Visual Analytics Using Graph Sampling and Summarization on Multitouch Displays

Nicholas G. Lipari, Christoph W. Borst, Mehmet Engin Tozal

School of Computing and Informatics
University of Louisiana at Lafayette, Lafayette, LA 70504 USA
{nlipari, cxb9999, metozal}@louisiana.edu

Abstract. Private industry datasets and public records contain more information than any algorithm can efficiently process or any person can reasonably interpret. This is a basic problem faced by researchers in visual analytics. Graph visualizations (a common large dataset representation) can organize relationships and entities in a visually accessible manner. Our work applies graph sampling and summarization to the interactive visualization of complex networks. We implemented several unbiased sampling techniques to facilitate large scale graph analysis. Moreover, we show biased sampling techniques can improve visualization by emphasizing key graph nodes. We combine algorithmic processing with human interpretations by allowing users to adjust sampling parameters, inspect sample graph visualizations, and compare sample distributions. Summarization also reduces graph complexity. By adjusting rendered graph density, users can navigate and maintain constant on-screen density.

1 Introduction

We are studying the combination of complex network analysis with multitouch interaction and visualization. A complex network is a class of graphs with non-trivial topological structures [1]. Researchers have studied these networks across different fields of science and engineering such as economics, biology, computer science, and physics. Examples include human disease networks [2], citation networks [3], online social networks [4], and Internet topology maps [5]. Disease networks represent diseases as graph nodes and connect nodes by links if they tend to co-occur in patient diagnostic records. Medical professionals study these graphs to discover the relations between multiple diseases. Nodes in citation networks denote academic publications, and a directed link between two nodes represents one paper referencing another. Citation networks reveal the important papers in a scientific domain and how authors reference those papers. Similarly, Internet topology maps at the autonomous system (AS) level encode such systems as nodes and their relations as edges. Network engineers and researchers use AS graphs to understand the Internet's dynamics and evolution.

Studying large-scale complex systems is often challenging. Storing an entire graph in memory, processing to extract useful information, and visualizing for decision mak-

This is an author-formatted version. The original publication is available at www.springerlink.com. ISVC 2016, Part I, LNCS 10072, pp. 462-471.

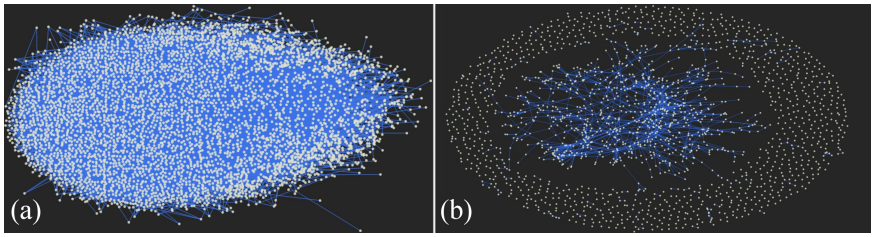

Fig. 1: Generic example of network dataset. (a) No meaningful information can be obtained from the complete graph. (b) A random path sampling exposes visible structure within the graph and allows more complex analysis to be conducted

ing are resource and time consuming tasks [1]. Moreover, accommodating human-computer interaction for cognitive analysis is difficult due to visually dense graph layouts, relatively few high importance nodes, and lack of good interaction techniques.

Graph sampling is a useful approach that provides the ability to select important components and relations of a large-scale source graph. As seen in Fig. 1, an otherwise densely crowded graph with no discernable structure (Fig. 1a) can be sampled to reveal examples of paths and neighborhoods (Fig. 1b) that were not immediately visible. Within some margin of error and for certain graph topologies [6], sampling can also preserve individual structural characteristics (e.g., degree distribution, path length distribution), select smaller and representative subgraphs from the source graph, and simplify challenges related to interaction and understanding.

In this study, we have increased the feasibility of large graph visualization and analysis by integrating several graph sampling and summarization routines into a multitouch graph analytics application. Sampling occurs as an offline or disk-based operation, not requiring the source graph to be loaded into memory. The sampling algorithms provide either biased or unbiased results, allowing users to quickly find nodes with high importance or estimate large-scale graph properties, respectively. Users can interactively adjust sampling parameters (e.g., percent of nodes from the source graph), highlight a range of nodes based on a centrality metric or structural characteristic, and control the on-screen graph density.

We begin by reviewing the relevant literature on graph sampling and visualization. Our system overview appears next, including organization and communication descriptions. We then detail the sampling, visualization, and summarization components. Last, we demonstrate the utility of our approach with an example use case by interactive visual analysis of a human disease co-occurrence dataset [2] consisting of 16,459 diseases labeled by their ICD-9 codes and 6,088,553 edges among them.

2 Related Work

The visual analytics field has produced several works on graph analysis. GraphViz [7] and Network Repository [8] are among those combining graph sampling with visual analytics. A web-based graph analytics system uses edge sampling with a sample and

hold strategy to support streaming graphs. The sampling algorithm used in [7] and [8] maintains a small amount of state and samples over the streaming graph.

A multi-touch interaction survey by Ingram et al. [9] reviewed multiple methods of selection and interaction, as well as multiple analogies for manipulating virtual objects. The authors suggest a framework centering around interaction feedback, physics-based virtual object movements, and an understanding of prior experience of users. TouchWave [10] and TouchViz [11] allow users to manually interact with time-series data in chart form. TouchWave attempts to improve the legibility of chart data, the ability to make comparisons, and the scalability to interpret dense datasets. Graphite [12] and Apolo [13] emphasize more micro-scale visualizations of graphs or networks. In Graphite, users draw a query pattern of nodes and edges, and the system returns exact or approximate subgraphs matches within a larger network. Our work allows users to participate in the sample creation process via touch interactions and presents visualizations such as graph diagrams and statistical plots.

The main form of graph rendering for the above solutions has been node-link diagrams. Hu [14] discusses several fundamental diagram layouts. Physics-based models arrange nodes according to some process found in nature (e.g., spring forces and electrically charged particles) [15]. While also used to improve the performance of layout generation algorithms [14], graph coarsening can simplify an existing layout based on topology and geometry. A common coarsening method collapses edges incident on less important nodes and requires summaries to be precomputed [16]. Another greedy approach to graph simplification [17] forms supernodes by examining neighbor lists and grouping the nearest matches between all pairs of nodes. Our work includes a summarization method that uses edge collapsing similar to [16] and arranges items in heaps as suggested by [17]. Unlike these, however, we compute summaries dynamically by storing collapsed graph components in a second heap.

Graph sampling has a wide range of applications [6], such as extracting key components of a graph, estimating structural properties of a graph, and visualizing large-scale graphs. Three classical examples of graph sampling are node sampling, edge sampling, and exploration based (traversal-based) sampling [18]. Node sampling and edge sampling involve selecting nodes and edges according to some predefined distributions. Traversal-based sampling approaches start from one or more initial nodes and sample the graph by expanding toward the neighbors of the current nodes. Many traversal-based sampling approaches use well-known graph exploration algorithms including Breadth-First Search, Depth-First Search, and Random Walk.

3 System Overview

Before detailing the sampling and visualization contributions of our research, we give an overview of our framework's organization and internal communication. To assist users in navigating through datasets and finding structure from noise, we developed a graph analytics system wherein users interactively explore large, dense graph datasets while requesting different samples and analysis. As illustrated in Fig. 2, our solution is

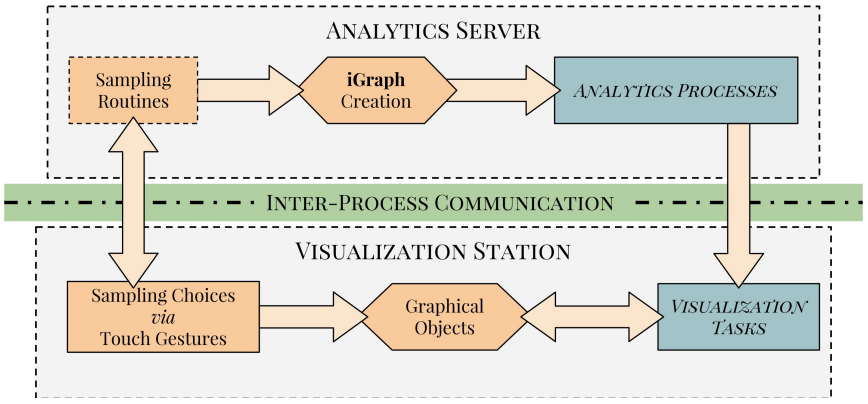

Fig. 2. Analytics Server, Visualization Station, and component communication. Users interact with touch gestures to select sampling algorithms and parameters. Sampled graphs are stored in an igraph object on the Analytics Server. Users may also perform various visualization tasks, which spawn corresponding analytics processes. The Visualization Station renders the results as graphical objects (e.g., node-link diagrams and ECDF plots)

separated into two main components: an analytics server and a visualization station, with an inter-process communication protocol using a message-passing approach.

The analytics server is responsible for reading graphs from a data store (e.g., SQLite, MongoDB, Neo4j), sampling graphs, and computing topology metrics. The igraph software suite [19] stores sampled datasets, manages graph metadata (e.g., node labels and positions), computes centrality measures, and generates graph layouts. Several different sampling techniques can reduce the source dataset as users tune the sampling parameters (e.g., percentage of nodes or edges) for best visualization and analysis of the resulting graph. The sampling routines operate in worker threads and communicate progress to the visualization station.

The visualization station renders graph objects, presents analytics results in forms understandable to users, and processes touch input from users. Interactions (touch inputs or mouse clicks) may be visualization specific, such as navigation gestures, or require communication to and from the analytics server. The system interprets these actions and issues requests to the analytics server. In a common sequence of actions, a user specifies a dataset choice and sampling type, which the station sends to the analytics server. Upon sample completion, the visualization station receives and stores the sample's topological information (e.g., node degree and sorted neighbor lists) and graph layouts in a form efficient for rendering.

4 Methods

Currently, we have implemented inter-process communication over a network socket to match specialized hardware with memory-intensive (i.e., analytics) or graphics-

intensive tasks. Communication could also be implemented on a single computer with a shared memory object. Multiple graphics workstations can connect to a single high-performance analytics server over the network. The analytics server maintains a connection to the on-disk data source and manages the in-memory samples. Separate worker threads compute graph samples, topological metrics, and layout positions.

In the following section, we present how sampling and summarization can reduce the computational, cognitive, and rendering complexity of large graphs. Sampling types are distinguished by their bias or lack of bias. We discuss the benefits of both cases and show how selected algorithms compare in this regard.

4.1 Graph Sampling Approaches

In general, a sample graph generated through an arbitrary sampling procedure does not preserve all characteristic (degree, betweenness, and clustering coefficient) distributions of the population graph. As a result, one needs to align the sampling procedure with respect to the characteristic to be preserved. In recent years, researchers have proposed several solutions, e.g., [20], to construct randomly uniform sample graphs. These unbiased sampling strategies are effective for various characteristic estimation problems. On the other hand, biased sampling techniques may be useful for analysis and visualization purposes. To illustrate, biased sampling is a cost effective solution to extract nodes, edges, or paths similar to each other in terms of a centrality measure in a graph. Similarly, visualizing a graph obtained through a biased sampling allows users to focus on important nodes, edges, or paths in the graph.

We implemented several disk-based sampling algorithms including random node sampling, induced random edge sampling, random path sampling, random walk, and Metropolis-Hastings sampling. Random node sampling generates a sample graph by selecting a number of vertices with equal probability from a population graph. This technique is effective in estimating local properties of nodes in a population graph. Similarly, induced random edge sampling generates a sample graph by randomly selecting edges and the endpoint nodes (induced nodes) from a population graph. Induced edge sampling has bias towards high degree nodes. Those nodes have more edges to be sampled compared to low degree nodes. Ideally, random path sampling generates a sample graph by selecting paths at random among all possible shortest paths in a graph. However, generating all shortest sample paths in a graph is a costly operation. We approximated random path sampling by randomly selecting a source and a destination node in the population graph and computing the shortest path(s) between them. This technique produces good estimations for path-based node properties, e.g., betweenness. Random walk sampling generates a sample graph by first selecting seed nodes and randomly selecting an edge toward one neighbor at each step. This approach is also biased toward high degree nodes. Metropolis-Hastings sampling removes the bias in random walk sampling by reweighting the neighbor selection procedure to reduce transitions toward high degree neighbors.

4.2 Visualization and Interaction

A visual analytics system for large graphs should create meaningful graph representations and draw a user's attention to important features. The system should also mitigate the impact of graph scale on visual performance. We designed and implemented the multitouch visualization station (Fig. 3) in order to provide interactive graph analysis to users. The basic navigation and selection tasks follow standard, intuitive gestures such as tap to select and pinch to zoom. Aside from these, we have tested several multi-touch interactions to improve the exploration of graphs. Our application-specific interactions include sampling algorithm selection, adjustment of sample size, Empirical Cumulative Distribution (ECDF) plot management, ECDF range selection, and interactive summarization, all accessed through a touch interface.

Graph sampling controls allow users to interact with the analytics server and achieve the most appropriate sample for their needs. For example, to achieve a sample graph with high betweenness nodes (which roughly are the nodes appearing on many paths), users can select a path sampling or walk-based sampling algorithm with bias towards high-betweenness nodes. The sampling thread updates progress percentage and estimated time during execution.

When the sampling algorithm completes, the visualization station receives the node and edge lists from the analytics server. We currently support two graph-rendering methods: pre-rendered and direct-rendered. The pre-rendered method creates a high-resolution texture of the graph drawing to allow faster redraw and interaction rates. We primarily choose the rendering method based on node and edge counts; graph samples are pre-rendered when the number of graphical objects would cause rendering rates below real-time interaction constraints (about 60 fps). Except, when users update graph summarization (discussed below), we direct-render intermediate layouts and reevaluate the rendering method choice with the summarized graph.

ECDFs objectively reflect the structure and connectivity in the graph sample. Users can analyze the sampled graphs by their centrality metric distribution. We aggregate per-node metrics from igraph and render an empirical CDF plot (Fig. 3b-d, right). A two-finger pinch gesture on the plot can specify a range of plot values to highlight or filter the relevant nodes. To facilitate this, we maintain a hash-map that relates empirical values to lists of node labels. The visualization station highlights nodes in the range, rendering a label above each node's location. The analytics server can create a subgraph induced by these vertices.

Summarization is a related but alternative method to reduce the size and complexity of graphs. While a sample graph is a subset of the population graph, a summary graph is a transformation of the population. That is, the nodes or edges of a sample graph also appear in the population graph, whereas the nodes and edges in a summary graph may not appear in the population. We implemented a summarization approach based on concepts from [16] and [17]. Users can simplify the graph by removing unimportant nodes and edges, reducing visual clutter, dynamically updating the graph based on interactions (e.g., pinch to zoom), and maintaining a constant visual density during navigation.

Users adjust two parameters to contract or expand nodes with respect to their neighbors. A user-defined threshold metric (e.g., node degree) is directly updated by

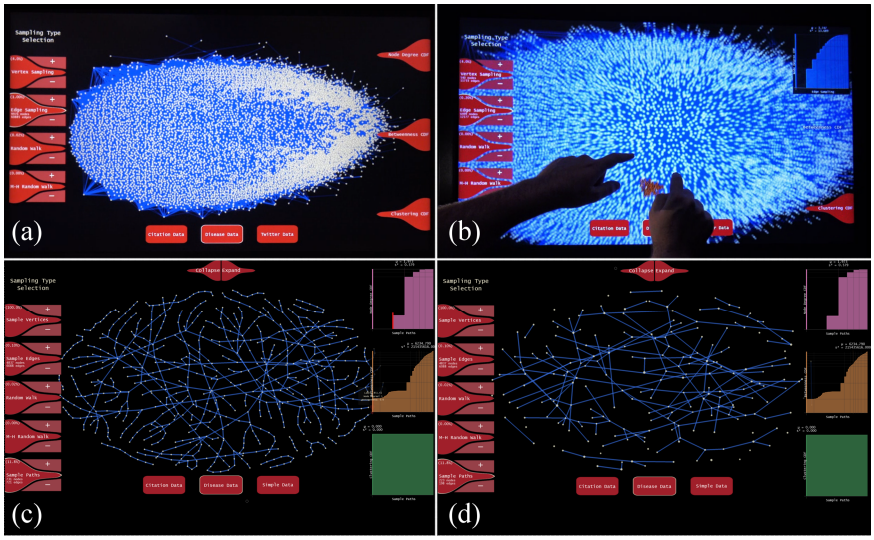

Fig. 3. Graph sample creation. (a) Graph with one percent of edges. The sample has approximately 10,000 nodes and 60,000 edges. Sampling parameter can also be adjusted up (+) and down (-) by a constant ratio. (b) Two-finger navigation allows users to narrow the scope of graph to be rendered. (c) Path sampling with 731 nodes and 723 edges, also described by ECDFs to the right. (d) The path-sampled graph after two summarization levels

touch input, and the zoom level (i.e., scale) is extracted from the navigation interactions. To choose the graph nodes that will collapse into others, we compare each node's metric against the product of these two parameters. We collapse all (source) nodes below the threshold onto the neighbor (sink) with the largest metric value. Node movement animates along the edge connecting the two nodes. Other summarization methods, e.g., [16], require costly, graph-wide computations. We are able to summarize the graph by only visiting the nodes and edges that the routine will update. To maintain interactive rates, the routine uses two heaps to determine which nodes to collapse or expand, based on changes of the threshold.

Nodes move back and forth between collapsed and visible heaps based on the current setting of the threshold and scale (zoom) parameters. Two animation steps translate nodes and edges in the collapse and expand queues. Once the animations complete, we update the heaps based on the summarized graph's topology. Through summarization, users can interactively remove less important nodes and reduce the clutter caused by unneeded edges. By controlling summarization with two parameters, the graph can be explored with a given density regardless of navigation choices. The algorithm is currently implemented in our application and does not suffer any large slowdowns for even dense graphs such as Fig. 3a.

5 Use Case Example

We now provide an example of our visual analytics system with a large dataset from the medical field. We have used several real world datasets for visualization, including a citation network [3] and a disease co-occurrence graph [2], as well as synthetic graphs (e.g., Erdos-Renyi, Albert-Barabasi, Watts-Strogatz) [21] for metric distribution comparison. Rendering these large graphs as node-link diagrams and performing analytics tasks can tax even the most advanced workstation.

The disease co-occurrence graph [2] being manipulated in Fig. 3a contains approximately 10,000 nodes and 60,000 edges (a one percent edge sample). Our analytics and visualization modules mediate the impact that large datasets have on the system's responsiveness. Fig. 3b demonstrates that, even after scaling and navigating the graph, nodes are still tightly clustered and individual edges cannot be distinguished. Due to the sample's topology, automatic layout generation also fails to reveal patterns within the graph. We are unable to see any apparent structure such as the most highly connected nodes or distinct paths of nodes connected by edges.

The path-sampling used in Fig. 3c is biased towards high betweenness nodes (middle ECDF plot), known to be of relative importance in many datasets. The sampling also produces a coarse approximation of the larger graph's degree ECDF (upper-right plot within Fig. 3c). The resulting graph has about 750 nodes and edges, and the smaller graph size allows fast estimation of betweenness centrality. The automatic layout contains readily visible hubs and paths, placing branching structures and long paths throughout the graph with minimal overlap. The shorter, isolated paths surround the graph's border without occluding other structures.

Fig. 3d shows how summarization further simplifies the sampled graph by collapsing low degree nodes onto their largest neighbors. After two levels of summarization (i.e., collapsing nodes with degree two and lower), a clearer image appears of the important nodes in the sample. Long paths with no branches are simplified to single edges, larger connected components are more clearly linked by paths, and we emphasize those nodes containing collapsed neighbors with an increase in size. Users can visit and expand the collapsed neighborhoods by navigating and adjusting scale. Users can more clearly understand where the changes to the graph occur with this gradual visual feedback.

6 Conclusion and Future Work

Among our contributions to graph visualization and interaction, we have demonstrated how biased and unbiased sampling simplify finding information about a dense graph dataset. With biased sampling, users can visualize nodes with similar centrality metric values, such as nodes with high betweenness. Unbiased sampling allows us to find large-scale graph properties more quickly without requiring the entire graph be loaded into memory or waiting a prohibitive amount of time. Some metrics require visiting every node in the graph (degree distribution), while others require the measurement of all paths in the graph (betweenness). By producing an unbiased sample with Metropo-

lis-Hastings Random Walk, we can more efficiently estimate the degree distribution of the population graph.

Summarization further reduces clutter in graph renderings, maintaining a constant on-screen density. The graph visualization can temporarily hide nodes of low importance, based on user interactions (threshold and scale). With the combination of sampling and summarization, users can attain a clearer understanding of their data and request more detailed analysis.

We plan to incorporate other interactive visualization methods and provide users with greater control of the various sampling methods. Users will be able to access summaries and detailed subgraphs to help understand analytics results. Users will specify an area of the sample and request further analysis from the original dataset (e.g., several random walks or paths starting at the area of interest). To help bring the user's attention to these areas of interest, we will perform a highlighting of subgraphs based on topology metrics. Instead of manually selecting ECDF regions and performing time-consuming search tasks, we can highlight nodes based on the biases of the selected sampling method or similarities to nodes and subgraphs the user has selected. Finally, we plan to extend the sampling and analytics modules by introducing additional sampling techniques and centrality measures, respectively.

Acknowledgments

This research was funded by the Center for Visual and Decision Informatics (CVDI), an Industry/University Cooperative Research Center of the National Science Foundation, members from an Industry Advisory Board, and University matching funds. Information about CVDI can be found at nsfcvdi.org. We thank Dr. Raju Gottumukkala, Sivarama K. Venna, and Maryam H. Beisafar for earlier Analytics Server work.

References

1. Cohen, R., Havlin, S.: Complex Networks: Structure, Robustness and Function. 1st edn. Cambridge University Press, New York (2010)
2. Hidalgo, C.A., Blumm, N., Barabási, A.-L., Christakis, N.A.: A Dynamic Network Approach for the Study of Human Phenotypes. PLoS Comput. Biol. 5 (2009) e1000353. doi:10.1371/journal.pcbi.1000353.
3. Gehrke, J., Ginsparg, P., Kleinberg, J.: Overview of the 2003 KDD Cup. ACM SIGKDD Explorations 5 (2003) 149–151
4. Jiang, J., Wilson, C., Wang, X., Sha, W., Huang, P., Dai, Y., Zhao, B.Y.: Understanding latent interactions in online social networks. ACM Trans. Web 7 (2013) Article 18
5. Tozal, M.E.: The Internet: A System of Interconnected Autonomous Systems. In: Rassa, B. (ed.): Proceedings of the 11th Annual IEEE Systems Conference. Orlando, FL (2016) 1–8
6. Cem, E., Tozal, M.E., Sarac, K.: Impact of sampling design in estimation of graph characteristics. In: Wang Y., Xu, K. (eds.): Proceedings of the 32nd International Performance Computing and Communications Conference (IPCCC). San Diego, CA (2013) 1–10

7. Ahmed, N.K., Rossi, R.A.: Interactive Visual Graph Analytics on the Web. In: Hamilton, C. (ed.): Proceedings of the International 9th AAAI Conference on Web and Social Media. Oxford, UK (2015) 566–569
8. Rossi, R.A., Ahmed, N.K.: An Interactive Data Repository with Visual Analytics. ACM SIGKDD Explorations Newsletter 17 (2015) 37–41
9. Ingram, A., Wang, X., Ribarsky, W.: Towards the Establishment of a Framework for Intuitive Multi-touch Interaction Design. In: Tortora, G., Levialdi, S., Tucci, M. (eds.): Proceedings of the International Working Conference on Advanced Visual Interfaces, Capri Island, Italy (2012) 66–73
10. Baur, D., Lee, B., Carpendale, S.: TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs. In: Shaer, O., Shen, C. (eds.): Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, Cambridge, MA (2012) 255–264
11. Drucker, S.M., Fisher, D., Sadana, R., Herron, J., Schraefel, M.: TouchViz: A Case Study Comparing Two Interfaces for Data Analytics on Tablets. In: Mackay, W.E. (ed.): Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France (2013) 2301–2310
12. Chau, D.H., Faloutsos, C., Tong, H., Hong, J., Gallagher, B., Eliassi-Rad, T.: GRAPHITE: A Visual Query System for Large Graphs. In: Wu, X. (ed.): Proceedings of the IEEE International Conference on Data Mining Workshops, Las Vegas, NV (2008) 963–966
13. Chau, D.H., Kittur, A., Hong, J., Faloutsos, C.: Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning. In: Tan, D. (ed.): Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC (2011) 167–176
14. Hu, Y.: Algorithms for Visualizing Large Networks. In: Naumann, U., Schenk, O. (eds.): Combinatorial Scientific Computing. 1st edn. Berkeley, CA (2012) 525–545
15. Eades, P.: A heuristic for graph drawing. Congressus numerantium 42 (1984) 146–160
16. Hendrickson, B., Leland, R.W.: A Multi-Level Algorithm for Partitioning Graphs. In: Karin, S. (ed.) Proceedings of the ACM/IEEE Conference on Supercomputing, San Diego, CA (1995) Article 28
17. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph Summarization with Bounded Error. In: Lakshmanan, L., Ng, R.T., Shasha, D. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, Vancouver, BC (2008) 419–432
18. Hu, P., Lau, W.C.: A survey and taxonomy of graph sampling. CoRR 1308.5865 (2013)
19. Csardi G., Nepusz, T.: The igraph software package for complex network research. InterJournal: Complex Systems (2006) Article 1695
20. Kurant, M., Markopoulou, A., Thiran, P.: Towards Unbiased BFS Sampling. IEEE J. Sel. Area Comm. 29 (2011) 1799–1809
21. Albert, R., Barabási, A.-L.: Statistical mechanics of complex networks. Rev. Mod. Phys. 74 (2002) 47–97