Chenhui Li · George Baciu · Yunzhe Wang

Module-based visualization of large-scale graph network data

Received: 30 November 2015 / Revised: 3 June 2016 / Accepted: 7 June 2016

Abstract The efficient visualization of dynamic network structures has become a dominant problem in many big data applications, such as large network analytics, traffic management, resource allocation graphs, logistics, social networks, and large document repositories. In this paper, we present a large-graph visualization system called *ModuleGraph*. ModuleGraph is a scalable representation of graph structures by treating a graph as a set of modules. The main objectives are: (1) to detect graph patterns in the visualization of large-graph data, and (2) to emphasize the interconnecting structures to detect potential interactions between local modules. Our first contribution is a *hybrid modularity measure*. This measure partitions the cohesion of the graph at various levels of details. We aggregate clusters of nodes and edges into several modules to reduce the overlap between graph components on a 2D display. Our second contribution is a k-clustering method that can flexibly detect the local patterns or substructures in modules. Patterns of modules are preserved by the ModuleGraph system to avoid information loss, while sub-graphs are clustered as a single node. Our experiments show that this method can efficiently support large-scale social and spatial network visualization.

Keywords Network visualization · Module grouping · Graph drawing · Information visualization · Community detection

1 Introduction

Large network analysis and visualization have become an important challenge with the rapid increase in network data, such as data generated from social networks, modern transportation networks, document referencing, and academic citations. Graph connectivity patterns are of interest in network data analytics for detecting unexpected connectivity and clustering features. This would allow us to discover and isolate points of interest in massive networks which are otherwise difficult to spot visually on current displays. However, the effect of large network visualization is often limited to the size of the display (Zinsmaier et al. 2012).

A practical approach to displaying a large-scale network is partitioning the network according to well-defined domain-dependent attributes. However, graph visualization in the presence of incomplete information is an open challenge and applications in this area can be found in abundance. To better visualize and

An earlier version of the paper was presented at the SIGGRAPH Asia 2015 Symposium on Visualization in High Performance Computing, 2–5 November, Kobe, Japan, 2015.

C. Li · G. Baciu (✉) · Y. Wang
Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, People's Republic of China
E-mail: csgeorge@comp.polyu.edu.hk

understand patterns in large-graph discovery, we focus on the representation of local patterns. This is a critical step in deciding the structural components of a graph visualization.

In this paper, we present the *ModuleGraph*, a flexible graph visualization framework that aggregates the components of large graphs into modules to help users to address graph-size limitations and effectively gain insights into attributed graph data. The main idea of this paper is to partition the graph into several clusters and visualize the relationships between clusters. By visualizing these clusters, we can find missing and/or relevant associations. By increasing the level-of-detail, we can further emphasize graph patterns within clusters and allow users to understand the structure of a large graph in detail.

We first address the problem of module-based graph visualization by introducing a *graph modularity measure*. Second, a graph simplification method is adopted to accelerate the module detection. Third, each graph pattern in the module is analyzed using the k-clustering method to enhance the module visualization. Fourth, the visualization of *ModuleGraph* is constructed by (a) building a module-based graph that indicates the basic structure of the original graph and (b) identifying the sub-graph patterns. We use a symbolic signature instead of a simple node in the community graph. Finally, our experiments on real social networks and spatial networks show that *ModuleGraph* can effectively transform large-graph data into recognizable patterns and shapes that reveal significant structural and topological information.

The remainder of our paper is organized as follows. The related work is discussed in Sect. 2. We then present the details of our ModuleGraph construction, including the mathematical model and the visual design, in Sect. 3. We describe the experimental study to demonstrate the effectiveness of our method in Sect. 4. In Sect. 5, we further discuss the performance of our methods. Finally, in Sect. 6, we conclude our work and present future research directions.

2 Related work

The primary objective of large network visualization is the understanding of global and local patterns in dynamic graphs, such as connectivity bundles, clustering, boundary formations, and expansions. Zinsmaier et al. (2012) are inspired by the level-of-detail (LOD) techniques in the computer graphics discipline and introduce a straight-line graph drawing that can be rendered interactively with different levels of detail to visualize large-scale graphs. Because display systems have finite area and are physically limited both in size and spatial dimensions for visualizing large-graph structures, a multi-screen solution is adopted in Chae et al. (2012). However, the multi-screen approach imposes restrictions on spatial layouts and interactions.

Various methods for graph simplification have been developed. The most prominent ones are the graph summarization (Tian et al. 2008), HiMap (Shi et al. 2009), and K-Core (Dorogovtsev et al. 2006). The graph summarization method in Tian et al. (2008) allows the user to interactively control the resolution of each aggregation in a large graph. HiMap (Shi et al. 2009) is presented to visualize large-scale social networks through a hierarchical summarization. Complexity reduction via K-Core (Dorogovtsev et al. 2006) is used to remove nodes with less than k links. Zhang et al. (2015) proposed a new model to calculate the core structure with a million or more nodes. Unlike the k-core method, this model can convert low-degree vertices into core nodes. For further graph simplification, a dimensionality reduction method is proposed in van den Elzen et al. (2016). This method projects the abstract information into points in the two-dimensional space.

Our work is related to the community detection algorithm. It is aimed at partitioning a large graph into modules. Newman (2006) converts the problem into an optimization problem using a modularity-based method called *Modularity Classes*. This method can measure the performance of the algorithm on the community detection problem. Blondel et al. (2008) further present an acceleration method called *Louvain* to reduce the computational complexity of the community detection problem. They demonstrate the performance of *Louvain* on several large data sets. Moreover, dynamic graph evolution is achieved in Vehlow et al. (2015) using community detection and a stream representation.

For the representation of community networks, Blondel et al. (2008) and Rosvall and Bergstrom (2008) discuss in detail *community aggregation*. They aggregate the nodes of a community into a super-node. A machine learning method, such as *Belief Propagation* (Horng et al. 2011), is adopted to explore large graphs. In addition, Dunne and Shneiderman (2013) improve the graph visualization readability by drawing different glyphs. Wu et al. (2015) utilize Voronoi maps to enhance the community visualization. Compared with the previous methods cited above, our method focuses on designing a module for anticipating the visual effects of the final visualization layout for large-scale networks.

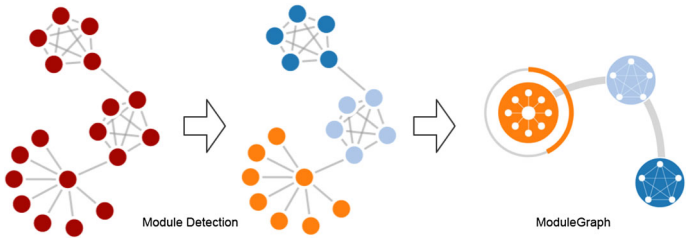


Fig. 1 Simple example of module detection and ModuleGraph

3 Visualizing graph modules

We define the visualization of large-scale networks as a problem of module detection and clustering. Figure 1 is a simple example of module detection. The input of our framework is the network data and the corresponding number of edges and nodes. First, we aggregate the nodes into modules through modularity measuring. Second, we abstract the features of detected modules and assign the modules with different patterns using a k-clustering method. In this step, we design five patterns to approximatively represent the structure of the module. Finally, the detected information is visualized according to the module design.

Our method, ModuleGraph, can be viewed as an abstraction of a large graph. The right part of Fig. 1 is an example of ModuleGraph. The topology of a large graph is denoted by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ..., \mathbf{v}_n^T], (\mathbf{v_i} \in \mathbb{R}^2)$ denotes the vertices and $\mathbf{E}$ denotes the undirected edges. The ModuleGraph is defined as $\mathbf{Mg} = (\mathbf{M}, \mathbf{E})$, where $\mathbf{M} = [\mathbf{m}_0^T, \mathbf{m}_1^T, ..., \mathbf{m}_n^T], (\mathbf{m_i} \in \mathbb{R}^2)$ denotes a set of modules and $\mathbf{E}$ denotes a set of undirected edges, where each edge has a weight of $w = |E|$. Each $\mathbf{m}$ is a vector with several features, such as the module pattern, sub-node count, and sub-link count.

Overall, each module can be considered as a community in the graph. Hence, a community detection method called *Modularity Classes*, as described in Newman (2006), can be adopted to find the communities from a large graph. The method of community detection is better than the k-means clustering method, because assignment of k is not required in advance. A further improved method for community detection, called the *Louvain*, reduces the calculation of the modularity. This has been shown to be effective on large-scale community detection by Blondel et al. (2008). The process of finding the modules consists of maximizing the modularity of the linked nodes. The modularity measure Newman (2006) can Q be defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[W_{ij} - \frac{k_i k_j}{2m} \right] \lambda(s_i, s_j) \tag{1}$$

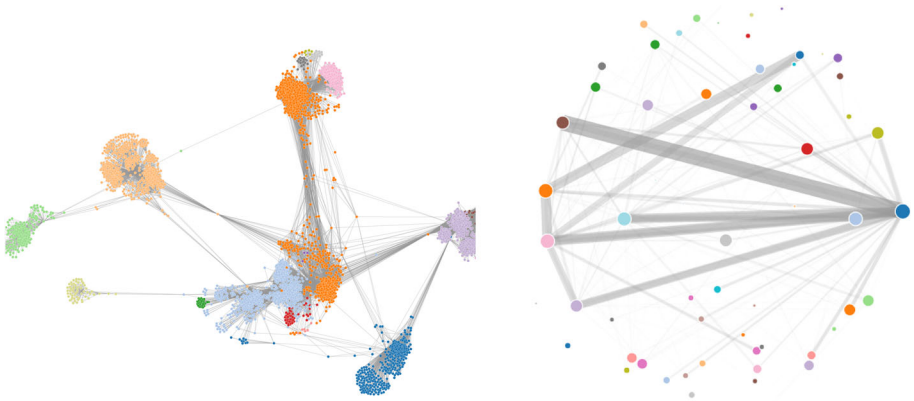
where W_{ij} indicates the linking weight of node i and node j; k_i and k_j denote the counts of linked nodes on node i and j, respectively; m indicates the sum of all linking weights; and the function $\lambda(i, j)$ denotes whether two nodes belong to the same community. The range of W_{ij} is [0.0, 1.0]. Parameter $\lambda(i, j)$ is set to 1 when node i and j belong to the same community, or 0 when node i and node j belong to different communities.

Historically, researchers have paid less attention to community detection on spatial networks. An example of a spatial network is a geographical network, such as airlines around the world. We further consider the distance feature for a spatial network by modifying Eq. (1). The link with a longer length will contribute less modularity. Hence, the improved modularity measure Q_d can be defined as follows:

$$Q_d = \frac{1}{2m} \sum_{i,j} \left[\mathrm{W}_{ij} \mathrm{dist}_{\mathrm{ij}}{}^{-\alpha} - \frac{k_i k_j}{2m} \right] \lambda(s_i, s_j). \tag{2}$$

In Eq. (2), the value of α indicates the network type. The type could be 0 or 1, respectively, denoting a non-spatial network or a spatial network. dist_{ij} denotes the Euclidean distance of two nodes in a spatial network. In a non-spatial network, the value of dist_{ij} is 1.

The calculation of *Modularity Classes* (Newman 2006) for a large-scale graph is time consuming. Although the *Louvain* method (Blondel et al. 2008) achieves a high performance, its precision is lower than that of *Modularity Classes*. We present a hybrid modularity-based method to utilize the two addressed methods. Our method is based on the modularity and is suitable for calculating the modules of a large-scale graph. We can define the hybrid modularity Q_h as follows:



(a) Original network **(b)** Community-based aggregation

Fig. 2 Visualization of a simplified large network

$$Q_h = \begin{cases} Q_d, (c_v + c_l) \leq \varepsilon \\ Q_d', (c_v + c_l) > \varepsilon. \end{cases} \tag{3}$$

In Eq. (3), c_v denotes the vertex count of the graph, c_l denotes the edge count of the graph, and $Q_d{}'$ is a high-speed community detection measure that removes the function of $\lambda(s_i, s_j)$. Although $\lambda(s_i, s_j)$ removal will slightly affect the accuracy of community detection, it makes sense of the community detection of a large-scale graph. $\varepsilon \in (0, +\infty)$ is a parameter that decides which measure will be used. The setting of ε is performed based on the computer configuration, namely the available CPU and RAM. In our experimental environment, we set ε as 0.1 million to achieve interactive visualization when the input network is a large-scale network. If the experimental computer is more powerful than ours, as shown in Sect. 5, the user can increase ε to achieve more accurate results.

Using the hybrid modularity-based method, we can effectively detect the communities of a large-scale graph, as shown in Fig. 2 (left). Different communities are assigned different colors. To observe the graph clearly, each community can be replaced with a single node, as shown in Fig. 2 (right). A simplified community graph allows the user to easily realize the clear structure of the original graph.

When the module sizes become very large, the simplified community graph may remain unsatisfactory on a limited display. Our *Module Detection* method can be extended using the hierarchical module method. A hierarchical module means that the detected communities in the previous detection process can be considered as the input of a new community detection process.

For hierarchical module detection, we define the iteration of module detection as i, with a default value of 1. Screen width and height are defined as w and h. If the number of nodes in the input network is far greater than ηwh, we increase the module iteration of i until the node count in the new graph is equal to or less than ηwh. η is a free parameter that reflects the blank space of the module visualization in the display. In our experiment, we set η as 0.3, w as 1440, and h as 900.

3.1 Pattern definition

To further gain insight into modules, we analyze the module patterns. Wernicke (2006) presented a method to detect motifs in a large network. Motifs indicate the link patterns in the graph. Unlike motifs, we calculate several pattern factors to analyze each module type according to its internal graph, as well as other significant features, such as module size, linking weight, visibility, and occupation percentage. An internal graph represents the structure of the module.

To simplify the module representation, we classify the modules into five agent patterns. The design of agent patterns is according to the most common network structures as presented in network structure theory (Xu 2013) and communication theory (Theory 2010). Figure 3 shows our designed patterns: tree-connection, concentration, circular, low-connectivity, and full-connectivity. The tree-connection pattern indicates that the graph is extended from a node and that some children of the root still contain the children. The

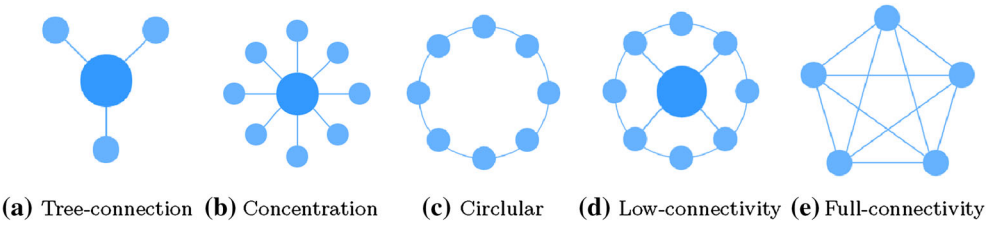


Fig. 3 Five types of graph modules

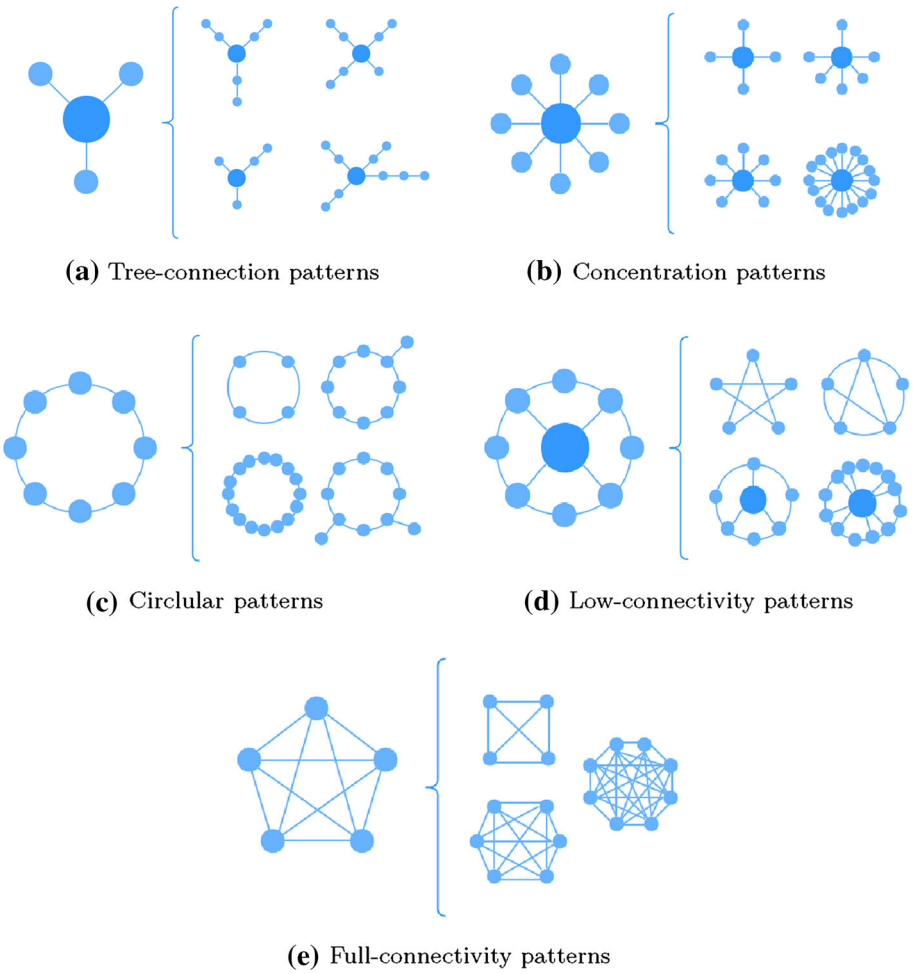


Fig. 4 Examples of graph modules: for each example, we show the agent pattern on the *left*, and example cases on the *right*

concentration pattern indicates that all children nodes are connected to the root and that each child does not have other connections. The circular pattern indicates that each node in the graph only has two links. The full-connectivity pattern is a complete graph. The remaining structure of the graph will be classified as a low-connectivity pattern. We define the patterns in Fig. 3 as the agents to represent other similar structures. We show some cases of similar structures of the agent patterns in Fig. 4. The right part of each sub-figure shows some graph cases of the related patterns. The method of mapping the graph-to-agent patterns will be discussed in Sect. 3.2.

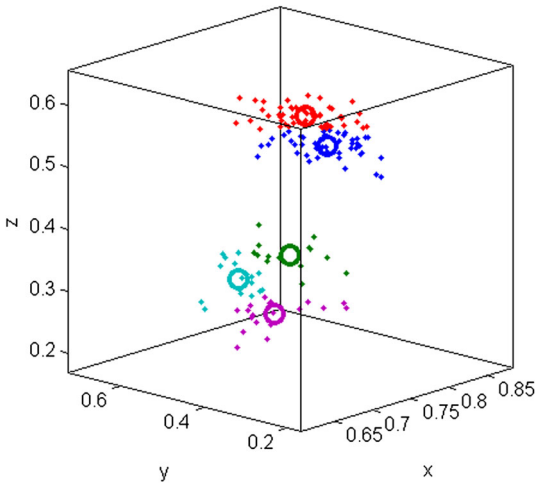


Fig. 5 Example of k-clustering. The *color* indicates the pattern type

3.2 Pattern detection via k-clustering

We present a k-clustering method to detect the pattern type of each module. To classify the module into the agent patterns that we have defined, we abstract three features for each module. The feature vector can be defined as $v_f = [\alpha, \beta, \delta]$. We define the number of connected nodes for a node as c. By following the pattern design, we can denote $\alpha = \frac{\max(e_i)}{\sum e_i}$, $\beta = \frac{\text{count}(e_i=2)}{n}$, and $\delta = \frac{1}{n}\text{count}(e_i > \frac{n}{2})$, where e_i denotes the neighbor count of node i and n denotes the node count of the module.

We propose a k-clustering method to approximately generate the clusters according to the module feature vector. Each cluster includes the modules with the same pattern. The proposed clustering method includes five steps. First, v_f is calculated for each module. We define each module as a node in our method and consider v_f as a three-dimensional position. Initially, all the nodes are not assigned to any cluster. Second, the cluster count (k) is initialized as 5, because we have designed 5 types of agent patterns for visualization. Third, 5 nodes that match 5 agent patterns are selected, and their positions are assigned as the cluster centers. The selection method can follow the agent pattern definition. Hence, we can obtain 5 initial clusters. Each initial cluster only contains one node. Fourth, the remain nodes will be assigned to the closest cluster by calculating the distance between its position and the other cluster center. Fifth, the mean position of the cluster's nodes is calculated and defined as the new cluster center.

The fourth and fifth steps should be repeated until the sum of the distances D is minimized. D is defined as $D = \sum_{i=0}^{i<k} \sum_{j=0}^{j<\text{count}(i)} \text{dist}(p_{ij}, c_i)$, where p_{ij} indicates node j in the cluster i, c_i indicates the cluster center of the cluster i, and dist is the distance calculation function. Figure 5 presents an example of 5 cluster distributions in the three-dimensional space after the k-clustering process is completed. Colored circles denote the cluster centers.

3.3 Visual design

The basic *ModuleGraph* layout can be calculated using a direct-force algorithm to ensure that each module will not be overlapping with other modules. The main idea of our visual design for *ModuleGraph* is to attempt to visualize the specific patterns instead of the crowd nodes in a large graph. The visual elements of each module include a circle with a different radius called m_c, an outside orbit with a different percentage called m_o, an icon called m_p that indicates the module pattern, and a set of edges called m_e that connect the related modules. Each edge between modules has a different width according to the weight. Figure 6 gives a description of the *ModuleGraph* design.

We choose the color themes from the *Google Color Palette* (Google 0000) to represent different modules. Because the color types are limited, the number in the center of the module can be used to indicate the id of the module. When the module graph remains large scale after simplifying the original graph, we design a chord diagram to assist in module visualization. A chord diagram is adopted to present the module relationship and distribution through the circle layout according to the importance weight of the module.

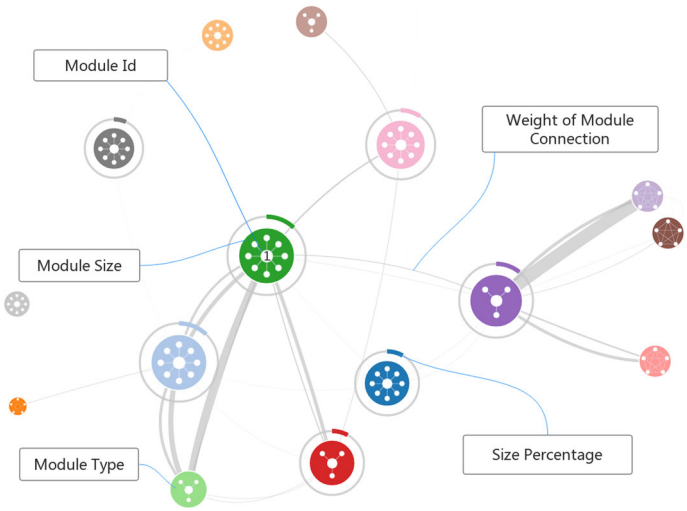


Fig. 6 Visualizing the ModuleGraph design

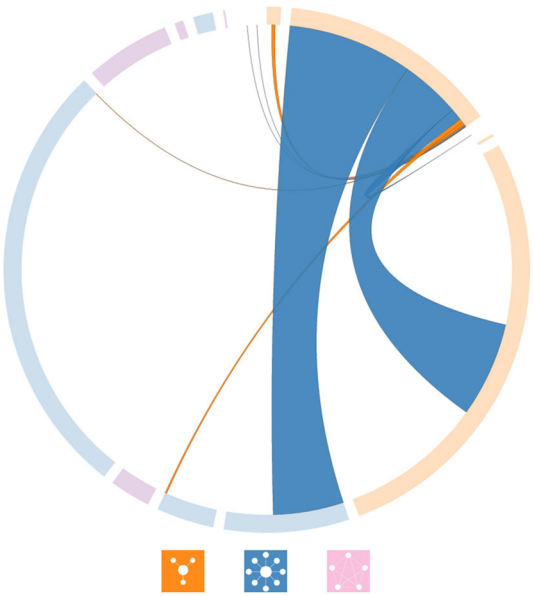


Fig. 7 Chord diagram representation of ModuleGraph

The importance weight can be defined according to the neighbor count of a module. The outside circle's color indicates the pattern type. The importance percentage of a module and the relationships among modules can be easily determined via a chord diagram, as shown in Fig. 7.

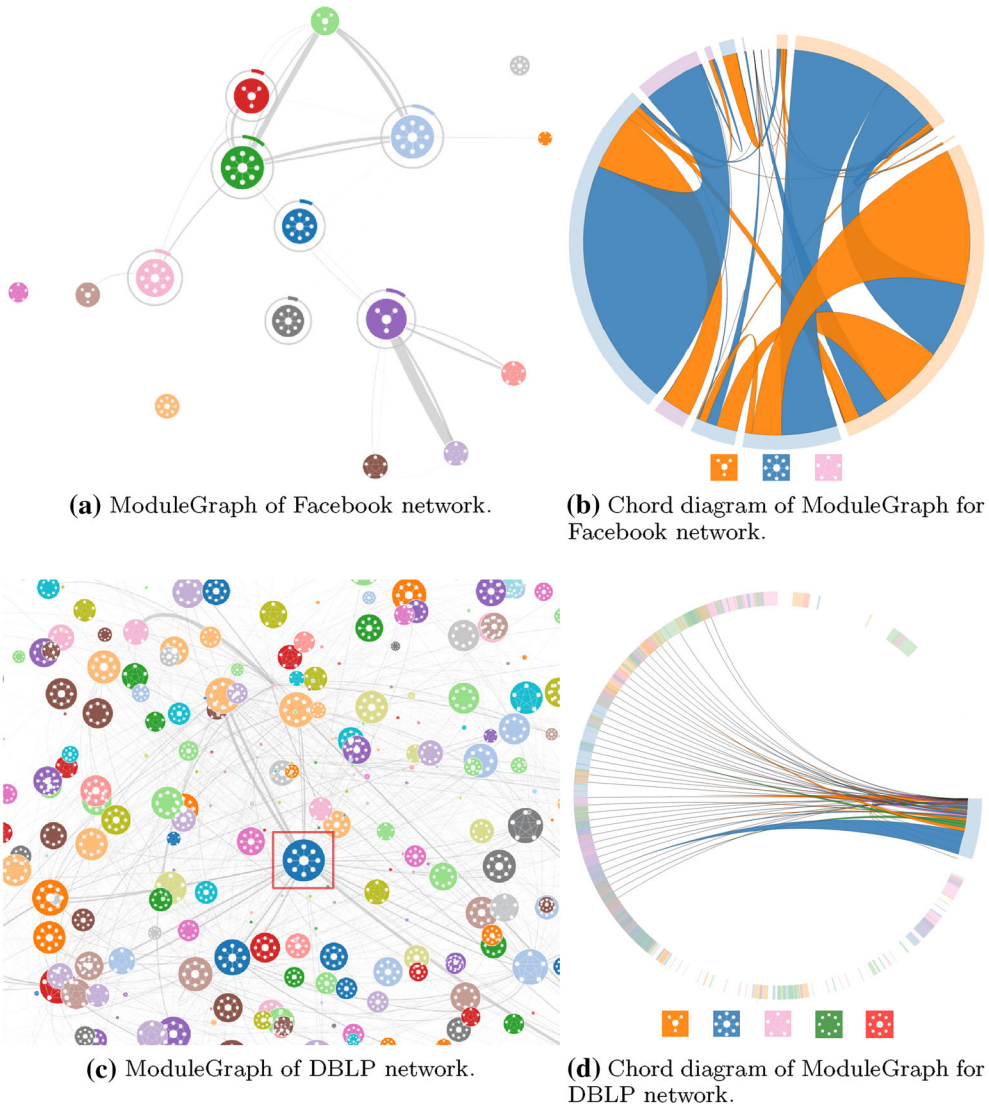
4 Experimental study

4.1 Social network

We select several large social network data sets from SNAP (Leskovec and Krevl 2014) which is a data library for analyzing large information networks. The selected social network data set is from Facebook, DBLP, YouTube, and Orkut, as summarized in Table 1. The visualization results of ModuleGraph for these

Table 1 Selected social network data sets

Data sets	Nodes	Edges	Module	Brief modules
Facebook	4,039	88,234	16	–
DBLP	317,080	1,049,866	442	–
YouTube	1,134,890	2,987,624	9821	8
Orkut	3,072,441	117,185,083	1969	244



(a) ModuleGraph of Facebook network.

(b) Chord diagram of ModuleGraph for Facebook network.

(c) ModuleGraph of DBLP network.

(d) Chord diagram of ModuleGraph for DBLP network.

Fig. 8 Visualizing social networks by ModuleGraph and chord diagrams

network data are shown in Fig. 8. From the results, we can find that the module patterns can be directly presented without any other interactions. From Fig. 8a, we can find that most of the module structure of the Facebook network is a concentration structure. A high percentage of concentration modules shows that most of the people on Facebook like to follow the hot people. Figure 8b is an assistant representation of Fig. 8a. The outer circle in Fig. 8b shows the percentage of each module type. Each chord at the outside circle represents a module. The inside part shows the connection of each module.

Figure 8c shows the ModuleGraph of DBLP data. The DBLP data include the paper citation relationships of the researchers. Because there are many modules in the final result, we hide the outside circle of the

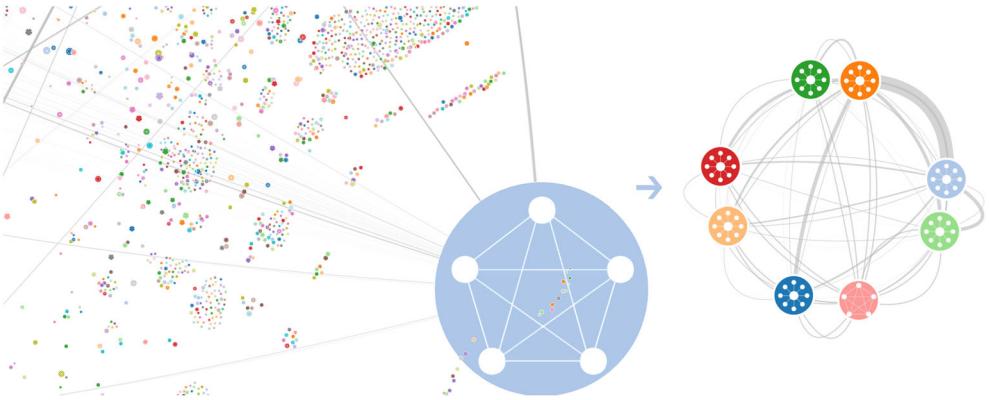


Fig. 9 Hierarchical ModuleGraph of YouTube social network

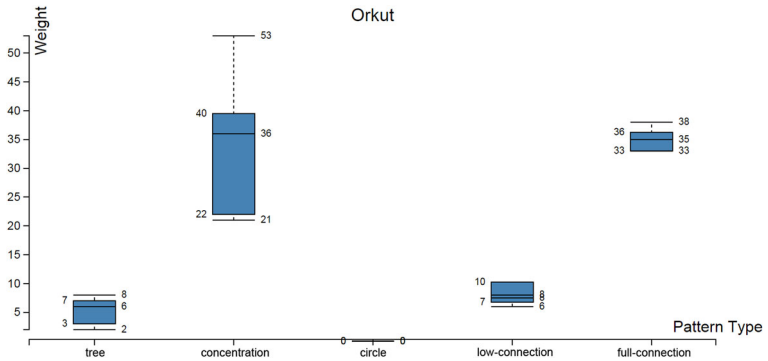


Fig. 10 Visualization of the size distribution of modules via *box plot* diagram for the Orkut network

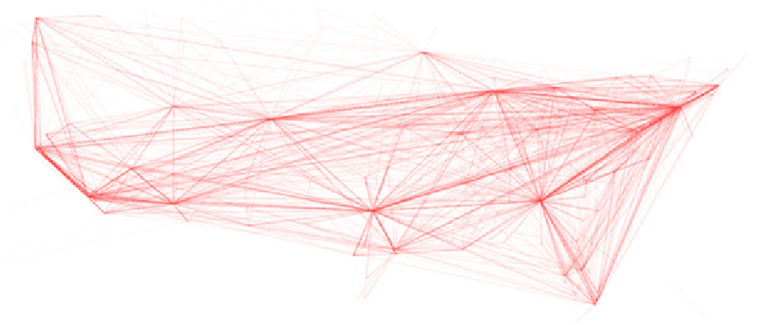
ModuleGraph. From Fig. 8c, we can find that the module pattern appearing most often is the tree-connection pattern. When the user clicks the module, as shown in the red rectangle of Fig. 8c, the related relationship of the selected module can be presented through a chord diagram, as shown in Fig. 8d. The bottom part of Fig. 8d presents the pattern types.

For large-scale networks, the hierarchical ModuleGraph representation as shown in Fig. 9 produces a clearer visual representation which can aid significantly in querying and visual selections. The left side of Fig. 9 shows the visual results without the hierarchical process. Because the detected modules are beyond the screen size, we create a new ModuleGraph by making the previous ModuleGraph as the input graph. The right side of Fig. 9 is the hierarchically processed result.

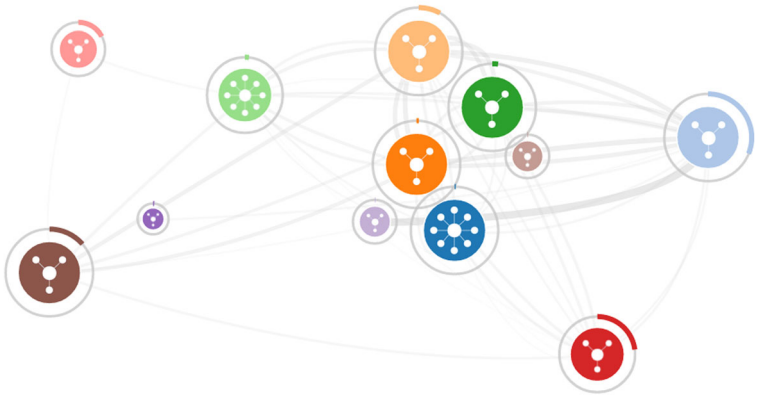
To further describe the information of the ModuleGraph, we adopt box plots to show the type distributions of the modules, as shown in Fig. 10. Figure 10 is related to the module information of the Orkut network. The five box plots in this Fig. 10 represent the five agent pattern statistics of the ModuleGraph. Each box plot includes five values: maximum, second maximum, median, second minimum, and minimum.

4.2 Spatial network

For the case study of a spatial network, we select a data set with a massive number of flight records in the US (Wickham 2011) in 2008. There are nearly 0.6 million flight lines, which are related to 3374 airports. Figure 11a is the original visualization of the airline network. By comparison with the simple representation, our method's result as shown in Fig. 11b represents the airport community distribution and the bundling airlines. In this case, the outside circle denotes the number of neighboring airports. The airline count inside the module is presented via the size of the inside circle. The pattern icon for each module represents the sub-network structure of the detected area. The information presented through ModuleGraph can help data analyzers acquire the important patterns from a large-scale network. For example, the air traffic northwest of New York State can be easily observed from Fig. 11b.



(a) Original visualization of the airline network.



(b) ModulGraph representation of the airline network.

Fig. 11 Spatial network of USA airline flights for a given period of time

Table 2 Comparison of module detection performance

Data sets	Community detection (Newman 2006)	Fast community detection (Blondel et al. 2008) (s)	Module detection method (s)	Module detection with k-clustering (s)
Facebook	1.1 (s)	0.2	1.2	1.2
DBLP	217.4 (s)	11.0	12.5	13.1
YouTube	Out of memory	25.3	27.4	27.7
Orkut	Out of memory	247.1	251.5	251.6
Flight Lines	54.8 (s)	4.2	5.7	5.74

5 Implementation and evaluation

We use D3 (Bostock et al. 2011) to visualize the results on a computer with 2.8 GHz Intel i7 CPU and 16 GB RAM. The visualizations are displayed in the Chrome browser (45.0) with a 1440 × 900 resolution. Module detection is implemented using the C++ language on the Windows 7 operating system.

Table 2 summarizes the performance of three module detection methods: community detection (Newman 2006), fast community detection (Blondel et al. 2008), and our module detection method. We also provide the module detection performance with k-clustering. From the summary, we can find that our method can effectively support large-scale network aggregation and visualization. Because a distance feature has been considered in our method, our method is slightly slower than the fast community detection (Blondel et al. 2008) for large-scale network processing. The time cost of k-clustering is ignorable, because it is only related to the number of detected modules.

Compared with the full network rendering, ModuleGraph achieves a satisfactory rendering performance for basic interactions. Timing results are shown in Table 3 where fps indicates frames per second. The rendering of the Orkut ModuleGraph is performed at high speed, because we adopted a hierarchical rendering strategy for the very-large ModuleGraph. Although in some experiments, the massive data sizes

Table 3 Comparison of the rendering performance

Data sets	Full network	ModuleGraph (fps)
Facebook	0.19 (fps)	58.82
DBLP	Out of memory	0.94
YouTube	Out of memory	0.55
Orkut	Out of memory	17.86
Flight Lines	0.15 (fps)	30.30

reduce the rendering speed, such as DBLP and YouTube, in Table 3, it is feasible to rewrite our ModuleGraph rendering applications directly on a GPU subsystem to achieve interactive rates.

6 Conclusion and future work

The *ModuleGraph* is an effective visualization tool for aggregating and representing large-graph data through clustering and interconnectivity pattern analysis. A module detection method is presented to integrate the nodes that belong to the same group into a new integral module. We further enhance our graph visualization method by representing the modules and the relationships among them. Our method can also be applied to spatial network visualization. In the future, we plan to extend *ModuleGraph* to visualize streaming networks. This would extend the ModuleGraph to a *Dynamic ModuleGraph* visualization system. The context change between modules can also be considered. Moreover, our method can be extended to visualize fuzzy networks that include overlapping communities.

Acknowledgments The authors would like to acknowledge the partial support of the Hong Kong Research Grants Council Grants, GRF PolyU 5100/12E, IGRF PolyU 152142/15E, and Project 4-ZZFF from the Department of Computing, The Hong Kong Polytechnic University.

References

Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008

Bostock M, Ogievetsky V, Heer J (2011) D3: data-driven documents. IEEE Trans Vis Comput Graph 17(12):2301–2309

Chae S, Majumder A, Gopi M (2012) HD-GraphViz: highly distributed graph visualization on tiled displays. In: Proceedings of the eighth indian conference on computer vision, graphics and image processing, ICVGIP '12, pp 43:1–43:8

Dorogovtsev SN, Goltsev AV, Mendes JFF (2006) *k*-core organization of complex networks. Phys Rev Lett 96:040601

Dunne C, Shneiderman B (2013) Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI '13. ACM, pp 3247–3256

Google: color palette. https://www.google.com/design/spec/style/color.html

Horng D, Chau P, Kittur A, Hong JI, Faloutsos C (2011) Apolo: making sense of large network data by combining rich user interaction and machine learning. In: Proceedings of the SIGCHI conference on human factors in computing systems, pp 167–176

Leskovec J, Krevl A (2014) SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data

Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582

Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci 105(4):1118–1123

Shi L, Cao N, Liu S, Qian W, Tan L, Wang G, Sun J, Lin CY (2009) Himap: adaptive visualization of large-scale online social networks. In: 2009 IEEE pacific visualization symposium (PacificVis), pp 41–48

Theory C (2010) Patterns of communication. http://communicationtheory.org/patterns-of-communication

Tian Y, Hankins RA, Patel JM (2008) Efficient aggregation for graph summarization. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data, SIGMOD '08, pp 567–580

van den Elzen S, Holten D, Blaas J, van Wijk J (2016) Reducing snapshots to points: a visual analytics approach to dynamic network exploration. IEEE Trans Vis Comput Graph 22(1):1–10

Vehlow C, Beck F, Auwärter P, Weiskopf D (2015) Visualizing the evolution of communities in dynamic graphs. Comput Graph Forum 34:277–288

Wernicke S (2006) Efficient detection of network motifs. IEEE/ACM Trans Comput Biol Bioinform 3(4):347–359

Wickham H (2011) Asa 2009 data expo. J Comput Graph Stat 20(2):281–283

Wu Y, Wu W, Yang S, Yan Y, Qu H (2015) Interactive visual summary of major communities in a large network. In: 2015 IEEE pacific visualization symposium (PacificVis), pp 47–54

Xu J (2013) Topological structure and analysis of interconnection networks, vol 7. Springer Science and Business Media

Zhang X, Martin T, Newman MEJ (2015) Identification of core-periphery structure in networks. Phys Rev E 91(3):1–10

Zinsmaier M, Brandes U, Deussen O, Strobelt H (2012) Interactive level-of-detail rendering of large graphs. IEEE Trans Vis Comput Graph 18(12):2486–2495