Visual Analysis of Complex Networks and Community Structure

Bin Wu, Qi Ye, Yi Wang, Ran Bi, Lijun Suo, Deyong Hu, and Shengqi Yang

Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China
wubin@bupt.edu.cn

Abstract. Many real-world domains can be represented as complex networks.A good visualization of a large and complex network is worth more than millions of words. Visual depictions of networks, which exploit human visual processing, are more prone to cognition of the structure of such complex networks than the computational representation. We star by briefly introducing some key technologies of network visualization, such as graph drawing algorithm and community discovery methods. The typical tools for network visualization are also reviewed. A newly developed software framework JSNVA for network visual analysis is introduced. Finally,the applications of JSNVA in bibliometric analysis and mobile call graph analysis are presented.

Keywords: Visual Analytics, Complex Network, Community Structure.

1 Introduction

An explosion of network research catalyses the emergence of network science, which not limited to physics, but captures biology, computer science, mathematics and the social sciences [1]. In network science, more efforts have been devoted to empirical studies of large-scale networks while classical random graph models, such as the Erdös& Rényi (ER) model, belong to pure theoretical study. Along with research of complex network, software packages have also been developed to assist scientists to tackle the real complex network datasets from modeling, analyzing to visualizing. Visualization can help make the complex network data easier to read or understand. The visualization of networks is possible to greatly improve the sense making of analytical results but also for the communication of results within and across disciplinary boundaries but also to the general public. Hence visualizations have played an important role in generating new insights in network science.

Visualization is often used as an additional or standalone data analysis method. Network visualization is a technique to graphically represent sets of network data. Visualization techniques can be applied to communicate the results of network measurement, modeling and validation or to visually compare the structure and dynamics of empirical and simulated networks [2]. Visualization has become an indispensable part of any network analysis toolkit. With respect to visualization, network analysis tools are used to change the layout, colors, size and other properties of the network representation. Large network visualization has recently received a lot of attention from researchers in both information visualization and network science communities. Although some newly available techniques are quite capable of visualizing large graphs of thousands to hundred thousands of nodes and edges, visualization of large graphs, is still a challenge problem due to the size and complexity of real networks and the diversity of the network science applications.

In this paper, we begin with a brief review of network visualization techniques and representative software packages. Since scalability is a key issue for network visualization paradigms, we overview the methods of graph clustering for large datasets, which can quickly discover the community structure embedded in a large graph and divide the graph into densely connected sub-graphs. A newly developed software framework JSNVA [3] for network visual analysis is also introduced. Finally, some applications of JSNVA ,such as investigating the structural features of scholarly networks and exploring temporal communications in mobile call graphs are discussed.

2 Network Visualization Basics and Typical Tools

2.1 Network Visualization Basics

A good visualization system for very large network should be a combination of three components including graph drawing, graph clustering and interaction. Network visualization techniques range from the layout of networks, to the visualization of network dynamics and the tight coupling of data analysis and visualization. The basics of visualization design include major matrix, tree, and graph layout algorithms. The visualization of network dynamics shows the evolution of networks in terms of attribute changes or structural changes. The tight coupling of data analysis and visualization has been more related to a new body of research: visual analytics [4], which is the science of analytical reasoning supported by highly interactive visual interfaces.

2.2 Algorithms for Drawing Large Graphs

Visualizing large graphs presents unique problems which require non-orthodox solutions [7]. Drawing which displays the entire graph has the advantage of showing the global structure of the graphs. In the last decade several algorithms that generate string-line drawings of general large graphs have been invented.

Fruchterman and Reingold citefr propose a heuristic force-directed algorithm based on the model of the Spring algorithm of Eades. This algorithm is widely used in many tools and framework, such as JUNG, Prefuse, Pajek etc. The naive algorithm is based on a model of pairwise repelling charged particles (nodes) and attracting springs (edges). Their layout algorithm needs $\Theta(|V|^2)$ time per iteration to calculate the repulsive forces between all pairs of nodes. In order to speed up this algorithm, they present a variation that called the grid-variant algorithm based on the technique in n-body simulations to approximate the distant bodies as a single pole. If the number of iterations is assumed to be constant, the best-case running time of grid-variant algorithm is $\Theta(|V| + |E|)$. The worst-case running time, however, remains $\Theta(|V|^2 + |E|)$ [5,12]. Frich et al. [6] present a randomized adaptive layout algorithm call GEM for nicely drawing undirected graphs that is based on the Spring algorithm and contains several new heuristics to improve the convergence, including local temperatures, gravitational forces and the detection of rotations and oscillations.

Faced with the problem of drawing a large graph, it is natural to associate with a hierarchy of graphs and produce a drawing starting with the smallest graph in the hierarchy, and drawing larger and larger graphs using previous drawing. Harel and Koren [9] use an $O(k|V|)$ algorithm that finds a 2-approximative solution of the k-center problem. They divide the graph into a sequence of subgraphs which is determined by the approximative solution of the k-center problem. They use a variation of the algorithm of Kamada and Kawai [10] as force-directed algorithm to draw each level. The asymptotic running time of their algorithm is $\Theta(|V||E|)$ and $\Theta(|V|^2)$ memory is needed to store the distances between all pairs of nodes [12]. Gajer et al. [7,8] propose a hierarchical force-directed layout algorithm called GRIP for drawing large graphs. The algorithm are composed of two phrases. A coarsening phase in which a sequence of coarse graphs with decreasing sizes is computed and a refinement phase in which successively drawing of finer graphs are computed, using the drawings of the next coarser graphs and a variant of suitable force directed algorithm. However, the GRIP algorithm is used for drawing the networks which contains only one component and it is hard to use it to get the overview of real-world networks directly.

Koren et al. [11] present a fast graph drawing algorithm ACE for drawing large graphs. ACE finds an optimal drawing by minimizing a quadratic energy function. The minimization problem is expressed as a generalized eigenvalue problem, which is solved rapidly using a novel algebraic multigrid technique instead of calculating the eigenvectors directly. The ACE algorithm is much faster than the force-directed algorithms for nearly all kind graphs. However, the authors do not give an upper bound on the asymptotic running time of ACE in the number of nodes and edges [12].

The ability to find and analyze such communities can provide invaluable help in understanding and visualizing the structure of networks [22]. The community detecting algorithms will great influence the results of network visualization, however, there are lots of metrics on community detecting has been proposed. Newman and Girvan [22] use the famous GN algorithm to discover natural divisions of co-authorship network. They get an overview of the relationships between different communities by applying the shortest-path betweenness version of the GN community structure algorithm. Palla et al. [18] use the k-clique-community detecting algorithm to get the overview of the protein-protein interaction network.

2.3 Network Visualization Typical Tools

There are various network visualization tools developed in resent years. These tools can be broadly divided into two categories: special-purpose tools or general-purpose tools. Those special-purpose tools are domain-specific. Social network analysis and biology are always the most popular domains. The typical tools include: NetDraw in UCINET [27], NetMiner, igraph, Cytoscape etc. Those general purpose tools can visualize arbitrary network data that can be expressed as a set of nodes, links, or paths. The typical software packages include: Pajek [26], JUNG [28], Prefuse [13], Graphviz, GUESS in Network Workbench etc.

NetMiner is an innovative software tool for Exploratory Analysis and Visualization of Network Data [29]. NetMiner allows user to explore social network data visually and interactively, and helps user to detect underlying patterns and structures of the network.It is a comprehensive package for the analysis of social network data, which can handle tens of thousands of nodes. Social network analysis methods include centrality measures, subgroup identification, role analysis, elementary graph theory, and permutation-based statistical analysis.

Cytoscape is an open-source, cross-platform network visualization and analysis application [19]. Cytoscape has it's roots in Systems Biology and is therefore well suited for analyzing data from high-throughput experimentation as well as other molecular state information. The central organizing metaphor of Cytoscape is a network (graph), with genes, proteins, and molecules represented as nodes and interactions represented as edges between nodes. The core functionality includes the visualization, layout, and manipulation of networks in addition to data handling services needed for importing, exporting, and managing network data.

Network Workbench [25] is a Large-Scale Network Analysis, Modeling and Visualization Toolkit for Biomedical, Social Science and Physics Research. This project will design, evaluate, and operate a unique distributed, shared resources environment for large-scale network analysis, modeling, and visualization, named Network Workbench (NWB). NWB integrates the excellent GUESS visualization tool to interactively explore and understand specific networks, as well as their interaction with other types of networks.

3 JSNVA: A Framework for Network Visual Analysis

To gain insight into today's large data resources, data mining extracts interesting patterns. To generate knowledge from patterns and benefit from human cognitive abilities, meaningful visualization of patterns are crucial [14]. Although information visualization technologies are indispensable in complex data analysis, widespread tools still need to be developed, as successful information visualization applications often require domain-specific customization.

In order to explore the link relations in real-world networks explicitly, we propose JSNVA (Java Straight-line Drawing Network Visual Analysis framework) [3], a lightweight framework for network visual analysis. The goal of JSNVA is to give simple and elegant solutions to analyze real-world networks vividly and intuitively, so we both focus on drawing graphs and analysis. To evaluate its function, we apply JSNVA to analyze real-world networks in different large scale data sets and we can get their underlying structure intuitively and quickly.

3.1 Architecture

A framework is a set of cooperating classes that make up a reusable design for a specific class of software [15]. JSNVA is written in Java programming language using Java2D graphics, and it provides a clear framework for graph based visual analysis. JSNVA draws from pioneering work on drawing graphs and existing system like JUNG, prefuse. To make JSNVA more flexible, reusable and understandable, we apply some design patterns in it and divide the framework into different modules based on their functions. JSNVA is mainly composed of the following core modules: Graph module, Layout module, Algorithm module and Paint module.

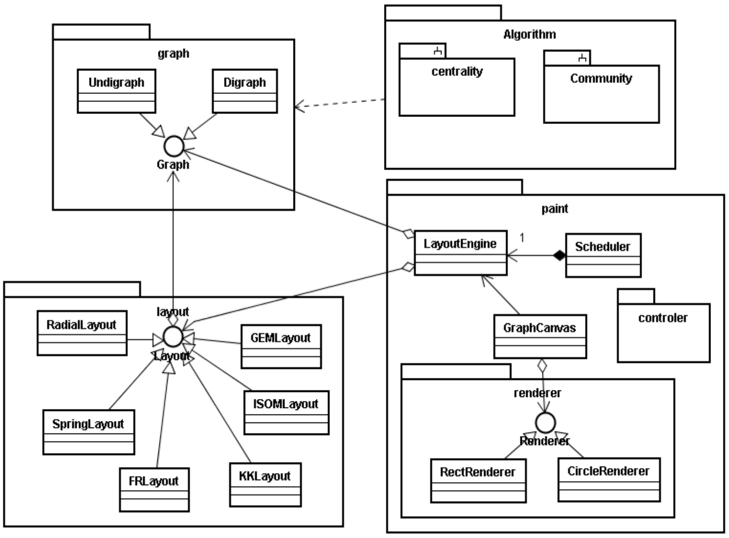

Fig. 1. The architecture of JSNVA

Graph Module. The Graph module implements the underlying data structure of the graph to be drawn or analyzed. The graph data structure can store attributes by different attribute keys. JSNVA provides both a directed graph interface and an undirected graph interface for abstract graphs. These graph interfaces enable us to handle general graph structures, so that we are not restricted to special graph structures such as general tree or rooted tree structures. In the same time, we can use the graph structures to analyze real-world network efficiently as a general graph library independent of other modules.

Layout Module. The Layout module is the core module of JSNVA which determines the positions of vertices under geometric constraints. In this module, developer can implement different layout algorithms strictly and don't need to connect them together, for which can be done in the Engine module. Layout module supplies different drawings algorithms to map the vertices' positions of abstract graph to the panel and store the positions of vertices in the graph structure. JSNVA implements many general layout algorithms, including forced directed layout [5,10], radical layout [16] etc.

Algorithm Module. The Algorithm module focuses on implementing the algorithms from graph theory. This module only use the basic graph structure in the Graph module, and it is still under development. We have implemented some basic algorithms on graph theory, such as Dijkstra algorithm, Kruskal algorithm and Prim algorithm etc. In order to get deep understanding of the different large-scale networks, we divide two sub-modules in the algorithm module, which are centrality sub-module and community sub-module. We have also implemented some community detecting algorithms, such as maximal clique detecting algorithm, GN algorithm [17], K-clique-community detecting algorithm [18] in the community sub-module. We also implement some well known centrality algorithms in the centrality sub-module such as degree centrality, betweenness centrality, closeness centrality etc.

Paint Module. The Paint module is the top one of the framework, and it use other modules to draw a network on a panel. In the Paint module there is a submodule called Controler module which enable JSNVA to handle interactive events, such as mouse event, saving images event etc. The GraphCanvas subclasses Swing's JPanel and it can be used in any Java Swing application. The LayoutEngine use a Scheduler pattern to provide dynamic visualization [24] which updates visual properties and redraws the network at regular time intervals. Following the Template Method pattern [15], the LayoutEngine defines the skeleton of drawing algorithm by using the layout algorithms provided by the Layout Module to display a network on the GraphCanvas. At each step of displaying the graph, the Scheduler in the LayoutEngine will call the layout method in the Layout interface to redraw the network on the GraphCanvas dynamically.

The vertices and edges are drawn by Renderer submodule. This submodule separate visual components from their rendering methods, allowing dynamic determination of visual appearances [24]. Changing the appearance of vertices and edges just requires overriding the paint methods in the submodule. The elements in network can be drawn as different styles, for example, for example, we can change the sizes, shapes, colors of vertices and we can also change the colors, thickness of edges. In the Render submodule there is a submodule called assignment module which is used to set visual attributes in the graph structure, such as sizes, colors, labels, fonts etc. for rendering the elements.

4 Applications

4.1 Co-authorship Network

Here we build a co-authorship network to investigate kinship patterns, community structure in it. We get the published papers written by the students and faculty at Beijing University of Posts and Telecommunications from 1998 to 2004, which are indexed by SCI, EI and ISTP. There are about 1990 papers in the data set. We extract authors' names from the files and put the first name before the surname manually and connect all the authors in each paper together. In the co-authorship network, the size of the giant component is 1453, filling 87.4% of the total volume. In the following parts of this paper, we only focus on characters of the giant component.

Community Detecting. Detecting the relations among communities is also crucial to the understanding of structural and functional properties of networks, and it is a useful way to handle information overload to get an overview of the network. There are two kinds of famous community detecting algorithms to find the clusters in network. One method is to divide the network into different separated sets of communities, and the other method is trying to detect sets of overlapping communities.

Separated Communities. Here we set the collaboration times between authors as the weights of edges, then use the algorithm provided by Newman [23] in the Algorithm module to detecting community structures in the weighted giant component, which is implemented in the Algorithm module. Through the algorithm, we get 34 communities in the component and all of them are in different colors. The thickness of edges in Fig. 2 indicates the collaboration times between authors and the size of vertices shows the productivity of authors. We choose the largest community which is to show the detailed view. Fig. 2 shows the largest community in the giant component. To avoid the graph image confused by authors' names, we label some famous authors in the community. We find some of the authors labeled are professors in the school of Telecommunication Engineering, while others are in the school of Science. PeiDa Ye is an academician from Chinese Academy of Sciences and JinTong Lin used to be a student of professor PeiDa Ye. Both of them are in the Optical Communication Center in the school. BoJun Yang and XiaoGuang Zhang et al. are professors in the school of Science. Although the authors are from different schools, they are in the same research field of Optical Communication.

Overlapping Communities. To uncover the stable communities, we will filter the unstable links in it and link these communities together to show their research fields. We use the k-clique-community detecting algorithm [18] in the Algorithm module to connect these highly overlapping cliques in this network. At last, we find 29 4-clique-communities in the stable network. We find out the most common used school name in each community to describe it. As shown in Fig. 3,

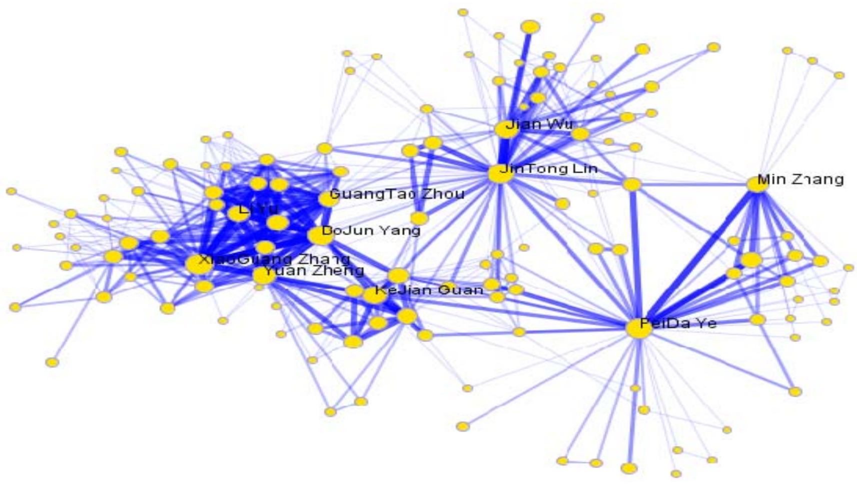

Fig. 2. The largest community in the giant component

the green vertices represent authors from the school of Telecommunication Engineering; the yellow ones are from the school of Science ; the dark blue ones are from the school of Computer Science; the purple ones are from the school of Electrical Engineering; the pink ones are from the school of Information Engineering; the red ones are the people belong to different communities. Through this figure, we find most of communities are from the school of Telecommunication Engineering which indicates the main research interest of the university is related to telecommunication. We can also find that the authors from the school of Science are closely connected with the authors from the school of Telecommunication Engineering because they are all have a main common research interest in Optical Communication.

4.2 VAST2008 Challenge

We explored the temporal communication patterns of Catalano/Vidro social network which is reflected in the mobile call data in the mobile call mini challenge of VAST 2008 contest [20]. Our team got the Cell Phone Mini Challenge Award on social network accuracy. In this challenge, we focus on detecting the hierarchy of the social network and try to get the important actors in it. We present our tools and methods in this summary. By using the visual analytic approaches, we can find out not only the temporal communication patterns in the social network but also the hierarchy of it.

Data Set. The mobile call records cover a ten-day period in June 2006, and are narrowed to about 400 unique mobile phones during this period. The data set includes records with the following fields: identifier for caller, identifier for receiver, time, duration and call origination cell tower. The challenge includes

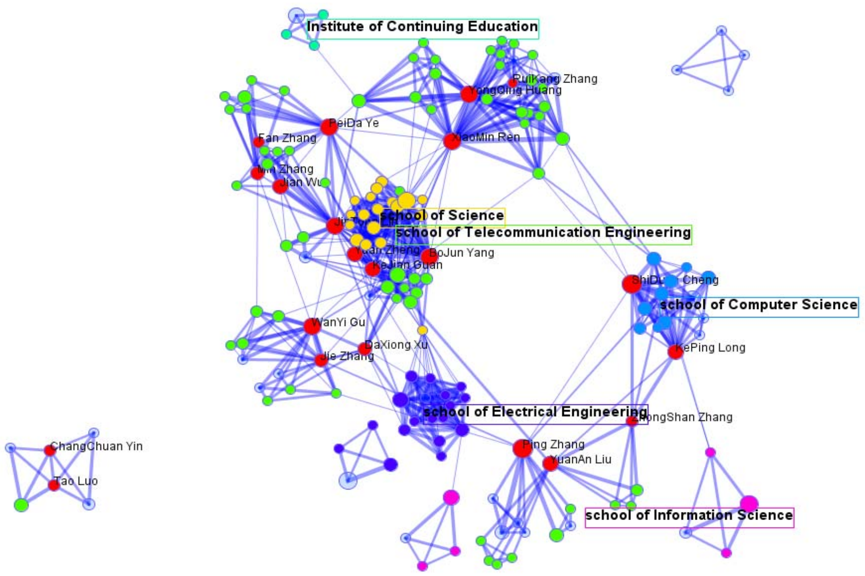

Fig. 3. An overview of the research fields of these communities at the university

questions about the social structure of the call network. Furthermore, we need to answer and characterize the changes to the social network over the ten days.

Temporal Relationship. In order to detect the communication patterns, we construct call graphs based on the call records. According to the mobile phone call records from June 1^{st} 2006 to June 10^{th} 2006, we find that the social network structure changed greatly on June 8^{th} 2006. In the first 7 days, the leader of the social network was 200. We use the PageRank value of each individual in the directed call graph formed in these 10 days to show the importance of the individual [21]. The sizes of vertices indicate the importance of vertices. We find an interesting pattern that all the neighbors of 200 vanished in the call graph formed on June 8^{th} 2006 and there were some actors with large degrees replaced their positions.

Equivalent Actors. The problem of detecting similar vertices in a network is not a new one. There are a lot of measures to show the similarity between data objects, such as simple matching coefficient, Jaccard coefficient, cosine similarity, Pearson's correlation coefficient etc. In order to get the similarity between two vertices in the call graph formed in these 10 days, we choose the Jaccard coefficient to measure how many common neighbors they share. Let $N(i)$ be the neighborhood of vertex i in a network and $|x|$ be the size of set x. Then the Jaccard coefficient of an actor pair (i, j), which is symbolized by $J(i, j)$, is given by the following equation:

$$J(i,j) = J(j,i) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}$$

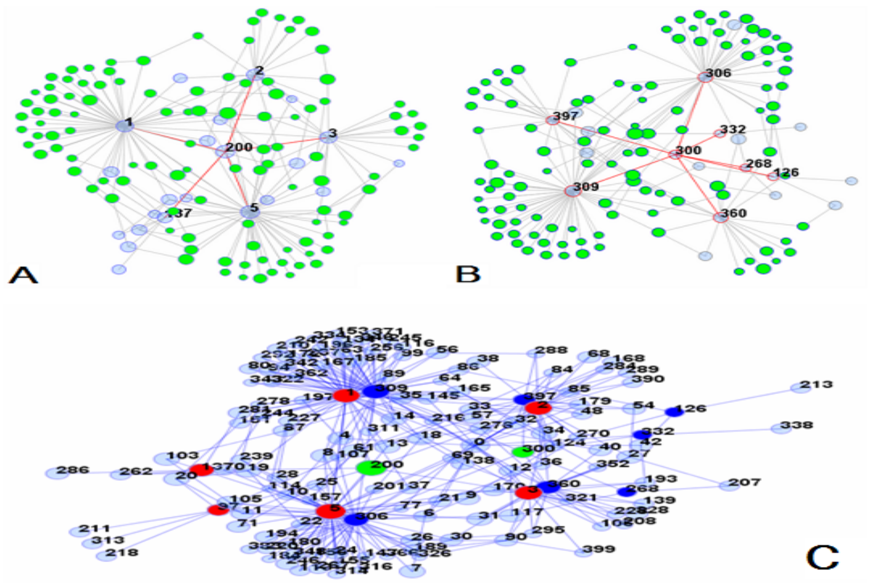

Fig. 4. The temporal social network (A) the first 7 days (B) the last 3 days. (C) The social network of the temporal call graph.

Table 1. The Jaccard coefficients of actor pairs

ID1	ID2	Jaccard Coefficient
1	309	0.75
5	306	0.68
3	360	0.60
2	397	0.59

Table 1 shows the Jaccard coefficients of these actor pairs with high degrees in the call graph during the 10 days. The equivalent actor pairs are (1, 309), (5, 306), (3, 360) and (2, 397).

Social Structure in the Call Graphs. We can get that there are two groups of people who communicate with a lot of people in the Catalano/Vidro social network. The first group is the neighbors of 200 and their identifiers are 1, 2, 3 and 5. The second group is the people who have the equivalent positions with the people in the first group whose identifiers are 309, 397, 360 and 306. All the people in the second group called the person whose ID is 300. We use the egocentric networks of 200 and 300 to show the social network structure. We get the social network by detecting the two egocentric networks whose roots are 200 and 300 in 2 hops. Fig. 4 (A) shows the social network of the first 7 days, and Fig. 4 (B) shows the social network of the last 3 days. The green nodes are the ones in both social networks. As shown in Fig. 4 (C), by merging these two temporal networks together, we can get the social network formed in these 10 days. In Fig. 4 (C), the green nodes are the core actors in the social network and the blue and red ones are their neighbors, respectively. We suspect that the network may reflect a social structure crackdown, and the positions of the 200, 1, 2, 3 and 5 may be replaced by 300, 309, 397, 360 and 306.

5 Conclusion

In this paper, network visualization technologies,such as layout algorithms for drawing large graphs and community detecting algorithms for graph clustering, are reviewed. Some typical network visualization tools are also introduced.A lightweight framework for network visual analysis JSNVA: Java Straight-line Drawing Network Visual Analysis framework is presented.

We present two applications to show its flexibility and performance. In the first application, we try to analyze the statistical and clustering features of a co-authorship network. By using focus+ detail views, we show an overview of the relations between different communities and the detailed relations between authors in the largest community. After finding the clustering and labeling the institute names in different stable communities, we can get an overview of the different research fields of the university and their relations. In the second application,we analyze the social network structure of an organism, based on the mobile call records of the mini challenge in VAST 2008. By visualizing the calling behaviors of the actors in the social network, we find the structure of the social network changes during a period observed. A reduced social network structure is constructed by exploring these communication patterns through JSNVA.

Acknowledgements. This work is supported by the National Natural Science Foundation of China under Grant No. 60402011, and the National Science and Technology Support Program of China under Grant No.2006BAH03B05. It is also is supported by IBM China Research Laboratory, the Specialized Research Fund for the Joint laboratory between Beijing University of Posts and Communications and IBM China Research Laboratory (Project No.JTP20071002-4,JTP200806014-3).

References

1. Barabási, A.-L.: Taming complexity. J. Nature Physics. 1, 68–70 (2005)
2. Börner, K., Sanyal, S., Vespignani, A.: Network Science. In: Cronin, B. (ed.) Annual Review of Information Science & Technology, vol. 41, ch. 12, pp. 537–607. Information Today, Inc., American Society for Information Science and Technology, Medford (2007)
3. Ye, Q., Wu, B., Wang, B.: JSNVA: a Java Straight-line Drawing Framework for Network Visual Analysis. In: 4th International Conference on Advanced Data Mining and Application, pp. 667–674 (2008)
4. Thomas, J.J., Cook, K.A.: A Visual Analytics Agenda. IEEE Computer Graphics and Applications 26(1), 10–13 (2006)

5. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. J. Software Practice and Experience 21, 1129–1164 (1991)
6. Frick, A., Ludwig, A., Mehldau, H.: A fast adaptive layout algorithm for undirected graphs. In: International Symposium on Graph Drawing, pp. 388–403. Springer, Heidelberg (1994)
7. Gajer, P., Goodrich, M.T., Kobourov, S.G.: A multi-dimensional approach to force-directed layouts of large graphs. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 211–221. Springer, Heidelberg (2001)
8. Gajer, P., Kobourov, S.G.: GRIP: Graph Drawing with Intelligent Placement. J. Graph Algorithms and Applications 6(3), 203–224 (2002)
9. Harel, D., Koren, Y.: A fast multi-scale method for drawing large graphs. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 183–196. Springer, Heidelberg (2001)
10. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. J. Information Processing Letters 31(5), 7–15 (1989)
11. Koren, Y., Carmel, L., Harel, D.: ACE: AFast Multiscale Eigenvector Computation for Drawing Huge Graphs. J. Multiscale Modeling and Simulation 1(4), 645–673 (2003)
12. Hachul, S., Jünger, M.: An experimental comparison of fast algorithms for drawing general large graphs. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 235–250. Springer, Heidelberg (2006)
13. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: ACM SIGCHI conference on Human factors in computing systems, pp. 421–430. ACM Press, New York (2005)
14. Assent, I., Krieger, R., Müler, E., Seidl, T.: VISA: Visual Subspace Clustering Analysis. J. ACM SIGKDD Explorations Special Issue on Visual Analytics 9(2), 5–12 (2007)
15. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Resuable Object-Oriented Sofrware. Addison-Wesley, Reading (1995)
16. Yee, K.P., Fisher, D., Dhamija, R., Hearst, M.: Animated Exploration of Dynamic Graphs with Radial Layout. In: IEEE International Conference on Information Visualization, pp. 43–50 (2001)
17. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. J. PNAS 12, 7821–7826 (2002)
18. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. J. Nature 435, 814–817 (2005)
19. Cline, M.S., et al.: Integration of biological networks and gene expression data using Cytoscape. Nature Protocols 2, 2366–2382 (2007)
20. Ye, Q., Zhu, T., Hu, D., Wu, B., Du, N., Wang, B.: Cell Phone Mini Challenge Award: Social Network Accuracy— Exploring Temporal Communication in Mobile Call Graphs. In: IEEE International Symposium on Visual Analytics Science and Technology, pp. 207–208 (2008)
21. Brin, S., Page, L.: The Anatomy of large-scale Hypertextual Web Search Engine. J. Computer Networks and ISDN Systems 30, 107–117 (1998)
22. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)
23. Newman, M.E.J.: Analysis of weighted networks. Physical Review E, 056131 (July 20, 2004), arXiv:cond-mat/0407503 v1
24. Jeffrey Heer, J., Agrawala, M.: Software Design Patterns for Information Visualization. IEEE Transactions on Visualization and Computer Graphics 12(5), 853–860 (2006)

25. Network Workbench, http://nwb.slis.indiana.edu/
26. Pajek, http://vlado.fmf.uni-lj.si/
27. NetDraw, http://www.analytictech.com
28. JUNG, http://jung.sourceforge.net
29. netminer, http://www.netminer.com