A Fast Algorithm for Balanced Graph Clustering

Mao Lin Huang and Quang Vinh Nguyen
Faculty of Information Technology, University of Technology, Sydney
{maolin@it.uts.edu.au, quvnguye@it.uts.edu.au}

Abstract

Scalability problem is a long-lasting challenge for both information visualization and graph drawing communities. Available graph visualization techniques could perform well for small or medium size graphs but they are rarely able to handle very large and complex graphs. One of effective approach to solve this problem is to employ graph abstraction; that is to hierarchically partitioning the complete graph into a clustered graph. A graph visualization technique is then applied to display the abstract view of this clustered graph with partially displayed detail of one or a few sub-graphs where the user is currently focusing on. This reduces the complexity of display and makes it easier for users to interpret, perceive and navigate the large scale information. In this paper, we propose a graph clustering method which can quickly discover the community structure embedded in large graphs and partition the graph into densely connected sub-graphs. The proposed algorithm can not only run fast, but also achieve a consistent partitioning result in which a graph is divided into a set of clusters of the similar size in terms of their visual complexity and the number of nodes and edges. In addition, we also provide a mechanism to partition very dense graphs in which the number of edges is much larger than the number of nodes.

Keywords--- **graph drawing, clustered graph, information visualization**.

1. Introduction

Graph visualization has been widely used in human-computer interaction. A graph commonly consists of a set of nodes and a set of edges representing entities and relationships between entities respectively. Graphs generated in real Information Visualization applications could be very large with typically thousands or perhaps millions of nodes, such as citation and collaboration networks and the World Wide Web. As the result of rapid increasing of the size in networks, the large scale visualization has been turned into one of the hottest topics in Information Visualization. Therefore, the question about how to comprehensively display large graphs on the screen becomes the key issue in graph visualization.

Although many techniques have been proposed and implemented the size of the datasets is still a key issue in graph visualization. Large graphs can decrease significantly the performance of a visualization technique which normally performs well on small or medium 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 impossible to discern between nodes and edges when a dataset of thousands of vertices and edges are displayed [1].

With the rapid growth of information in size, the classical graph model with a simple node-link diagram tends to be inadequate for large scale visualization with several thousands of items. The lack of formal hierarchical structures in those classical graph layout techniques could limit the conveying and perception of the complicated information. Therefore, a well established new graph model to accommodate the visualization of large graphs is urgently required. Clustered graph and compound graph models [3, 4] are typical new models proposed in late 90's to deal with such large graphs. They partition recursively the graph into the hierarchy of sub-graphs for clustered visualization, which simplifies the complex structures of large graphs through the global abstraction for easy interpretation, perception and navigation of large information spaces (see Figure 1). Next we will briefly review the graph clustering techniques.

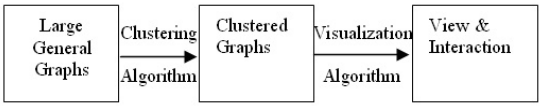

Figure 1. A conceptual model of visualizing large graphs.

1.1. Graph Clustering Techniques

The graph clustering (or graph partitioning) research in computer science has a long history with increasing interests in the study of large and very large networks,

such as citation networks, collaboration networks and even the World Wide Web. These techniques aimed to simplify the scalability and complexity of large graphs for better interpretation, perception and navigation. In general, graph clustering methods can be divided into two categories, including geometry-based clustering and none-geometry-based clustering.

Geometry-Based Clustering – graph partitioning techniques in which a graph is clustered based on the geometric attributes associated with vertices. Graphs that can be appropriately implemented in this type of the clustering are usually called the mesh (or planar) graphs. This type of graphs can usually be drawn on a plane without any edge crossing. Typical techniques in this approach are coordinate bisection, inertial bisection and geometric partitioning. The coordinate bisection is the simplest technique in which a graph is partitioned into two equal sub-graphs by finding the hyper-plane orthogonal to a chosen x- or y-coordinate [5]. On the other hand, the inertial bisection [6] method does not use a fixed x- or y-coordinate axis for the partitioning rather it chooses an axis which runs through the middle points of the graph. The geometric partitioning [7] is a more complicated method in which it searches a vertex separator for dividing the graph into 2 sub-graphs that maintains consistent with similar size. Although geometry-based clustering methods work quite well with mesh graphs, they do not handle other general types of graphs. Therefore, these techniques are not considered in our research because our methods should be able to handle general graphs.

None-Geometry-Based Clustering – graph partitioning techniques in which a graph is divided into sub-graphs based on its connectivity information, rather than its geometric information. Therefore, it has a wider range of applications and it is especially more useful for graph visualization in which most of the graphs do not have geometric property associated with vertices. There are many existing approaches in this category, such as graph growing and greedy clustering, spectral clustering and multilevel clustering. Graph growing and greedy clustering [8, 9] works simply by choosing a starting vertex, and then add other vertices gradually until the partition is big enough. More specifically, at each adding increment, greedy algorithms look for the best vertex in some sense while the graph growing algorithms look for a vertex that the resulting sub-graph grows in a certain way. Spectral clustering does not operate on a graph itself but it uses a mathematical representation of the graph. In other word, it makes use of the spectrum of the similarity matrix of data to cluster a graph. There is a variety of techniques in this approach, including spectral bisection [10], spectral quadrisection and octasection [11], and multilevel spectral bisection [12]. Multilevel partitioning [13, 14, 15, 16] is an alternative approach of multilevel spectral bisection. It transfers partitioning information between levels to improve the partition quality on each level. There are three major steps for implementing this method including coarsening graph, partitioning graph, and projecting and refining graph.

1.2. Balanced Graph Clustering

The graph clustering algorithm divides the graphs into smaller and smaller sub-graphs based on the density of connection within and between subgroups. Although this work does not require a very fast algorithm and it can run on a high-speed workstation, the computational cost of clustering algorithms should be controlled with the worst-case running time $O(n^2)$ on a sparse graph or faster 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 a NP-complete problem. Kernighan and Lin [8] and Newman and Girvan [17] 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 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. This algorithm is very fast and can handle graphs with hundred thousands of elements, its clustering results, however, 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 aesthetics criteria for graph drawing can be found from a book on Graph Drawing by Di Battista et al. [19] and the revised version from Ware et al. [20]. Among the aesthetics criteria, the evenly distribution of vertices and the display maximization of symmetry of graph structure are two important factors to ensure the quality of a graph visualization. These criteria are very much related to the quality of the balance of clusters produced by a clustering algorithm. In order word, 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 proposes a graph clustering method 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. [21], our clustering algorithm is more general case rather than using a binary space partition (BSP) clustering method as [21].

The balanced size of clusters could provide users with a clearer view of the clustered graph and thus it makes easier for users to visualize and navigate large graphs. Our balanced clustering technique also achieves the 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.

2. The Algorithm

A clustered graph $C = \{G, T\}$ consists of a general graph $G = \{V, E\}$ and a cluster tree T, where $T = \{G', r\}$ consisting of a hierarchical graph G' and a root r. A cluster tree T consists of a set of cluster nodes V' and a set of virtual links E', that is used to virtually present the clustering structure. A node group (or subgroup) is defined as a set nodes in C, 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 node n is defined as either a vertex v_i or an abstract cluster v'_i . A cluster $v'_i = C'$ is a sub clustered graph.

To simplify the problem, our clustering algorithm is applied only to those nodes connected with large subgroups. The algorithm first, groups none-connection nodes and nodes those connected with small node subgroups into two large clusters respectively. The clustering process is hierarchically applied through the entire cluster tree from upper level clusters to downer levels until each cluster has a reasonable small number of nodes. Formally, we carry out the following steps in our clustering process:

1. Group all isolated nodes into one node group.
2. Group all small connected subgroups of nodes, i.e. a connected subgroup of nodes with a few links, i.e. fewer than 40 nodes in our implementation, into clusters and merge these clusters into a large cluster.
3. For each large connected subgroup of nodes, apply clustering algorithm to partition it into small highly connected clusters.
4. Repeat step 3 for every cluster if the cluster is still large, i.e. more than 40 nodes.

Our clustering algorithm is a modification of Newman's algorithm [18] which considers the equality of size among clusters during the partitioning. The clustering algorithm is described as below:

1. Start with all N nodes belonging to N groups with a single element.
2. Calculate the increment ΔQ of the "modularity value" Q for all joints between any two connected subgroups.
3. Merge two subgroups into one large group in which the increment ΔQ is the largest when joining two subgroups.
4. Repeatedly step 2 and step 3 until all values of the increment ΔQ are negative when joining any two subgroups.

We next describe the calculation of the increment ΔQ of the modularity value. Similar to Newman's algorithm [18], our clustering algorithm also optimizes the quality function or the "modularity value" Q from [17] to find the best partitioning result. The quality function of joining two groups i and j is defined by formula:

$$Q = \sum_i (e_{ii} - a_i^2) \quad (1)$$

where e_{ij} is the fraction of edges in the graph that connect vertices in group i to vertices in group j, and $a_i = \sum_j e_{ij}$ is the fraction of edges in the graphs that connect from group i to all other groups.

Because of the increase of value Q means the better community structure, the process will find the maximum increment of Q among joining subgroups. Staring at each vertex being itself a subgroup, we then join all subgroups together in pair which has maximum value of the change in quality ΔQ. This process creates a dendrogram tree of the partitioning (or joining of group nodes) and the best partitioning result is location where the value of Q is maximized. From Newman [16], the change in Q when joining two communities is calculated by the formula:

$$\Delta Q = 2(e_{ij} - a_{ij}) \quad (2)$$

Although the optimization of the quality value Q calculated by Equation 3 performs well in some datasets, this technique could create unbalanced partitioning between clustered sub-graphs. This is because that the Newman's method does not consider the weight between merged subgroups. Figure 2 shows an example of a medium size clustered graph produced by Newman's algorithm [18]. The figure clearly shows an unbalanced partitioning result among clusters; the left cluster contains only 4 nodes while the right cluster has approximately 80 nodes, and this large cluster also contains a small bottom sub-cluster with 2 vertices and the large top sub-cluster. The figure also shows that the clustering process needs to iterate 3 times running through the cluster graph to archive a reasonable good partitioning result for a reasonable clear view.

To overcome the problem of unbalanced clustering, we introduce a weight variable w into the calculation of quality increment ΔQ for defining the modularity value Q. This weight variable ensures the balance between the connection strength and the node group size among clusters. Because that our method also optimize the modularity value Q, thus the increment in modularity

value Q for each pair of possible joining subgroups is calculated as below:

$$\Delta Q = 2(e_{ij} - a_{ij}) - wd_{ij} \quad (3)$$

where wd_{ij} defines the change of the unbalance value before and after the joining of two groups. This value indicates the dissimilarity in size among clustered groups from the partitioning process. The value of wd_{ij} is calculated by the formula below:

$$wd_{ij} = \left|wi + wj - aW^*\right| - 0.5\left|wi - aW\right| - 0.5\left|wj - aW\right| \quad (4)$$

where w_i and w_j are weights (or the fraction of number of nodes over the total nodes) of group i and group j respectively, aW^* is the average weight of all node groups after a group merging, and aW is the average weight of all node groups before the group merging.

Figure 3 shows an example of the clustering result when the balancing value among partitioned node groups is considered. The figure shows clearly that the new clustering algorithm provides a much better result in terms of balance, equality, consistence and perception of clusters. The unbalanced clustering problem remained in Newman's method [18] has been well addressed. Note that the width of edges in Figure 3 indicates the connection strength (or the number of edges) between clusters.

2.1. Highly Connected Graph Clustering

Although the above clustering algorithm performs well on general graphs, it often does not perform on very strong connected graphs in which the number of edges is very much larger than the number of nodes. This is because the clustering algorithm treats the very highly connected graph as a cluster without any further partitioning. This result could make it very hard for further analysis especially to those very large graphs. Figure 4 shows an example of very highly connected graph with more than 5,000 nodes and 50,000 edges. This figure illustrates the very low readability of the graph which is almost impossible to identify any useful detail or pattern in the graph except the indication of a very high connected graph.

In order to handle those strong connected graphs, we slightly modify our clustering algorithm that the partitioning process repeat step 2 and step 3 until all values of the increment ΔQ are negative or when joining any two subgroups or when the number of subgroups equals to 2. This modification ensures the highly connected graph is always divided into sub-graphs. Figure 5 shows the visualization of the clustered graph after using the modified algorithm. This figure uses the same dataset as Figure 4. However, the figure illustrates much more clearly the structure of the graphs which makes it very much easier to read and comprehend the information. Furthermore, this clustering result easily enables the navigation through the graph using a visualization and interaction technique.

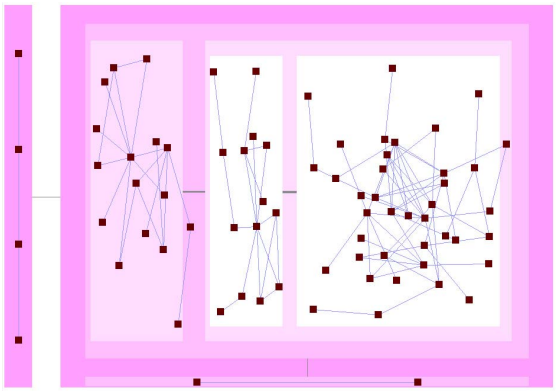

Figure 2. An example of a clustering produced by [18] on a medium graph in which very unequal sub-graphs are produced.

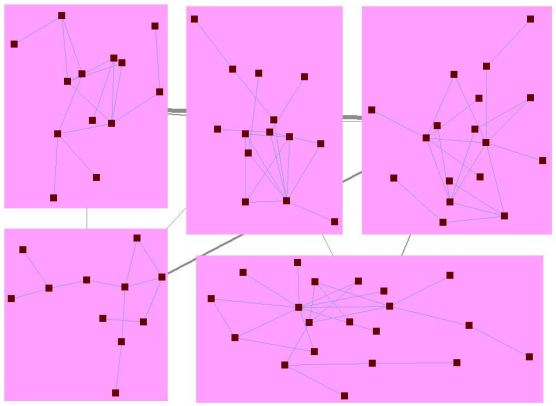

Figure 3. An example of a clustering produced by our algorithm using the same dataset as used in Error! Reference source not found. **in which clustered sub-graphs have quite similar size.**

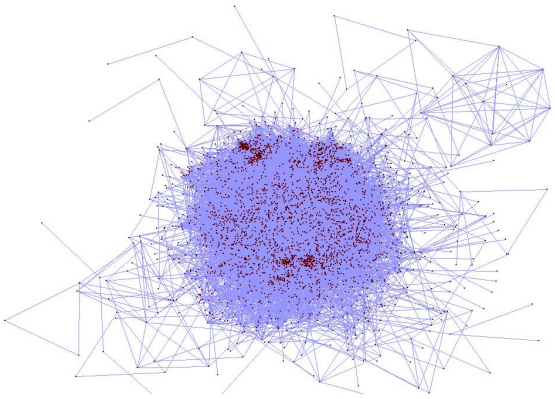

Figure 4. An example of a very highly connected graph which the original clustering algorithm cannot partition.

2.2. Computational Complexity

Newman has proofed in his method [18] that the computational complexity of his clustering method by optimizing the modularity value Q is $O((m+n)n)$ or $O(n^2)$ on a sparse graph. Our technique is also based on optimization of the modularity value Q. It is technically a modification the Equation 3 by adding the weight dissimilarity from Equation 4. Obviously, the calculation of the weight dissimilarity from Equation 4 is constant. Therefore, the computational complexity of our clustering algorithm is as same as Newman's algorithm [18] which is $O((m+n)n)$ or $O(n^2)$ on a sparse graph (where m is the number of edges and n is the number of nodes).

Our clustering process also includes a grouping step for all unconnected nodes and small connected node groups. This process is obviously linear. Therefore, the worst case time spent on merging groups and partitioning is within $O((m+n)n)$ or $O(n^2)$ on a sparse graph.

3. Applications

We have applied our clustering technique to several large datasets in the domains of citation and protein networks. In citation application, citation graphs are extracted from the social network of research papers and the terrorism network. In protein experiment, two very large graphs are extracted with large number of edges.

Figure 4 shows the example of a clustering of a citation graph of research papers extracted from the social network with over 2000 vertices and 4200 edges. The vertex presents a paper while a edge represents a citation. The grouping process uses our grouping algorithm while the partition of large related group at the right side uses Newman and Girvan's clustering algorithm [17]. The visualization immediately classifies the papers from social network into three node groups, including papers with no citations at the bottom-left side, papers with a few citations at the top-left side, and papers with a large number of citations at the right side. The run time for partitioning this dataset is a few hours on a PC with Pentium IV 1.6Ghz and 256Mbs RAM. In addition to the fast running time, the visualization also shows that the partitioning result is quite hard to follow with several clusters of unequal sizes.

Figure 5 shows a clustered visualization produced by our partitioning algorithm, performing on the same dataset as implemented in Figure 4. The running time for clustering the same dataset by using our new method is only a few seconds. The three citation groups shown in this figure are similar to those in Figure 4 because we use the same grouping algorithm. Nevertheless, Figure 5 illustrates a much better clustering than one produced in Figure 4. The clusters and sub-clusters generated in Figure 5 are quite balanced. This property could improve the perception of viewers when visualizing and navigating a very large dataset. For example, we can quickly identify graph structures from the right sub-

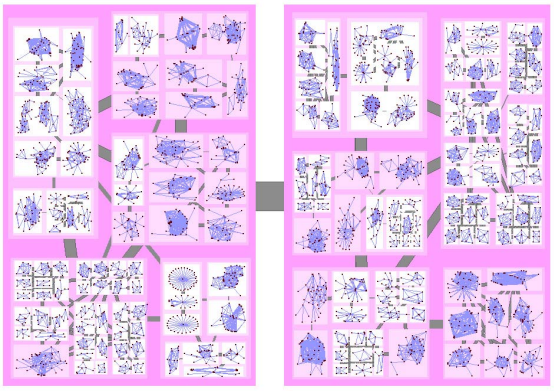

Figure 5. An example of a clustering produced by our algorithm using the same dataset as used in Figure 4.

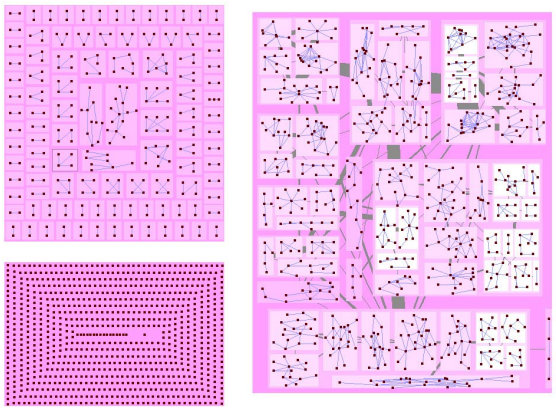

Figure 4. An example of Newman and Girvan's clustering algorithm [15] on a citation network of social network papers.

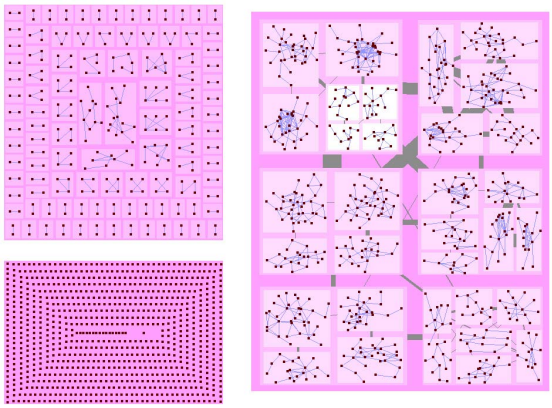

Figure 5. An example of our clustering algorithm on same dataset of the citation network with over 2000 vertices and 4200 edges as Figure 4.

graphs where the clusters at the top have much stronger connection than those at the bottom.

Figure 6 shows another example of applying our clustering algorithm in a terrorism related citation network. This graph contains 430 vertices and 590 edges. The vertex here presents a paper while an edge represents a citation. Similar to the previous visualization of social network, the visualization immediately classifies the papers into three citation groups in the terrorism network, including papers with no citations displayed at the top-right side, papers with a few citations displayed at the left side and papers with high connectivity displayed at the bottom-right side. The highly connected large node group is further partitioned into five sub-clusters. The visualization also partitions the clusters into similar size and we can see that the interconnection strength among the clusters is much weaker than the connections within each cluster.

Figure 7 illustrates an example of using our algorithm cluster a protein network with over 10,500 nodes and 27,500 edges. The vertices present the proteins while the edges represent their relations. The graph is hierarchically partitioned into clusters and sub-clusters along the cluster graph T from top root down to 4^{th} level. Our visualization could display quite clear structures and properties of the protein networks. This enables users to efficiently view, navigate and analyze the protein structure by using a space-efficient visual navigation technique which is a generalization of the hierarchical visualization *EncCon* algorithm [22]. For example, the figure shows that two largest clusters at the right and top-left side contain most of the vertices and links. The right cluster is much highly connected than the top-left cluster that with two sub-clusters. Nodes in the top-left cluster tend to have more leaves in comparison to the right cluster.

Figure 8 illustrates our clustering result on another very large protein graph with approximately 19,000 nodes and 75,000 edges. This figure shows a clear picture of the protein network that contains three groups of different properties. The largest cluster is at the right-hand side which shows a strong connectivity among its sub-clusters in the network. The running time for partitioning this dataset is about a few hours by using the same personal computer we used for other citation examples.

Conclusions

We have presented a graph clustering technique for discovering and representing the community structures in large datasets. Our method divides a graph into densely connected sub-graphs (clusters). Our clustering algorithm is a modification of Newman's clustering algorithm. However, our new method can not only run fast, but also partition a graph into similar size clusters. It achieves the objective of providing a clearer view for effective view comprehension and navigation of large graphs. We have also presented four experiments of

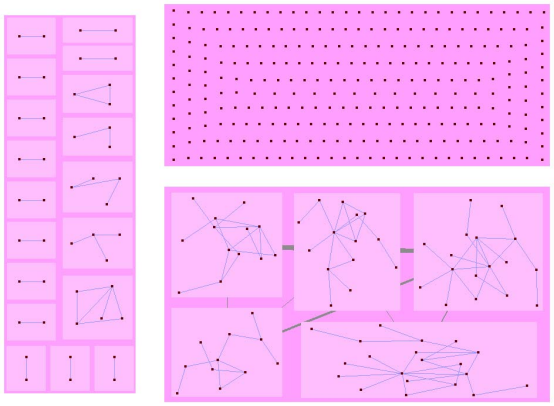

Figure 6. . An example of our clustering algorithm applying on a citation network of terrorism related papers.

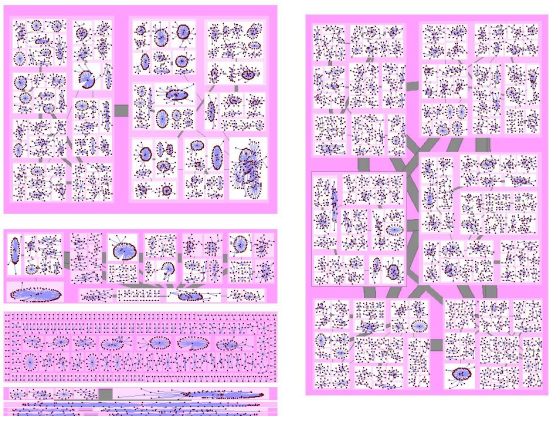

Figure 7. An example of our clustering algorithm applying on a dataset of protein networks with over 10,500 nodes and 27,500 edges.

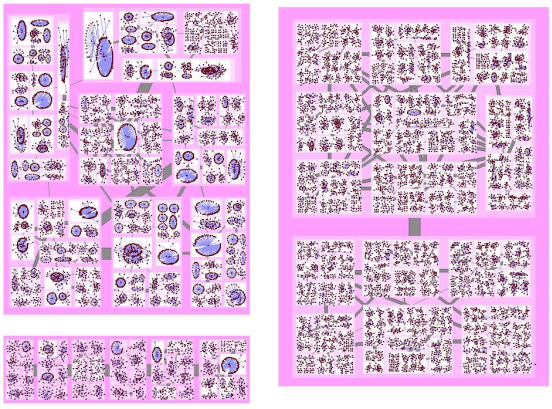

Figure 8. An example of our clustering algorithm applying on a dataset of protein networks with approximately 19,000 nodes and 75,000 edges.

clustering and visualizing large graphs in citation and protein networks. The experiments showed that our partitioning algorithm performs quite well on these datasets in comparison with the other existing method.

Now we are investigating a faster clustering algorithm to further reduce the computational time for clustering huge graphs with hundred thousands or even millions of items. Particularly we are investigating algorithms that could cluster huge graphs with the worst-case time of *O(nlog(n))*. In addition, the further optimization of the balance issue will also be listed in our research schedule.

Acknowledgements

This project is supported by Australia ARC Discovery research grant #DP0665463.

References

[1] Herman, I., Melancon, G. and Marshall, M.S. Graph visualization in information visualization: a survey. IEEE Transactions on Visualization and Computer Graphics, 6: 24-44, 2000.

[2] Eades, P. and Feng, Q. Multilevel visualization of clustered graphs. In Graph Drawing (GD'96), Springer, pages 101-112, 1996.

[3] Eades, P. and Huang, M. L. Navigating clustered graphs using force-directed methods. Journal of Computational and Graphical Statistics, 4(3): 157-181, 2000.

[4] Sugiyama, K. and Misue, K. Visualization of structural information: Automatic drawing of compound digraphs. IEEE Transactions on Systems, Man, and Cybernetics, 21(4): 867-892, 1991.

[5] Elsner, U. Graph partitioning: a survey. Technische Universitat Chemnitz, December 1997.

[6] Pothen, A. Graph partitioning algorithms with applications to scientific computing. In Parallel Numerical Algorithms, Kluwer, pages 323-368, 1997.

[7] Miller, G. L., Teng, S. H., Thurston, W. and Vavasis, S. A. Geometric separators for finite element meshes. SIAM Journal on Scientific Computing, 19(2): 364-386, march 1998.

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

[9] Chartrand, G. and Oellermann, O. R. Applied and Algorithmic Graph Theory. International series in pure and applied mathematics. McGraw-Hill, New York, 1993.

[10] Golub, G. H. and Loan, C. F., Matrix Computations. Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.

[11] Hendrickson, B. and Leland, R. An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing, 16(2): 452-469, 1995

[12] Barnard, S. T. PMRSB: Parallel multilevel recursive spectral bisection. In the 1995 ACM/IEEE Supercomputing Conference. ACM / IEEE, 1995, CD-ROM.

[13] Hendrickson, B. and Leland, R. A multilevel algorithm for partitioning graphs. In Supercomputing '95, ACM Press, 1995, CD-ROM.

[14] Karypis, G. and Kumar, V. Multilevel algorithms for multi-constraint graph partitioning. In Supercomputing '98, ACM SIGARCH and IEEE, 1998, CD-ROM.

[15] Delest, M., Fedou, J. M., and Melancon, G. A quality measure for multi-level community structure. In SYNASC 8th Internatinal Conference, 2006.

[16] Mancoridis, S., Mitchell, B. S., Chen, Y. and Gansner, E. R. Bunch: a clustering tool for the recovery and maintenance of software system structures. In IEEE Int'l Conf. Software Maintenance, pages. 50-59, 1999.

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

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

[19] Di Battista G, Eades P, Tamassia R, and Tollis IG. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall: New Jersey, 1999; 397pp.

[20] Ware, C., Purchase, H., Colpoys, L. and McGill, M. Cognitive measurements of graph aesthetics. Information Visualization, Palgrave, 1(2): 103-110, 2002.

[21] Duncan, C. A., Goodrich, M. T. and Kobourov, S. G. Balanced aspect ratio trees and their use for drawing large graphs. Journal of Graph Algorithms and Applications, 4(3), pp. 19-46, 2000

[22] Nguyen, Q. V. and Huang, M. L. EncCon: an approach to constructing interactive visualization of large hierarchical data. Information Visualization Journal, Palgrave, 4(1): 1-21, 2005.