Development of Interactive, Multi-Scale Network Navigation Method and Its Application to Functional Genomics Data

Thanet Praneenararat

Department of Computational Biology
Graduate School of Frontier Sciences

The University of Tokyo

"Where there is a will, there is a way."

— Dedicated to my beloved family, who are most precious to my life —

A Dissertation Presented

by

Thanet Praneenararat

Submitted to

Graduate School of Frontier Sciences, The University of Tokyo

in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

June 2012

Supervisor: Toshihisa Takagi

Department of Computational Biology

Abstract

In the post-genomic era, an exponential number of biological data are being produced at an accelerating pace by high-throughput technologies and available via online databases on the Internet. Among these, binary relationship data that can be described as sets of elements and 1-to-1 associations (connections) between them have become increasingly common. Co-expressed gene pairs and protein-protein interactions exemplify this data type. Network (graph) visualization, where nodes and edges correspond to the elements and the connections respectively, is widely used for representing binary relationship data because it is expected to be more interpretable than a long list of associations. However, when network data are large and complicated (e.g., >100 edges), the network representations often become cluttered with jumbles of nodes and edges, known as "hair-balls", and thus fail to convey information effectively. Therefore, one of the key challenges is how to develop network navigation approaches that can abstract data properly and interactively, and visualize the data insightfully at a right level of detail. By such methods, researchers would be able to explore and interpret their large-scale networks much more effectively. Until recently, many studies have used various methods to tackle the cluttered-visualization problem, but still cannot obtain satisfactory results–truly clean and intuitive visualizations.

Hierarchical clustering is a technique that meaningfully and recursively groups data elements based on a similarity measure, thereby producing a hierarchy or tree of clusters. This method works with many types of data, including networks, to create groups of data elements in a multi-scale fashion. In the hierarchy, higher levels contain fewer, larger clusters with more data elements, or nodes in case of networks, than lower levels. Such a hierarchy can be applied to abstract the network visualization by showing only high-level clusters, thereby reducing the number of elements on the screen. By showing the actual members of each cluster at a certain level of the hierarchy, detailed information can be displayed at a particular scale. However, existing network visualization methods that offer such multi-scale navigation still

have some drawbacks that hinder scientists from effectively and interactively exploring large biological network data, namely, (1) uses of clustering that depends upon user-provided information about hierarchies, (2) long running time (e.g., minutes to hours) required to abstract large networks, (3) inflexibility in navigation beyond fixed cluster boundaries, and (4) insufficiency of data abstraction, which leads to still cluttered network drawings.

In this dissertation, I present the *first* interactive, multi-scale navigation method for large and complicated biological networks and demonstrate its application to two types of functional genomics data, a yeast protein network dataset and an *Arabidopsis* gene co-expression dataset. The method is mainly composed of an ultrafast graph clustering technique that rapidly abstracts networks of about 100,000 nodes by recursively grouping densely connected portions and a biological-property-based clustering technique that uses property information provided for biological entities (e.g., Gene Ontology (GO) terms). It can rapidly and automatically abstract any region of large network data and produce biologically meaningful visualization with a manageable amount of information at all levels of detail. Apart from untangling large and complicated biological networks, it can be used to discover hidden knowledge in the networks readily and effectively as well. The method was first implemented as a stand-alone Java Swing application named NaviCluster (`http://navicluster.cb.k.u-tokyo.ac.jp`) and then integrated with Cytoscape as a plug-in, named NaviClusterCS (`http://navicluster.cb.k.u-tokyo.ac.jp/cs/`), to gain benefits from its usability and abundant useful features. I believe that the presented method will aid modern biologists in discovering knowledge from massive binary-relationship datasets more efficiently. In the final chapter, I anticipate the prospects for this research as four main directions: (i) clustering and implementation optimization, (ii) enhancement of functionalities, visualization, and user experiences, (iii) application to multiple types of networks, and (iv) integration with text mining toward interactive, systematic knowledge discovery.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor, Professor Toshihisa Takagi for giving me invaluable opportunities to pursue both my master's and Ph.D. degrees in his laboratory and for providing encouragements, inspiration, and support in many ways for these almost six years.

This work would not have been possible if I had not also been guided by Prof. Wataru Iwasaki. I sincerely thank him for his suggestions about several things, not only research. Advice on everyday matters and Japanese language is indeed indispensable for spending life as a graduate student in Japan. In addition, many heartfelt thanks go to all people in my laboratory.

I am immensely grateful to the committee of my thesis, Prof. Toshihisa Takagi, Prof. Takashi Ito, Asst. Prof. Masahiro Kasahara, Assoc. Prof. Mariko Okada, and Prof. Kentaro Shimizu, for reading my thesis and giving constructive comments. I wholeheartedly thank all professors in this department for providing wonderful environments to learn and share knowledge through many lectures and discussions, and giving me role models as a leading scientist. Besides, I am indebted to Prof. Kengo Kinoshita and Assoc. Prof. Takeshi Obayashi for their valuable comments and fruitful discussions about my research and future directions.

Moreover, I thank all my past mentors and supervisors in Thailand, who trained me and provided substantial knowledge on sciences. My beloved friends both in Thailand and Japan, especially those I know at Thai Students' Association in Japan, are warmly thanked for their great encouragement as well. Finally, I am profoundly indebted to my parents and family, who constantly raise me with love, and encourage me spiritually throughout my life. Without them, I could never have grown up, been educated, and completed this thesis. It is them to whom I wholeheartedly dedicate this work.

The studies in this dissertation and all activities as a graduate student were financially supported by the Research Fellowships from the Japan Society for the Promotion of Science for Young Scientists and partly by the Monbukagakusho scholarship of Ministry of Education, Culture, Sports, Science and Technology of Japan.

Contents

List of Figures

Chapter 1

Introduction

In the post-genomic era, an exponential number of biological data available via many online databases or high-throughput experiments are being produced at an accelerating pace. Among these data, binary relationship data, which can be described as sets of elements and 1-to-1 associations (connections) between the elements have become increasingly common. Genetic regulatory relationships, co-expressed gene pairs, protein-protein interactions (PPI) exemplify this data type. Conventionally, network (graph) visualization, where nodes (vertices) and edges denote the elements and the connections respectively, is widely used for representing binary relationship data because it is assumed to be more interpretable than a long list of associations [1, 2]. However, when networks become larger, such as those with >100 edges as shown in Figure 1.1 (the network consists of 331 nodes and 362 edges and was derived from [3], mainly involved with galactose utilization pathway), researchers have to face more difficulties in investigating such data and each individual element and connection. To make matter worse, when large and complicated networks, such as the human interactome as shown in Figure 1.2, are visualized, their representations often become cluttered with jumbles of edges and nodes, known as "hair-balls" [2], and hence fail to convey information effectively. The large size of the data and ineffective visualization approaches are considerably inhibiting scientists from making sense of the data and communicating them to others in a meaningful way [4]. To avoid such complicated network visualization, a great challenge is to devise network navigation approaches that can abstract data properly, and visualize them insightfully at a right level of detail [5–7], so that researchers can interactively explore and interpret their networks much more efficiently.

Hierarchical clustering is a technique that groups data elements based on a similarity measure meaningfully and recursively, thereby producing a hierarchy or tree of clusters [8] (Figure 1.3). This method works with many types of data, including networks, to create groups of data elements in a multi-scale fashion. In the hierarchy, higher levels

Figure 1.1: A sample molecular interaction network file bundled with Cytoscape v2.8

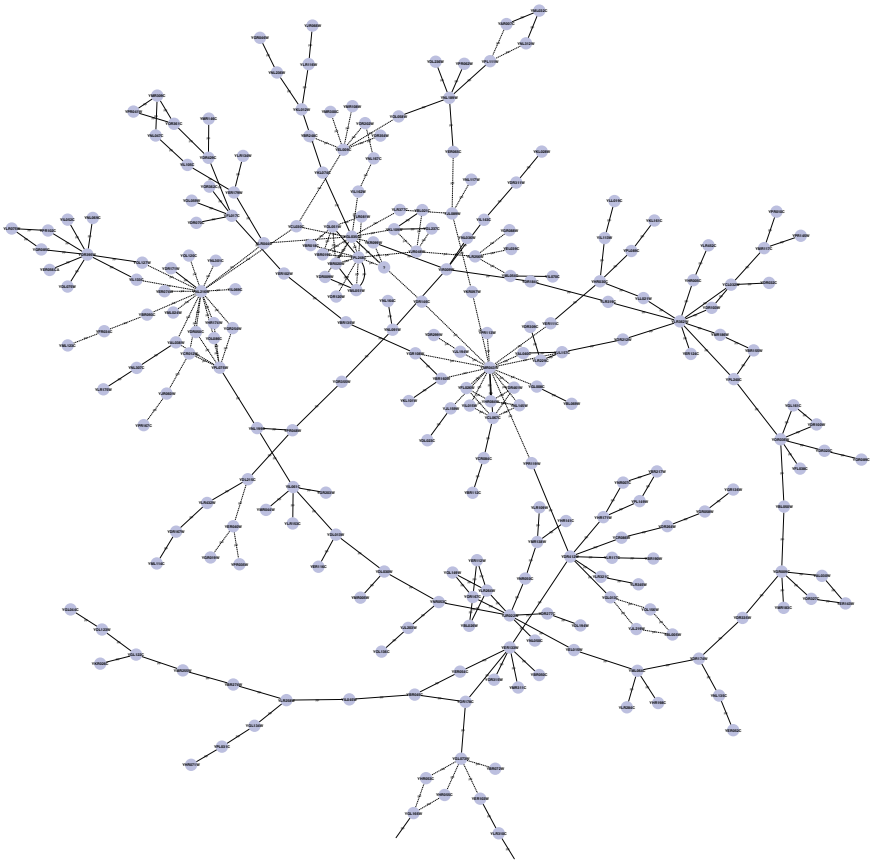


Figure 1.2: The human interactome obtained from www.cytoscape.org

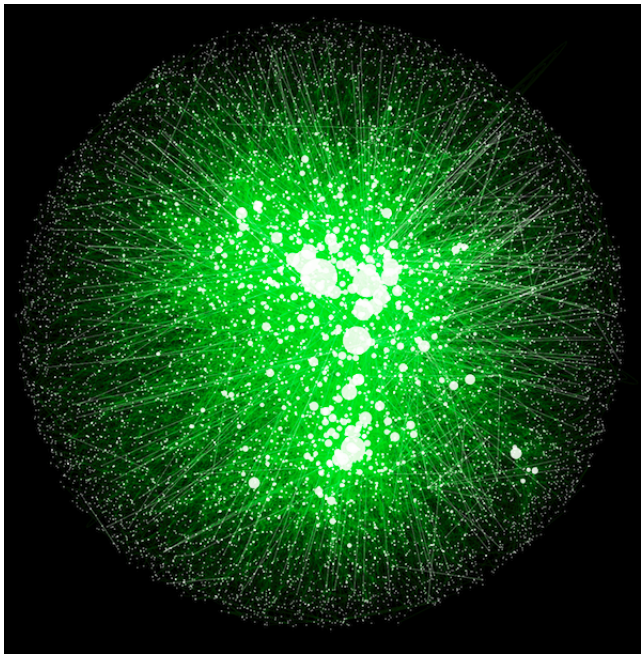
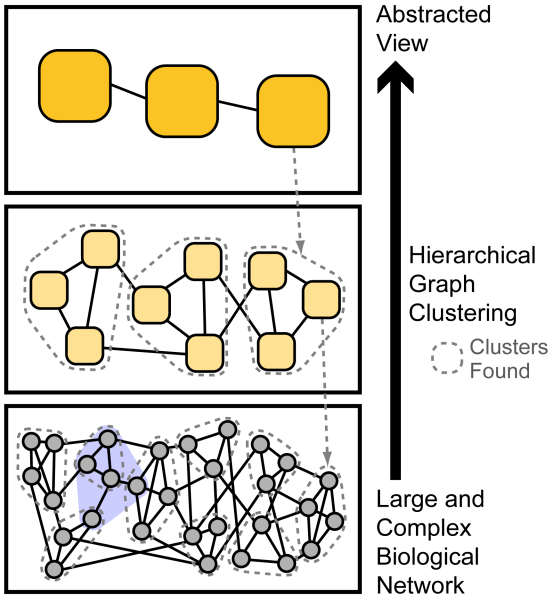


Figure 1.3: Hierarchical clustering adapted to visualization



contain fewer, larger clusters, each of which contains more data elements (or nodes, in case of networks), than lower levels. To solve the hair-ball problem, some methods [9–13] apply hierarchical clustering and display only the high-level clusters of the hierarchy to reduce the number of objects shown on the screen and abstract the network (the top panel of Figure 1.3). Descending the hierarchy and showing the members of each cluster at a particular level of the hierarchy can illustrate detailed information of the data at a certain scale (e.g., dotted arrows and regions in Figure 1.3). A recent study reported that natural networks possess hierarchical properties [14], so employing hierarchical clustering in abstracting biological networks thus makes sense and auspicious.

In spite of the advantages of hierarchical clustering, existing network visualization methods that offer multi-scale navigation via this technique still have some drawbacks that hinder scientists from effectively and interactively exploring large biological network data. First, some methods [10, 13, 15] need researchers to provide information on hierarchies or clustering, which usually not known to them in advance. Second, existing methods demand such long time that it is not suitable for interactive, real-time navigation. In the course of biological investigation, researchers often change their focus to generate different hypotheses, and thus require visualization of different sets of nodes/clusters. Existing methods that need long running times (minutes to hours) to produce each abstraction are therefore unacceptable in this regard [9, 11, 12, 16–19]. Methods that can provide appropriate abstractions of any given portion of the network rapidly and automatically, such as those that can abstract networks of 100,000 nodes in seconds, are thus crucial for efficient biological investigation. Third, existing visualiza-

tion systems do not allow for flexible navigation across fixed cluster boundaries; that is, researchers can explore members of only one cluster at a time [9–13]. Actually, members of interest to biologists may reside in different high-level nodes in the hierarchy (e.g., nodes in the light blue area in Figure 1.3). Those methods, which do not support flexible visualization of members of different clusters at the same time, are thus too limited to be used as an effective navigation approach. Last but not least, the clustering techniques exploited by existing methods are not sufficient for abstracting network data to a level that is clean and intuitive enough for immediate interpretation and investigation [9–12, 19, 20]. Recent investigations have revealed that for some common datasets, hub-like (high-degree) nodes tend to connect to low-degree nodes and most nodes interact with only few partners (e.g., yeast PPI networks) [21]. Large, densely connected regions of such networks are therefore quite few, but small, densely connected regions are rather frequently found. As a result, even if existing methods visualize the most abstracted view—the highest level of the hierarchy, which probably contains over 100 clusters, resulting representations still contain too much information and are difficult to manage. Therefore, means for further abstraction are needed to allow for intuitive visualization and navigation of large networks.

In this dissertation, I present an interactive, multi-scale network navigation method developed with three objectives: (i) the method can abstract network data without using a user-provided hierarchy; (ii) the method can rapidly, automatically, and interactively produce abstractions of any region of the network, including nodes/clusters belonging to different ancestral clusters in the hierarchy; (iii) the method can reliably produce intuitive visualization with a manageable amount of information at every step of navigation. Moreover, the effectiveness of this method was confirmed using two types of functional genomics data, namely, a real yeast protein network dataset and a *Arabidopsis* gene co-expression dataset, whose results are presented in Chapter 4. This approach will aid modern biologists in interpreting and analyzing large and complicated biological networks toward effective hypothesis generation and knowledge discovery.

Chapter 2

Methods

The method is composed of three main components, two clustering components (an ultrafast graph clustering component and a property-based clustering component) and one visualization component, which are executed in sequence (Figure 2.1).

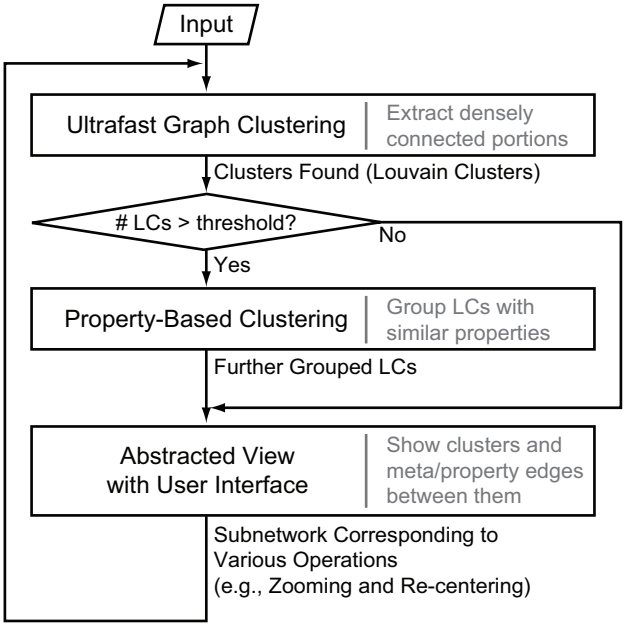
2.1 Ultrafast Graph Clustering Component

Given an input network, the method firstly abstracts the entire network based on an ultrafast graph clustering method, named the Louvain clustering. It iteratively groups topologically densely connected portions of the network, which may be equivalent to biologically meaningful clusters, such as protein complexes, with rapid speed. For networks of about 100,000 nodes, the component can finish clustering within a few seconds, thus it is especially imminent to be used to solve the problem of interactive navigation of large and complicated biological networks [22]. Besides, this ultrafast graph clustering technique brings another advantage if the next component, the property-based clustering, has to be executed afterwards, in that it can considerably reduce the number of input clusters of the slower property-based clustering.

2.1.1 Graph Clustering and Modularity

Graph clustering detects clusters (communities) in networks by identifying topologically densely connected sets of nodes where weighted connections of nodes *within* the sets are stronger/denser than weighted connections *between* nodes inside and outside of the set. This metric is called the modularity or Q function [23]. Its definition is $Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta\left(c_i, c_j\right)$, where A_{ij} is the weight of the edge between node i and node j, $m = \frac{1}{2}\sum_{ij} A_{ij}$, $k_i = \sum_j A_{ij}$ is the sum of the weights of all edges connected to node i, c_i is the community to which node i is assigned and $\delta\left(u, v\right) = 1$ if $u = v$ and 0 otherwise, that is, only nodes within the same cluster are considered. In words, the weight of *real*

Figure 2.1: A diagram of the presented work. Both clusterings aim for producing biologically meaningful clusters.



edges connecting any two nodes in each cluster minus the probability of edges connecting the two nodes in a *random* graph is computed. The sum of these quantities of all clusters yields the modularity.

2.1.2 Modularity-Based Graph Clustering Algorithms

Numerous algorithms for identifying clusters have been developed based on modularity optimization [22–26]. A survey about application of clustering algorithms to biological problems can be obtained via [8].

The Newman-Girvan (NG) algorithm is a well-known, pioneering one that iteratively removes edges most likely to lie between clusters (edges with high "betweenness"), splitting the clusters into two, until no edges remain [23]. "Edge betweenness" used in this algorithm is derived from "vertex betweenness" originally used in the social network analysis. The betweenness of a given edge is described as the number of shortest paths between pairs of nodes running through the edge. In this regard, the algorithm tackles the clustering problem with different strategies from others, which usually consider edges that are most central to clusters first. A resulting dendrogram is used to identify the best partition of the given input network dataset. In particular, the split that yields the highest modularity value is selected as the solution. This algorithm requires a traversal of all remaining edges at every step. As a result, it demands a high computational cost, thus apparently it is not appropriate to be applied to large networks. The dendrogram

itself can be used to generate hierarchical clusters but the hierarchy would be deep with two members at each level. The time complexity of the NG algorithm is $O(m^2 n)$ in general case and $O(n^3)$ for sparse graphs, where m = number of edges and n = number of nodes. Dunn et al. have employed this algorithm to find clusters of interconnected proteins in PPI datasets [27]. They also reported the distribution of Gene Ontology (GO) terms used to annotate proteins within each cluster and confirmed the significant correlations to functional annotations.

After this work [23], many studies about finding communities in graphs are being conducted progressively. A number of them improved the NG algorithm in aspects of performance (speed) [24, 25] or quality (modularity) [28, 29] or both [22, 26]. The Newman Fast (NF) algorithm [25] greedily optimizes the modularity function directly and works in a bottom-up style. It demands the time complexity of $O(n^2)$ for a sparse graph as compared to $O(n^3)$ of the NG algorithm. CNM [24] improves the NG algorithm in the same way as NF [25] did by optimizing modularity function but they used more elaborate data structures and some shortcuts in implementation, thereby boosting the speed of the algorithm to $O\left(n \log^2 n\right)$. The more delicate implementation makes the algorithm more practical with larger networks, but no improvements in the aspect of theory were achieved.

Until recently, the best known algorithm developed to overcome the shortage of high computational cost was presented by Wakita and Tsurumi (WT) [26]. Their algorithm has a near-linear time complexity, much improved from the NG algorithm. The algorithm is devised based on the observation that CNM tends to produce very large clusters and this behavior slows down the process much. Instead, the WT algorithm thus tries to balance cluster sizes in each step, resulting in about seven-time faster speed than CNM. However, the speed of this algorithm was still insufficient and the modularity of the produced clusters still had room for improvement when incorporated into an interactive navigation of large networks (e.g., human gene networks of $>$ 20,000 nodes) [22]. Recently, Blondel et al. developed a breakthrough algorithm for rapidly finding communities (clusters) with high modularity in huge networks of about 100,000 nodes [22]. The authors called this algorithm the "Louvain" method, named for their institution. As shown in Table 2.1, the performance of this algorithm overwhelms many existing clustering algorithms, such as the CNM and WT algorithms. It goes beyond the work of WT, by capable of handling graphs with up to 118 million nodes and 1 billion edges, as compared with the 5.5-million-node scale of the WT algorithm. Blondel et al. also pointed out that balancing the cluster sizes as done in the WT algorithm actually deteriorates the modularity. I found that this algorithm for finding meaningful communities in large and complicated networks could be applied to the problem of interactive navigation.

Table 2.1: Comparison of the Louvain clustering with others. Each cell shows the modularity of the resulting partition and the time used for each dataset and each algorithm. Column and row headers stand for datasets and algorithms, respectively. This table is derived from [22, Table 1]. See [22] for details about the datasets.

Dataset Name (#Nodes/#Edges)	Internet (70k/351k)	Web nd.edu (325k/1M)	Phone (2.04M/5.4M)	Web WebBase 2001 (118M/1B)
CNM [24]	0.692/799 s	0.927/5034 s	-/-	-/-
WT [26]	0.667/62 s	0.898/248 s	0.553/367 s	-/-
Louvain [22]	0.781/1 s	0.935/3 s	0.76/ 44 s	0.984/152 min.

2.1.3 The Louvain Algorithm

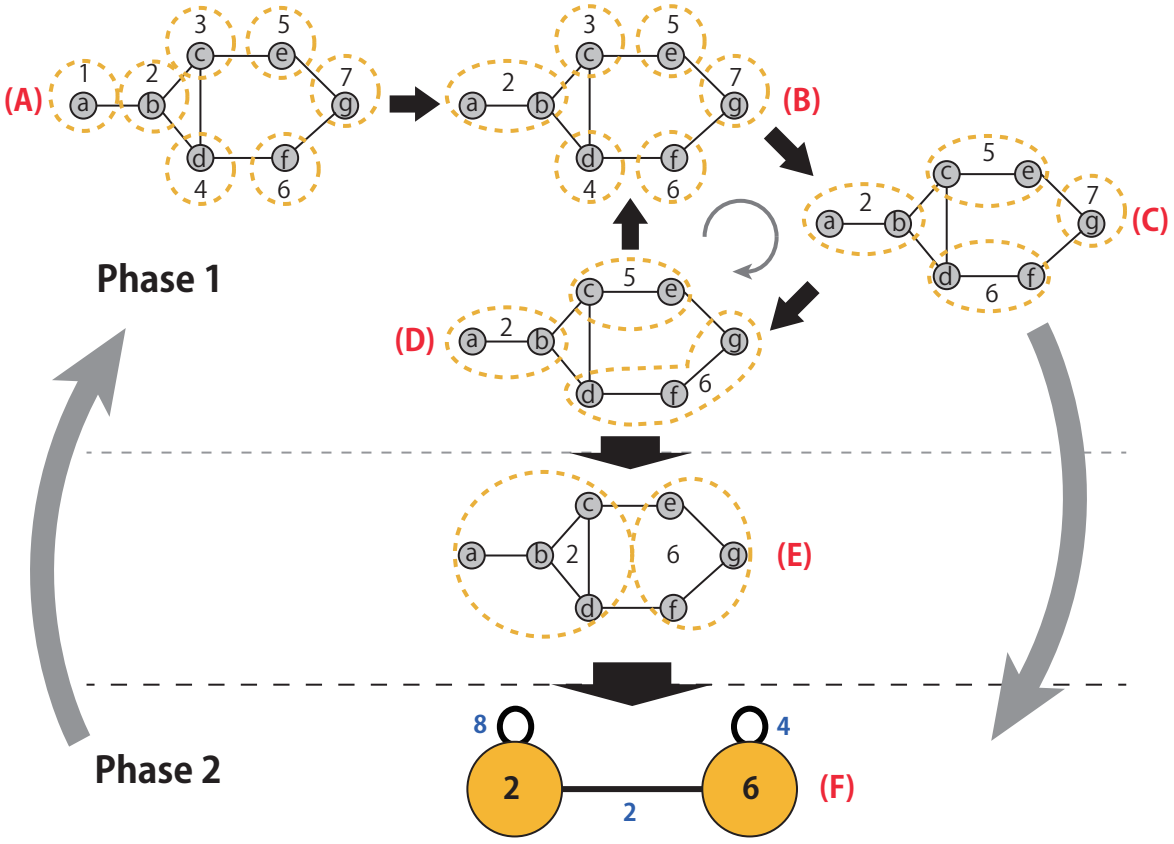
The Louvain algorithm consists of two phases running in sequence, resulting in one pass, which are repeated until no additional changes of modularity are made.

Phase 1 Let each node belong to a cluster different from every other node. For each node the algorithm considers its neighbors' clusters and see whether it should move the node to a neighboring cluster. It moves the node only when there are movements to neighboring clusters that result in a positive gain in modularity. Or else, the node is left in its own 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. This process is repeated until no members are added to/removed from any clusters and yields clusters with the maximum local modularity.

Phase 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 new nodes to represent corresponding edges between the members of the nodes.

The output of phase 2 is then fed back to phase 1 and the algorithm iteratively runs these two phases (one pass) until no additional changes are made. Using a graph of seven nodes as an example, a diagram of this algorithm is shown in Figure 2.2. More details of this algorithm and its application to many types of huge networks can be found in [22]. The Louvain algorithm can finish clustering networks of 70,000 nodes in one second [22]. 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.

Figure 2.2: A diagram of the Louvain algorithm, composed of Phase 1 and Phase 2 using a graph of seven nodes as an example. Circles in gray represent nodes and letters inside nodes represent node names. Circles in yellow represent clusters and numbers besides nodes are cluster names. (A) Initially, each of seven nodes belongs to a cluster different from others. (B) After node a is considered, it is moved to cluster 2 of node b, its only neighbor. (C) The state of the graph after considering node a to node f. (D) The state of graph after considering all nodes in one round. The algorithm starts from node a again and iterates through all nodes. (E) This procedure is repeated until no further changes, thereby reaching the maximum local modularity (F) The algorithm steps to Phase 2, which creates a new graph by using the clusters from Phase 1 as new nodes in the new graph. A meta-edge is drawn accordingly between the two new nodes, as there are edges between node c of cluster 2 with node e of cluster 6, and between node d of cluster 2 and node f of cluster 6. Self-loops are drawn on new nodes as well to denote edges between the members of the new nodes. According to the algorithm, self-loop weights need to be doubled.



As for the quality of the resulting clusters, the algorithm produces clusters (called Louvain Clusters or LCs in this presented work) with high modularity. It has been shown that clusters characterized by high modularity in biological networks correspond to biologically functional units (e.g., protein complexes in PPI networks and transcriptional modules in gene regulatory networks [27]). Thus, the LCs are expected to be intuitive to biologists and meaningful groups in navigation of biological networks.

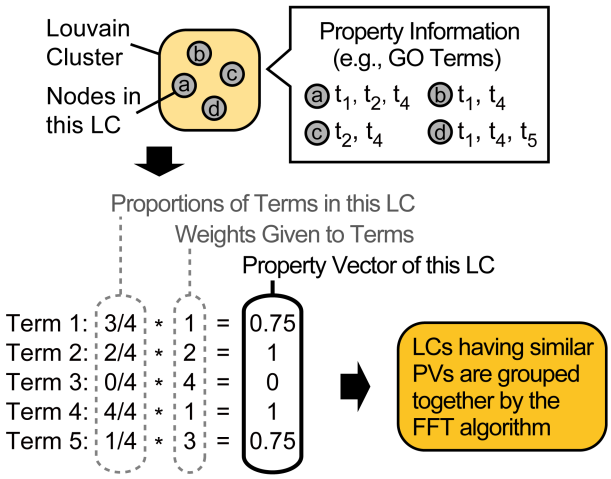
Note that another feasible means for decreasing the number of clusters on the screen is to continue the Louvain algorithm, even if the gain in modularity becomes negative. However, in such cases, the biological intuitiveness of the resultant clusters would be lowered due to the low modularity [27]. Thus, it would be better to adopt another reliable source of information at this stage, in addition to the topology of the networks.

To confirm the robustness and efficiency of the Louvain algorithm, Wallace et al. [30] applied it to co-citation networks and argue that the resulting clusters are coherent and unambiguous. At the time of this writing, according to my empirical survey, there are more than 100 publications both in physical sciences and life sciences citing, using, or modifying the Louvain algorithm [31–35]. In addition, Newman, one of the authors of the NG algorithm and also an expert in the field of community detection research, mentioned the Louvain algorithm in his recent review about communities and modules in networks published in Nature Physics, stating that this is one of the *best* available algorithms tested against computer-generated benchmark problems [34].

2.2 Property-Based Clustering Component

In case the abstraction is insufficient due to the characteristics of the biological network, the property-based clustering further groups the resulting clusters from the ultrafast graph clustering component (LCs) to a degree that can be visually interpreted. It automatically groups clusters with similar biological properties defined by the property information often assigned to biological entities, such as Gene Ontology (GO) terms (Figure 2.3). The new resultant clusters generated by this component are used instead of the previous ones to reduce the number of clusters on the screen, thereby decreasing the visual complication of the network representation and allowing for more effective navigation. Notably, VisANT 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 [36]. In contrast, the presented property-based clustering *automatically* generates clusters having similar properties to achieve interactive navigation.

Figure 2.3: a diagram of the property-based clustering



There are two main advantages of the property-based clustering: (i) the property-based clustering lets researchers control the number of clusters shown on the screen freely via a parameter K of its underlying algorithm, thereby allowing them to decrease the number of clusters on the screen to an extent they can manage to interpret. Further, as 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; and (ii) the generated clusters from this component are supposed to be highly intuitive, as they are formed using the property information that carries biological meaning.

In short, property vectors are created for LCs based on all property terms annotated to all nodes in the LCs. Then, the Farthest First Traversal K-center or FFT algorithm, the variant of K-means algorithm, is run using dot products between property vectors of LCs as a similarity measure to achieve new K clusters. The definitions of property vectors and the pseudocode of the FFT algorithm are given below.

2.2.1 Basic Definitions (Node, Term, Weight)

Number of Nodes Let N be the number of nodes of an input graph processed by the ultrafast graph clustering component.

Number of LCs Let L be the number of LCs.

Node For each n, where $1 \leq n \leq N$, node v_n has a set of terms $T(v_n)$.

Term $T(v_n)$ denotes the properties of node v_n (e.g., a set of GO terms).

Weight A weight $w(t)$, is given to a given term t to quantify its importance (e.g., properties that are rare and/or of particular interest to researchers may be given higher weights).

Set of Terms Let T_{all} be the set of all terms appearing in all nodes of the given input graph, so $T_{all} \equiv \bigcup_{1 \leq n \leq N} T(v_n)$ and $T_{all} = \{t_j | 1 \leq j \leq |T_{all}|\}$.

2.2.2 Property Vector and Related Definitions

Proportion Factor Let $Prop(t, \mathrm{LC}_l) = |\{v \in \mathrm{LC}_l | t \in T(v)\}| / |\mathrm{LC}_l|$, for each LC, LC_l, $1 \leq l \leq L$.

Property-Term Score The score of term t_j is calculated as $w(t_j)\, Prop(t_j, \mathrm{LC}_l)$, where $1 \leq j \leq |T_{all}|$.

Labeling In the current implementations (see Chapter 3), a property term to be used for labeling final clusters on the screen is the term t_h which $w(t_h)\, Prop(t_h, \mathrm{LC}_l) \geq w(t_j)\, Prop(t_j, \mathrm{LC}_l)$, $\forall j, 1 \leq j \leq |T_{all}|$.

Property Vector of LC The property vector for LC_l or $\mathbf{PV}(\mathrm{LC}_l)$ is a $|T_{all}|$-dimensional vector whose j-th element is the property-term score of term t_j.

Similarity of LCs or PVs The similarity between two LCs, $Sim(\mathrm{LC}_a, \mathrm{LC}_b)$, is given as a normalized dot product of the two property vectors; that is $Sim(\mathrm{LC}_a, \mathrm{LC}_b) = \mathbf{PV}(\mathrm{LC}_a) \cdot \mathbf{PV}(\mathrm{LC}_b) / |\mathbf{PV}(\mathrm{LC}_a)|\, |\mathbf{PV}(\mathrm{LC}_b)|$. Interchangeably, $Sim(\mathbf{PV}(\mathrm{LC}_a), \mathbf{PV}(\mathrm{LC}_b)) = Sim(\mathrm{LC}_a, \mathrm{LC}_b)$.

Similarity of PV to set of PVs The similarity of a PV, $\mathbf{PV}(\mathrm{LC}_\lambda)$, to a set of PVs, Σ, is defined as $Sim(\mathbf{PV}(\mathrm{LC}_\lambda), \Sigma) = \max(Sim(\mathbf{PV}(\mathrm{LC}_\lambda), \mathbf{PV}(\mathrm{LC}_l)))$, for all $\mathbf{PV}(\mathrm{LC}_l) \in \Sigma$

Similarity of LC to PC The similarity of an LC, LC_l, to a PC, PC_k, or $Sim(\mathrm{LC}_l, \mathrm{PC}_k)$, is given as a normalized dot product of the property vector of LC_l and of the center of PC_k, which in turn is an averaged vector among the property vectors of all LCs inside PC_k.

Then LCs having similar property vectors are grouped by the FFT algorithm [8] modified for generating property clusters.

Algorithm 2.1 Farthest First Traversal K-Center Algorithm

Input: A set of L Louvain clusters, $\mathbb{X}_{\mathrm{LC}} = \{\mathrm{LC}_1, \ldots, \mathrm{LC}_L\}$; A set of corresponding property vectors of LCs, $\Upsilon = \{\mathbf{PV}(\mathrm{LC}_1), \ldots, \mathbf{PV}(\mathrm{LC}_L)\}$; The number of property clusters K

Output: A set of K property clusters, $\mathbb{X}_{\mathrm{PC}} = \{\mathrm{PC}_1, \ldots, \mathrm{PC}_K\}$

Steps:

1: Arbitrarily choose a vector $\mathbf{PV}(\mathrm{LC}_m)$ and move it from Υ to an empty set Σ
2: Construct a property cluster PC_1, assign LC_m to be the center of PC_1, and add PC_1 to $\mathbb{X}_{\mathrm{PC}}$
3: **while** $|\Sigma| \neq K$ **do**
4: Let LC_m be a Louvain cluster whose property vector $\mathbf{PV}(\mathrm{LC}_m)$ is least similar to Σ, i.e., $m = \arg\min_\lambda (Sim(\mathbf{PV}(\mathrm{LC}_\lambda), \Sigma))$, $\mathbf{PV}(\mathrm{LC}_\lambda) \in \Upsilon$
5: Move $\mathbf{PV}(\mathrm{LC}_m)$ to Σ
6: Construct a new property cluster PC_k, assign LC_m to be the center of PC_k, and add PC_k to $\mathbb{X}_{\mathrm{PC}}$
7: **end while**
8: /*Assign the rest of LCs to suitable property clusters, judged by their similarities*/
9: **for all** the rest of $\mathbf{PV}(\mathrm{LC}_l)$ in Υ **do**
10: Assign LC_l to PC_κ where $\kappa = \arg\max_k (Sim(\mathrm{LC}_l, \mathrm{PC}_k))$, $\mathrm{PC}_k \in \mathbb{X}_{\mathrm{PC}}$
11: **end for**
12: **return** a set of K property clusters $\mathbb{X}_{\mathrm{PC}}$

2.2.3 The FFT algorithm

The Farthest First Traversal K-center (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. The FFT algorithm that is modified to use with property vectors is shown in Algorithm 2.1.

2.3 Visualization Component

Thirdly, the resulting clusters/nodes are instantly displayed with meta-edges and property edges, which represent the number of edges that exist between any members of the two clusters and the similarities between their properties, respectively. In the case that the number of clusters is less than the parameter K, the biggest cluster is greedily selected and split in order of decreasing size. This is repeated until either the number of clusters is equal to K or breaking only one more cluster makes the cluster number larger

than K. While showing the abstracted view, the researcher can interactively zoom, move laterally beyond cluster boundaries, focus on a particular set of clusters/nodes, etc. Any portions of the whole network of interest to the researcher can be fed into the clustering processes and the abstracted view of that cluster is displayed. The overall processes finish 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, allowing for truly interactive navigation of large biological networks.

Chapter 3

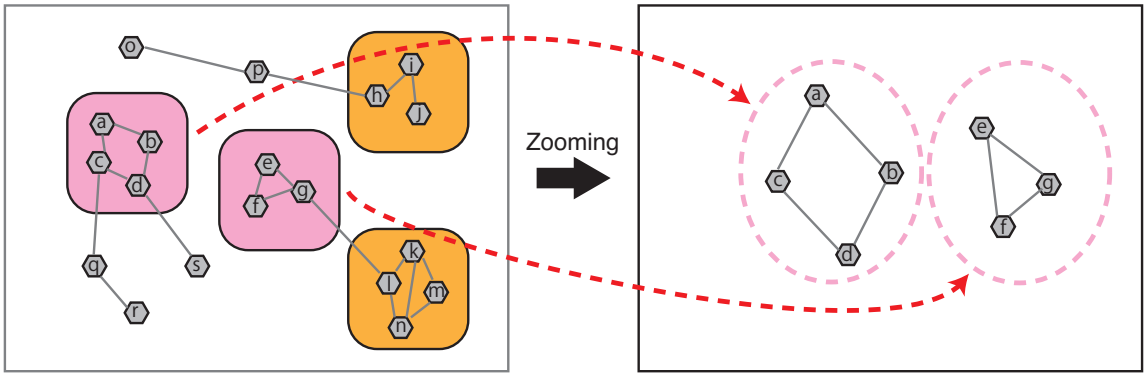
Implementations

I implemented the proposed method as two types of applications, a Java Swing application named NaviCluster (`http://navicluster.cb.k.u-tokyo.ac.jp/`) and a Cytoscape [15] plug-in named NaviClusterCS (`http://navicluster.cb.k.u-tokyo.ac.jp/cs/`). The basic concepts of these two versions are the same. As for the Java Swing application, the user interface was created using the Swing framework and the JUNG (Java Universal Network/Graph Framework) library (`http://jung.sourceforge.net`) was employed to create network drawings. It can be run on any computers with Java version 6 or later installed. On the other hand, the Cytoscape plug-in version can be run on any computers that have Cytoscape v2.8.x installed. The advantage of this version is that it can make use of useful functions of Cytoscape, such as import-export functions, and flexibly utilize extended features via other plug-ins in the future.

3.1 Requirements for Running Applications

Three input files are required to run the applications: a node list file, an edge list file, and a property information file. (1) 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 and TAIR for *Arabidopsis*). The database information is used to provide URL links. (2) The edge list file contains connected pairs of node names and the weights of connections (weights describe how strongly the nodes are connected). (3) 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 files bundled with the tools, 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

Figure 3.1: An example of zooming on a network. Nodes and clusters are represented by hexagons and round rectangles, respectively. The left view is an initial network consisting of four clusters and five nodes. Selected two clusters are highlighted in pink. After the zooming function is invoked, their members composed of seven nodes are displayed altogether.



cellular component). By using the namespace information, for instance, 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. Examples of these three files can be found in Appendices A and B. The programs have a minimum memory requirement of 1 GB for networks of about 100,000 edges and 1.5 GB-memory is recommended for running the programs smoothly.

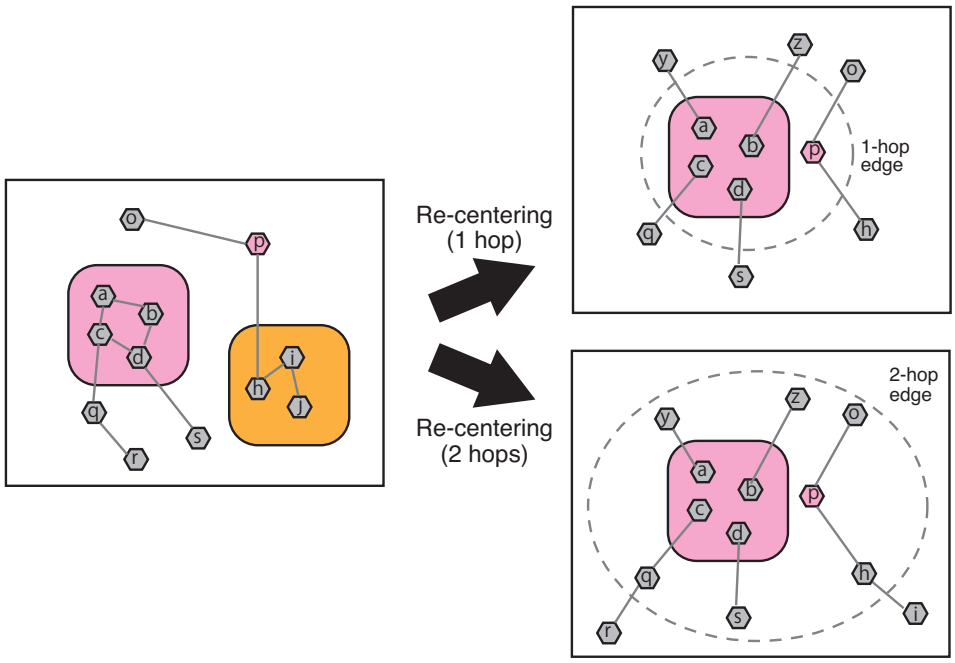
3.2 Features and Functions

3.2.1 Main Features

The presented method provides zooming and re-centering functions, which imitate the functions of common Web mapping services such as Google Maps. The zooming function, as reflected by its name, shows the details (members) inside selected clusters. It executes the two-stage clustering instantly, using all node members of the selected clusters as input, and displays the abstracted network afterwards (Figure 3.1). Note that the zooming operation can be run on more than one cluster at a time, useful for cases where the nodes of interest belong to different clusters.

Because zooming is always performed on *all* members of selected clusters, it alone is insufficient to provide flexible enough navigation for researchers to explore relationships between *some* nodes that may belong to different clusters. To fulfill this requirement, the re-centering function was designed and implemented. Given researcher-selected

Figure 3.2: An example of re-centering on a network. Nodes and clusters are represented by hexagons and round rectangles, respectively. The left view is an initial network consisting of two clusters and five nodes. Selected cluster and node are highlighted in pink. After the re-centering function is invoked with a threshold of one hop, their six direct neighbors are displayed altogether, as shown in the top right view. Note that nodes y and z, which were not in the initial view, are gathered in this view as well. The bottom right view shows the result from re-centering the network in the left view with a threshold of two, composed of eight direct/indirect neighbors.

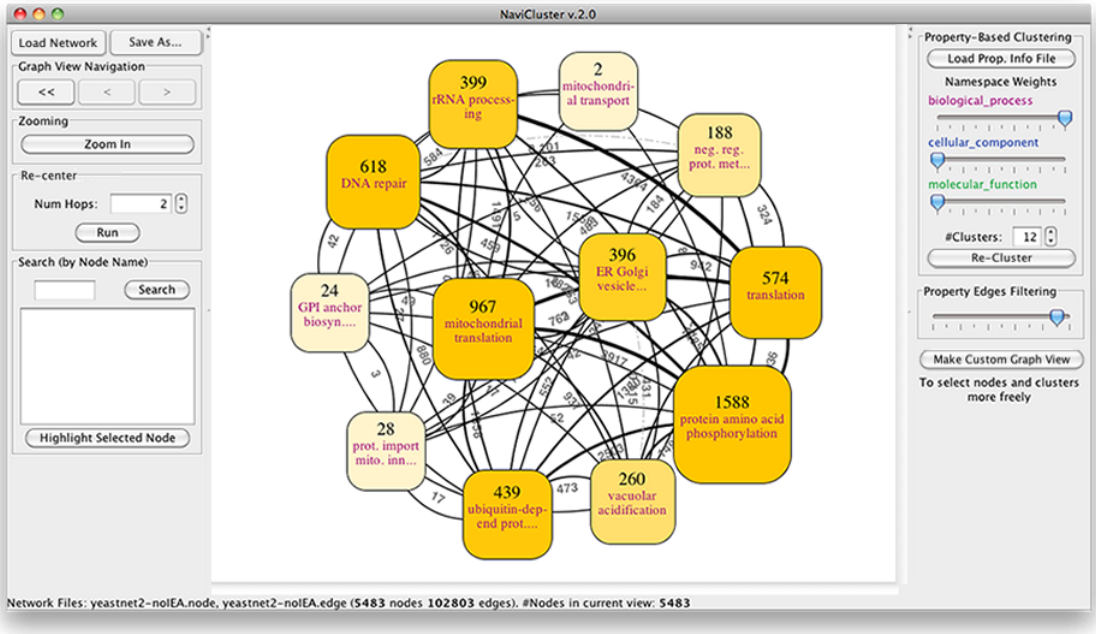


nodes/clusters, it runs the clustering on all nodes in the entire network whose geodesic distances to the selected clusters/nodes are less than/equal to a provided value (Figure 3.2). The neighbor nodes do not need to be in the current view to be included and clustered. This function corresponds to the panning function that Web mapping services provide to see surrounding regions. The resulting view shows the clusters/nodes of interest in the center of the display, as well as the nearby "neighbors". By changing the geodesic distance value, both fine and rough visualization centered on the clusters/nodes of interest can be obtained. Both functions correspond to seeing a map of several cities in different countries in Web mapping services and were not possible with previously existing methods.

3.2.2 Additional Functions

In addition to the highlighted features, both implemented tools have many additional functions designed for interactive network navigation. Users can finely create views containing only nodes/clusters they want to explore as well. Undo/Redo functions allow users to easily move backwards and forwards between previously created network views. As previously mentioned, users can adjust the namespace factors to change their importance, which changes the weights of all terms in the same namespaces, as long as at least one namespace weight is not zero. Furthermore, the number of clusters resulting from the property-based clustering, 12 by default, can be freely changed to match the individual preferences of the user and the view will be adjusted accordingly to show the result with the new number of clusters. The user can trace a node of interest easily with the search function; the cluster in the current graph view that contains the node of interest can be highlighted. In addition, users can customize the view according to their preferences in many ways; for example, the property edges can be filtered based on the similarity value of property vectors. Besides, visual appearances are designed to intuitively describe the different characteristics of the nodes/clusters and edges (e.g., node/cluster labels, cluster sizes, and edge thicknesses). Context-specific popup menus, which contain links to external databases, are also available to accelerate knowledge discovery and hypothesis generation as much as possible. See Section 3.3 for User Interfaces of the two implementations and Appendices A and B for User Manuals.

Figure 3.3: User Interface of NaviCluster, composed of two panels and one canvas. The left panel allows for network loading, export as images, graph view navigation, zooming, re-centering, and searching. The right panel allows for property information file loading, namespace weight adjustment, network re-clustering, property edge filtering, and custom graph view generation.



3.3 User Interfaces

3.3.1 NaviCluster - Java Swing Application

NaviCluster is mainly composed of three parts, the left and right panels, and the center canvas (Figure 3.3). The left panel allows users to load networks, export the canvas as image files, navigate views back and forth, zoom in on selected clusters, re-center the network on clusters/nodes of interest, and search for and highlight a node of interest via its name (see Appendix A for details). After loading NaviCluster, the program automatically loads and generates abstracted visualization of the default network data, YeastNet v.2 (Section 4.1), bundled with the tool. The right panel allows users to load property information files, adjust namespace weights, re-cluster the network based on new weights and preferred numbers of clusters, refine the filtering threshold of property edges, and create custom graph views.

To load a new network, ones start with loading a node list file and an edge list file via the *Load Network* button of the left panel. Then, the network is clustered by the two clustering components (Chapter 2) and visualized immediately. A resultant abstracted view is illustrated in Figure 4.1. When the clustered network is displayed, the users can

navigate the network in many different ways. Double-clicking on a cluster zooms in on the cluster. Selecting clusters and clicking the *Zoom In* button zooms in on those clusters at once. Selecting nodes and/or clusters and clicking the *Run* button of *Re-Centering* re-centers the network on the selected nodes/clusters (see Section 3.2.1). Apart from that, the users can go back and forth between the network views created in the past by using the "<<", "<" and ">" buttons. The resultant networks on the canvas can be exported as EPS, JPG, SVG, PS and PNG via the *Save As* button.

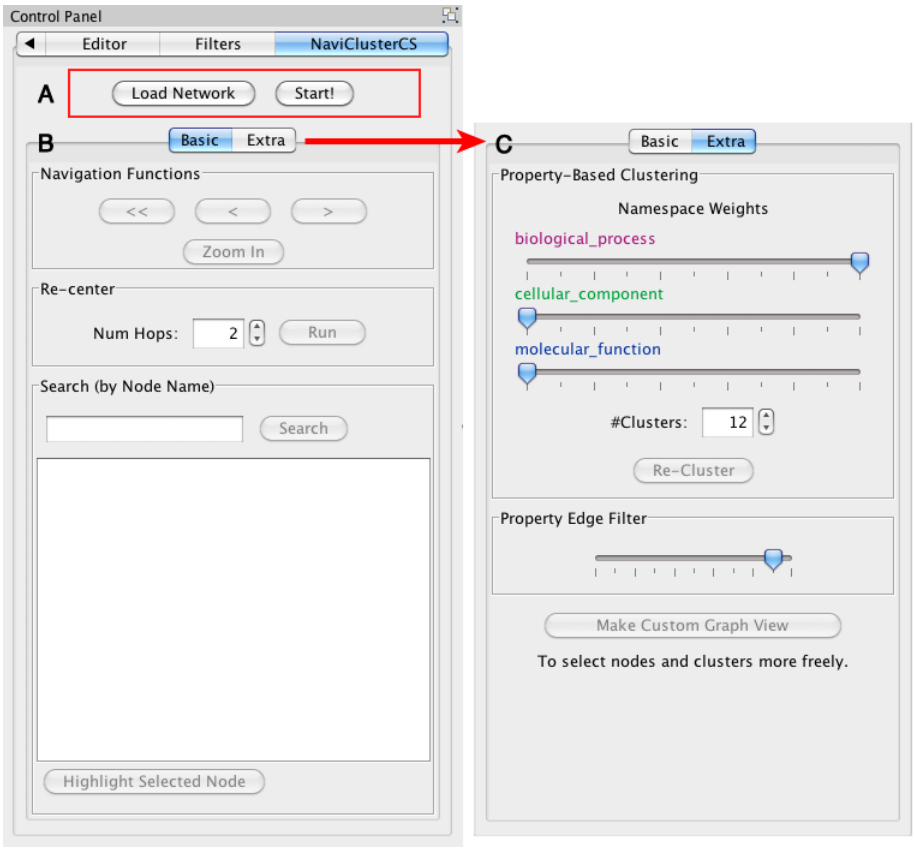
Besides, when users right-click on clusters/nodes/edges, context-sensitive menus appear. For nodes, users can see top ten property terms assigned to the nodes and open web pages containing related information on the nodes. For clusters, apart from property terms annotated to the node members of clusters, users can investigate the names of node members with their highest score terms, and information on the edge density of the clusters. As for edges, users can see details of the edges, which vary according to the types of edges (edge, meta-edge, or property edge). See Appendix A and the online manual at `http://navicluster.cb.k.u-tokyo.ac.jp/` for details.

3.3.2 NaviClusterCS - Cytoscape Plug-In

The control panel of NaviClusterCS comprises two buttons, the *Load Network* button and the *Start* button, and two tabs, the *basic* tab and the *extra* tab (Figure 3.4). The *Load Network* button is used to load new network data and the *Start* button invokes the two-stage clustering on the network. The *basic* tab allows users to navigate views back and forth, zoom in on clusters of interest, re-center the network on clusters/nodes of interest, and search for a node of interest (see Appendix B for details). In the *extra* tab, users can adjust namespace weights, re-cluster the network, refine the filter of property edges, and create new custom views. Data Panel of Cytoscape is hidden by default to give more space for resultant networks generated by NaviClusterCS.

Users can start with loading a node list file and an edge list file via the *Load Network* button of the *basic* tab or via the import functions of Cytoscape. Besides, users can use their networks loaded onto Cytoscape as well, provided that there exist all required attributes attached to the nodes and edges of the networks, such as an attribute named "weight" for edges. For detail, see the online manual at `http://navicluster.cb.k.u-tokyo.ac.jp/cs/`. After loading, the users can run the two-stage clustering on the network by clicking the *Start* button (Figure 3.4). A resultant abstracted view is illustrated in Figure 4.6. When the clustered network is displayed, the users can navigate the network in many different ways. Double-clicking on a cluster zooms in on the cluster. Selecting clusters and clicking the *Zoom In* button zooms in on

Figure 3.4: User Interface of the control panel of NaviClusterCS, which consists of two buttons for loading network data and starting clustering the network (A), and two tabs for performing various operations on the network. The *basic* tab allows for graph view navigation, zooming, re-centering, and searching (B). The *extra* tab allows for namespace weight adjustment, network re-clustering, property edge filtering, and custom graph view creation (C).



those clusters at once. Selecting nodes and/or clusters and clicking the *Run* button of the *Re-Centering* panel re-centers the network on the selected nodes/clusters (see Section 3.2.1). Apart from that, the users can go back and forth between the network views created in the past by using the "<<", "<" and ">" buttons. The resultant networks on the canvas can be easily saved as image via Cytoscape's Export menu.

Furthermore, users can configure the settings of NaviClusterCS and switch between domains of interest via the NaviClusterCS menu. General Settings allows for changing the directory that contains domain information (e.g., property information file used in the property-based clustering), a VizMapper file (a file used by Cytoscape to describe visual appearances of nodes, edges, background, etc.), and a graph layout algorithm used when loading networks. Switch Domains allows for selecting domain of interest, changing a property information file, and specifying information about external databases used to create context-specific menus for nodes. See Appendix B and the online manual at `http://navicluster.cb.k.u-tokyo.ac.jp/cs/` for details.

Chapter 4

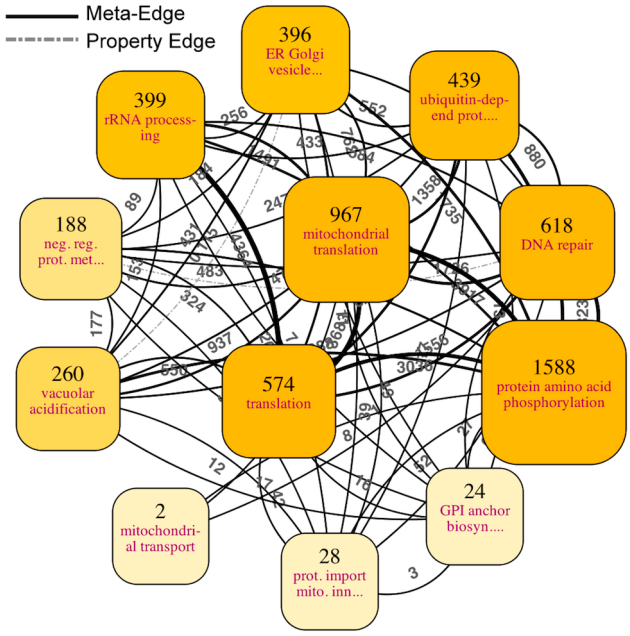
Results

In this chapter, the results from applying the proposed method to interactively navigate two types of functional genomics datasets are presented and discussed. The first one is a protein network dataset of *Saccharomyces cerevisiae*, named YeastNet v.2 [37]. The second one is a gene co-expression dataset of *Arabidopsis*, named ATTED-II [38]. The two network datasets were loaded, visualized, and analyzed in the implemented tools, NaviCluster and NaviClusterCS, respectively. Mainly shown here are the revelation of hierarchical structures of both networks and the use of the proposed method to uncover interesting elusive facts about genes/proteins of interest embedded in the networks.

4.1 Application to YeastNet v.2

YeastNet v.2 [37] is a probabilistic functional gene network that integrates heterogeneous functional genomics and proteomics data, e.g., direct physical interactions, regulatory interactions, protein complex membership, into objective models of cellular systems of yeasts. It was developed as a version 2 with the improvements over the previous version in that (1) the bias toward the prevailing gold standard reference during training was reduced, (2) a probabilistic model for calculating confidence in protein physical interaction and genetic interaction datasets was employed, and (3) thresholds for improving the derivation of functional linkages from DNA microarrays were imposed. Compared to 4,681 nodes and 34,000 linkages of the first version, YeastNet v.2 covers 5,483 yeast proteins (nodes) and 102,803 linkages (edges) . The log-likelihood score (LLS) scheme was assigned to each linkage to standardize the contributions from each genomic dataset to reflect different utilities for inferring functional linkages. High positive LLS scores indicate that rich information for discovering functional linkages can be found in the genomic dataset; a score of zero means that the dataset is not informative than random

Figure 4.1: An abstracted view for the entire network of YeastNet v.2, a probabilistic functional network containing 5,483 yeast proteins and 102,803 linkages. The associated log-likelihood scores were adopted as edge weights and used in the ultrafast graph clustering. NaviCluster directly generated this drawing with the preferred number of clusters on the screen set at 12.



expectation. This section shows the visualizations of YeastNet v.2, which were generated by the stand-alone software NaviCluster.

4.1.1 Abstracted View of the Whole Network

Figure 4.1 illustrates the abstracted visualization of the entire *Saccharomyces cerevisiae* protein network YeastNet v.2 of 5,483 nodes and 102,803 edges. The log-likelihood scores indicating the probabilities of true functional linkages in YeastNet were adopted as the edge weights for the ultrafast graph clustering component. Edges with high weights connote that the protein pairs of the edges have strong relationships and thus highly likely to be grouped in the same clusters in the ultrafast graph clustering. GO terms assigned to the proteins in the SGD database (`http://downloads.yeastgenome.org/` (access date: 29 March 2010)) were used as property information for the property-based clustering. The property-based clustering was configured to focus only on the biological process namespace, in order to group proteins involved in the same biological process. This can be done by excluding all terms of the other two namespaces in the property-based clustering—which can be easily performed using the namespace sliders in the tool (see Appendix A.8).

The numbers displayed above clusters represent the numbers of nodes within the clusters. The labels following them are the abbreviated property terms that can best describe the properties of the clusters, providing insight into the biological meaning of the clusters. Such property terms are the ones that are shared by most proteins in the clusters and meanwhile most specific (deepest in the GO hierarchy) among all terms annotated to the proteins in the clusters (see Section 2.2.2 for details). In case there are more than one cluster labeled with the same property term, the next highest score term of each cluster is additionally displayed in brackets to discriminate the cluster from the others. The display sizes of the clusters are proportional to linear normalizations of the numbers of the clusters ($nsize$) over the interval from the minimum ($nmin$) and the maximum numbers ($nmax$) among all clusters $((nsize - nmin) / (nmax - nmin))$. The color saturations of the clusters also reflect the number of proteins inside.

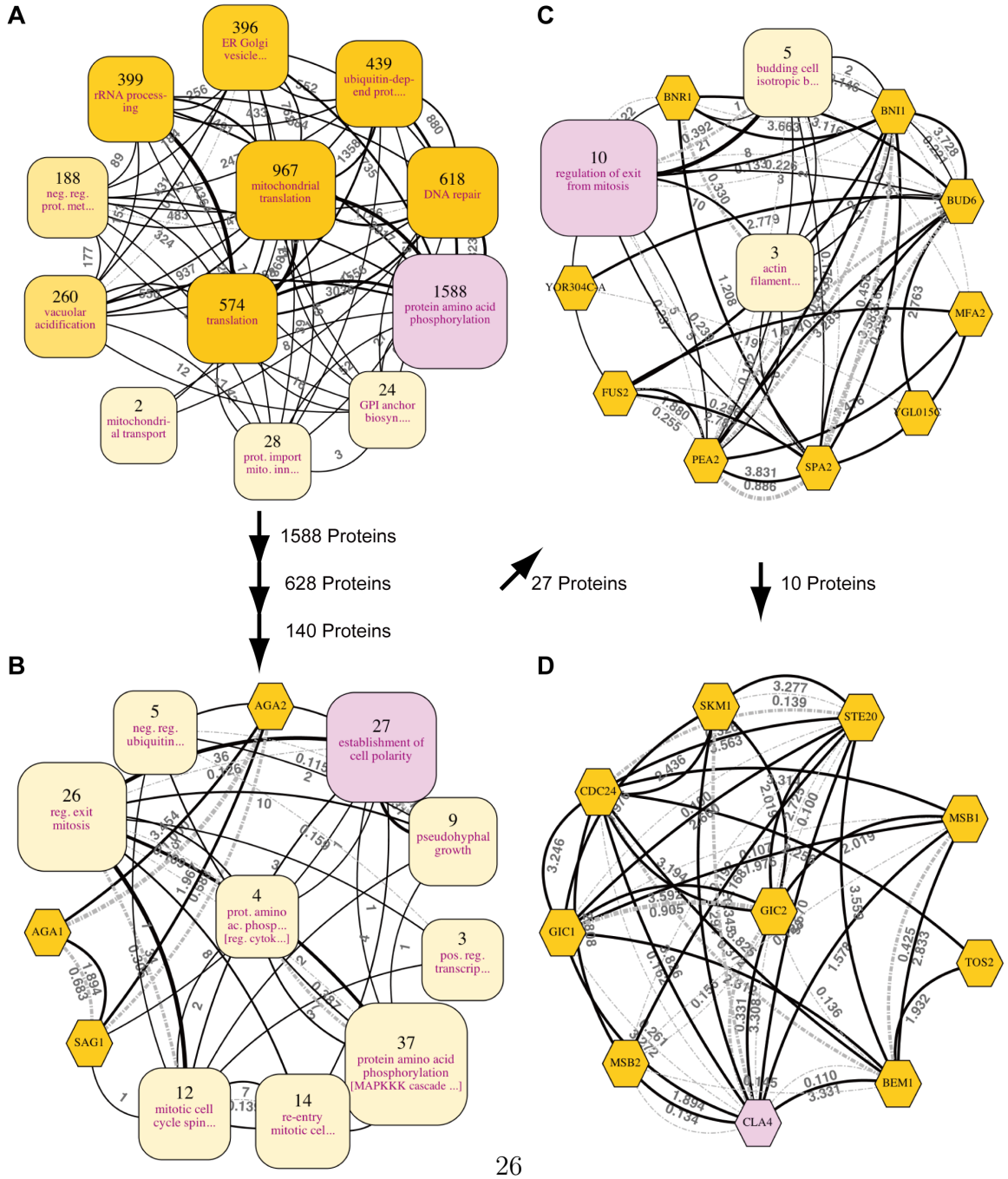
Meta-edges are drawn between any two clusters that have at least one edge between at least one member in each of the two clusters (solid lines in Figure 4.1). The gray number next to each meta-edge is the total numbers of all edges existing between the members of the two clusters, which are also reflected in the thickness of the meta-edge. In addition, a property edge is drawn between every pair of clusters if the similarity between their property vectors, represented by the associated gray number, is larger than a specified threshold (dashed lines in Figure 4.1; the threshold is 0.1).

4.1.2 Revelation of the Hierarchical Structure of the Network (Based on Cla4)

Figure 4.2 demonstrates how the zooming and searching functions of NaviCluster can be employed together to lead researchers to the protein of interest, in this case Cla4. Cla4 is a p21-activated protein kinase that acts as an effector of Cdc42. It has been implicated in many important biological processes such as cell polarization [39–43], cytokinesis [42, 43], and exit from mitosis [44–48]. In Figure 4.2, the clusters containing Cla4 are highlighted in all views; the granularities of detail in the views vary from coarsest to finest. Each zooming operation (a solid arrow) performs clustering on the member proteins of the highlighted cluster and immediately shows 12 more detailed clusters.

In the first view, which abstracts the whole network, the clusters are labeled with broad biological processes such as *DNA repair*, *rRNA processing*, and *translation* (Figure 4.2A). Cla4 is grouped under the *protein amino acid phosphorylation* cluster, which is highlighted. The members of this cluster are mostly involved in phosphorylation processes, which is also true for Cla4, whose function is to phosphorylate proteins. After zooming in on the *protein amino acid phosphorylation* cluster, one can find clusters of

Figure 4.2: The hierarchical organization of the clusters encompassing Cla4, a protein of interest, which are highlighted in pink. The numbers by the arrows indicate the numbers of proteins contained in the processed sub-networks. Hexagons represent proteins that are not contained in any cluster. **(A)** The most abstract view. Cla4 belonged to the highlighted *protein amino acid phosphorylation* cluster. After zooming in on this cluster, the clusters containing Cla4 of two deeper views were also the clusters labeled *protein amino acid phosphorylation* and were excluded for conciseness. **(B)** At this level, Cla4 was contained in the *establishment of cell polarity* cluster. Clusters labeled with the same property terms as those of others are discriminated by additionally displaying the next highest score terms in brackets. **(C)** Cla4 was contained in the *regulation of exit from mitosis* cluster. **(D)** The most specific view. Cla4 was clustered together with Ste20, Gic1, Gic2, Cdc24, Bem1, Skm1, Msb1, Msb2 and Tos2.



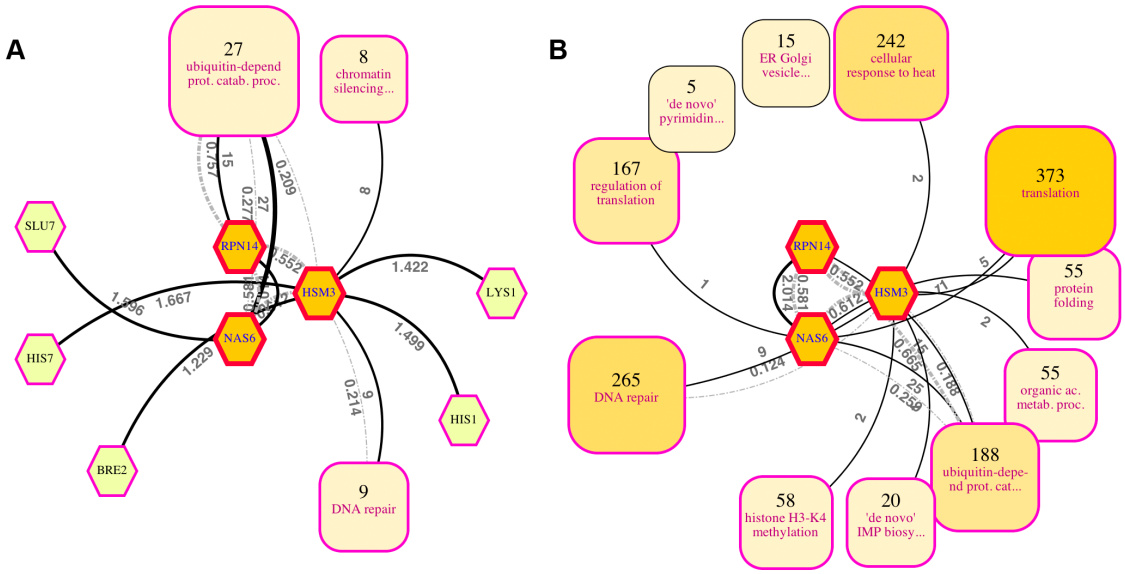
more specific processes, such as *mitotic cell cycle spindle assembly checkpoint*, *pseudohyphal growth*, and *establishment of cell polarity* (Figure 4.2B). In this view, Cla4 is a member of the *establishment of cell polarity* cluster, which is consistent with previous studies suggesting the role of the protein [39–41]. Some clusters found deeper in this cluster are characterized by actin filaments or budding processes (Figure 4.2C); as expected, they are related to cell polarity. Cla4 is found in the cluster of *regulation of exit from mitosis* at this level. After zooming in on this cluster to get the final view, illustrated are the relationships between Cla4 and other proteins such as Gic1, Gic2, and Ste20 (Figure 4.2D). In fact, these proteins were found to be implicated in mitotic exit by three different mechanisms [48]. Cla4 was discovered to promote mitotic exit and cytokinesis by activating a guanine nucleotide exchange factor, Lte1, which in turn causes the activation of Tem1, thereby terminating the M phase of the cell cycle [46–48]. In addition, Gic1 and Gic2 were also proposed to be involved in stimulating mitotic exit in parallel with Cla4 by inhibiting GTPase-activating proteins of Tem1, both of which result in activation of Tem1 [45, 48], which in turn signals to terminate the M phase of the cell cycle. Moreover, Cdc24 and Bem1, which play important roles upstream in the regulation of mitotic exit [39–41, 48], are also clustered together with Cla4 in this most detailed view.

4.1.3 Discovery of Unknown Roles of Nas6, Rpn14, Hsm3 as Chaperones

Figure 4.3A shows the view resulting from selecting the proteins Nas6, Rpn14, and Hsm3 and invoking the re-centering function to gather all direct interactors. Recent studies have reported that these three proteins, previously known to bind to regulatory particles of proteasomes, actually function as chaperones, assisting in the regulatory particle assembly in yeast [49–52]. In Figure 4.3A, the fundamental roles of the three proteins as proteasome-related proteins are delineated explicitly; specifically, they interact with many proteins involved in the *ubiquitin-dependent protein catabolic process*.

It is also possible to investigate the relationships between Nas6, Rpn14, and Hsm3 and other proteins that are farther from them, to understand their roles in a broader aspect (Figure 4.3B; the abstracted view collecting all proteins within two hops from the three selected proteins). This view discloses additional relationships with other clusters of more general processes. Among them, the relationships with the *protein folding* and *cellular response to heat* clusters actually suggest the recently reported roles of the three proteins as chaperones. This is because proteins denatured due to heat are rescued by

Figure 4.3: Re-centered views for discovering hidden facts from both direct and indirect neighbors of proteins of interest, Nas6, Rpn14, and Hsm3. Edges not connected to the center proteins are omitted for clarity. **(A)** Re-centered with the geodesic distance from the central nodes set to one. Twenty seven of the 49 direct neighbors of the three proteins form a *ubiquitin-dependent protein catabolic process* cluster, nine form a *DNA repair* cluster and eight form a *chromatin silencing at telomere* cluster. **(B)** Re-centered with the geodesic distance set to two. This view illustrates the more general functions of the proteins (see the main text)



chaperones, which help fold and assemble proteins. It should be noted that these clusters were formed without using any property information from the three selected proteins.

4.1.4 Function Elucidation of Nsi1 (Ydr026c), a new rDNA Silencing Factor

Figure 4.4 shows the hierarchical organization of the yeast protein network, focused upon Nsi1 or Ydr026c. Previously known as Ydr026c, Nsi1 was very recently confirmed the role as an rDNA silencing factor that contributes to rDNA stability and lifespan extension in yeast [53]. Under normal conditions, rDNA silencing is one of the key mechanisms to protect yeast cells from accumulating toxins, a major cause that leads to aging, so this process is critical for the growth and survival of yeasts. By previous studies, Nsi1 was suggested to associate with ribosomes and physically interact with Fob1, which in turn is required for rDNA silencing by recruiting other two proteins (Net1 and Sir2) to participate in the mechanism [53–55]. Moreover, the association profile of Nsi1 with rDNA is closely similar to that of Fob1 [56], meaning that their functions may be highly similar. Confirmed by the experiments [53], Nsi1 was recently proved to really play a role in rDNA silencing, partly because of the fact that the loss of Nsi1 decreases the activity of the mechanism.

Figure 4.4 presents the results from zooming in on the YeastNet dataset, focused on Nsi1 in the same way as of Figure 4.2—arrows denote zooming operations. In Figure 4.4A, Nsi1 is grouped under the *rRNA processing* cluster, relevant to the role of Nsi1, whose function is related to the activities of rDNA in the nucleolus, where the synthesis and processing of rRNAs take place. After zooming in on the *rRNA processing* cluster, one can find clusters of other specific processes, such as *RNA modification*, *tRNA methylation*, and *transcription from RNA polymerase III promoter* (Figure 4.4B). In this view, Nsi1 is a member of the *transcription from RNA polymerase III promoter* cluster, suggesting that the protein might be involved in this process. In Figure 4.4C, Nsi1 is found in the cluster of *transcription of nuclear large rRNA transcript from RNA polymerase I promoter* at this level, which is again consistent with the recently found role as involved with rDNAs, the places where rRNAs are transcribed. After zooming in on this cluster to get the final view, illustrated are the relationships between Nsi1 and RNA polymerase proteins, namely Rpa34 and Rpa49, which all assist in the synthesis of rRNA transcripts [57] (Figure 4.4D). Interestingly, all created views are very consistent in that the protein of interest (Nsi1) might strongly relate to various mechanisms of DNAs and RNAs.

In addition, when investigating the relationships between Nsi1 and its neighbors via the re-centering function, as shown in Figure 4.5, the view discloses relationships

Figure 4.4: The hierarchical organization of the clusters encompassing Ydr026c (Nsi1), a protein of interest, which are highlighted in pink. The numbers by the arrows indicate the numbers of proteins contained in the processed sub-networks. Hexagons represent proteins that are not contained in any cluster. **(A)** The most abstract view. Nsi1 belonged to the highlighted *rRNA processing* cluster. After zooming in on this cluster, the cluster containing Nsi1 of a deeper view was also labeled the same as the first view and was excluded for conciseness. **(B)** At this level, Nsi1 was contained in the *transcription from RNA polymerase III promoter* cluster. **(C)** Nsi1 was contained in the *transcription of nuclear large rRNA transcript from RNA polymerase I promoter* cluster. **(D)** The most specific view. Nsi1 was clustered together with Rpa34, Rpa49, Rpa14, Rpa43, and Rpa190.

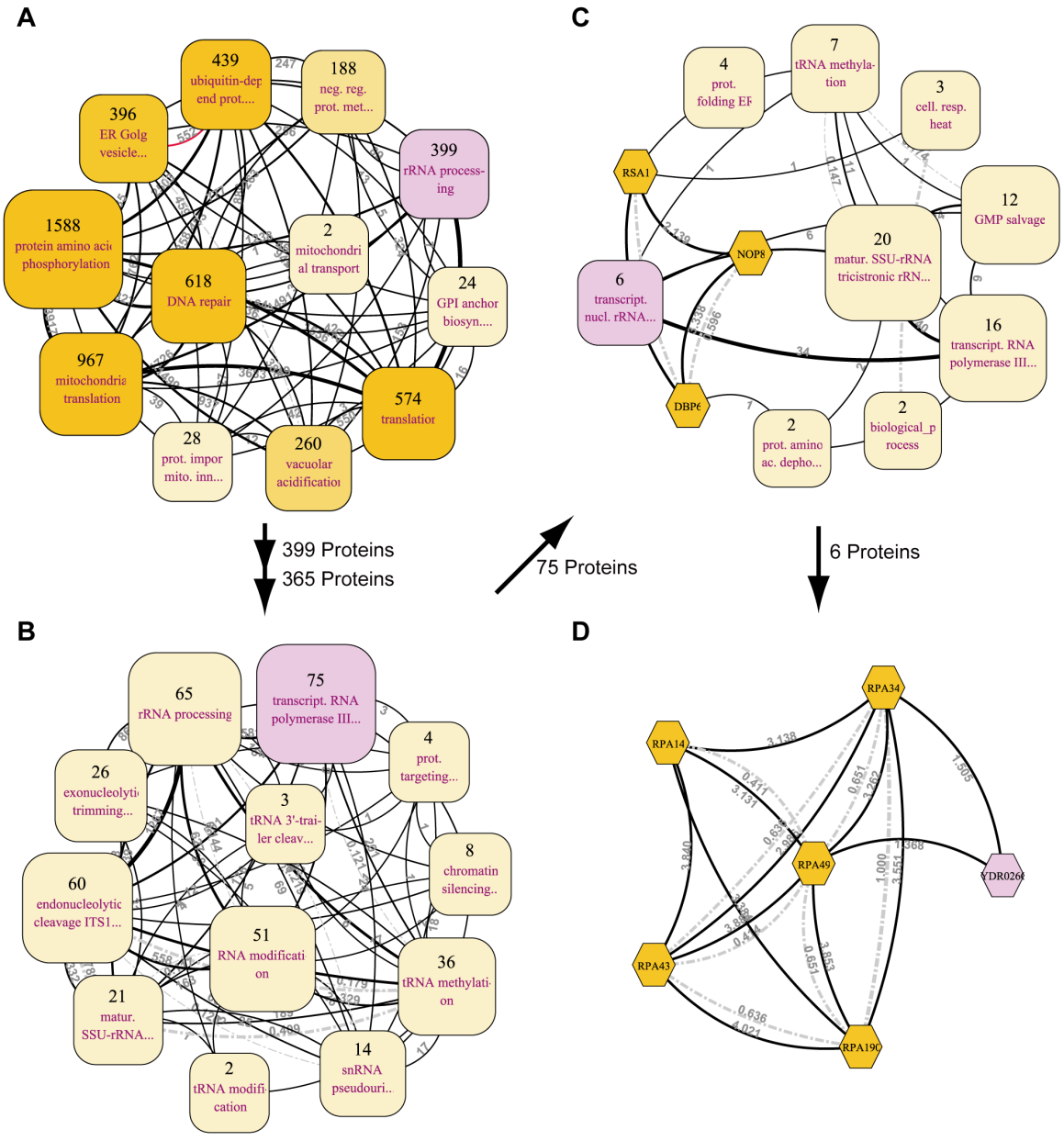
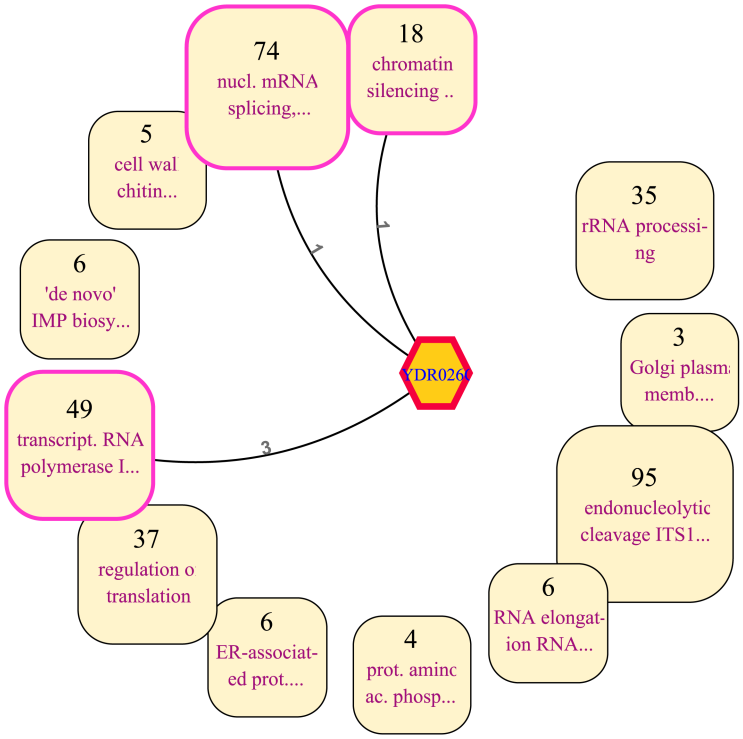


Figure 4.5: Re-centered view on Ydr026c (Nsi1) with a threshold set to two. The re-centering function allows researchers to intuitively grasp what types of proteins exist around the protein(s) of interest. Edges not connected to the center protein are omitted for clarity. All of the proteins that are within two hops from Nsi1 were clustered to a visually interpretable level. One can immediately see clusters related to *chromatin silencing at telomere*, *transcription from RNA polymerase III promoter*, and *nuclear mRNA splicing, via spliceosome*, around the protein of interest, implying that Nsi1 are probably involved with RNA synthesis and processing.



with clusters implicated in *chromatin silencing at telomere*, *transcription from RNA polymerase III promoter*, and *nuclear mRNA splicing, via spliceosome*. Although the total direct relationships of Nsi1 with other proteins are only five, this view can provide a general idea of what kinds of processes Nsi1 get involved with, via the biological meaning of the surrounding clusters. This makes researchers be able to focus on only a few biological processes to start with and validate, instead of randomly formulating hypotheses.

4.2 Application to ATTED-II

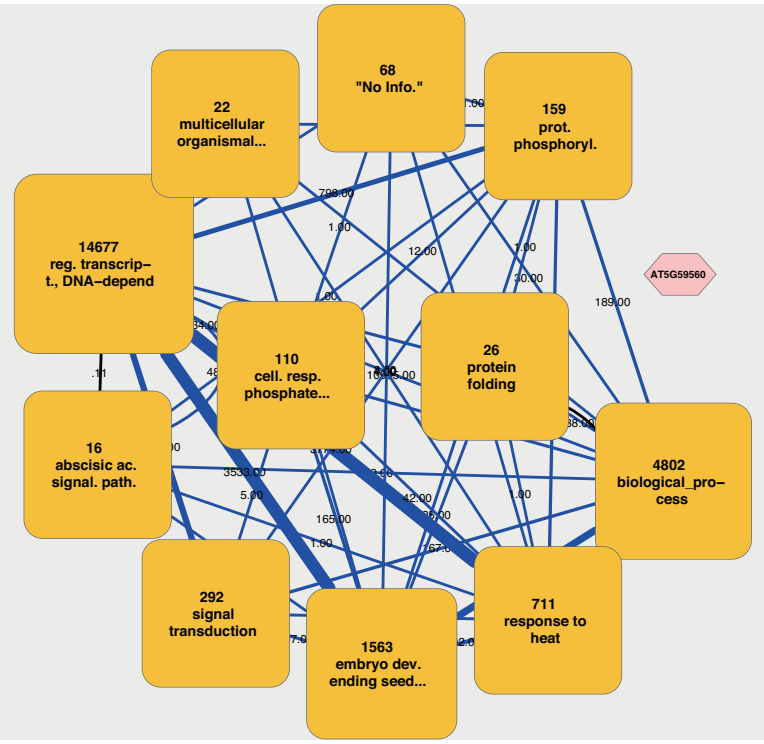
ATTED-II [38] or *Arabidopsis thaliana trans*-factor and *cis*-element prediction database is an *Arabidopsis* gene co-expression network dataset deduced from microarray data and predicted *cis* element. As a secondary database, it was designed to help researchers retrieve information on gene-to-gene relationships. Originally, the database adopted Pearson's correlation coefficients (PCCs) to quantify the similarity of gene expression profiles. Reported in 2009, however, the updated version of the database employs mutual ranks (MRs) instead of PCCs, to resolve the problem that many functionally relevant co-expressed gene pairs have low PCCs [58]. MR is calculated as the geometric mean of the correlation rank of gene A to gene B and of gene B to gene A—a small MR value means that the gene pair has high correlation.

I ran NaviClusterCS on a subset of ATTED-II, composed of 22,447 nodes and 189,546 edges, which correspond to genes and co-expressions, respectively. Because not all pairs of genes are regarded as important co-expressions, only edges with small MRs enough were included. Empirically, I used 30 as a threshold value as the authors of ATTED-II also indirectly treat co-expressions with MRs more than 30 as not so important edges by making them thin edges in their visualization system [58]. Furthermore, as the clustering algorithms used in the proposed method regard high values of edge weights as highly related connections, compared to the "low value, high correlation" scheme of MR, we need to reverse the values of MRs before assigning them to edges. I accomplished this by using 30 minus MR for each edge instead; again, 30 is the maximum value used for filtering edges.

4.2.1 Abstracted View of the Whole Network

Illustrated in Figure 4.6 is the abstracted view of the whole ATTED-II network dataset composed of 22,447 nodes and 189,546 edges. The figure was generated by NaviClusterCS, the Cytoscape plug-in. The number of clusters to be displayed was set at 12. The GO terms assigned to genes were obtained from TAIR (`ftp://ftp.arabidopsis.`

Figure 4.6: An abstracted view of the ATTED-II dataset with 22,447 nodes and 189,546 edges. The rounded squares are clusters and the hexagon is a gene that is not contained in any cluster. Because the gene AT5G59560 has no connections with any genes in this network, it appears as an unconnected node. 30 minus Mutual Ranks (MRs) were adopted as edge weights. The number of clusters/nodes to be shown on the screen was set to 12. See the main text for details.



`org/home/tair/Ontologies/Gene_Ontology`), a database of genetic and molecular biology data for *Arabidopsis thaliana*, and only biological process terms were used for the property-based clustering in this visualization. The numbers shown above the clusters represent the number of nodes within the clusters, and the labels following the numbers are the abbreviated GO terms that get the highest score and, thus, best describe the properties of the clusters. Meta-edges are drawn as blue lines between any two clusters that have at least one edge between at least one member in each of the two clusters, with the numbers next to each edge representing the total numbers of all edges existing between the members of the two clusters. The total numbers are also reflected by the thicknesses of the meta-edges. In addition, a property edge is drawn as a black line between every pair of clusters if they share significant numbers of GO terms; the similarity can be adjusted as a filter.

4.2.2 Revelation of the Hierarchical Structure of the Network (Based on *AT5G13320* (*PBS3*))

Similar to Section 4.1.2, this section demonstrates how the zooming function can lead a researcher to the gene of interest (*AT5G13320* (*PBS3*) in this case). *PBS3* encodes a member of an auxin-responsive GH3 family of acyl-adenylate/thioestor-forming enzymes, some of which have been shown to catalyze hormone-amino acid conjugation. In Figure 4.7, the clusters containing *PBS3* are highlighted in all of the views; the granularities of detail in each view vary from coarsest to finest.

Figure 4.7A depicts the whole network, composed of the clusters that are labeled with broad biological processes such as *regulation of transcription*, *DNA-dependent*, *response to heat*, and *embryo development ending in seed development*. *PBS3* is found under the cluster of *signal transduction*, which is highlighted. Indeed, *PBS3* is involved in a signal transduction cascade of systemic acquired resistance or SAR, which is a systemic immune response in plants against a pathogen [59]. Zooming in on the *signal transduction* cluster reveals that the *systemic acquired resistance* cluster is highlighted, meaning that *PBS3* is grouped under this cluster (Figure 4.7B). This result is in accordance with many sources stating that *PBS3* affects SAR [59, 60]. Zooming in on the *systemic acquired resistance* cluster at this stage indicates that *PBS3* is a member of the *defense response to bacterium* cluster (Figure 4.7C), which obviously related to defense mechanisms. Finally, Figure 4.7D illustrates the deepest view after zooming in on the *defense response to bacterium* cluster. At this stage, the relationships between *PBS3* and other genes related to SAR, such as *AT1G74710* (*ICS1*), *AT1G33960* (*AIG1*), and *AT1G19250* (*FMO1*), are depicted [59, 61, 62]. This example illustrates that clusters comprising co-expressed genes of similar functions are sensibly created and their roles are indicated informatively and correctly.

4.2.3 Function Elucidation of *WRKY28* and *WRKY46*, Key Players of Systemic Acquired Resistance

Shown in Figure 4.8 is a view centered on the genes of interest whose functions were unknown (in this case, *WRKY28* (*AT4G18170*) and *WRKY46* (*AT2G46400*)). By invoking the re-centering function, a user can grasp what types of genes exist around the genes of interest in this network. In Figure 4.8, the clusters related to the defense response, such as *response to chitin*, *response to salicylic acid*, and *response to abscisic acid stimulus*, surround the two genes, suggesting that these genes are also involved in this function. Actually, they have recently been confirmed to be positive regulators of *ICS1* and *PBS3*, which are key players of systemic acquired resistance (SAR) [60]. It is

Figure 4.7: The hierarchical organization of the clusters containing *AT5G13320* (*PBS3*), a gene of interest, which are highlighted in light yellow. **(A)** The most abstracted view. *PBS3* belonged to the *signal transduction* cluster. **(B)** At this level, *PBS3* was found in the *systemic acquired resistance* cluster. **(C)** *PBS3* was included in the *defense response to bacterium* cluster. (D) The most specific view. *PBS3* was clustered together with ten other genes, such as *AIG1*, *ICS1*, and *FMO1*.

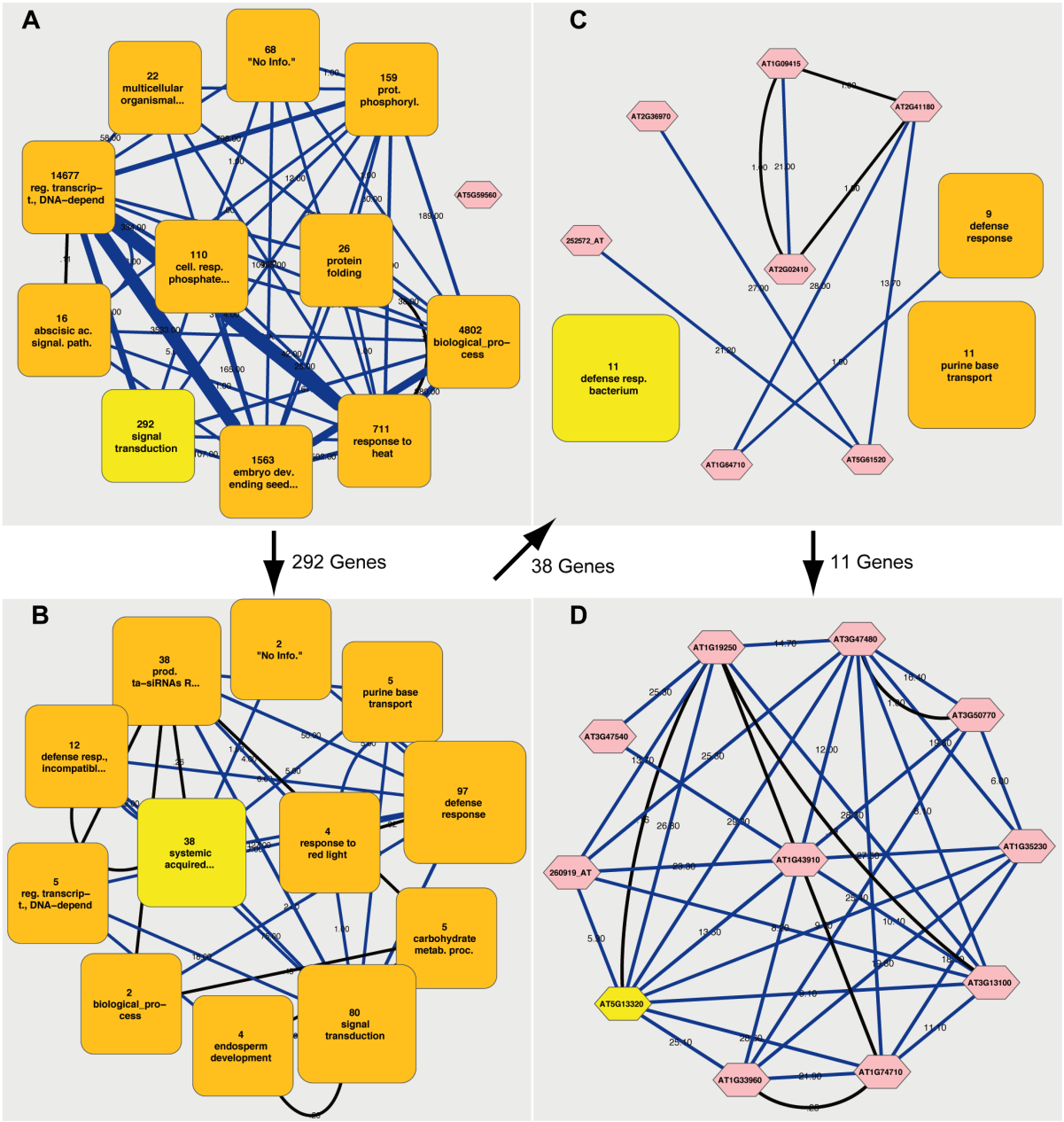
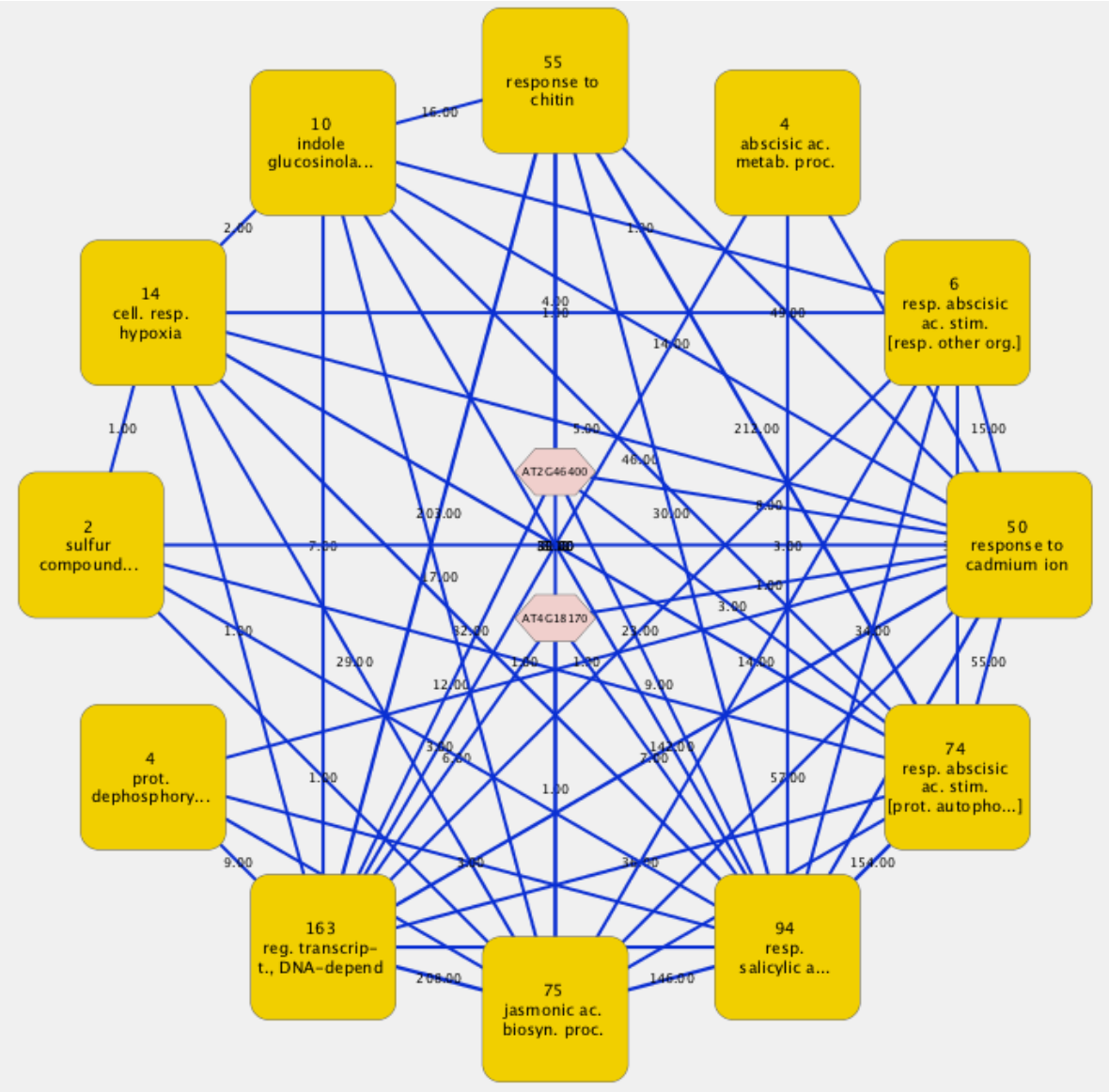


Figure 4.8: Re-centered view on *AT4G18170* and *AT2G46400* with a threshold set to two. The re-centering function allows researchers to intuitively grasp what types of genes exist around the genes of interest in large networks. All of the genes that are within two hops from these two genes were clustered to a visually interpretable level. One can immediately see clusters related to *response to chitin*, *response to salicylic acid*, and *response to abscisic acid stimulus*, around the genes of interest, implying that *AT4G18170* and *AT2G46400* are involved in defense response mechanisms.



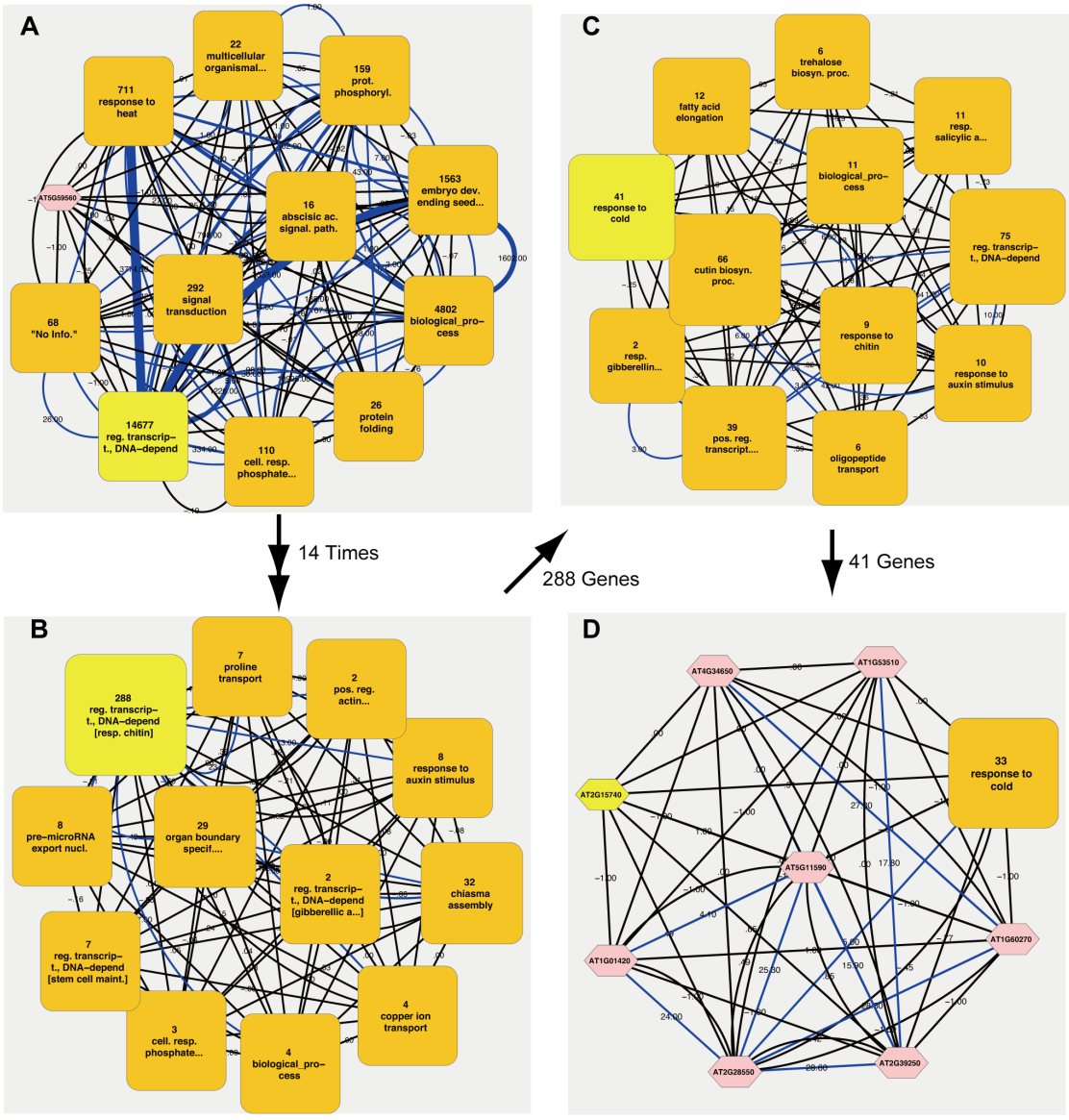
extremely difficult to infer this kind of knowledge from huge network data if a cluttered visualization is employed. This example clearly proves again how the proposed visualization approach can provide meaningful and interpretable information for researchers.

4.2.4 Discovery of a New Role of *AT2G15740* (*GsZFP1*) as a Positive Regulator of Plant Tolerance to Cold Stress

Figure 4.9 shows some abstracted views at different levels of granularity of the *Arabidopsis* gene co-expression network in the case *AT2G15740* is searched for. The clusters containing *AT2G15740* (*GsZFP1*) are highlighted in all views and the granularities of detail in each view vary from coarsest to finest. *AT2G15740* is one of the C2H2-type zinc-finger proteins (ZFPs) that have been implicated in different cellular processes involved in plant development and stress responses [63–65]. Specifically, it is an alkaline (NaHCO3)-responsive ZFP gene, which was originally identified in soybeans, *Glycine soja*. In [63], it was introduced as a transgene into *Arabidopsis* for testing under various environments such as cold and drought. Overexpression of the gene in *Arabidopsis* resulted in a greater tolerance to cold and drought stress and increased the expression of many stress-response marker genes that play a role in response to cold and drought, thereby indicating that *AT2G15740* may be a positive regulator of plant tolerance to cold and drought stress [63].

In Figure 4.9A, *AT2G15740* is grouped under the *regulation of transcription, DNA-dependent* cluster, the largest cluster in which most genes reside. After zooming in on this cluster 14 times, one can find clusters of more specific processes, such as *proline transport*, *chiasma assembly*, and *response to auxin stimulus* (Figure 4.9B). In this view, *AT2G15740* is still clustered under the *regulation of transcription, DNA-dependent* cluster with an additional label as *response to chitin*, suggesting that the gene might be involved in this process. In Figure 4.9C, *AT2G15740* is a member of the cluster of *response to cold* at this level, which is consistent with the recently found role as involved with tolerance to cold stress [63]. After zooming in on this cluster, the final view reveals property edges between *AT2G15740* and genes implicated in various biological processes (e.g., *organ morphogenesis*, *cortical microtubule*, and *regulation of transcription, DNA-dependent*) and one cluster related to *response to cold* (Figure 4.9D). This shows that zooming in on the network by searching for *AT2G15740* can also suggest the role of the gene interestingly and correctly.

Figure 4.9: The hierarchical organization of the clusters focused on *AT2G15740* or *GsZFP1*, a gene of interest. The numbers by the arrows indicate the numbers of gene contained in the processed sub-networks or the number of zooming operations. Hexagons represent genes that are not contained in any cluster at a view. **(A)** The most abstracted view. *AT2G15740* belonged to the highlighted *regulation of transcription, DNA-dependent* cluster. After zooming in on this cluster, the clusters containing *AT2G15740* of 13 deeper views were also the clusters labeled *regulation of transcription, DNA-dependent* and were excluded for the sake of conciseness. **(B)** At this level, *AT2G15740* was contained in the *regulation of transcription, DNA-dependent* cluster with *response to chitin* in a bracket. Clusters labeled with the same property terms as those of others are discriminated by additionally displaying the next highest score terms in brackets. **(C)** *AT2G15740* was contained in the *response to cold* cluster. **(D)** The most specific view. *AT2G15740* was clustered together with the cluster of *response to cold*, *AT4G34650*, *AT1G53510*, *AT1G60270*, *AT5G11590*, *AT2G39250*, *AT2G28550*, *AT1G01420*.



Chapter 5

Discussion

5.1 Comparison with Existing Network Visualization Methods

Table 5.1 shows the comparison of the proposed method with related network visualization tools in aspects of cluster generation means, multi-scale navigation support, purpose of use, development architecture, measures for handling overwhelming visualization (insufficient clustering), and support for flexible navigation beyond cluster boundaries. Stand-alone applications (NaviCluster, BioLayout Express^{3D}, jClust/Medusa, RobinViz, VisANT) [10, 11, 20, 36, 66, 67], web-based tools (Cellular Overview) [68], and Cytoscape plug-ins (NaviClusterCS, CyOog, clusterMaker, CyClus3D, NeMo, MODEVO, GenePro) [12, 13, 17–19, 69] are included here.

Most of them support automatic cluster generation using various underlying clustering or community identification algorithms, except for VisANT [36], Cellular Overview [68], and GenePro [13]. VisANT [36] provides multi-scale visualization; however, the user must manually create metanodes (equivalent to clusters) themselves. CyOog (Power Graphs [12]) 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. Concerning the multi-scale visualization, GenePro [13], BioLayout Express^{3D} [10], and jClust [11, 67] can visualize only a single level of clusters; they do not support visualization of recursive clustering.

Some provide multi-scale network navigation in each own way. For instance, in contrast to the intuitive zooming style adopted by the presented method, clusterMaker [69] support multi-scale navigation via dendrograms, whose parts, when selected by users, are reflected in the network view of Cytoscape. Whereas most tools aim at visualizing various types of biological networks, MODEVO [17], RobinViz [20], Cellular

Overview [68], and GenePro [13], are tailored for specific types of networks, such as interaction networks and metabolic networks.

Among all the tools shown in Table 5.1, only the proposed method and RobinViz [20] offer flexible navigation across created clusters. RobinViz implements the 2-hop neighborhood operation, where neighbors that are 2-hop far from selected nodes are gathered and visualized. The proposed method, however, goes beyond this by allowing for gathering and visualizing more than 2-hop neighbors, via the re-centering function. This can be done without producing cluttered visualization, as it automatically clusters the resultant networks after the re-centering function is invoked. Notably, adopting the property-based clustering, the presented method is the only tool that imposes measures for handling the case of hard-to-manage and complicated visualization of the overwhelming numbers of nodes and edges, due to insufficient clustering.

5.2 Utility of the Presented Method

The benefits of the main features, zooming and re-centering, are shown previously in Chapter 4, via many examples. In this section, I summarize those use cases and clarify the utility of the presented method. By the zooming function, as can be seen in Figures 4.2, 4.4, 4.7 and 4.9, proteins or genes of related biological processes can be clustered together sensibly and their roles are indicated insightfully, correctly, and appropriately based on the granularity of each view. The amount of information displayed is kept tractable by showing coarse and fine information on the abstracted and detailed views, respectively. Every searching-and-zooming step is rendered in a matter of seconds, which allows researchers to gather interesting information at the desired degree of detail easily, rapidly, and effectively, thereby fully supporting interactive biological investigation of massive datasets.

In addition to the multi-scale navigation of large networks, the examples shown in Figures 4.3, 4.5 and 4.8 indicate that the proposed method can also be used to intuitively and effectively retrieve knowledge on nodes of interest from large and complicated networks. In Figure 4.3, abstraction of the three proteins' neighbors is necessary because, even if it is possible for biologists to manually investigate all 49 direct neighbors shown in Figure 4.3A, it is not practical to do the same for the 1,443 indirect neighbors shown in Figure 4.3B. In the same way, it is not trivial to manually investigate the relationships between two *WRKY* genes and 501 neighbors, as shown in Figure 4.8. These two examples confirm the importance of the re-centering function, especially in the case of Nas6, Rpn14, Hsm3, as it is the indirect information that suggested the chaperone role—only information on relationships with the direct neighbors was not enough. However, it was

not possible to *interactively* create these views using previous methods that depend on fixed cluster hierarchies.

In the case of Nsi1 (Ydr026c), as discussed in [53], the protein was actually confirmed to be a true player of chromatin silencing at rDNA. Notably, however, after getting the results showing that Nsi1 is involved in chromatin silencing at rDNA, Ha et al. then also proceeded to confirm whether it is involved in chromatin silencing at telomere [53] and found a *negative* result. This seems to contradict with what are delineated in Figure 4.5. However, in the view shown in Figure 4.5, when ones hover the pointer over the *chromatin silencing at telomere* cluster, they can see that additionally the second highest score property term of this cluster is *chromatin silencing at rDNA*. In this case, although the results from experiments are against the highest score property term, I argue that instead, this in fact shows the presented method provides researchers with interesting candidates that make sense enough to be picked up for verification by scientists. Besides, the biological processes appearing as labels of the clusters of Nsi1 in each view, namely *rRNA processing*, *transcription from RNA polymerase III promoter*, and *transcription of nuclear large rRNA transcript from RNA polymerase I promoter*, shown in Figure 4.4, deserve validation as well, because of high probability that proteins residing in the same clusters would be implicated in many overlapping functions (the so-called guilt-by-association approach [70–72]). It should be noted that the surrounding clusters in Figure 4.5 were formed without using any property information of Nsi1; actually, in this dataset, Nsi1 was annotated with only the "*biological_process*" term, ultimately illustrating that the proposed method can really be used for suggesting unknown functions of poorly annotated proteins.

Similar to Nsi1, we have little information about *AT2G15740* too. In the ATTED-II dataset used in Section 4.2, in fact, there existed no edges connecting gene *AT2G15740* and other genes at all. Despite this, only unraveling the network via zooming could also delineate and suggest the relationships of the gene to *response to cold* correctly in the two most detailed views (Figure 4.9), showing that the proposed method is suitable for both exploring the hierarchical organization of large networks and investigating unknown functions of genes at the same time. Note that because the gene had no connections to others, we cannot use the re-centering function with this gene, as this will result in a view composed of only one node, *AT2G15740* itself.

Interestingly, compared to the yeast dataset, the *Arabidopsis* network in this study was clustered into a large number of levels due to the characteristics of the network. Most of the genes were grouped under the *regulation of transcription, DNA-dependent* clusters. Consequently, this resulted in cumbersome tasks for users to zoom in on the clusters many times before reaching the lowest level. Whether to preserve the modularity of

clustering strictly or to balance the hierarchy of clusters for better organizations and the subsequent ease of navigation is still be controversial and needs careful consideration[22, 26]. In my opinion, an obvious workaround is to devise a navigation approach that lets researchers "jump" between graph views more freely without zooming in on clusters in sequence or "skip" the clusters that are labeled the same as previous views. This should be highly feasible, thanks to the speed of the Louvain algorithm. Nonetheless, as one can see that users cannot get much information from zooming in on the clusters labeled the same for fourteen times, I am also interested in the idea of compromising the modularity of clustering up to an *acceptable* level, if that leads to graph views with more informative and useful information, such as provision of overlapping clusters. Of course, this issue partly depends on the dataset generation methods as well. Unlike PPI networks, in many co-expression datasets, data values for *all* pairs are provided and can be used to generate networks in numerous ways. In this research, I used mutual ranks as suggested by the authors of ATTED-II and also filtered co-expressions whose ranks are larger than 30. Generating alternative meaningful networks based on the same original dataset may lead to different and easy-to-navigate abstracted views. Moreover, in an empirical study, I was also faced with the similar issue, when applying the method to networks of publications extracted from PubMed (`http://www.pubmed.org`) based on a specific query, using MeSH (Medical Subject Headings) (`http://www.nlm.nih.gov/mesh/`) terms from the MEDLINE database, as property information and cosine similarities between publications as the similarity measure. In particular, the resulting abstracted views were unfavorably deep and need a lot of zooming operations to unravel the network. As future work, although not trivial, all of these issues will be researched to find more satisfactory solutions.

5.3 Biological Validation of Clustering Methods

Up until now, research publications on biological data clustering have reported and validated their results in various ways. Some emphasized the qualities of the results, whose definition themselves are quite broad. Many studies focused on using well-defined mathematical measures, such as modularity (Q), to evaluate their results [12, 28, 73–75], whereas others used comparison against their designated gold standards, such as KEGG (Kyoto Encyclopedia of Genes and Genomes) (`http://www.genome.jp/kegg/`) or GO (`http://www.geneontology.org`) [10, 12, 27]. Some studies validated their results using the speed of clustering as well [22, 26]. However, as stated in Chapter 1, none of the existing methods are suitable to be incorporated into the *interactive* and *multi-scale* navigation scheme, due to the limited speed and scalability problems.

The validation of clustering biological data would still be controversial, actively discussed, and would have room for improvement for years. Each work commonly claimed that it was better than others in its *own* benchmark. This is partly due to the various purposes of performing clustering and types of the data; for instance, expected results from clustering DNA microarray data may be different from clustering protein interaction networks. In addition, the undisputed standards for assessment still do not exist; hence numerous distinct directions of research and validation methods are feasible to conduct. The goal of clustering has still being evolved in these recent years, as the abundant amount of data is still being produced, thereby increasing the demand of high-performance clustering and intuitive visualization of the data.

Above all, the interactive and multi-scale characteristics of the presented method has *never* existed before in any existing studies. Although no one could claim that a certain method completely overcomes *all* other approaches in *all* aspects, the utility and benefits of the proposed method when applied to different types of functional genomics data have been explicitly demonstrated in this dissertation. Thus, it is promising to be used with various kinds of biological networks as well and I argue that the presented method is indeed crucial to effective investigation of biological data in this big-data era. Furthermore, as discussed in the last chapter, this kind of interactive navigation would pave the road to interactive biological hypothesis generation, and finally accelerating the pace of interactive knowledge discovery drastically.

5.4 Extendibility of the Method

It should be noted that the proposed 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 publications 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.

Table 5.1: A comparison table of related network visualization tools.

Tool / Features	Cluster Generation	Multi-Scale Navigation	Purpose of Use	Architecture	Flexible Navigation Beyond Cluster Boundaries	Measures for Handling Insufficient Clustering
NaviCluster [66]/ NaviClusterCS	Automatic and Extremely Fast	Yes	Generic	Stand-Alone/ Cytoscape Plug-in	Yes (Re-Centering)	Property-Based Clustering
CyOog (Power Graph) [12]	Automatic	Yes (Power Nodes)	Generic	Cytoscape Plug-in	No	No
clusterMaker [69]	Automatic	Yes (Dendrogram)	Generic	Cytoscape Plug-in	No	No
CyClus3D [18]	Automatic	No	Generic	Cytoscape Plug-in	No	No
NeMo [19]	Automatic	No	Generic	Cytoscape Plug-in	No	No
BioLayout Express^{3D} [10]	Automatic	No	Generic	Stand-Alone	No	No
jClust/Medusa [11, 67]	Automatic	No	Generic	Stand-Alone	No	No
MODEVO [17]	Automatic	No	Protein Interaction Network	Cytoscape Plug-in	No	No
RobinViz [20]	Automatic	No	Protein Interaction Network	Stand-Alone	Yes (2-Hop Neighborhood)	No
VisANT [36]	Manual	Yes (Metanodes)	Generic	Stand-Alone	No	No
Cellular Overview [68]	Manual and Stored in Databases	Yes (Zooming User Interface)	Metabolic Network	Web-Based	No	No
GenePro [13]	Manual	No	Interaction Network	Cytoscape Plug-in	No	No

Chapter 6

Conclusions and Prospects

To summarize, this dissertation presents the *first* method for interactive and multi-scale navigation of large, complicated biological networks that displays appropriately abstracted views at all levels of detail. The importance and novelty of the research question raised in this dissertation have recently been confirmed by very recent publications citing this work [76–78]. The paper was referred to as an active, pioneering research topic in bioinformatics, in almost the last part of Discussion of those publications, indicating that effective visualization and navigation of large and complicated biological networks are still to be gained much attention by bioinformaticians in this very near future.

Reflected in both implemented tools, the specially designed features and interfaces, particularly the re-centering function, enable flexible navigation across cluster boundaries. Application to real functional genomics data, YeastNet and ATTED-II, demonstrates how this research achieves the goal of providing effective, intuitive, and interactive navigation and why the method is eminently suitable for large biological networks. As shown by analysis of Cla4, Nsi1, *PBS3* and *GsZFP1*, every time the researcher zooms in, clusters were constructed on the fly and visualized immediately along with meaningful labels. All views were informative as to the hierarchical structure of the network. As for Nas6, Rpn14, and Hsm3, their roles as chaperones were suggested in the view created by the re-centering function, which encompassed the indirect neighbors of these nodes. Similar elucidations of functions can also be obtained in the case of *WRKY* genes and Nsi1. This outcome indicates that, in addition to the usual network exploration accomplished by displaying direct interactors, the presented methods can also uncover interesting hidden facts in large, complicated networks via both main features. These will help researchers formulate new hypotheses more easily and systematically, even if they possess little experience in bioinformatics. I believe that the presented method

will aid modern biologists in discovering knowledge from massive binary-relationship datasets, which are accumulating at an accelerating pace.

As stated previously, the research presented in this dissertation is comparatively new to the field, thus there are various future directions and extensions of the work that can be conducted. In this section, I raise four main directions that seem highly feasible and promising to follow.

6.1 Clustering and Implementation Issues

To begin with, we can improve the proposed method in aspects of clustering processes and implementation. (1) As the Louvain clustering is a greedy algorithm, the orders of input nodes fed into the algorithm can affect the final results and their modularity, so it is expected to see research on improving the results via input patterns. (2) The proposed method sometimes works significantly slower than normal, when faced with some kinds of networks, such as publication networks generated from PubMed. This is attributed to the deteriorated performance of the property-based clustering. Currently, it depends coherently on the FFT algorithm, whose performance can dramatically decrease when used with ultimately high dimensional property vectors (data points), such as the case of MeSH terms annotated to publication networks. Therefore, devising techniques for evading from "the curse of dimensionality" is explicitly necessary. I plan to investigate this issue more, starting from the survey about algorithms in the same family, such as k-medoid, k-means++ [8, 79]. Furthermore, as performed by Clauset et al. [24], some customization of data structures or implementation optimization might be an alternative way to do first to improve the performance without developing new algorithms. (3) Utilizing property information more during the course of clustering may be promising. For instance, as the FFT algorithm is also greedy, the input patterns influence the resulting clusters. Thus, when the property-based clustering is executed, the property information may be taken into consideration to choose the first Louvain cluster to be processed that gives more favorable results. In addition, as demonstrated in the previous chapters, cosine similarity gives satisfactory results and is generally suitable for any kinds of annotations. However, in life sciences where researchers heavily rely on GO, this similarity scheme might be too simple. In fact, there are many studies and research on similarity measures of GO published every year, showing that this is also a very hot field [80–83]. Incorporating knowledge about the GO hierarchy should give rise to a more sophisticated and appropriate similarity measure to be used with GO as property information. (4) Stated in Chapter 3, NaviClusterCS was developed to make use of abundant features of Cytoscape and it partly achieved the goal already. However, to

make it even more compatible with other plug-ins and functions of Cytoscape, more work is needed. I plan to improve it to run more smoothly and be compatible with the new version (Cytoscape 3). Besides, as HTML5 has many powerful features and can be seamlessly used in any platforms, it draws attention from developers around the world. This is also the case for bioinformaticians and Cytoscape developers. In my opinion, Cytoscape Web, developed based upon HTML5, will be the killer application of the field in the near future—of course, especially if it is bundled with the presented method.

6.2 Enhancement of Functionalities, Visualization, and User Experiences

Secondly, functionalities, visualization schemes, and user experiences still have room for improvement. (1) As the clustering components of the method automatically split clusters to fulfill the canvas, sometimes it is hard to quickly see what kinds of clusters originally existed in the view—the user needs to investigate the property information of each node one-by-one to get the overall picture. Therefore, it may be interesting to let users manually expand/collapse some clusters in graph views, provided that the overall number of objects on the screen is still less than a threshold. (2) Although "Make Custom Graph View" and zooming on more than one clusters can assist in flexibly generating new views based on user-selected clusters/nodes, it is sometimes not trivial to pinpoint genes/proteins of interest residing in large clusters. Reforming the searching and selecting functions obviously improve user experiences and let researchers do their work more readily; for example, this can help them quickly search for multiple nodes in different clusters before performing re-centering, through the improved method. (3) According to some feedbacks from users, information on edges or relationships is unfortunately less elucidated in visualization for now. Despite the fact that many researchers are interested in how their proteins/genes of interest are *located* in the whole network or how they form *clusters* with others, *relationships* or *edges* with other proteins/genes are important for some researchers. In the current implementation, information on nodes and clusters is predominantly visualized via labeling and context-sensitive menus; however, for example, showing the weights of property edges as cosine similarity values may not be good enough to be truly user-friendly and informative for some researchers. (4) There are usually more than one interesting property terms for each node or cluster. Showing other reliable statistical values (e.g., p-value) along with those terms and/or effectively showing multiple property terms at the same time should make biologists more confident of using the information.

6.3 Multiple Types of Networks, Different Kinds of Requirements

Thirdly, up to now, I applied the method to only functional genomics datasets that are undirected networks (graphs). It is undisputedly interesting to apply the presented method to other kinds of networks, such as signal transduction pathways, metabolic pathways and other omics data as well. Recently, the Louvain method has been successfully applied to homologous sequences already [32]. As a plan, I would like to apply the method to integrated networks of multiple types of omics data (e.g., proteome, genome, metabolome, and phenome). In my opinion, through interactive navigation, this would stimulate better holistic understanding of organisms under study. However, much care and improvements of the method are needed to handle with different types of networks, because they may need distinct similarity measures, visualization schemes or knowledge suggestion approaches. For instance, Ma et al. claimed that the guilt-by-association method was not proper for gene-drug networks, as it was too coarse to make quantitative inference about drug effects at the system level [84]. On the other hand, in publication networks, showing main keywords or important figures found in the papers instead of their authors' names or titles may be more practical and intuitive.

6.4 Toward Interactive, Systematic Knowledge Discovery

Finally, my ultimate goal when conducting this research is to create *interactive*, *systematic* hypothesis generation and knowledge discovery systems in the future. Apart from developing better function prediction or knowledge suggestion approaches for proteins/genes, as an example, acquiring unknown subcellular localization (e.g., based on the cellular component namespace of GO) is of interest as well, but cannot be done easily for nodes with no annotations and connections. This is because they would be gathered together in one separate cluster from others at first. Even when the re-centering function is invoked, they will be visualized alone. To me, the most promising technique to be used here is integration of the implemented tools with knowledge bases and text mining/processing engines, which have gained much popularity in biology and biomedicine [85–91]. Specifically, those that store and extract information from full-text articles are favorable [92–94], because of many studies reporting that the information density and coverage of full texts are considerably larger than abstracts [95–97]. There is much knowledge embedded in a great number of literatures that is still not discovered and utilized by researchers. Using this information for suggesting new knowledge or

formulating hypotheses are thus reasonable and more likely to succeed than using the datasets provided by users alone. Not surprisingly, nevertheless, the bottleneck of the overall workflow for now is the text mining/processing systems, which need dramatic improvements in performance to come up with the rapid visualization method already accomplished in this research. I believe that, when the improvements of both methods successfully reach their peaks, truly *interactive*, *systematic* knowledge discovery systems would enormously benefit a wide range of researchers and change the world of research completely.

References

[1] Merico D, Gfeller D, Bader GD: **How to visually interpret biological data using networks**. *Nat Biotechnol* 2009, **27**(10):921–924.

[2] Suderman M, Hallett M: **Tools for visually exploring biological networks**. *Bioinformatics* 2007, **23**:2651–2659.

[3] Ideker T, Thorsson V, Ranish JA, Christmas R, Buhler J, Eng JK, Bumgarner R, Goodlett DR, Aebersold R, Hood L: **Integrated genomic and proteomic analyses of a systematically perturbed metabolic network**. *Science* 2001, **292**:929–34.

[4] Evanko D: **Supplement on visualizing biological data**. *Nat Methods* 2010, **7**(3s):S1.

[5] Gehlenborg N, O'Donoghue S, Baliga N, Goesmann A, Hibbs M, Kitano H, Kohlbacher O, Neuweger H, Schneider R, Tenenbaum D, Gavin A: **Visualization of omics data for systems biology**. *Nat Methods* 2010, **7**(3 Suppl):S56–68.

[6] Hu ZJ, Mellor J, Wu J, Kanehisa M, Stuart JM, DeLisi C: **Towards zoomable multidimensional maps of the cell**. *Nat Biotechnol* 2007, **25**(5):547–554.

[7] O'Donoghue S, Gavin A, Gehlenborg N, Goodsell D, Hériché J, Nielsen C, North C, Olson A, Procter J, Shattuck D, Walter T, Wong B: **Visualizing biological data-now and in the future**. *Nat Methods* 2010, **7**(3 Suppl):S2–4.

[8] Andreopoulos B, An AJ, Wang XG, Schroeder M: **A roadmap of clustering algorithms: finding a match for a biomedical application**. *Brief Bioinform* 2009, **10**(3):297–314.

[9] Abello J, van Ham F, Krishnan N: **ASK-GraphView: a large scale graph visualization system**. *IEEE Trans Vis Comput Graph* 2006, **12**(5):669–676.

[10] Freeman TC, Goldovsky L, Brosch M, van Dongen S, Mazière P, Grocock RJ, Freilich S, Thornton J, Enright AJ: **Construction, visualisation, and clustering of transcription networks from microarray expression datas**. *PLoS Comput Biol* 2007, **3**(10):2032–2042.

[11] Pavlopoulos GA, Moschopoulos CN, Hooper SD, Schneider R, Kossida S: **jClust: a clustering and visualization toolbox**. *Bioinformatics* 2009, **25**(15):1994–1996.

[12] Royer L, Reimann M, Andreopoulos B, Schroeder M: **Unraveling protein networks with power graph analysis**. *PLoS Comput Biol* 2008, **4**(7):e1000108.

[13] Vlasblom J, Wu S, Pu S, Superina M, Liu G, Orsi C, Wodak SJ: **GenePro: a Cytoscape plug-in for advanced visualization and analysis of interaction networks**. *Bioinformatics* 2006, **22**(17):2178–2179.

[14] Clauset A, Moore C, Newman MEJ: **Hierarchical structure and the prediction of missing links in networks**. *Nature* 2008, **453**(7191):98–101.

[15] Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T: **Cytoscape: a software environment for integrated models of biomolecular interaction networks**. *Genome research* 2003, **13**(11):2498–504.

[16] Enright A, Van Dongen S, Ouzounis C: **An efficient algorithm for large-scale detection of protein families**. *Nucleic Acids Res* 2002, **30**(7):1575–1584.

[17] Wozniak M, Tiuryn J, Dutkowski J: **MODEVO: exploring modularity and evolution of protein interaction networks**. *Bioinformatics* 2010, **26**(14):1790–1.

[18] Audenaert P, Van Parys T, Brondel F, Pickavet M, Demeester P, Van de Peer Y, Michoel T: **CyClus3D: a Cytoscape plugin for clustering network motifs in integrated networks**. *Bioinformatics* 2011, **27**(11):1587–8.

[19] Rivera CG, Vakil R, Bader JS: **NeMo: Network Module identification in Cytoscape**. *BMC Bioinformatics* 2010, **11 Suppl 1**:S61.

[20] Aladag AE, Erten C, Sozdinler M: **Reliability-oriented bioinformatic networks visualization**. *Bioinformatics* 2011, **27**(11):1583–4.

[21] Yamada T, Bork P: **Evolution of biomolecular networks: lessons from metabolic and protein interactions**. *Nat Rev Mol Cell Biol* 2009, **10**(11):791–803.

[22] Blondel V, Guillaume J, Lambiotte R, Lefebvre E: **Fast unfolding of communities in large networks**. *J Stat Mech* 2008, **2008**(10 (October, 2008)):P10008.

[23] Newman MEJ, Girvan M: **Finding and evaluating community structure in networks**. *Phys Rev E* 2004, **69**(2):026113.

[24] Clauset A, Newman MEJ, Moore C: **Finding community structure in very large networks**. *Phys Rev E* 2004, **70**(6):066111.

[25] Newman MEJ: **Fast algorithm for detecting community structure in networks**. *Phys Rev E* 2004, **69**(6):066133.

[26] Wakita K, Tsurumi T: **Finding community structure in mega-scale social networks**. *ArXiv Computer Science* 2007, **e-prints**:0702048.

[27] Dunn R, Dudbridge F, Sanderson CM: **The use of edge-betweenness clustering to investigate biological function in protein interaction networks**. *BMC Bioinformatics* 2005, **6**:39.

[28] Luo F, Yang Y, Chen CF, Chang R, Zhou J, Scheuermann RH: **Modular organization of protein interaction networks** 2007, **23**(2):14.

[29] Xiang J, Hu K, Tang Y: **A class of improved algorithms for detecting communities in complex networks**. *Physica A* 2008, **387**(13):3327–3334.

[30] Wallace ML, Gingras Y, Duhon R: **A new approach for detecting scientific specialties from raw cocitation networks**. *J Am Soc Inf Sci Technol* 2009, **60**(2):240–246.

[31] De Vico Fallani F, Chessa A, Valencia M, Chavez M, Astolfi L, Cincotti F, Mattia D, Babiloni F: **Community structure in large-scale cortical networks during motor acts**. *Chaos, Solitons & Fractals* 2012, **45**(5):603–610.

[32] Miele V, Penel S, Daubin V, Picard F, Kahn D, Duret L: **High-quality sequence clustering guided by network topology and multiple alignment likelihood**. *Bioinformatics* 2012, **28**(8):1078–1085.

[33] Fortunato S: **Community detection in graphs**. *Physics Reports-Review Section of Physics Letters* 2010, **486**:75–174.

[34] Newman MEJ: **Communities, modules and large-scale structure in networks**. *Nature Physics* 2011, **8**:25–31.

[35] Tamminen M, Virta M, Fani R, Fondi M: **Large-scale analysis of plasmid relationships through gene-sharing networks**. *Mol Biol Evol* 2012, **29**(4):1225–1240.

[36] Hu ZJ, Hung JH, Wang Y, Chang YC, Huang CL, Huyck M, DeLisi C: **VisANT 3.5: multi-scale network visualization, analysis and inference based on the gene ontology**. *Nucleic Acids Res* 2009, **37**(Web Server):W115–W121.

[37] Lee I, Li Z, Marcotte E: **An improved, bias-reduced probabilistic functional gene network of baker's yeast, *Saccharomyces cerevisiae***. *PLoS One* 2007, **2**(10):e988.

[38] Obayashi T, Kinoshita K, Nakai K, Shibaoka M, Hayashi S, Saeki M, Shibata D, Saito K, Ohta H: **ATTED-II: a database of co-expressed genes and *cis* elements for identifying co-regulated gene groups in *Arabidopsis***. *Nucleic Acids Res* 2007, **35**(Database issue):D863–9.

[39] Bi E, Chiavetta J, Chen H, Chen G, Chan C, Pringle J: **Identification of novel, evolutionarily conserved Cdc42p-interacting proteins and of redundant pathways linking Cdc24p and Cdc42p to actin polarization in yeast**. *Mol Biol Cell* 2000, **11**(2):773–93.

[40] Bose I, Irazoqui J, Moskow J, Bardes E, Zyla T, Lew D: **Assembly of scaffold-mediated complexes containing Cdc42p, the exchange factor Cdc24p, and the effector Cla4p required for cell cycle-regulated phosphorylation of Cdc24p**. *J Biol Chem* 2001, **276**(10):7176–86.

[41] Gulli M, Jaquenoud M, Shimada Y, Niederhäuser G, Wiget P, Peter M: **Phosphorylation of the Cdc42 exchange factor Cdc24 by the PAK-like kinase Cla4 may regulate polarized growth in yeast**. *Mol Cell* 2000, **6**(5):1155–67.

[42] Benton B, Tinkelenberg A, Gonzalez I, Cross F: **Cla4p, a *Saccharomyces cerevisiae* Cdc42p-activated kinase involved in cytokinesis, is activated at mitosis**. *Mol Cell Biol* 1997, **17**(9):5067–76.

[43] Cvrcková F, De Virgilio C, Manser E, Pringle J, Nasmyth K: **Ste20-like protein kinases are required for normal localization of cell growth and for cytokinesis in budding yeast**. *Genes Dev* 1995, **9**(15):1817–30.

[44] Bosl W, Li R: **Mitotic-exit control as an evolved complex system**. *Cell* 2005, **121**(3):325–33.

[45] Höfken T, Schiebel E: **Novel regulation of mitotic exit by the Cdc42 effectors Gic1 and Gic2**. *J Cell Biol* 2004, **164**(2):219–31.

[46] Jensen S, Geymonat M, Johnson A, Segal M, Johnston L: **Spatial regulation of the guanine nucleotide exchange factor Lte1 in *Saccharomyces cerevisiae***. *J Cell Sci* 2002, **115**(Pt 24):4977–91.

[47] Seshan A, Bardin A, Amon A: **Control of Lte1 localization by cell polarity determinants and Cdc14**. *Curr Biol* 2002, **12**(24):2098–110.

[48] Tiedje C, Sakwa I, Just U, Höfken T: **The Rho GDI Rdi1 regulates Rho GTPases by distinct mechanisms**. *Mol Biol Cell* 2008, **19**(7):2885–96.

[49] Funakoshi M, Tomko RJ, Kobayashi H, Hochstrasser M: **Multiple assembly chaperones govern biogenesis of the proteasome regulatory particle base**. *Cell* 2009, **137**(5):887–99.

[50] Le Tallec B, Barrault M, Guérois R, T C, Peyroche A: **Hsm3/S5b participates in the assembly pathway of the 19S regulatory particle of the proteasome**. *Mol Cell* 2009, **33**(3):389–99.

[51] Roelofs J, Park S, Haas W, Tian G, McAllister F, Huo Y, Lee B, Zhang F, Shi Y, Gygi S, Finley D: **Chaperone-mediated pathway of proteasome regulatory particle assembly**. *Nature* 2009, **459**(7248):861–5.

[52] Saeki Y, Toh-E A, Kudo T, Kawamura H, Tanaka K: **Multiple proteasome-interacting proteins assist the assembly of the yeast 19S regulatory particle**. *Cell* 2009, **137**(5):900–13.

[53] Ha CW, Sung MK, Huh WK: **Nsi1 plays a significant role in the silencing of ribosomal DNA in *Saccharomyces cerevisiae***. *Nucleic Acids Res* 2012, **40**(11):4892–4903.

[54] Mohanty BK, Bastia D: **Binding of the replication terminator protein Fob1p to the Ter sites of yeast causes polar fork arrest.** *J Biol Chem* 2004, **279**(3):1932–1941.

[55] Uetz P, Giot L, Cagney G, Mansfield TA, Judson RS, Knight JR, Lockshon D, Narayan V, Srinivasan M, Pochart P, Qureshi-Emili A, Li Y, Godwin B, Conover D, Kalbfleisch T, Vijayadamodar G, Yang M, Johnston M, Fields S, Rothberg JM: **A comprehensive analysis of protein-protein interactions in *Saccharomyces cerevisiae***. *Nature* 2000, **403**(6770):623–627.

[56] Huang J, Moazed D: **Association of the RENT complex with nontranscribed and coding regions of rDNA and a regional requirement for the replication fork block protein Fob1 in rDNA silencing.** *Genes Dev* 2003, **17**(17):2162–2176.

[57] Beckouet F, Labarre-Mariotte S, Albert B, Imazawa Y, Werner M, Gadal O, Nogi Y, Thuriaux P: **Two RNA polymerase I subunits control the binding and release of Rrn3 during transcription.** *Mol Cell Biol* 2008, **28**(5):1596–1605.

[58] Obayashi T, Hayashi S, Saeki M, Ohta H, Kinoshita K: **ATTED-II provides coexpressed gene networks for *Arabidopsis***. *Nucleic Acids Res* 2009, **37**(Database issue):D987–91.

[59] Eulgem T, Somssich IE: **Networks of WRKY transcription factors in defense signaling**. *Curr Opin Plant Biol* 2007, **10**(4):366–71.

[60] van Verk MC, Bol JF, Linthorst HJ: **WRKY transcription factors involved in activation of SA biosynthesis genes**. *BMC Plant Biol* 2011, **11**:89.

[61] Wang Z, Li X: **IAN/GIMAPs are conserved and novel regulators in vertebrates and angiosperm plants**. *Plant Signal Behav* 2009, **4**(3):165–7.

[62] Mishina TE, Zeier J: **The *Arabidopsis* flavin-dependent monooxygenase FMO1 is an essential component of biologically induced systemic acquired resistance**. *Plant Physiol* 2006, **141**(4):1666–75.

[63] Luo X, Bai X, Zhu D, Li Y, Ji W, Cai H, Wu J, Liu B, Zhu Y: **GsZFP1, a new Cys2/His2-type zinc-finger protein, is a positive regulator of plant tolerance to cold and drought stress**. *Planta* 2011, **235**(6):1141–1155.

[64] Liu K, Wang L, Xu Y, Chen N, Ma Q, Li F, Chong K: **Overexpression of OsCOIN, a putative cold inducible zinc finger protein, increased tolerance to chilling, salt and drought, and enhanced proline level in rice**. *Planta* 2007, **226**(4):1007–1016.

[65] Ciftci-Yilmaz S, Mittler R: **The zinc finger network of plants**. *Cell Mol Life Sci* 2008, **65**(7-8):1150–1160.

[66] Praneenararat T, Takagi T, Iwasaki W: **Interactive, multiscale navigation of large and complicated biological networks**. *Bioinformatics* 2011, **27**(8):1121–7.

[67] Pavlopoulos GA, Hooper SD, Sifrim A, Schneider R, Aerts J: **Medusa: A tool for exploring and clustering biological networks**. *BMC Res Notes* 2011, **4**:384.

[68] Latendresse M, Karp PD: **Web-based metabolic network visualization with a zooming user interface**. *BMC Bioinformatics* 2011, **12**:176.

[69] Morris JH, Apeltsin L, Newman AM, Baumbach J, Wittkop T, Su G, Bader GD, Ferrin TE: **clusterMaker: a multi-algorithm clustering plugin for Cytoscape**. *BMC Bioinformatics* 2011, **12**:436.

[70] Li SS, Xu K, Wilkins MR: **Visualization and analysis of the complexome network of *Saccharomyces cerevisiae***. *J Proteome Res* 2011, **10**(10):4744–4756.

[71] Janga SC, Díaz-Mejía JJ, Moreno-Hagelsieb G: **Network-based function prediction and interactomics: the case for metabolic enzymes**. *Metab Eng* 2011, **13**:1–10.

[72] Schwikowski B, Uetz P, Fields S: **A network of protein-protein interactions in yeast**. *Nat Biotechnol* 2000, **18**(12):1257–1261.

[73] King A, Przulj N, Jurisica I: **Protein complex prediction via cost-based clustering.** *Bioinformatics* 2004, **20**(17):3013–20.

[74] Vlasblom J, Wodak SJ: **Markov clustering versus affinity propagation for the partitioning of protein interaction graphs**. *BMC Bioinformatics* 2009, **10**.

[75] Wang J, Li M, Chen J, Pan Y: **A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks**. *IEEE/ACM Trans Comput Biol Bioinform* 2011, **8**(3):607–620.

[76] Trujillo C, Cooper MM, Klymkowsky MW: **Using graph-based assessments within socratic tutorials to reveal and refine students' analytical thinking about molecular networks.** *Biochem Mol Biol Educ* 2012, **40**(2):100–107.

[77] Fung DC, Li SS, Goel A, Hong SH, Wilkins MR: **Visualization of the interactome: what are we looking at?** *Proteomics* 2012.

[78] Tuikkala J, Vähämaa H, Salmela P, Nevalainen OS, Aittokallio T: **A multilevel layout algorithm for visualizing physical and genetic interaction networks, with emphasis on their modular organization**. *BioData Min* 2012, **5**:2.

[79] Arthur D, Vassilvitskii S: **k-means++: The advantages of careful seeding**. *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 2007, :1027–1035.

[80] Spasic I, Ananiadou S, McNaught J, Kumar A: **Text mining and ontologies in biomedicine: making sense of raw text.** *Brief Bioinform* 2005, **6**(3):239–251.

[81] Schlicker A, Domingues FS, Rahnenführer J, Lengauer T: **A new measure for functional similarity of gene products based on Gene Ontology.** *BMC Bioinformatics* 2006, **7**:302.

[82] Rokhlenko O, Shlomi T, Sharan R, Ruppin E, Pinter RY: **Constraint-based functional similarity of metabolic genes: going beyond network topology**. *Bioinformatics* 2007, **23**(16):2139–2146.

[83] Wang JZ, Du Z, Payattakool R, Yu PS, Chen CF: **A new method to measure the semantic similarity of GO terms.** *Bioinformatics* 2007, **23**(10):1274–1281.

[84] Ma H, Zhao H: **iFad: an integrative factor analysis model for drug-pathway association inference.** *Bioinformatics* 2012.

[85] Cohen AM, Hersh WR: **A survey of current work in biomedical text mining.** *Brief Bioinform* 2005, **6**:57–71.

[86] Hossain MS, Gresock J, Edmonds Y, Helm R, Potts M, Ramakrishnan N: **Connecting the dots between PubMed abstracts.** *PLoS One* 2012, **7**:e29509.

[87] Papanikolaou N, Pafilis E, Nikolaou S, Ouzounis CA, Iliopoulos I, Promponas VJ: **BioTextQuest: a web-based biomedical text mining suite for concept discovery.** *Bioinformatics* 2011, **27**(23):3327–3328.

[88] Andronis C, Sharma A, Virvilis V, Deftereos S, Persidis A: **Literature mining, ontologies and information visualization for drug repurposing**. *Brief Bioinform* 2011, **12**(4):357–368.

[89] Arighi CN, Roberts PM, Agarwal S, Bhattacharya S, Cesareni G, Chatr-Aryamontri A, Clematide S, Gaudet P, Giglio MG, Harrow I, Huala E, Krallinger M, Leser U, Li D, Liu F, Lu Z, Maltais LJ, Okazaki N, Perfetto L, Rinaldi F, Sætre R, Salgado D, Srinivasan P, Thomas PE, Toldo L, Hirschman L, Wu CH: **BioCreative III interactive task: an overview.** *BMC Bioinformatics* 2011, **12 Suppl 8**:S4.

[90] Hirschman L, Burns GA, Martin K, Arighi C, Cohen KB, Valencia A, Wu CH, Chatr-Aryamontri A, Dowell KG, Huala E, Lourenço A, Nash R, Veuthey AL, Wiegers T, Winter AG: **Text mining for the biocuration workflow**. *Database* 2012, **2012**.

[91] Garten Y, Coulet A, Altman RB: **Recent progress in automatically extracting information from the pharmacogenomic literature**. *Pharmacogenomics* 2010, **11**(10):1467.

[92] Ramakrishnan C, Patnia A, Hovy E, Burns GA: **Layout-aware text extraction from full-text PDF of scientific articles**. *Source Code Biol Med* 2012, **7**:7.

[93] Garten Y, Altman RB: **Pharmspresso: a text mining tool for extraction of pharmacogenomic concepts and relationships from full text**. *BMC Bioinformatics* 2009, **10**(Suppl 2):S6.

[94] Muller HM, Kenny EE, Sternberg PW: **Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature**. *PLoS Biol* 2004, **2**(11):e309.

[95] Shah PK, Perez-Iratxeta C, Bork P, Andrade MA: **Information extraction from full text scientific articles: where are the keywords?** *BMC bioinformatics* 2003, **4**:20.

[96] Schuemie MJ, Weeber M, Schijvenaars BJA, van Mulligen EM, van der Eijk CC, Jelier R, Mons B, Kors JA: **Distribution of information in biomedical abstracts and full-text publications**. *Bioinformatics* 2004, **20**(16):2597–2604.

[97] Cohen KB, Johnson HL, Verspoor K, Roeder C, Hunter LE: **The structural and content aspects of abstracts versus bodies of full text journal articles are different.** *BMC Bioinformatics* 2010, **11**:492.

Appendix A

NaviCluster User Manual

An online manual is also available and updated at `http://navicluster.cb.k.u-tokyo.ac.jp/`.

A.1 Download & Run

Prerequisite: Java 6 or later must be installed in your computer.

1. Download and Install Java version 6 (with update later than 10) or later via `www.java.com`. You can check the version of Java installed in your computer by typing `java-version` in the Terminal (Mac OS X) or Command Prompt (Microsoft Windows). You should get the message saying something like, `java version "1.X.0_YY"`. X should be at least 6, meaning that your installed Java is version 6 or above, and YY denotes numbers that represent the update version.
2. Download `NaviCluster-bin.zip` via `http://navicluster.cb.k.u-tokyo.ac.jp/download.html` and decompress it. The compressed file contains `NaviCluster.jar`, the `lib` folder (libraries necessary for running NaviCluster), a property information file (`gene_ontology-100329-short.txt`), and a sample dataset composed of a node list file (`yeastnet2-noIEA.node`), which contains information about nodes and an edge list file (`yeastnet2-noIEA.edge`), which contains a list of edges and their weights.

   (a) The property information file is Gene Ontology (GO) derived from the OBO format available on its website (`http://www.geneontology.org`) at 29 March 2010.

   (b) The sample dataset is the *Saccharomyces cerevisiae* protein network YeastNet v.2 with 5,483 nodes and 102,803 edges.

(c) GO annotations for nodes in the node list file are obtained from *Saccharomyces* Genome Database (SGD), `http://www.yeastgenome.org/`.

3. Double-click `navicluster.sh` (Mac OS X) or `navicluster.bat` file (Windows). You can also run NaviCluster via the command

```
java -Xmx1024m -jar NaviCluster.jar
```

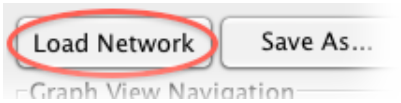
4. NaviCluster runs the two-stage clustering algorithms on the sample dataset and shows the main window. At this stage, you can start navigating the network.

Note: For Linux users, please type `-Dawt.useSystemAAFontSettings=on` between `java` and `-jar` in the above command to force Java to use anti-alias rendering for all GUI components.

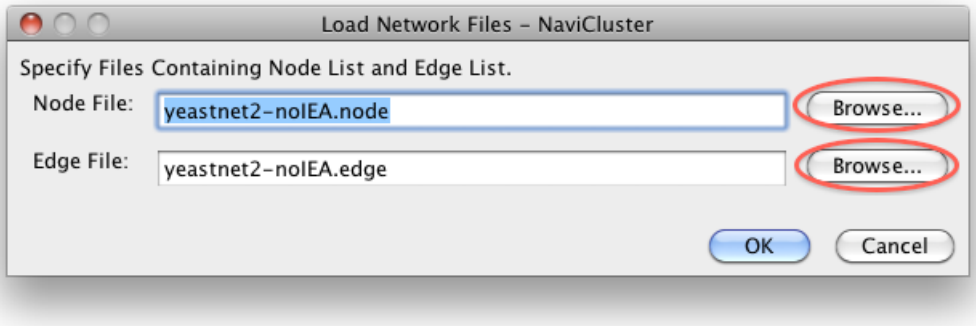
A.2 Network Loading

To load a new network, you need two files describing the network: a node list file and an edge list file. See Sections A.2.1, A.2.2 for details about the node list file format and edge list file format.

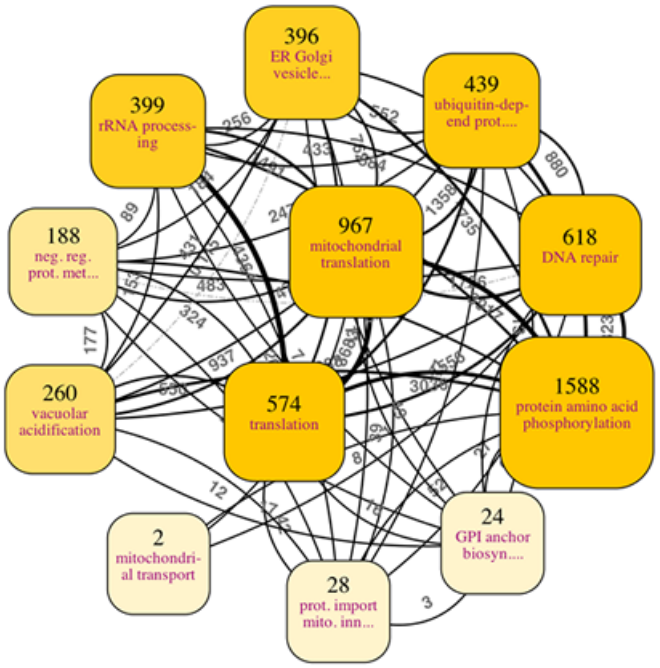
1. Click the Load Network button.



2. Click Browse... to select your node list file. Then, click Browse... to select your edge list file. A node list file must have `.node` as its extension. An edge list file must have `.edge` as its extension.



3. Click OK. NaviCluster then loads, clusters, and visualize your abstracted network.



A.2.1 Node List File Format

A node list file contains a list of nodes and their information. Together with an edge list file, they describe a network. To create a network that can be loaded in NaviCluster, you have to make your node list file comply with the following guidelines:

yeastnet2-nolEA.node

```
display_name    source_db_name  source_db_id    prop_info
CYR1    SGD     S000003542      GO:0005739|GO:0004016|GO:0004016|GO:0007265|GO:0007188|GO:0005739|GO:0007568|GO:0005886
SNA3    SGD     S000003687      GO:0008150|GO:0016020|GO:0000328|GO:0000328|GO:0003674
LSM5    SGD     S000000948      GO:0006399|GO:0000956|GO:0005732|GO:0005730|GO:0046540|GO:0005688|GO:0000398|GO:0003723|GO:
FOL2    SGD     S000003499      GO:0006807|GO:0005737|GO:0005634|GO:0003934|GO:0009396|GO:0005737|GO:0006082|GO:0006519|GO:
YGR054W SGD     S000003286      GO:0005840|GO:0022626|GO:0006413|GO:0006413|GO:0022627|GO:0003743|GO:0003743
SEC13   SGD     S000004198      GO:0006900|GO:0031080|GO:0043547|GO:0005198|GO:0090114|GO:0090114|GO:0006999|GO:0006999|GO:
RPS17A  SGD     S000004486      GO:0003735|GO:0022627|GO:0006417|GO:0000028|GO:0006412
SDH2    SGD     S000003964      GO:0006121|GO:0031966|GO:0008177|GO:0006099|GO:0005749|GO:0009117|GO:0051186|GO:0055114|GO:
GO:0008219|GO:0005739
YHR045W SGD     S000001087      GO:0008150|GO:0003674|GO:0005783
TCB3    SGD     S000004537      GO:0005739|GO:0008289|GO:0005933|GO:0008150|GO:0005739
TMS1    SGD     S000002512      GO:0005774|GO:0016020|GO:0003674|GO:0008150
BNA6    SGD     S000001943      GO:0034354|GO:0004514|GO:0005737|GO:0005634
CDA1    SGD     S000004298      GO:0005631|GO:0030476|GO:0030476|GO:0004099|GO:0004099
```

1. The first row of the file must be the column header names: `display_name`, `source_db_name`, `source_db_id`, and `prop_info`. Each name is separated by `"\t"`.
2. Node data starts from the second row. You can add your nodes in any order–they are not needed to be in an alphabetical order of their names. Each data item is separated by `"\t"`.

3. The meaning of each column:

   (a) `display_name`: the name of node. This is used to label the node on the canvas.

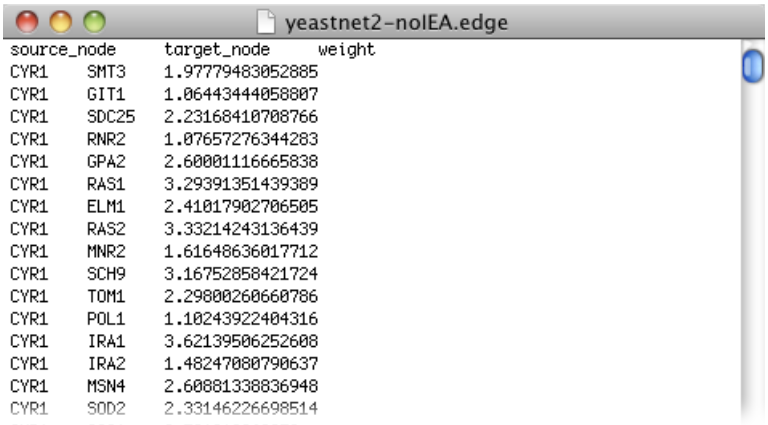
   (b) `source_db_name`: the name of source database where the node relies on. NaviCluster provides queries on the NCBI Entrez Gene database (`www.ncbi.nlm.nih.gov/gene`) using the name of node as input. In case you specifies the source database as SGD (*Saccharomyces* Genome Database), NaviCluster alternatively links the node to the SGD website `http://www.geneontology.org`.

   (c) `source_db_id`: The identifier of node in the source database. In case of SGD, the node is queried using this ID instead of its name.

   (d) `prop_info`: a list of property terms' identifiers annotated to the node. They are separated by a vertical bar `"|"`. Each property term must be described in the property information file.

A.2.2 Edge List File Format

An edge list file contains a list of edges and their weights. Together with a node list file, they describe a network. To create a network that can be loaded in NaviCluster, you have to make your edge list file comply with the following guidelines:



1. The first row of the file must be the column header names: `source_node`, `target_node`, and `weight`. Each name is separated by `"\t"`.

2. Edge data starts from the second row. You can add your edges in any order–they are not needed to be in an alphabetical order of their names. Each data item is separated by `"\t"`.

3. The meaning of each column:

   (a) `source_node`: the first node of the edge (specified by the name that exists in the node list file).

   (b) `target_node`: the second node of the edge (specified by the name that exists in the node list file).

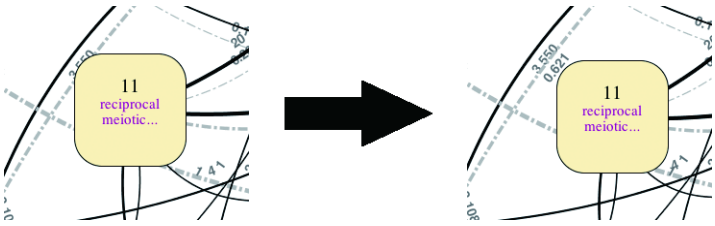
   (c) `weight`: the weight of the edge (can be any nonnegative number).

A.3 Basic Network Navigation

Once your network is loaded, you can start navigating the network in several ways, click, shift-click, double-click, and right-click. Apart from these, you can go back and forth between the created views by using the set of Graph View Navigation buttons in the left panel (Figure 3.3).

Click: You can click to select a node, a cluster, or an edge/meta-edge/property edge. You can then drag the node/the cluster freely to rearrange the network in the current graph view to, for example, reveal an edge weight which might have been hidden on the canvas (Figure A.1). Once you select a node or a cluster, only edges/meta-edges/property edges that directly connect to the node or the cluster are keep shown (Figure A.2).

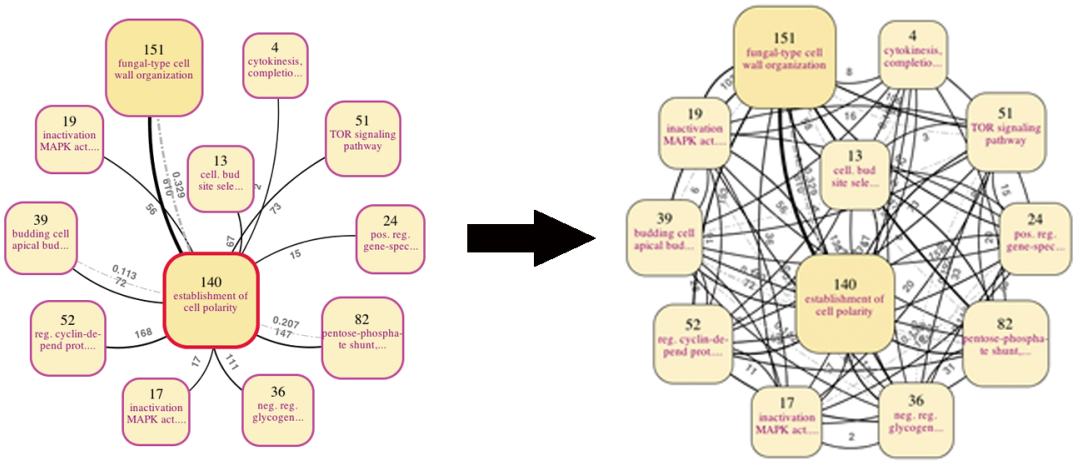
Figure A.1: Dragging a cluster to reveal hidden objects. Before dragging a cluster shown on the left, the weights of two edges were hidden. After dragging the cluster to the right, the weights of two edges are revealed.



Shift-Click: To select more than one object at a time, press `SHIFT` key and then click to select items (nodes, clusters, edges/meta-edges/property edges) one by one. Or you can drag your mouse at the canvas space to create a rectangle that covers objects you want to select.

Double-Click: Double-click on a cluster in NaviCluster means zooming in on the cluster to reveal its members. You can alternatively select the cluster and click the Zoom In button located in the left panel. To zoom in on more than one cluster, selecting

Figure A.2: When selecting a cluster, the tool shows only directly connected edges. After selecting the *establishment of cell polarity* cluster, only edges connecting to the members of the cluster are shown.

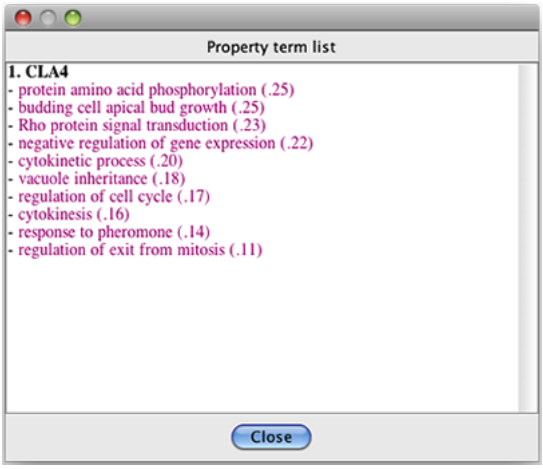


multiple clusters (by shift-click) and click the Zoom In button. Double-click on other items that are not currently selected has no effects.

Right-Click: Right-click shows a context-sensitive menu depending on what are currently selected (Figure A.3). For detail, see Section B.4.1.

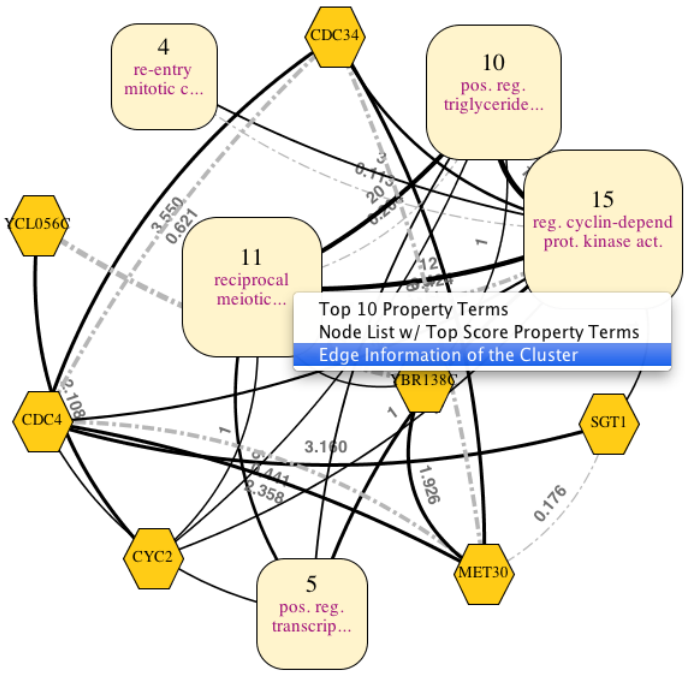
A.3.1 Context-Sensitive Menus

Nodes:



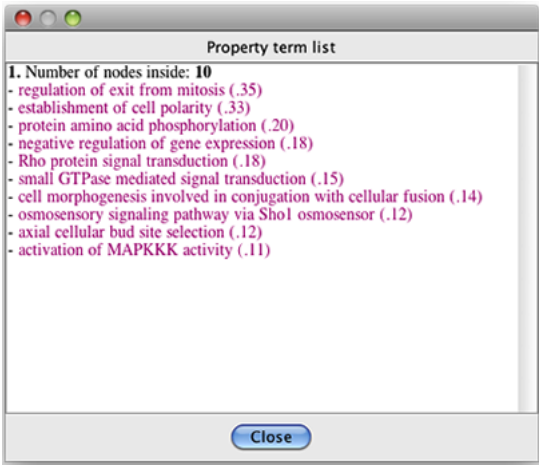
1. **Top 10 Propery Terms**: Show the ten highest-score property terms of node
2. **Show Information from Relevant Online Database**: Retrieve and show information from the online database using a node name as a query. The database name is obtained from the node list file. If NaviCluster does not know the database

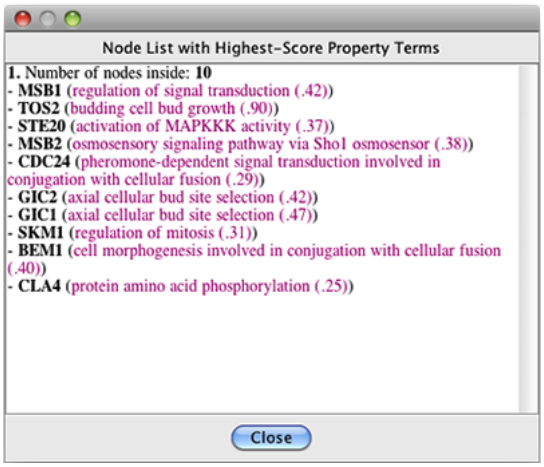
Figure A.3: Right-click shows a context-sensitive menu.

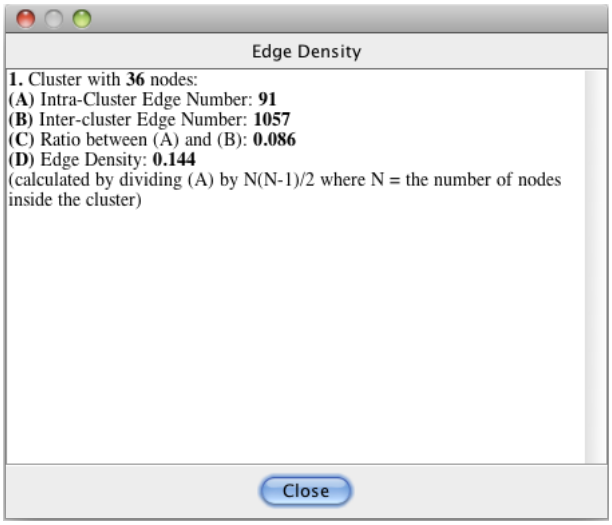


name specified in the node list file, it uses the NCBI Entrez Gene database (`www.ncbi.nlm.nih.gov/gene`) instead.

Clusters:

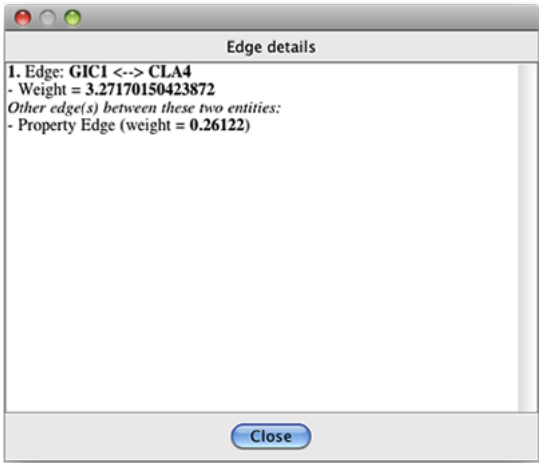


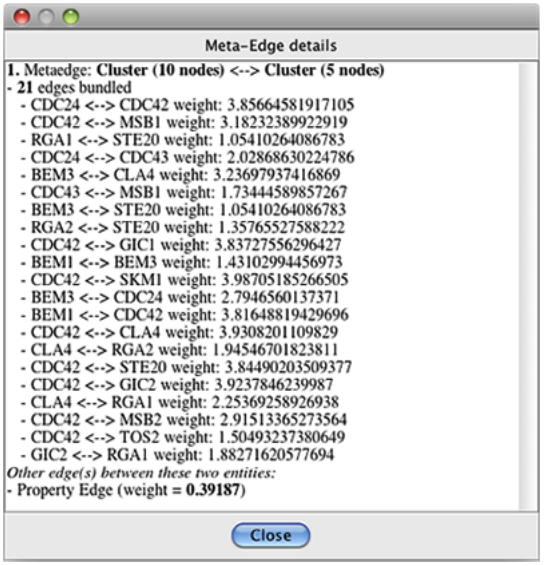




1. **Top 10 Propery Terms**: Show the ten highest-score property terms among all the terms annotated to the node members of cluster.
2. **Node List(s) w/ Top Score Property Terms**: Show the members of the cluster followed by their highest score terms.
3. **Edge Density**: Provide information about edge density of the cluster: **(A)** the number of edges within the cluster, **(B)** the number of edges connecting the nodes *within* the cluster and nodes *outside* the cluster, **(C)** the ratio between (A) and (B), **(D)** edge density calculated as the ratio between (A) and $N(N-1)/2$ where N = the number of nodes within the cluster.

Edges, Meta-Edges, Property Edges:

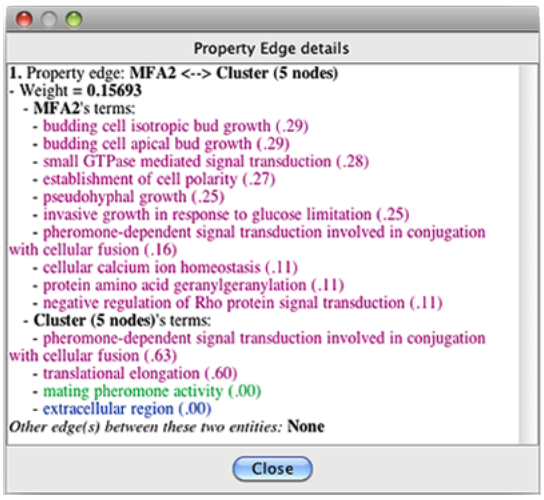




View Edge Details: Show the details of edge (a pair of end nodes and an edge weight) with brief information about other types of edges existing between the two ends.

View Meta-Edge Details: Show the details of meta-edge (a pair of end nodes and the number of edges bundled) with brief information about other types of edges existing between the two ends.

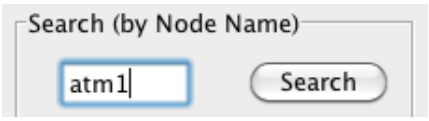
View Property Edge Details: Show the details of property edge (a pair of end nodes and the similarity value of property edge) with brief information about other types of edges existing between the two ends.



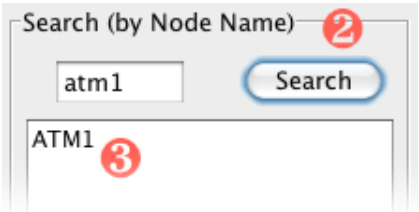
A.4 Search & Highlight

To facilitate the navigation to your node of interest, NaviCluster provides the search-and-highlight function. With this function, you can get to the node more easily and quickly.

1. Type the name of your node of interest (case-insensitive).

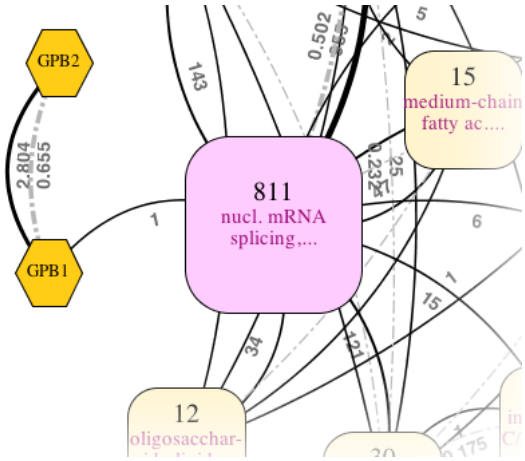


2. Click Search or press `ENTER` on the keyboard.



3. Nodes whose names or their parts match with your query are shown in the text area below.

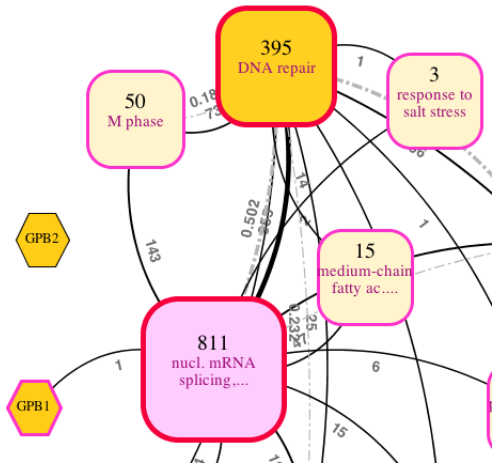
4. Double-click on the name of node you want to trace (or select the name, and press Highlight Selected Node). The cluster in the current view containing the node is then highlighted in pink.



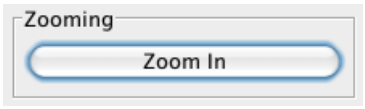
A.5 Zoom

Zooming allows you to reveal members of selected clusters and navigate deep down to a lower level of the hierarchy. In NaviCluster, You can select *more than one* cluster to zoom in on. All the members of selected clusters will be fed into the two-stage clustering (Chapter 2 and Section 3.2.1) before visualization.

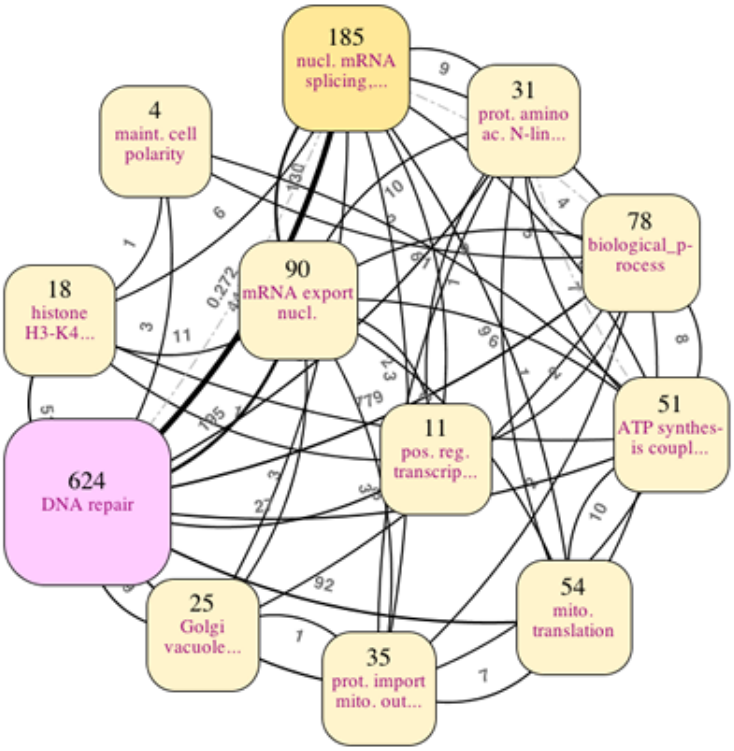
1. Select a cluster or many clusters you want to zoom in on.



2. Click Zoom In. In case of one cluster, you can double-click on it directly as well.



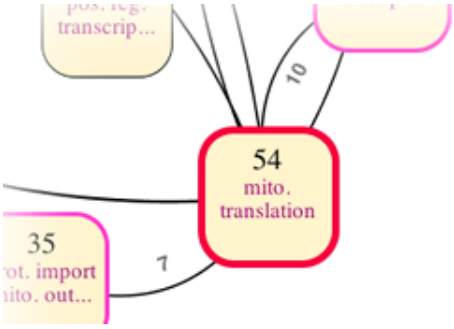
3. NaviCluster then clusters all nodes of the selected cluster(s) and visualizes the result, which may be composed of only nodes, only clusters, or a mix of clusters and nodes.



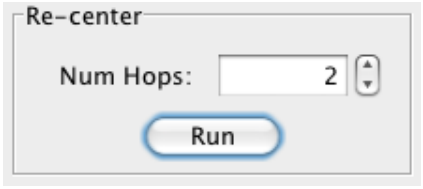
A.6 Re-Center

Re-centering runs the two-stage clustering (Chapter 2 and Section 3.2.1) on all nodes of the entire network whose geodesic distances to selected nodes/clusters are less than or equal to a value specified by users (number of hops).

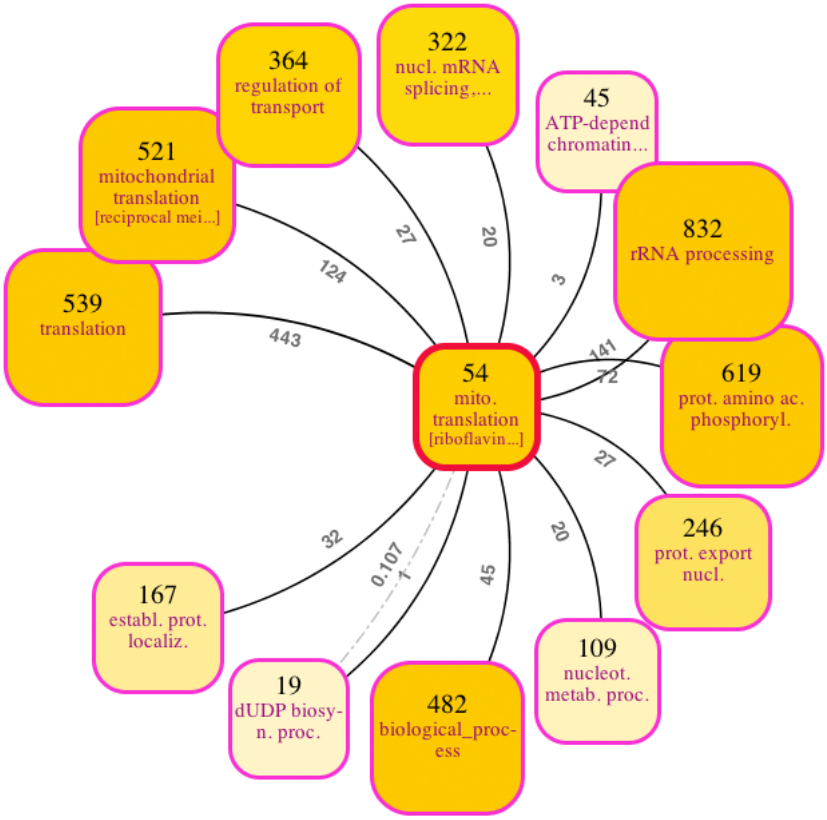
1. Select nodes/clusters you want to re-center the network on.



2. Specify the number of hops (geodesic distance) in the textbox (by typing or adjusting the spinner on the right of the box) and click Run to execute the re-centering function.



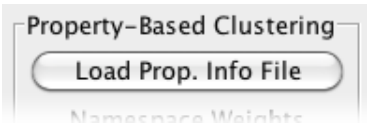
3. Re-centered nodes/clusters are displayed surrounded by their neighbors, organized as clusters.



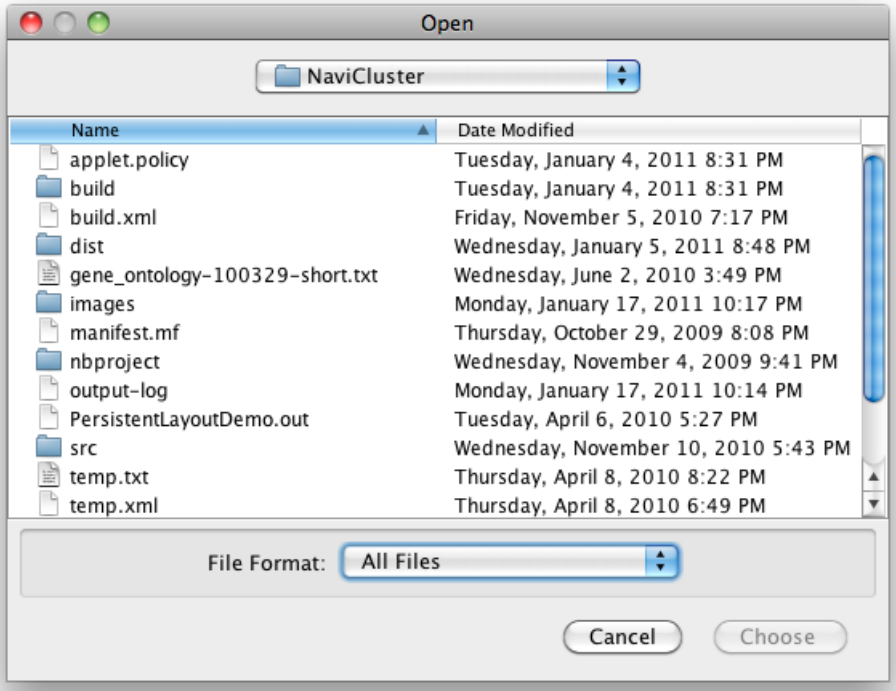
A.7 Property Information File Loading

The property-based clustering component of NaviCluster relies on the propery information file, which describes properties annotated to the nodes of network. When loading a new network, you can change the property information file to describe properties used in the new network files accordingly.

1. Click the Load Prop. Info File button.

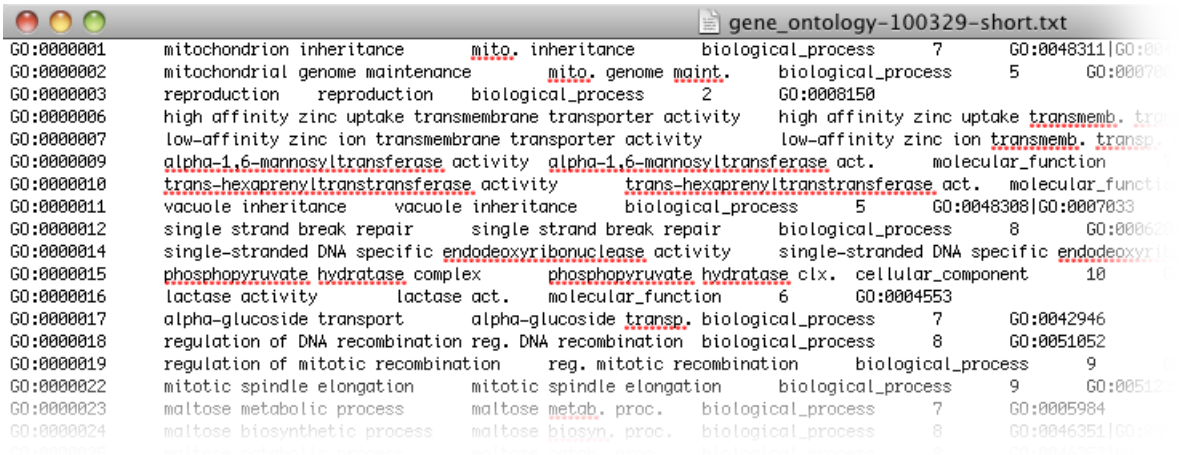


2. Select a new property information file. Any text file complying with the format, described in Section B.8.1, should be fine. Then, click OK to load the file.



A.7.1 Property Information File Format

As mentioned above, the property information file describes all property terms annotated to the nodes of the input network. To create such a file, you have to make your property information file comply with the following guidelines (Gene Ontology (GO) is used in the figure):

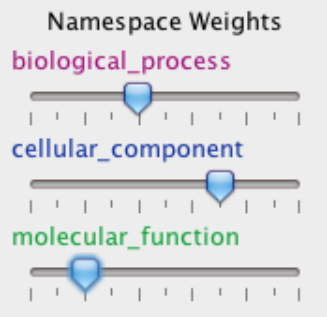


1. Data starts from the first row. There are six items per row needed. Each item is separated by "`\t`". Each row is referred to by its ID (the first column), so you can enter data rows in any order–there is no need to be in an alphabetical order.
2. The meaning of each word in the row: (The meaning in case GO is used are shown in brackets.)
   (a) `ID`: the identifier of property term [GO ID].
   (b) `name`: the name of property term [GO term's name]
   (c) `display_name`: the name that will be used in labelling clusters in abstracted views. [It is a short version of the term's name, used to save the space.]
   (d) `namespace`: the namespace (category) of term [GO namespace of the term: Biological Process, Molecular Function, or Cellular Component]
   (e) `weight`: the weight of term used in the property-based clustering. Terms with higher weight affect the meanings (properties) of nodes more than terms with lower weights. [the depth of the term in the GO hierarchies]
   (f) `parents`: the list of parent of the term (in case the property information is organized in a hierarchy). Parent terms are separated by vertical bars, "`|`". [All parent terms of the term based on the GO hierarchies]

A.8 Network Re-Clustering

As the property-based clustering of NaviCluster (Section 2.2.1) makes use of namespace weights, you can change the aspect of network navigation by adjusting the weights differently and re-cluster the network.

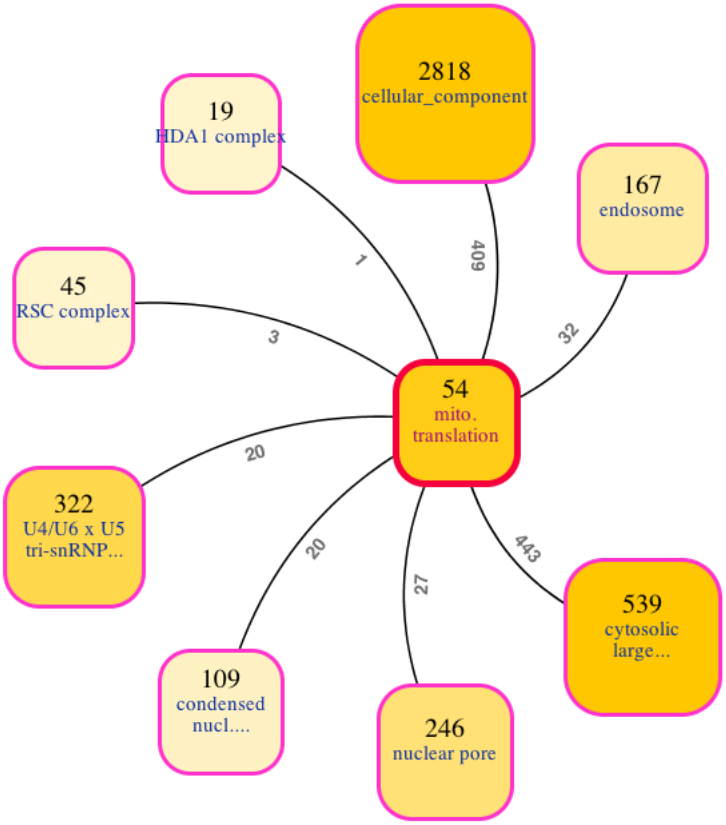
1. Adjust the sliders of namespace weights as you wish. The minimum value is on the left (0.0) and the maximum value is on the right (1.0). The higher weight, the more the terms belonging to the namespace contribute to the result of clustering.



   NOTE: All namespace weights must NOT be set to zero at the same time. As NaviCluster guarantees the manageable amount of information on the screen, in case the number of clusters produced by the Louvain algorithm is not small enough, property terms whose namespace weight is not zero have to be used. Therefore, putting all weights to zero has no meaning in NaviCluster and is not intended to produce a view generated by only the Louvain algorithm.

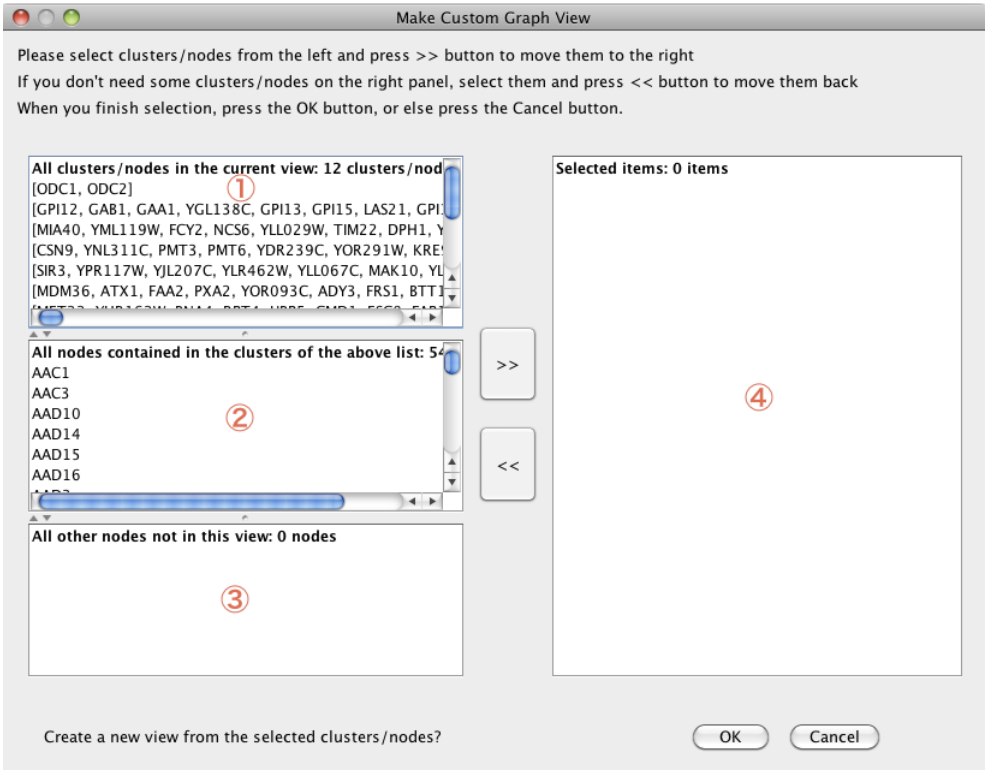
2. Specify the preferred number of clusters to be shown on the screen.

3. Click Re-Cluster. NaviCluster then re-clusters the network using the new values you have adjusted.



A.9 Make Custom Graph View

When clicking the Make Custom Graph View button located in the right panel (Figure 3.3), the dialog shown in the below figure appears. This function lets you select clusters/nodes to be shown on the screen more specifically by choosing them from the lists on the left and clicking the >> button to move them to the list on the right. If you want to cancel your selection, select the corresponding clusters/nodes on the right and click the << button to move them back.



Panel 1 shows the list of all clusters in the current view and the nodes appearing explicitly in the view.

Panel 2 shows the list of all the nodes contained in the clusters of this view. This assists you in selecting some nodes inside the clusters of interest.

Panel 3 shows other nodes in the input network that do not appear in the current view.

Panel 4 shows the list of selected clusters/nodes to be used to create a new view.

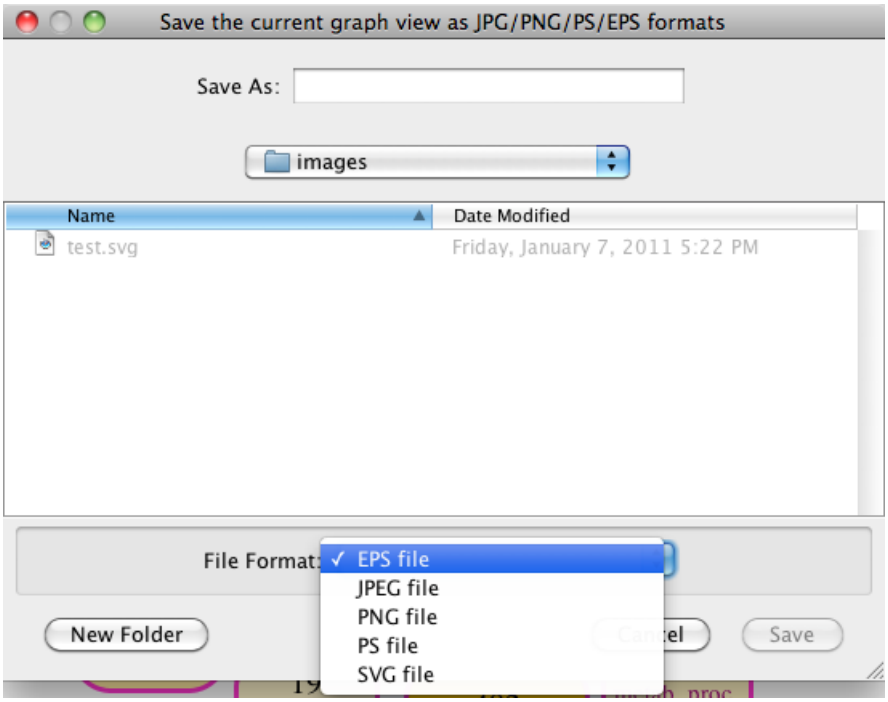
After finishing the selection, press the OK button to create a new view from your selected clusters/nodes. If you want to cancel this operation, press the Cancel button.

A.10 Export as Images

You can save a graph view you explore as an image via the Save As... button in the left panel. Supported file formats are EPS, JPG, PNG, PS, and SVG.

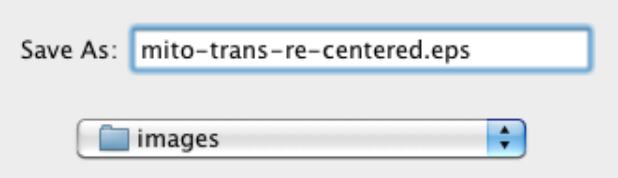
1. Click the Save As... button.

2. Select your preferred file type. The default selection is EPS, which is a vector graphics format.

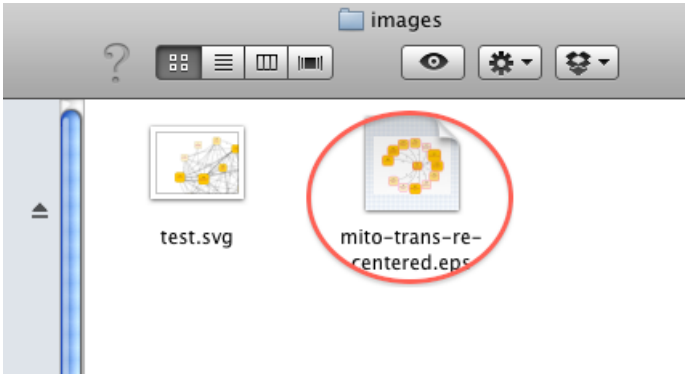


NOTE: Vector graphics can be expanded to any sizes while retaining the image quality, which is in contrast to raster graphics

3. Type a name for the file.



4. Click OK. The canvas is then exported as an image and ready for use.



A.11 Conversion from PSI-MITAB to Node & Edge List files

PSI-MITAB format is a standard format supported by many databases, e.g., DIP, MINT, IntAct, and BioGRID (`http://www.psidev.info/`). A script that converts a network in the PSI-MITAB format to node and edge list files for NaviCluster is provided to support users who want to import networks in PSI-MITAB.

Download: `PsiMiTabConverter.zip` from `http://navicluster.cb.k.u-tokyo.ac.jp/PsiMiTabConverter.zip`

Instruction:

1. Extract the file into your favorite place in your computer
2. Type the following command in the terminal/command prompt:

```
java -Xmx1024m -jar PsiMiTabConverter.jar <PsiMitabFilePath>
                <AnnotationFilePath> -needIEA
```

   (a) `PsiMiTabFilePath` is the path to the network in PSI-MITAB format you want to convert.

   (b) `AnnotationFilePath` is the path to an annotation file that complies with GO Annotation File Format Guide (`http://www.geneontology.org/GO.format.annotation.shtml`).

   (c) `-needIEA` (`t` or `f`): type `t` if you want to include annotations whose evidence is IEA (Inferred from Electronic Annotation); Or else type `f`.

Appendix B

NaviClusterCS User Manual

An online manual is also available and updated at `http://navicluster.cb.k.u-tokyo.ac.jp/cs/`.

B.1 Download & Run

Prerequisite: Cytoscape v2.8 or later must be installed in your computer. Cytoscape can be downloaded at `http://www.cytoscape.org/download.html`. User manual of Cytoscape can be found at `http://cytoscape.org/manual/Cytoscape2_8Manual.html`.

1. Download `NaviClusterCS.jar` via `http://navicluster.cb.k.u-tokyo.ac.jp/cs/files/NaviClusterCS.jar`.
2. Locate your Cytoscape folder. If you use Cytoscape v2.8.1 in Mac OS X, the path to the Cytoscape folder should be `/Applications/Cytoscape_v2.8.1/` by default.
3. In Windows, this should be `C:\Program Files\Cytoscape_v2.8.1` by default. If you use other versions of Cytoscape, the version number in the Cytoscape path varies accordingly.
4. Place `NaviClusterCS.jar`, which is a plugin file, into the `plugins` folder under the Cytoscape folder.
5. Launch Cytoscape as usual and wait until the initialization is finished.
6. The NaviClusterCS panel appears as the first tab in the control panel of Cytoscape. From now, you can load network files and start using NaviClusterCS.

Note: It is recommended to adjust the maximum memory that Cytoscape can use to 1024 MB or larger, depending on an available memory size of your computer:

1. Open the file `Cytoscape.vmoptions` in the Cytoscape folder, e.g., `/Applications/Cytoscape_v2.8.1/` in Mac using any text editors.
2. In the file `Cytoscape.vmoptions`, change the number after the word `"-Xmx"` in the second line to `"1024m"` or larger number.

Here, `1024m` means the maximum memory that Cytoscape can use is 1024 MB or 1 GB. If your computer has more memory, try a larger memory size, e.g., `2048m`. This should make NaviClusterCS run even more smoothly.

B.2 Sample Dataset

1. Download `Sample.zip` via `http://navicluster.cb.k.u-tokyo.ac.jp/cs/files/Sample.zip` and decompress it. The compressed file contains a property information file `gene\_ontology-150411.txt`, and a sample dataset composed of a node list file (`atted.node`), which contains information about nodes and an edge list file (`atted.edge`), which contains a list of edges and their weights.

   (a) The property information file is Gene Ontology (GO) derived from the OBO format available on its website (`http://www.geneontology.org`) at 15 April 2011.

   (b) The sample dataset is the *Arabidopsis* gene co-expression network ATTED-II with 22,447 nodes and 189,546 edges.

   (c) GO annotations for nodes in the node list file are obtained from the TAIR database, `http://www.arabidopsis.org/`.

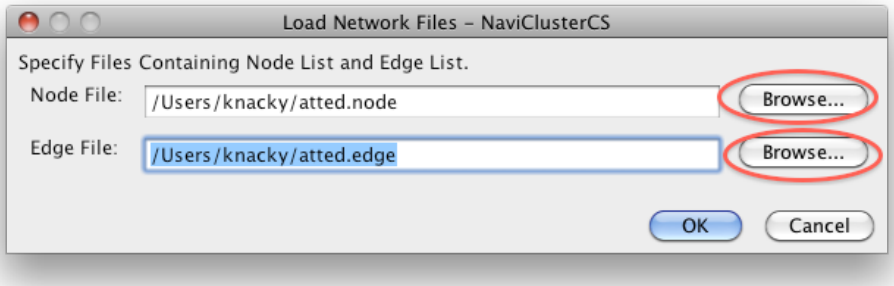
2. In Cytoscape, click the Load Network button in the NaviClusterCS panel.
3. Select `atted.node` and `atted.edge`, downloaded in step 1, as node and edge list files respectively and click OK. Cytoscape then loads the network dataset.
4. Click Start!. NaviClusterCS runs the two-stage clustering on the sample dataset and shows the result on the canvas of Cytoscape. At this stage, you can start navigating the network.

B.3 Network Loading

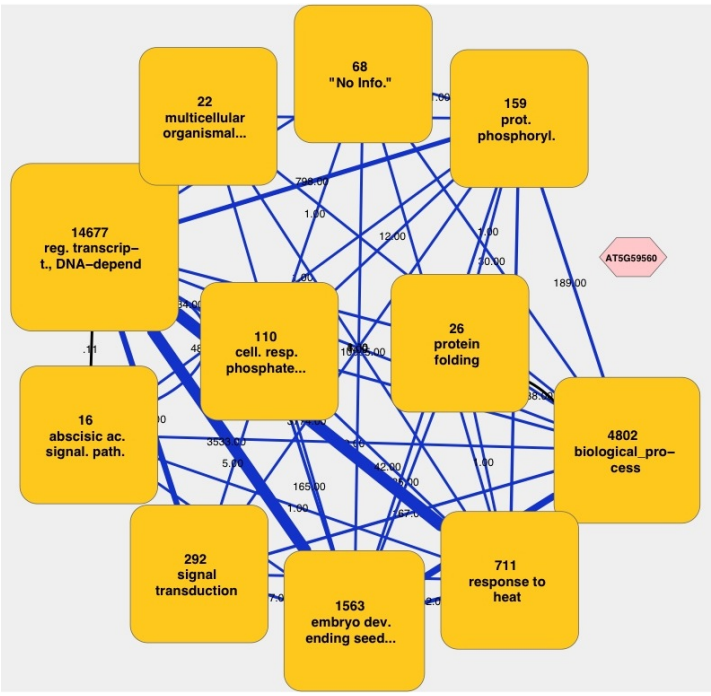
To load a network via NaviClusterCS interface, you need two files describing the network: a node list file and an edge list file. See Sections B.3.1, B.3.2 for details about the node list file format and edge list file format.

1. Click the Load Network button on the NaviClusterCS tab of the Control Panel.

2. Click Browse... to select your node list file. Then, click Browse... to select your edge list file. A node list file must have `.node` as its extension. An edge list file must have `.edge` as its extension.

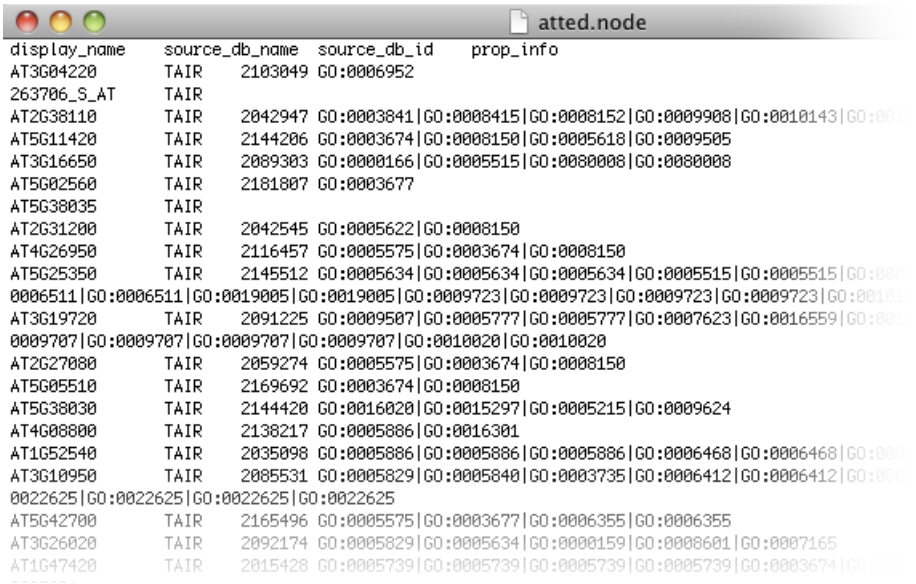


3. Click OK. NaviClusterCS then loads, clusters, and visualize your abstracted network.



B.3.1 Node List File Format

A node list file contains a list of nodes and their information. Together with an edge list file, they describe a network. To create a node list file, follow the below guidelines:

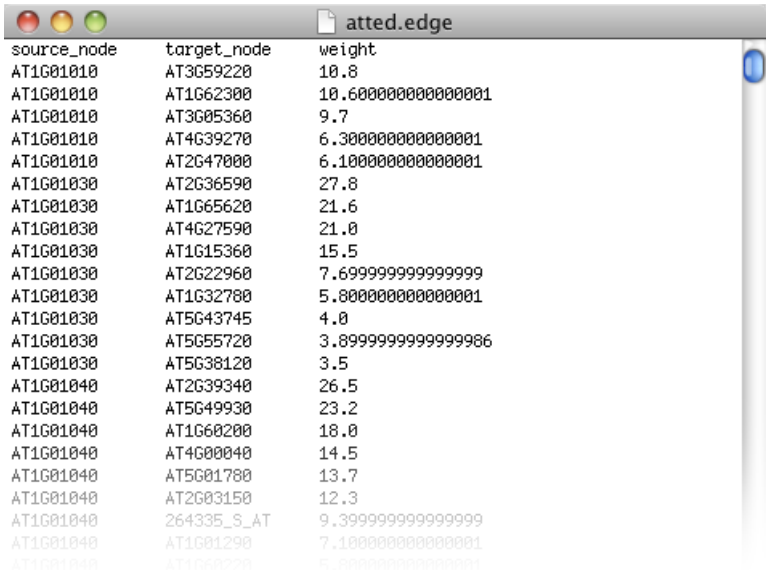


1. The first row of the file must be composed of four column headesr: `display_name`, `source_db_name`, `source_db_id`, and `prop_info`. Each name is separated by "`\t`".
2. Node data starts at the second row. You can add your nodes in any order–they are not needed to be in an alphabetical order of their names. Each data item is separated by tab "`\t`".
3. The meaning of each column:
   (a) `display_name`: the name of node. This is used to label the node on the canvas.
   (b) `source_db_name`: the name of source database where the node relies on. NaviCluster provides queries on the NCBI Entrez Gene database (`www.ncbi.nlm.nih.gov/gene`) using the name of node as input by default. Default databases and their URLs, which are bundled with the sample file, are SGD, TAIR, UniProt, FlyBase, HGNC, RGD, and Entrez Gene. You can specify your new source databases and their URLs for queries by selecting "NaviClusterCS" → "General Settings" from the menu bar.
   (c) `source_db_id`: The identifier of node in the source database.

(d) `prop_info`: a list of property terms' identifiers annotated to the node. They are separated by a vertical bar "`|`". Each property term must be described in the property information file.

B.3.2 Edge List File Format

An edge list file contains a list of edges and their weights. Together with a node list file, they describe a network. To create an edge list file, please follow the below guidelines:


atted.edge

source_node	target_node	weight
AT1G01010	AT3G59220	10.8
AT1G01010	AT1G62300	10.600000000000001
AT1G01010	AT3G05360	9.7
AT1G01010	AT4G39270	6.300000000000001
AT1G01010	AT2G47000	6.100000000000001
AT1G01030	AT2G36590	27.8
AT1G01030	AT1G65620	21.6
AT1G01030	AT4G27590	21.0
AT1G01030	AT1G15360	15.5
AT1G01030	AT2G22960	7.6999999999999999
AT1G01030	AT1G32780	5.800000000000001
AT1G01030	AT5G43745	4.0
AT1G01030	AT5G55720	3.8999999999999986
AT1G01030	AT5G38120	3.5
AT1G01040	AT2G39340	26.5
AT1G01040	AT5G49930	23.2
AT1G01040	AT1G60200	18.0
AT1G01040	AT4G00040	14.5
AT1G01040	AT5G01780	13.7
AT1G01040	AT2G03150	12.3
AT1G01040	264335_S_AT	9.399999999999999
AT1G01040	AT1G01290	7.100000000000001

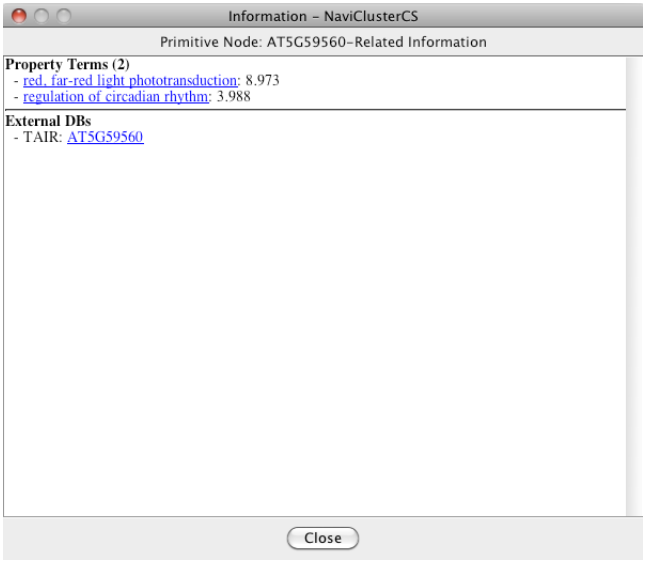
1. The first row of the file must be composed of three column headers: `source_node`, `target_node`, and `weight`. Each name is separated by "`\t`".
2. Edge data starts at the second row. You can put your edges in any order–they are not needed to be in an alphabetical order of their names. Each data item is separated by "`\t`".
3. The meaning of each column:
   (a) `source_node`: the first node of the edge (specified by the name that exists in the node list file).
   (b) `target_node`: the second node of the edge (specified by the name that exists in the node list file).
   (c) `weight`: the weight of the edge (can be any nonnegative number).

B.4 Basic Network Navigation

For the information about how to manipulate nodes and edges on the canvas of Cytoscape, see the Cytoscape User Manual (`http://cytoscape.org/manual/Cytoscape2_8Manual.html#Basic%20Network%20Navigation`). You can select clusters generated by NaviClusterCS as usual Cytoscape nodes. Note that when you select a node/cluster, only edges/meta-edges/property edges connecting to that node/cluster are shown.

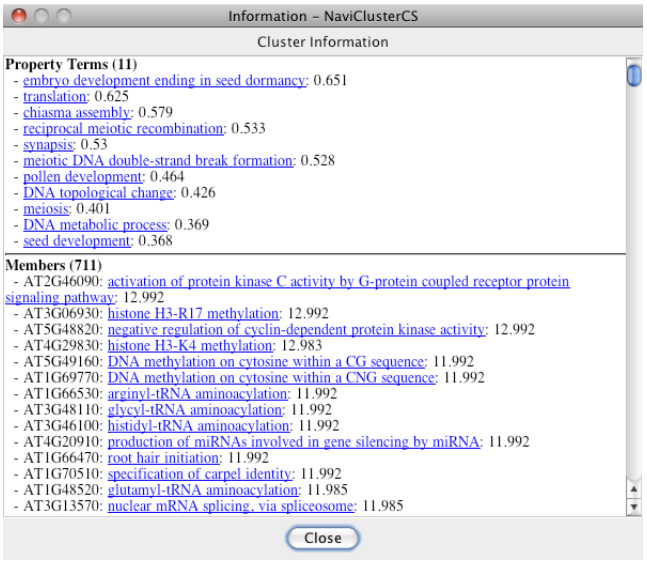
B.4.1 Context-Sensitive Menus

Nodes:



1. **Propery Terms**: Show the property terms of node
2. **External DBs**: Show link to the information stored in online databases with the node's name as a query.

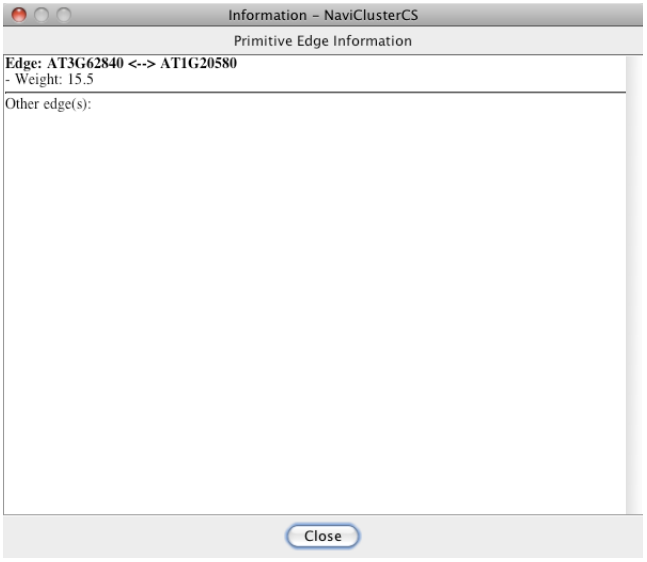
Clusters:



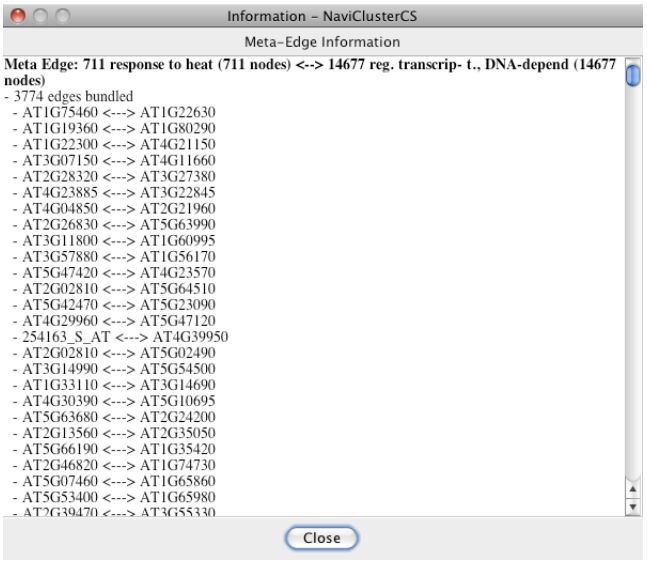
1. **Propery Terms**: Show the property terms among all terms of all the node members of the cluster
2. **Members**: Show the members of cluster followed by their highest score terms.

Edges, Meta-Edges, Property Edges:

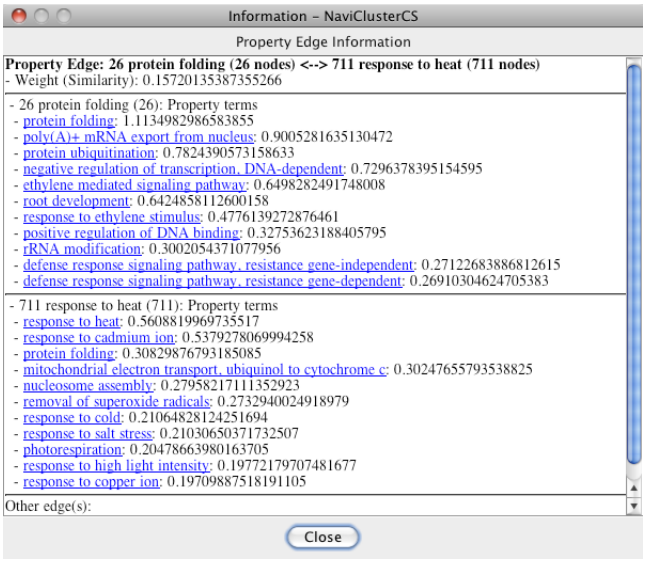
Edge Information: Show the details of edge (a pair of end nodes and an edge weight) with brief information about other types of edges existing between the two ends.



Meta-Edge Information: Show the details of meta-edge (a pair of end nodes and the number of edges bundled) with brief information about other types of edges existing between the two ends.



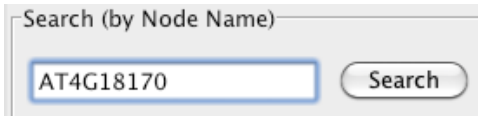
Property Edge Information: Show the details of property edge (a pair of end nodes and the similarity value of property edge) with brief information about other types of edges existing between the two ends.



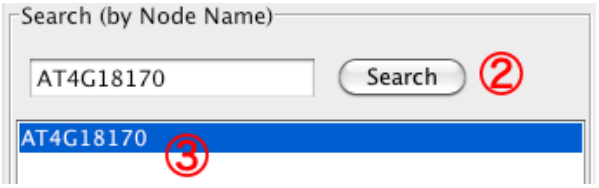
B.5 Search & Highlight

To facilitate the navigation to your node of interest, NaviClusterCS provides the search-and-highlight function. With this function, you can get to the node more easily and quickly.

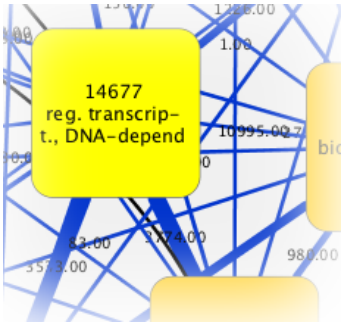
1. Type the name of your node of interest (case-insensitive).



2. Click the Search button.



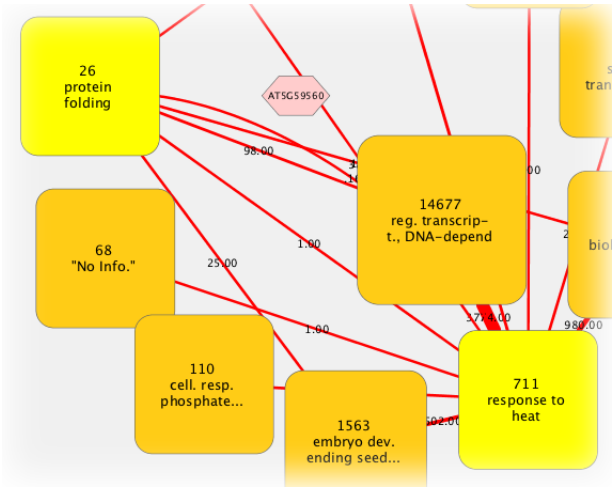
3. Nodes whose names or their parts match with your query are shown in the text area below.
4. Double-click on the name of node you want to trace (or select the name, and press Highlight Selected Node). The cluster in the current view containing the node is then highlighted.



B.6 Zoom

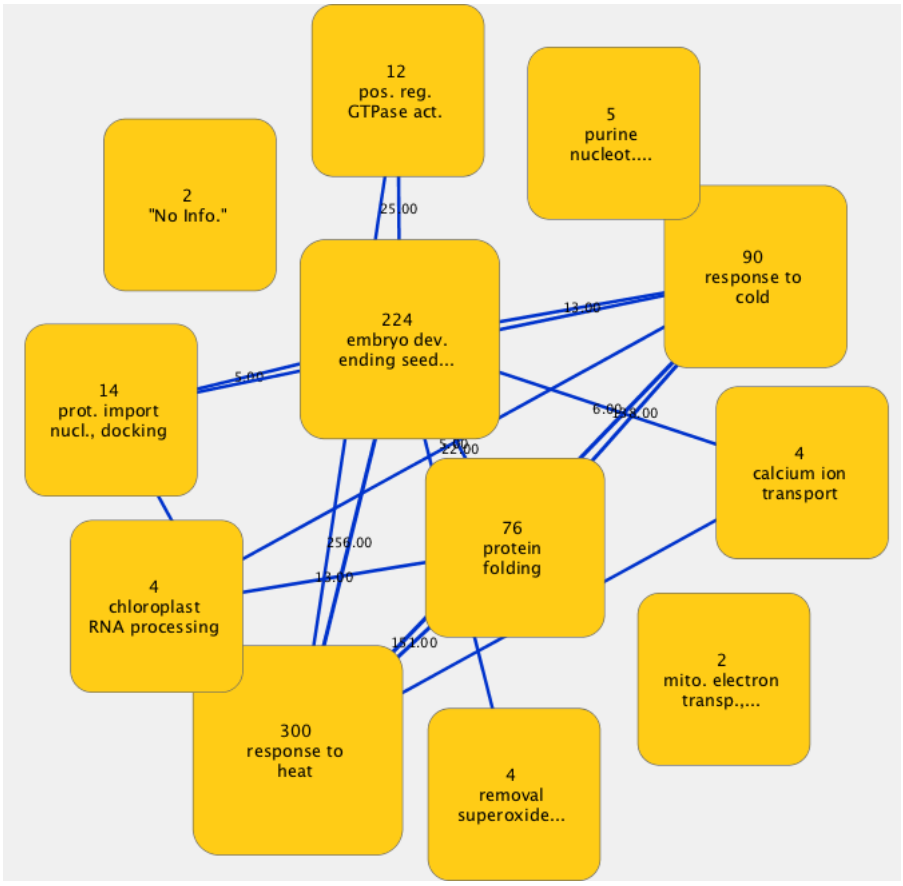
Zooming allows you to reveal members of selected clusters and navigate deep down to a lower level of the hierarchy. In NaviClusterCS, You can select *more than one* cluster to zoom in on. All the members of selected clusters will be fed into the two-stage clustering (Chapter 2 and Section 3.2.1) before visualization.

1. Select a cluster or many clusters you want to zoom in on.



2. Click Zoom In. In case of one cluster, you can double-click on it directly as well.

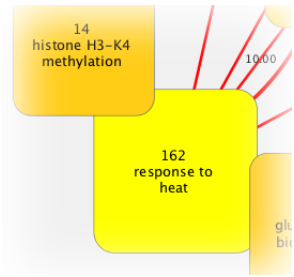
3. NaviClusterCS then clusters all nodes of the selected cluster(s) and visualizes the result, which may be composed of only nodes, only clusters, or a mix of clusters and nodes.



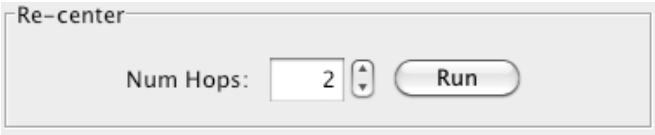
B.7 Re-Center

Re-centering runs the two-stage clustering (Chapter 2 and Section 3.2.1) on all nodes of the entire network whose geodesic distances to selected nodes/clusters are less than or equal to a value specified by users (number of hops).

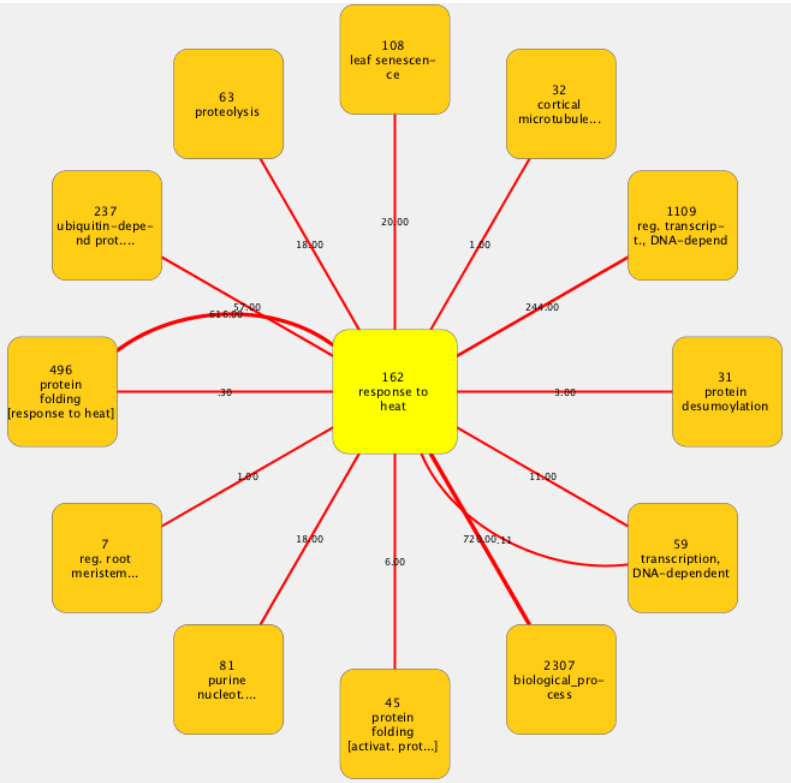
1. Select nodes/clusters you want to re-center the network on.



2. Specify the number of hops (geodesic distance) in the textbox (by typing or adjusting the spinner on the right of the box) and click Run to execute the re-centering function.



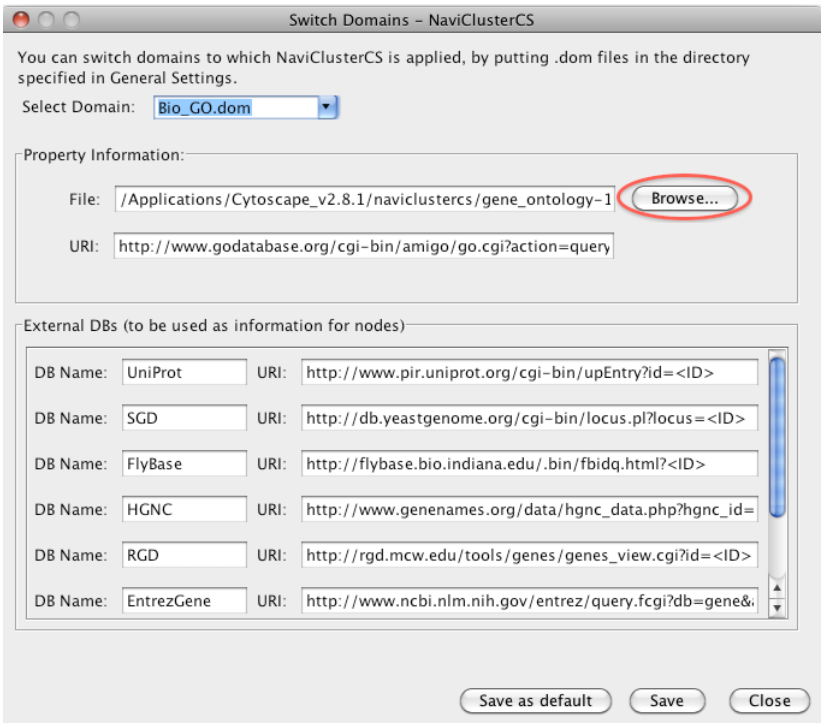
3. Re-centered nodes/clusters are displayed surrounded by their neighbors, organized as clusters.



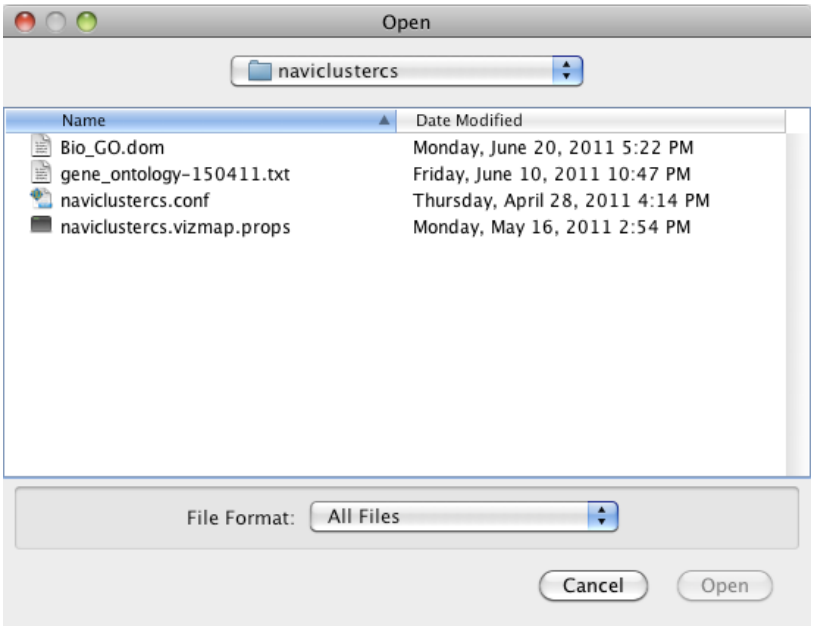
B.8 Property Information File Loading

The property-based clustering component of NaviClusterCS relies on the propery information file, which describes properties annotated to the nodes of network. When loading a new network, you can change the property information file to describe properties used in the new network files accordingly.

1. Select menu NaviClusterCS → Switch Domains... on the menu bar.
2. In the Switch Domains dialog, click the Browse.. button in the Property Information panel.



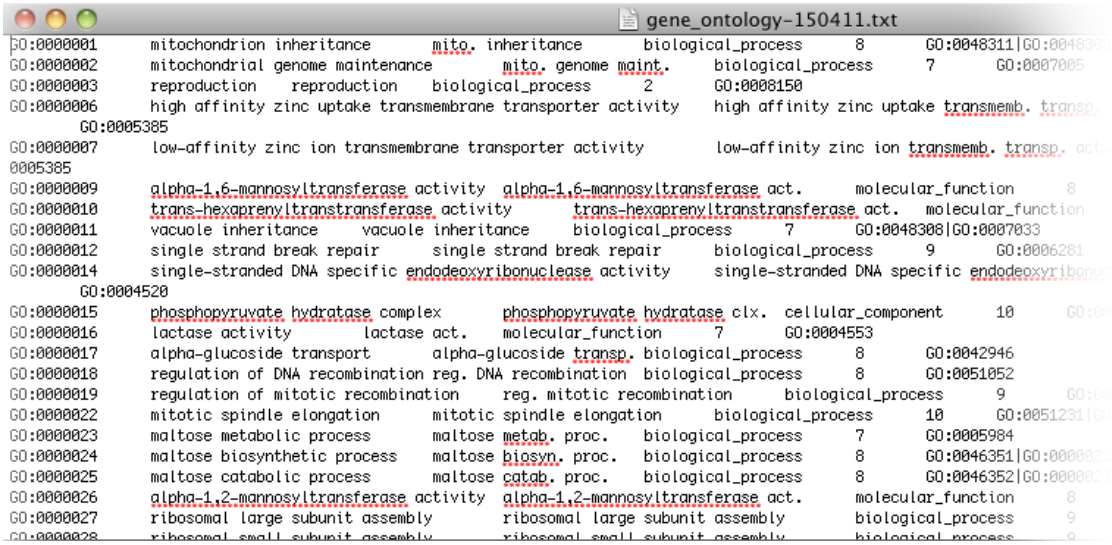
3. Select a new property information file. Any text file complying with the format, described in Section B.8.1, should be fine. Then, click OK to load the file.



B.8.1 Property Information File Format

As mentioned above, the property information file describe all property terms annotated to the nodes of the input network. To create such a file, you have to make your property

information file comply with the following guidelines (Gene Ontology (GO) is used in the figure):

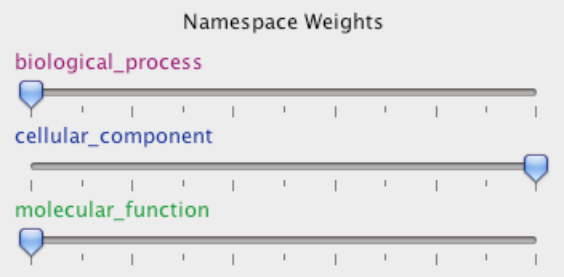


1. Each row represents the information of each property term. There are six items per row. Each item is separated by tab "`\t`". Each row is referred to by its ID (the first column), so you can enter data rows in any order–there is no need to be in an alphabetical order.
2. The meaning of each word in the row: (The meaning in case GO is used are shown in brackets.)
   (a) `ID`: the identifier of property term [GO ID].
   (b) `name`: the name of property term [GO term's name]
   (c) `display_name`: the name that will be used in labelling clusters in abstracted views. [It is a short version of the term's name, used to save the space.]
   (d) `namespace`: the namespace (category) of term [GO namespace of the term: Biological Process, Molecular Function, or Cellular Component]
   (e) `weight`: the weight of term used in the property-based clustering. Terms with higher weight affect the meanings (properties) of nodes more than terms with lower weights. [the depth of the term in the GO hierarchies]
   (f) `parents`: the list of parent of the term (in case the property information is organized in a hierarchy). Parent terms are separated by vertical bars, "`|`". [All parent terms of the term based on the GO hierarchies]

B.9 Network Re-Clustering

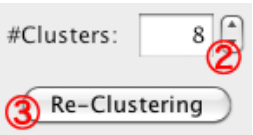
As the property-based clustering of NaviCluster (Section 2.2.1) makes use of namespace weights, you can change the aspect of network navigation by adjusting the weights differently and re-cluster the network.

1. Adjust the sliders of namespace weights as you wish. The minimum value is on the left (0.0) and the maximum value is on the right (1.0). The higher weight, the more the terms belonging to the namespace contribute to the result of clustering.

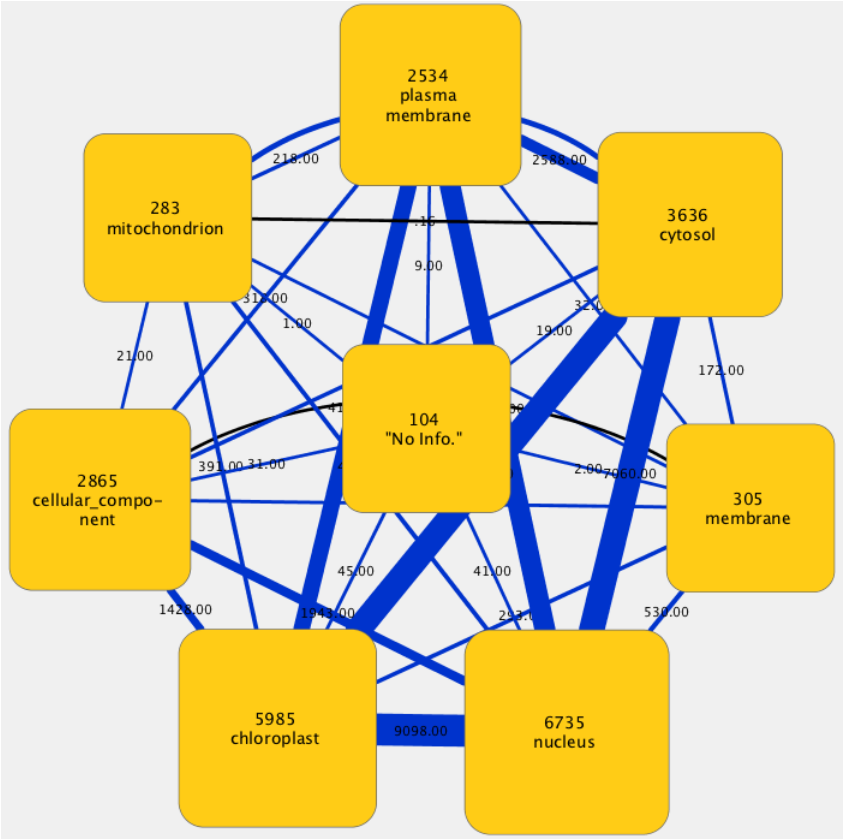


   NOTE: All namespace weights must NOT be set to zero at the same time. As NaviClusterCS guarantees the manageable amount of information on the screen, in case the number of clusters produced by the Louvain algorithm is not small enough, property terms whose namespace weight is not zero have to be used. Therefore, putting all weights to zero has no meaning in NaviClusterCS and is not intended to produce a view generated by only the Louvain algorithm.

2. Specify the preferred number of clusters to be shown on the screen.

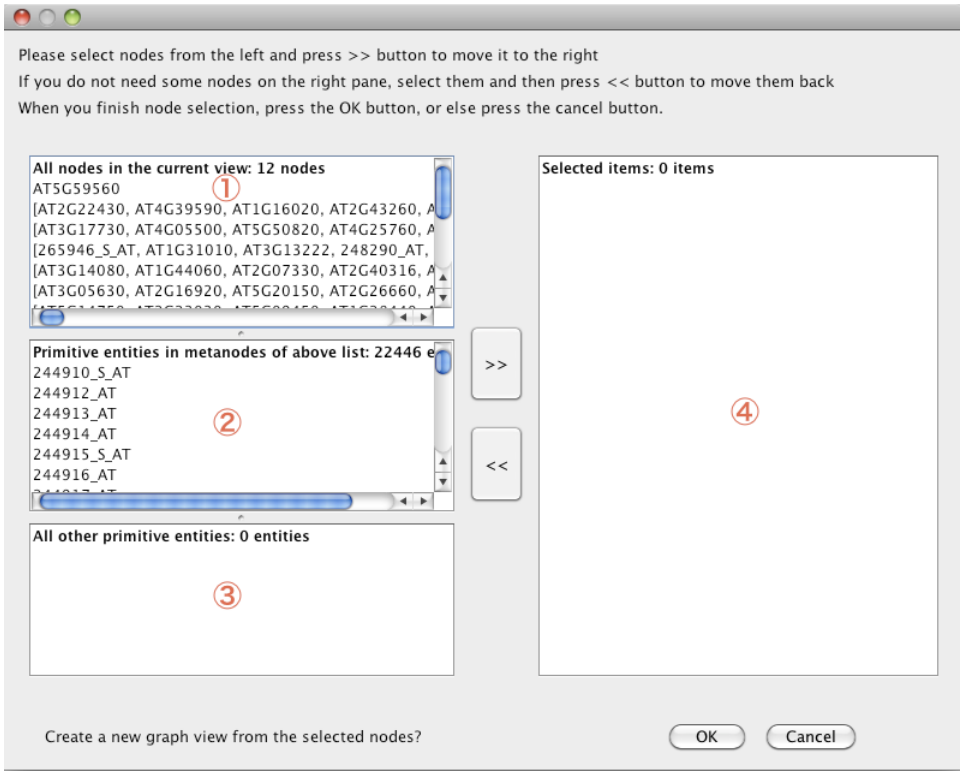


3. Click Re-Cluster. NaviClusterCS then re-clusters the network using the new values you have adjusted.



B.10 Make Custom Graph View

When clicking the Make Custom Graph View button located in the right panel (Figure 3.3), the dialog as shown in the below figure appears. This function lets you select clusters/nodes to be shown on the screen more specifically by choosing them from the lists on the left and clicking the >> button to move them to the list on the right. If you want to cancel your selection, select the corresponding clusters/nodes on the right and click the << button to move them back.



Panel 1 shows the list of all clusters in the current view and the nodes appearing explicitly in the view.

Panel 2 shows the list of all the nodes contained in the clusters of this view. This assists you in selecting some nodes inside the clusters of interest.

Panel 3 shows other nodes in the input network that do not appear in the current view.

Panel 4 shows the list of selected clusters/nodes to be used to create a new view.

After finishing the selection, press the OK button to create a new view from your selected clusters/nodes. If you want to cancel this operation, press the Cancel button.