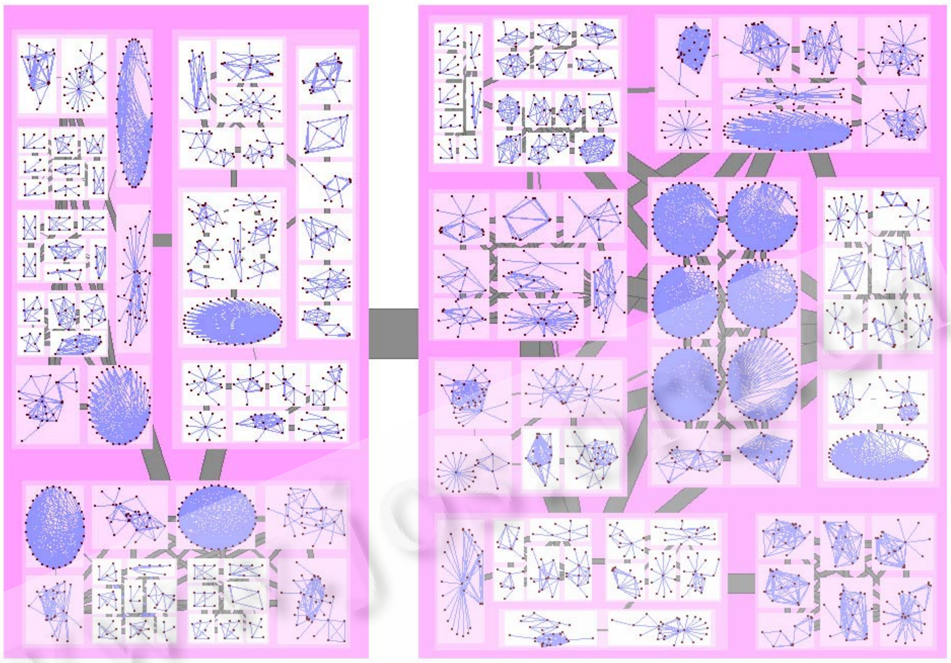


Fig.6 An example of clustering on a protein dataset with very strong relations produced by our algorithm

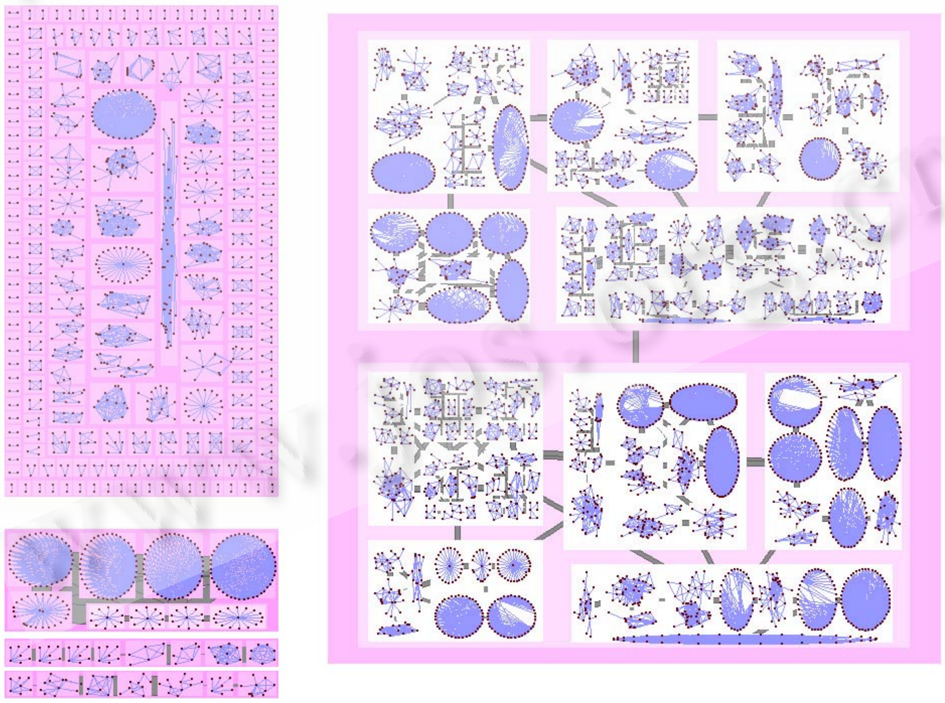


Fig.7 An example of clustering on a protein dataset with our algorithm cluster a protein network with over 15 000 nodes and 40 000 edges

In our prototype, we use a *multiple-views* technique[22], including a *main view* and *context views*, to achieve the *focus+context* view navigation of large clustered graphs. The *main view* shows as much detail of information as possible allowing users to efficiently perform their interaction and visual analysis on this area. The *context views*, which are displayed in the small areas at the left-hand side, are only responsible for displaying a several levels of the contextual information (or the history of navigation) during the navigation. Therefore, a large amount of the detail in these context views is filtered and they remain only the main structures and important landmarks in an abstract manner for guiding the navigation. These simplified structures could preserve user's mental map of where they are, where they come from, and where they have been during the navigation.

The navigational views allow the exploration of data hierarchically as well as across multiple selected clusters to quickly focus on the interest parts of data. Therefore, users can semantically zoom-in one or many areas of interest or sub-clusters while retaining simplified context views. It effectively uses both focus and context views to enable the interactive investigation of very large data. We provide three views to assist viewers to obtain the best understanding of the dataset and its relational structure they are currently exploring. These views are: 1) *a full context view*, 2) *a current context views* and 3) *a main view* (see Figs.8 and 9).

Full context view is displayed as a small panel located at the top-left side of the visualization. This view displays the entire context of a large clustered graph in high-level abstraction with little details. This enables users to always maintain an overall view of the information. The *full-context view* only shows three levels of the clustered graph from the root in a simplified display. This is because too much information displaying at a small area would create the overcrowded and distraction to users. The view updates automatically according to changes occurred in the database. It highlights the context of the current sub-graph the user is interacting, so that users can always identify the where they are and where they have been in a large information space. For example, in Figure 8 we can easily identify the *current-context view* which is on the top-left region of the *full-context view* and the *main-view* which is the right sub-graph inside the top-left region of the *full-context view*.

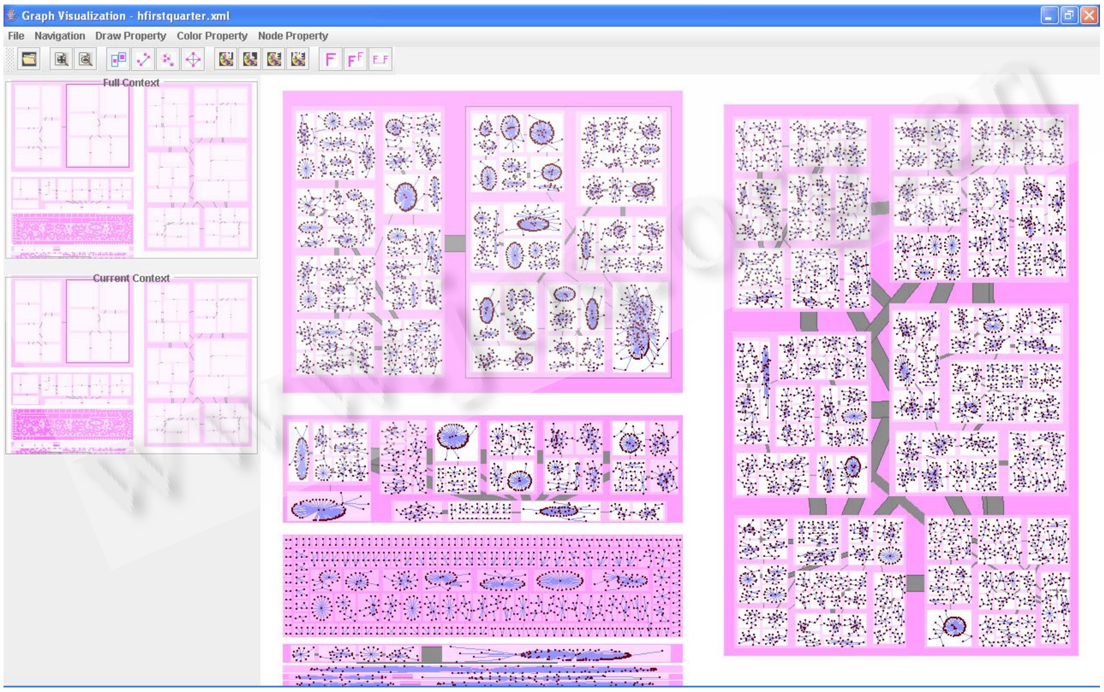

Graph Visualization - hfirstquarter.xml
File Navigation Draw Property Color Property Node Property
Full Context
Current Context

Fig.8 A clustered visualization of a very large protein dataset with three views: 1) a full-context, 2) a current-context view and 3) a main view

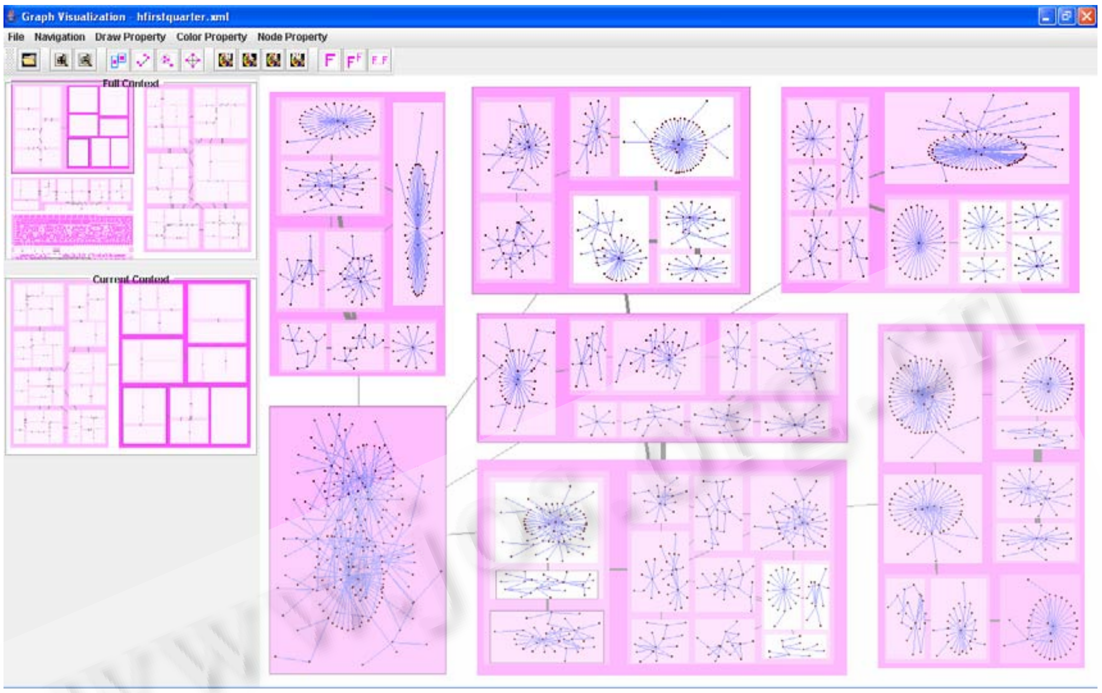

Graph Visualization - hfirstquarter.xml
File Navigation Draw Property Color Property Node Property
Full Context
Current Context

Fig.9 A navigational view of a selected sub-graph of the same data set as Fig.8 produced by our visualization

Current context view is displayed as a small panel locating under the *full-context view* at the left-hand side. This view displays the immediate context or the up-level view of the main view. Similar to the *full-context view*, this view displays up to three levels of clustered graph from the root in an abstraction manner. The *current-context* panel also updates automatically and highlights a focused sub-graph which to be viewed in the main view in details. This enables users to easily identify their focused region at the current context (see Fig.9).

Main view is displayed in a large area of the visualization. It displays the detail of a focused sub-graph. It displays the structure of the clustered graph in an enclosure manner with color brightness. This view also shows the connectivity strength between clusters using the width of edges. Furthermore, the size of nodes in the main-view is automatically rescale based on the number of visible nodes.

5.3 Final display

Some graphical attributes are employed to in final display of clustered graphs to assist viewers to quickly identify the domain specific properties of data and the hierarchical structure of the clustered graph. Colors are employed to assist viewers to quickly identify the hierarchical structure of the clustered graph. In our prototype, the local regions of nodes at different levels are painted with a same color but at different brightness. This drawing property aims to provide a pleased view while retaining the clarity between sub-graphs. Although the use of brightness of colors for background can theoretically apply to several levels of hierarchies, we also apply this drawing property to first four hierarchical levels to avoid the confusion with too many colors.

Width or thickness of the edge is employed to represents the weight of the edge (or the number of connections between two nodes). For example, in Fig.4(b), there are 3 links between the left cluster and bottom-right cluster. However, Fig.4(a) displays only one thick abstract link between these clusters. We can see that edges among leaf nodes are drawn with light-green color and edges among non-leaf nodes (or between two clusters) are drawn with light-gray color. Furthermore, to avoid the overlapping due to the over-thickness of edges, the thickness of an edge is limited. Therefore, if the weight of an edge between two clusters is greater than a limited number in our implementation, the edge is drawn at its limited maximum-width with darker boundary.

Although theoretically the area partitioning algorithm can optimize display space of the entire graph layout, a small portion of display space is reserved as gaps among clusters for showing abstract links between clusters and these abstract links indicate the connectivity strength between clusters.

6 Conclusions

We have presented a new visualization technique that could handle the visualization of very large graphs with up to tens thousands or even hundred thousands of elements on an ordinary Personal Computer. Our method includes two independent steps: clustering and visualization. The clustering step aims to reduce the visual complexity and enhance the comprehension of large graph layouts through the use of visual abstraction. We first discover an optimized community structure in a graph and divide it into densely connected clusters. We then use three levels of visual abstraction: 1) the full context view, 2) the current context view and 3) the main view, to display the large graph. The visualization uses a new space-efficient layout and navigation technique that can visualize effectively the large clustered graphs with several thousands of elements on a limited display space. We use a *multiple view* technique to archive the *focus+context* view navigation. The interaction Animation is also employed to preserve the user mental maps during the interaction.

References:

[1] Herman I, Melancon G, Marshall MS. Graph visualization in information visualization: A survey. IEEE Trans. on Visualization and Computer Graphics, 2000,(6):24−44.

[2] Marshall S. Methods and tools for the visualization and navigation of graphs [Ph.D. Thesis]. Bordeaux: University Bordeaux I, 2001.

[3] Auber D. Tulip: a huge graph visualization framework. In: Mutzel P, Junger M, eds. Graph Drawing Software, Mathematics and Visualization. Springer-Verlag, 2003, 105−126.

[4] Eades P, Feng Q. Multilevel visualization of clustered graphs. In: North SC, ed. Graph Drawing (GD'96). California: Springer, 1996. 101−112.

[5] Feng Q. Algorithms for drawing clustered graphs [Ph.D. Thesis]. Newcastle: University of Newcastle, 1997.

[6] Ware C. Information Visualization: Perception for Design. San Francisco: Morgan Kaufmann Publishers, 2004.

[7] Wilkins AJ. Visual Stress. Oxford: Oxford University Press, 1995.

[8] Harel D, Koren Y. Graph drawing by high-dimensional embedding. Journal of Graph Algorithms and Applications, 2004,8(2): 195−214.

[9] Walshaw C. A multilevel algorithm for force-directed graph drawing. Mathematics Research Report 00/IM/60, University of Greenwich, 2000.

[10] Abello J, van Ham F, Krishnan N. ASK-GraphView: A large scale graph visualization system. IEEE Trans. on Visualization and Computer Graphics, 2006,12(5):669−676.

[11] Harel D, Koren Y. A fast multi-scale method for drawing large graphs. In: Marks J, ed. Graph Drawing. Colonial Williamsburg: Springer-Verlag, 2000. 183−196.

[12] Hachul S, Junger M. An experimental comparison of fast algorithms for drawing general large graphs. In: Healy P, Nikolov NS, ed. Graph Drawing. Limerick: Springer-Verlag, 2005. 235−250.

[13] Auber D, Chiricota Y, Jourdan F, Malancon G. Multiscale visualization of small world networks. In: IEEE Symp. on Information Visualization. Seattle: IEEE Computer Society, 2003. 75−81.

[14] Auber D, Jourdan F. Interactive refinement of multi-scale network clusterings. In: Int'l Conf. on Information Visualization. London: IEEE, 2005. 703−709.

[15] Archambault D, Munzner T, Auber D. TopoLayout: Multi-Level graph layout by topological features. IEEE Trans. on Visualization and Computer Graphics, 2007,13(2):305−317.

[16] Fekete JD, Wang D, Dang N, Aris A, Plaisant C. Overlaying graph links on treemaps. In: IEEE Symp. on Information Visualization –Symp. Poster Compendium. Seattle: IEEE Computer Society, 2003. 82−83.

[17] Johnson B, Shneiderman B. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures. In: Kaufman AE, ed. 1991 IEEE Visualization. IEEE Computer Society, 1991. 284−291.

[18] Newman MEJ. Fast algorithm for detecting community structure in networks. Physical Review E, 2004,69:066133.

[19] Nguyen QV, Huang ML. EncCon: An approach to constructing interactive visualization of large hierarchical data. Information Visualization Journal, 2005,4(1):1−21.

[20] Nguyen QV, Huang ML. Space-Optimized tree: A connection+enclosure approach for the visualization of large hierarchies. Information Visualization Journal, 2003,2(1):3−15.

[21] Eades P. A heuristic for graph drawing. Congressus Numerantium, 1984,42:149−160.

[22] Robert JC. On encouraging multiple views for visualisation. In: Int’l Conf. on Information Visualization. London: IEEE, 1998. 8−14.

[23] Kernighan G, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970,29(2): 291−307.

[24] Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004,69:026113.

[25] di Battista G, Eades P, Tamassia R, Tollis IG. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999.

[26] Ware C, Purchase H, Colpoys L, McGill M. Cognitive measurements of graph aesthetics. Information Visualization Journal, 2002, 1(2):103−110.

[27] Huang ML, Nguyen QV. A fast algorithm for balanced graph clustering. In: Proc. of the Int’l Conf. on Information Visualization. Zürich: IEEE, 2007. 46−52.

[28] Duncan CA, Goodrich MT, Kobourov SG. Balanced aspect ratio trees and their use for drawing large graphs. Journal of Graph Algorithms and Applications, 2000,4(3):19−46.

HUANG Mao-Lin is a senior lecturer and doctoral supervisor at the Faculty of Information Technology, University of Technology, Sydney, Australia. His research areas are graph drawing and information visualization.

NGUYEN Quang Vinh is a Post-Doctoral Fellow at the Faculty of Information Technology, University of Technology, Sydney, Australia. His current research interests include information visualization.