Vol.2010-BIO-22 No.5
2010/7/29

Interactive, Multi-Scale Navigation of Large and Complicated Biological Networks

Thanet Praneenararat[†], Toshihisa Takagi[†, ††, †††], and Wataru Iwasaki[†]

Networks are widely used to represent many biological data. However, their visualization often becomes too complicated to be interpreted, particularly if they contain hundreds or thousands of entities. Here, we present an interactive multi-scale navigation method for large and complicated biological networks, powered by an ultrafast graph clustering technique and a biological-property-based clustering. Similar to Google Maps, this method provides appropriately abstracted views at any magnification ratio and enables researchers to effectively discover knowledge from network data.

1. Introduction

Biologists in the post-genomic era face difficulties managing and utilizing the volume of biological data that are available via the Internet or obtained through high-throughput experiments, which are becoming increasingly common. In many cases, binary relationships (sets of elements and 1-to-1 associations between them) are used to represent such datasets, which often include protein-protein interactions (PPI), correlatively expressed gene pairs, genetic regulatory relationships, and signal transduction/metabolic reactions.

These relationship data are conventionally presented using network visualization where nodes (vertices) and edges correspond to elements and associations, respectively [1][2]. Network visualization is used because it is typically assumed to be more interpretable by humans than a long list of associations. High quality visualization should allow for effective investigation of the information, hypothesis generation, and biological discovery [1]. Unfortunately, network representations often fail to effectively convey information to readers in cases where the networks are large and complicated (e.g., > 100 edges). The drawings of such networks, referred to as “hair balls” [2], occur frequently when analyzing high-throughput biological data and fail to aid biologists. Effective navigation approaches are required to realize the full potential of large, binary-relationship data sets [3].

Hierarchical clustering is a technique used with many types of data, including networks or graphs, that meaningfully groups data elements in a recursive manner, thereby producing a hierarchy, or tree, of clusters [4] (Figure 1). Higher levels in the hierarchy contain fewer, larger clusters, each of which encompasses more data elements (or nodes, in the case of networks) than lower levels. In the case of hair-balls, some methods [5][6][7][8][9] use hierarchical clustering to create an interpretable visualization by displaying only the high level clusters, thereby reducing the number of elements in the figure and abstracting the networks (e.g., the top panel in Figure 1). By descending the hierarchy and showing the actual members of each cluster, detailed information can still be intuitively shown at a particular scale (e.g., the dotted arrows and regions in Figure 1). A recent study reported that natural networks display hierarchical properties [10], suggesting that hierarchical clustering of biological networks is both reasonable and promising.

[†] Department of Computational Biology, the University of Tokyo, Kashiwa, Chiba, Japan
[††] Database Center for Life Science, Research Organization of Information and Systems, Bunkyo-ku, Tokyo, Japan
Information Processing Society of Japan
[†††] National Institute of Genetics, Research Organization of Information and Systems, Mishima, Shizuoka, Japan

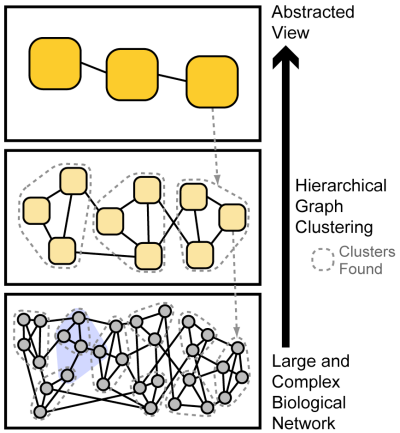

Figure 1 Application of hierarchical graph clustering to network navigation

Despite the advantages of hierarchical clustering, existing visualization methods using this technique have some drawbacks that hinder effective investigation of large biological datasets. First, some methods [6][7][11] require researchers to provide information on hierarchies or clusters, data which is usually not known in advance. Second, existing methods do not allow for flexible navigation beyond fixed cluster boundaries [5][6][7][8][9]. In other words, they can visualize the members of one cluster at a time but do not support visualization and navigation of members of different clusters, despite the fact that nodes/clusters of interest to biologists may belong to various high level nodes in the hierarchy (e.g., nodes in the light-blue area in Figure 1). Third, existing methods are inappropriate for interactive, real-time navigation. Researchers frequently change their focus in the course of biological investigation to generate hypotheses and need to visualize different sets of nodes/clusters. Thus the long running times (minutes to hours) needed to produce the abstractions are unacceptable [5][8][9]. Methods that can provide appropriate abstractions of any given portion of the network rapidly and automatically, such as those that process about 100,000 nodes in seconds, are therefore necessary for efficient, interactive biological investigation. In addition to the previously mentioned problems, the clustering techniques employed by existing methods are often insufficient for abstracting large networks to a level that is simple enough for interpretation [5][8][9]. Recent investigations have revealed that in some common biological datasets, hub-like nodes tend to connect with low-degree nodes and the majority of nodes interact with only few partners (e.g., yeast PPI networks) [12]. Large, densely connected regions of the network are therefore quite few in such networks; instead, small, densely connected modules are more frequently found. Consequently, even the highest level of the created hierarchies can contain over 100 clusters, resulting in cluttered and difficult to manage visualizations. Therefore, means for further abstraction are required to allow for effective navigation of large biological networks.

We developed an interactive, multi-scale network navigation method with three advantages: 1) our method can work without a user-provided hierarchy, 2) the method can rapidly, automatically, and interactively produce abstractions of any region of the network, including nodes/clusters belonging to different ancestors in the hierarchy, and 3) an intuitive visualization with a manageable amount of information is reliably produced at every step of navigation. The effectiveness of our method was confirmed using real yeast protein network data. Our approach will aid modern biologists faced with large and complicated network data.

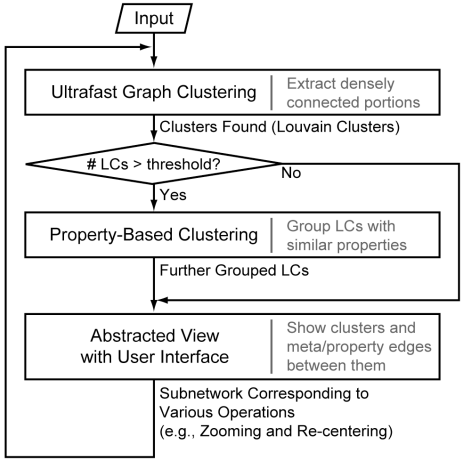

Figure 2 Overview of the method

2. Methods

2.1 Overview

Our method consists of three components: an ultrafast graph clustering component, a property-based clustering component, and an interface that presents an abstracted view and permits researchers to flexibly choose nodes/clusters (Figure 2). First, the method abstracts

the whole network using the ultrafast graph clustering component. It detects topologically dense, connected regions, which may correspond to biologically meaningful clusters, such as protein complexes. Second, in case the abstraction is insufficient because of the characteristics of the biological network, the property-based clustering component further abstracts the network to an extent sufficient for visual interpretation. This component groups clusters with similar biological properties by utilizing the fact that biological entities are often assigned property information, such as Gene Ontology (GO) terms. Third, the resultant clusters/nodes are immediately displayed with meta-edges and property edges, which represent the numbers of edges that exist between any members of two clusters and the similarities between their properties, respectively. While showing the abstracted view, the interface allows researchers to interactively zoom, move laterally beyond cluster boundaries, focus on an arbitrary set of clusters/nodes, etc. Any subset of the entire network of particular interest to the researcher can be fed into the clustering components and the abstracted view of that cluster is displayed. This cycle can be completed in a few seconds on a typical PC with a CPU of about 2 GHz and a memory of about 1 GB for datasets with 100,000 nodes, permitting truly interactive navigation of large biological networks.

2.2 Ultrafast Graph Clustering

The ultrafast graph clustering component detects clusters in networks by finding densely connected sets of nodes where connections of nodes *within* the sets are stronger/denser than connections *between* nodes inside and outside of the set. This metric is called the modularity, or Q function [13], and numerous graph clustering algorithms have been developed to identify clusters that optimize modularity [13][14][15][16][17][18]. The Newman-Girvan is a well-known, pioneering algorithm that iteratively removes edges most likely to lie between clusters, splitting the clusters into two, until no edges remain [13]. This process results in a dendrogram (a tree showing the order of the splits) and the best clustering can be identified from this tree by choosing the split with the highest modularity. This algorithm has a high computational cost, as it requires a traversal of all remaining edges at every step. Until recently, the best known algorithm developed to overcome this shortage with near-linear time complexity was devised by Wakita and Tsurumi [16]. However, the speed of this algorithm was still insufficient and the quality of the clusters produced had room for improvement when incorporated into an interactive navigation of large networks (e.g., human gene networks of > 20,000 nodes) [17]. Recently, Blondel et al. developed a breakthrough algorithm for quickly identifying high modularity clusters in huge networks of about 100,000 nodes (the Louvain algorithm [17]). We found that this algorithm for finding meaningful communities in large and complicated networks could be applied to the problem of interactive navigation.

The Louvain algorithm works in two phases, as follows:

1. Starting from the state that each node belongs to a cluster different from every other node, for each node the algorithm considers its neighbors' clusters and moves the node to a neighboring cluster. The cluster to be joined is determined by choosing the movement that results in the highest positive modularity gain among all possible movements to the node's neighboring clusters. If no movements result in a positive gain in modularity, the node is not moved. This process is repeated until no members are added to/removed from any clusters and yields clusters with the maximum local modularity.
2. Every cluster from phase 1 is then treated as a new node. For each pair of new nodes, an edge connecting them exists if there is at least one edge between any member of one of the new nodes and any member of the other. Edge weights are determined based on the number of previous edges. Self-loops are drawn on nodes to represent corresponding inter-cluster edges.

The output of phase 2 is then fed back to phase 1 and the algorithm iteratively runs these two phases until no additional changes are made.

This algorithm can finish clustering networks of 70,000 nodes in one second [17]. Thus, it works swiftly on many biological networks that generally contain less than 100,000 nodes (e.g., yeast or human PPI networks). The ultrafast speed of the algorithm is essential for accomplishing the goal of truly interactive navigation of large networks. Additionally, if slower property-based clustering is to be executed afterwards, the graph clustering method displays another advantage in that it significantly reduces the number of clusters to be input to the next clustering.

The clusters produced by this algorithm, which we call Louvain Clusters or LCs, are characterized by high modularity. It has been shown that clusters with high modularity in biological networks correspond to biologically functional units (e.g., protein complexes in PPI networks and transcriptional modules in gene regulatory networks [19]). Thus, the LCs are expected to be intuitive and meaningful groups in navigation of biological networks.

2.3 Property-Based Clustering

The property-based clustering component aims to decrease the complexity remaining after the application of the Louvain algorithm by further grouping LCs based on property information typically associated with the nodes (Figure 3). The visualization step displays the clusters resulting from the property-based clustering instead of those generated by the graph clustering approach, thereby reducing the number of clusters on the screen. The VisANT tool works similarly to our property-based clustering and offers integrated visualization of the GO hierarchy and user-specified networks, but it requires the user to manually create clusters

containing the same GO terms [20]. In contrast, our property-based clustering *automatically* generates clusters having similar properties to achieve interactive navigation.

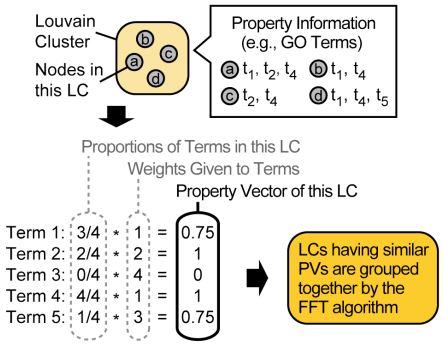

Figure 3 Property-based clustering

Let N be the number of nodes in the original input graph and L be the number of LCs. For each n, where $1 \le n \le N$, node v_n has a set of terms, $T(v_n)$, that denotes the properties of the node (e.g., a set of GO terms). A weight, $w(t)$, is given to each term t to quantify its importance (e.g., properties that are rare and/or of particular interest to researchers may be given higher weights). Let $T_{all} \equiv \bigcup_{1 \le n \le N} T(v_n)$ and $T_{all} = \{t_j \mid 1 \le j \le |T_{all}|\}$. For each LC, LC_l, where $1 \le l \le L$, let $Prop(t, \mathrm{LC}_l) = |\{v \in \mathrm{LC}_l \mid t \in T(v)\}| / |\mathrm{LC}_l|$. Then the property vector for LC_l or $\mathbf{PV}(\mathrm{LC}_l)$ is a $|T_{all}|$-dimensional vector whose j-th element is the score of term j, which is calculated as $w(t_j) Prop(t_j, \mathrm{LC}_l)$. Next, the similarity between two LCs, LC_a and LC_b, is given as a normalized dot product of the two property vectors; that is, $Sim(\mathrm{LC}_a, \mathrm{LC}_b) = \mathbf{PV}(\mathrm{LC}_a) \cdot \mathbf{PV}(\mathrm{LC}_b) / |\mathbf{PV}(\mathrm{LC}_a)| |\mathbf{PV}(\mathrm{LC}_b)|$. Then LCs having similar property vectors are grouped by the Farthest First Traversal *K*-center (FFT) algorithm [4]. The FFT algorithm is a complexity-reducing variant of the *K*-means algorithm, where initial *K* cluster centers are chosen as follows. The first center (vector) is chosen randomly and each remaining center is determined by greedily choosing a vector farthest from the set of already chosen centers. The rest of the vectors are assigned to the cluster to which they are most similar.

There are two main advantages of the property-based clustering. First, the property-based clustering allows researchers to directly control the number of clusters shown on the screen through the parameter *K* of the FFT algorithm. To solve the problem of the cluttered visualization produced by applying only the graph clustering, the number of clusters shown on the screen must be decreased to an extent that biologists can manage to interpret. In addition, because the preferred numbers of clusters on the screen might differ according to the circumstances, it is important that biologists be allowed to adjust the number of clusters displayed. Second, because the clusters generated by the property-based clustering are based on the property information that carries biological meaning, the clusters are expected to be highly intuitive. Note that it is also possible to continue Louvain clustering to further reduce the number of clusters, even if the gain in modularity becomes negative. However, in such cases, the biological intuitiveness of the clusters produced might be lowered due to the decreased modularity [19]. Therefore, it might be better to adopt another reliable source of information, in addition to the topology of the networks, at this stage.

3. Results and Discussion

The proposed method was implemented as a Java 6 Swing application with a graphical interface for flexible navigation (see the Results section for detail). The JUNG (Java Universal Network/Graph Framework) library [21] was employed to create the visualization. Three input files are required to run the application: a node list file, an edge list file, and a property information file. The node list file describes node names, property terms annotated with the nodes, and database names and IDs used in those databases (e.g., SGD for yeast proteins). The database information is used to provide URL links. The edge list file contains connected pairs of node names and the weights of the connections (weights describe how strongly the nodes are connected). The property information file describes the property terms in the node list file: terms' IDs, names, display names (used in labeling clusters in abstracted views), namespaces, default weights, and their parent terms. In the case of the GO property information file bundled with the software, the default weights are terms' depths in the GO hierarchy. This treats more specific terms as more important properties. In addition, each term belongs to one of three namespaces (biological process, molecular function, or cellular component). By using the namespace information, researchers can put heavier weights on all biological process terms at once if they want to group nodes having similar biological process

terms, rather than other namespace terms. If parent terms are provided for each term, they are automatically assigned to the nodes that the term annotates as well. The *is_a* and *part_of* relationships in GO are handled by this entry. The implemented software, NaviCluster, is available at http://navicluster.cb.k.u-tokyo.ac.jp/, and works on any platform that can run Java 6. The program has a minimum memory requirement of 1 GB for networks of about 100,000 edges.

To confirm the novelty and capabilities of the present method, we compared it to other existing visualization methods such as GenePro [6], Power Graphs [8], VisANT [20], BioLayout Express^{3D} [7], and jClust [9] (Table 1). Our method was the only one capable of representing the hard-to-manage and complicated visualization of the overwhelming numbers of nodes and edges and providing the capability to navigate networks beyond cluster boundaries. GenePro, BioLayout Express^{3D}, and jClust can visualize clusters of nodes, but only at a single level. They do not support visualization of recursive clustering. VisANT provides multi-scale visualization; however, the user must manually create metanodes (equivalent to clusters) themselves. CyOog (Power Graphs) hierarchically visualizes power nodes (equivalent to clusters) created by the Power Graph algorithm, but the speed is not fast enough to be used for interactive navigation of large biological networks.

It should be noted that our method is highly extendable in several ways. First, any type of property information, not just GO categories, can be used in the property-based clustering. For example, if a researcher is interested in diseases, she/he can use disease names associated with proteins derived from disease databases to investigate PPI networks by clustering proteins related to similar diseases. Second, because the clustering components of the present method can abstract any sub-network very rapidly, any interactive function for producing network views of interest can be achieved if modules for selecting appropriate clusters/nodes are implemented. For example, it is easy to devise a module that interactively produces networks of genes regulated by a selected regulatory factor, given information on gene regulatory relationships. Third, the presented method is not limited to biological applications. In fact, it is general enough to be tailored to network data from other sources as well, as long as information adequately describing the properties of the nodes is provided. For example, citation networks of biomedical research articles can be explored with the MeSH (Medical Subject Headings) vocabularies that are stored in the MEDLINE database and friendship networks of university students can be explored with information of class names they attend.

To summarize, we present the first method for interactive and multi-scale navigation of large and complicated biological networks that displays appropriately abstracted views at all levels of detail. The specially designed interface enables flexible navigation across cluster boundaries. We believe that the method described here will aid modern biologists in discovering knowledge from massive binary-relationship datasets, which are accumulating at an accelerating pace. The implemented NaviCluster software is freely available at http://navicluster.cb.k.u-tokyo.ac.jp/

Table 1 A comparison table of existing approaches

Methods	Cluster generation	Multi-scale navigation support	Measures for handling insufficient clustering	Flexible navigation beyond cluster boundaries
Presented Method (NaviCluster)	Automatic (Extremely Fast)	Yes	Property-Based Clustering	Yes (via Re-centering)
GenePro	Manual	No	No	No
CyOog (Power Graphs)	Automatic	Yes (Power Nodes)	No	No
VisANT	Manual	Yes (Meta-Nodes)	No	No
BioLayout Express^{3D}	Automatic	No	No	No
jClust	Automatic	No	No	No

References

1) Merico D, Gfeller D, Bader GD: How to visually interpret biological data using networks, Nat Biotechnol, Vol.27, pp.21-924 (2009).
2) Suderman M, Hallett M: Tools for visually exploring biological networks, Bioinformatics, Vol.23, pp.2651-2659 (2007).
3) Hu ZJ, Mellor J, Wu J, Kanehisa M, Stuart JM, et al.: Towards zoomable multidimensional maps of the cell, Nat Biotechnol, Vol.25, pp.547-554 (2007).
4) Andreopoulos B, An AJ, Wang XG, Schroeder M: A roadmap of clustering algorithms: finding a match for a biomedical application, Brief Bioinform, Vol.10, pp.297-314 (2009).
5) Abello J, van Ham F, Krishnan N: ASK-GraphView: a large scale graph visualization system, IEEE Trans Vis Comput Graph, Vol.12, pp.669-676 (2006).
6) Vlasblom J, Wu S, Pu S, Superina M, Liu G, et al.: GenePro: a Cytoscape plug-in for advanced visualization and analysis of interaction networks, Bioinformatics, Vol.22, pp.2178-2179 (2006).
7) Freeman TC, Goldovsky L, Brosch M, van Dongen S, Mazière P, et al.: Construction, visualisation, and clustering of transcription networks from microarray expression data, PLoS Comput Biol, Vol.3,

pp.e206 (2007).
8) Royer L, Reimann M, Andreopoulos B, Schroeder M (2008) Unraveling protein networks with power graph analysis. PLoS Comput Biol 4: e1000108.
9) Pavlopoulos GA, Moschopoulos CN, Hooper SD, Schneider R, Kossida S: jClust: a clustering and visualization toolbox. Bioinformatics 25: pp.1994-1996 (2009).
10) Clauset A, Moore C, Newman MEJ: Hierarchical structure and the prediction of missing links in networks, Nature, Vol.453, pp.98-101 (2008).
11) Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, et al.: Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome Res, Vol.13, pp.2498-2504 (2003).
12) Yamada T, Bork P: Evolution of biomolecular networks: lessons from metabolic and protein interactions, Nat Rev Mol Cell Biol, Vo.10, pp.791-803 (2009).
13) Newman MEJ, Girvan M: Finding and evaluating community structure in networks, Phys Rev E, Vol.69, pp.026113 (2004).
14) Newman MEJ: Fast algorithm for detecting community structure in networks, Phys Rev E, Vol.69, pp.066133 (2004).
15) Clauset A, Newman MEJ, Moore C: Finding community structure in very large networks, Phys Rev E, Vol.70, pp.066111 (2004).
16) Wakita K, Tsurumi T: Finding community structure in mega-scale social networks, ArXiv Computer Science e-prints, pp.0702048 (2007).
17) Blondel V, Guillaume J, Lambiotte R, Lefebvre E: Fast unfolding of communities in large networks, J Stat Mech, Vol.2008, pp.P10008 (2008).
18) Danon L, Duch J, Diaz-Guilera A, Arenas A: Comparing community structure identification, J Stat Mech Vol.2005, pp.P09008 (2005).
19) Dunn R, Dudbridge F, Sanderson CM: The use of edge-betweenness clustering to investigate biological function in protein interaction networks, BMC Bioinformatics, Vol.6, pp.39 (2005).
20) Hu ZJ, Hung JH, Wang Y, Chang YC, Huang CL, et al.: VisANT 3.5: multi-scale network visualization, analysis and inference based on the gene ontology, Nucleic Acids Res, Vol.37, pp.W115-W121 (2009).
21) Java Universal Network / Graph Framework (JUNG). http://jung.sourceforge.net.