

# A Hybrid Space-Filling and Force-Directed Layout Method for Visualizing Multiple-Category Graphs

Takayuki Itoh\*  
Ochanomizu University

Chris Muelder†  
University of California, Davis

Kwan-Liu Ma‡  
University of California, Davis

Jun Sese§  
Ochanomizu University

## ABSTRACT

Many graphs used in real-world applications consist of nodes belonging to more than one category. We call such graph "multiple-category graphs". Social networks are typical examples of multiple-category graphs: nodes are persons, links are friendships, and categories are communities that the persons belong to. It is often helpful to visualize both connectivity and categories of the graphs simultaneously. In this paper, we present a new visualization technique for multiple-category graphs. The technique firstly constructs hierarchical clusters of the nodes based on both connectivity and categories. It then places the nodes by a new hybrid space-filling and force-directed layout algorithm to clearly display both connectivity and category information. We show layout results using our hybrid method and compare it with other methods, and present a case study using an active biological network dataset.

**Keywords:** Information visualization, graph visualization, multiple-category graphs, clustering, force directed layout, space-filling hierarchical data visualization, active biological network.

## 1 INTRODUCTION

In many real-world applications, graphs are used to represent relations between entities and one common task is to find hidden structures of the graph. When entities have too many associations, the task becomes more difficult. For example, in a social context people can belong to different communities, and they are connected to others through their friends. In this case, the resulting social network consists of a set of nodes corresponding to people, with multiple categories corresponding to communities, and a set of links corresponding to the friendship. Many researchers in social networks are interested in the analysis of relationships between communities and friendships. In this paper, we call graphs consisting of nodes belonging to more than one category "*multiple-category graphs*".

Graphs or networks are widely used in various information visualization applications. For example, social network visualization represents people as nodes, and gene visualization represents the genes as nodes. Since nodes of general graphs do not have positional information, graph layout techniques are very important to obtain readable visualization results. Force-directed layout [12] is one of the most popular approaches for calculating the positions of nodes. It is used in many applications because it can take edge weights into consideration and generates appealing results. However, it is computationally expensive: computation time of traditional force-directed layouts may be proportional to the square of the number of nodes in the worst cases. Treemap based methods [11] can more quickly layout the large graphs. The method firstly hierarchically clusters of nodes based on their connectivity,

and then places the nodes by Treemap [10]. In addition to its high performance, it is useful for various graph analytics because the method clearly represents clusters of the graphs.

If all nodes of a graph belong to only one category, a tree can be constructed from the graph by simply grouping the nodes. In this special case we can apply existing visualization techniques, which firstly place the nodes base on the tree structure, and then draw the connectivity between the nodes [7]. However, people usually join multiple communities in their social life; therefore we cannot generally assume that the nodes can construct a category-based tree.

We consider the following requirement for the visualization of multiple-category graphs:

**Requirement 1:** Place nodes which belong to the common categories close each other, because users may want to observe such nodes simultaneously.

We also consider the following general graph drawing requirements:

**Requirement 2:** Reduce the sum of lengths of edges, or number of intersections among edges.

**Requirement 3:** Avoid the cluttering of nodes, because it may prevent the clickability and readability.

**Requirement 4:** Maximize screen space utilization, because we often want to look the whole data in a limited screen space.

**Requirement 5:** Reduce the computation time.

This paper presents a new technique for effectively visualizing multiple-category graphs. The technique firstly applies hierarchical clustering to a given graph, and generates clusters of nodes based on both their categories and connectivity. It then visualizes the hierarchy by applying space-filling layout, as existing Treemap based work does [11]. During the process, our new technique applies hybrid space-filling and force-directed layout so that it can satisfy the above requirements. Force-directed layout is good at requirements 1 and 2, because it flexibly stabilize the distances of nodes. On the other hand, space-filling layout is good at requirements 3, 4, and 5. Our hybrid layout demonstrates the features of the two layout techniques so that it can satisfy the all above requirements.

This paper also presents an application of the technique for the visualization of active biological networks [14]. The networks consist of large number of genes connected based on known protein-protein interactions, and the genes are categorized based on expression or repression conditions which can be used to extract densely connected subnetworks. Section 5 of this paper shows that the presented technique can reveal interesting and important features of the multiple-category active biological networks.

## 2 RELATED WORK

### 2.1 Force-Directed Graph Layout

The force-directed graph layout is one of the most popular graph layout techniques. It assumes that edges of a graph have spring-like forces attempting to keep stable lengths. In addition, the layout can

\*e-mail: itot@is.ocha.ac.jp

†e-mail: cwmuelder@ucdavis.edu

‡e-mail: ma@cs.ucdavis.edu

§e-mail: sesejun@is.ocha.ac.jp

flexibly stabilize the distances of nodes by adjusting the weights of edges. Many implementations attempt to minimize the sum of energy of the edges by iterative calculation [12].

Force-directed layouts are preferred in many applications because they provide appropriate distances between pairs of connected nodes. That layout's drawback, however, is computation time. It requires iterative calculations, and the worst case running time is proportional to the square of number of nodes. To solve the problem, many accelerated force-directed techniques have recently been presented [6].

## 2.2 Graph Clustering

Graph clustering has many uses in the field of graph/network analysis. Many graph clustering techniques are based on agglomerative clustering algorithms, which firstly treat each node as a separate cluster and then progressively merge clusters. The fast modularity community structure inference algorithm [5], which we apply in the current implementation, is a recent technique based on an agglomerative clustering algorithm. Alternatively, divisive clustering algorithms firstly treat the entire graph as a cluster and then recursively split the cluster into smaller clusters. The recently developed k-SNAP [17] technique is an example of recursive split algorithms.

## 2.3 Space-Filling Hierarchy Layout

The space-filling hierarchy layout is one of the most common approaches for visualizing tree or hierarchical data. It features an all-in-one visualization of lower-level data items in hierarchical data. Treemap [10] is the most popular space-filling layout technique. It recursively subdivides the display space into rectangular regions, resembling nested bar charts. The Quantum Treemap [1] is one of the most valuable variations of the Treemap because it represents all leaf-nodes as equally sized and shaped icons.

We have also presented a space-filling hierarchy layout technique [8] [9], which is somewhat analogous to Treemap because both techniques subdivide screen space into rectangular regions. Unlike the various Treemap algorithms that distribute hierarchical data by a top-down algorithm, our technique deals with data by a bottom-up algorithm. Experiments described in [8] discuss the trade-offs between Quantum Treemap and our Treemap-like technique. Our Treemap-like technique has a better aspect ratio of sub-regions and better layout stability with similar hierarchical data, while Quantum Treemap has better computation time and better screen space utilization. In addition, our Treemap-like technique has a unique feature called "templates", which describe ideal positions of data items on a screen space (as described in Section 5 of the reference [8].) The technique presented in this paper applies the templates for the hybrid force-directed and space-filling layout.

## 2.4 Hierarchy Based Graph Visualization

Hierarchically clustered graphs are an effective data structure for information visualization because they are suitable for overview, zoom, and filtering operations. Various techniques for hierarchical graph visualization have been presented over the last ten years [13], including two types of hierarchically clustered graphs. In many cases hierarchy is defined independent of edges. For example, we can treat Web pages of a Web site as a hierarchical graph if we suppose nodes are Web papers and edges are hyperlinks between pairs of Web pages. In this case, we may want to hierarchically cluster the nodes based on domains or directories, independent of their hyperlinks. On the other hand, we can also generate hierarchical graphs from the results of connectivity-based graph clustering.

Bourqui et al. [3] presented a multi-resolution force-directed layout technique. It applies the force-directed layout to the each level of the data hierarchy and divides the screen space with a Voronoi-based algorithm. It relates to the technique presented in this paper

because both techniques use a force-directed layout technique and avoid overlapping clusters by dividing up screen space.

Several Treemap-based hierarchical graph visualization techniques have recently been presented. This approach is appropriate for large graphs because of its properties, such as small computation time, reduced clutter, and efficient screen space usage. For example, Zhao et al. [20] presented a technique which interchangeably allows graph diagrams and Treemap to represent parts of a tree.

We presented a technique to quickly visualize clustered graphs [11]. It begins by hierarchically clustering a graph based on its connectivity, and then, it spreads out the nodes using a Treemap layout. The technique was successfully applied to the so called "California" graph, which contains 9663 nodes and 16151 edges. In addition to displaying meaningful clusters, it was able to achieve its result at 13 frames per second.

Several other techniques focus on drawing edges in a given tree structure. Holten [7] presented a technique to draw edges of the tree data as bundles. This technique is also useful for understanding the brief structure of edges.

Note that there are no techniques introduced in this section that take multiple categories into consideration.

## 2.5 Multiple Categorization

As discussed in Section 1, visualization of multiple-category graphs is useful in various fields. Generally, relationships among categories and nodes can be represented as DAG. In the case of social networks, communities are parent nodes, participants are children nodes, and the children may have multiple parents. DAG visualization is still a complicated problem even though various related methods have been presented [15]. The Treemap-based DAG visualization method [18] is relevant to our technique, but the method may have to deal with NP-hard problems. Moreover, it does not take connectivity of nodes into consideration. We do not expect connectivity among nodes to be well-represented by simply drawing edges on such DAG representations.

## 2.6 Visualization of Biological Networks

This paper applies the presented technique toward the visualization of biological networks. Bioinformatics is an important area which various information visualization techniques have been recently applied [16]. There have been many visualization techniques for gene or protein interaction networks, and some of them are relevant to our techniques since they represent experimental values (e.g. expression values obtained by microarray experiments) by coloring nodes. Osprey [4] and PSIMAP [2] are typical examples that have visualized gene/protein interaction networks and their experimental values. However, their examples do not provide an optimal layout of large-scale networks.

## 3 A HYBRID LAYOUT

This section presents a new technique for visualizing multiple-category graphs. As discussed in Section 1, the technique is a combination of force-directed graph layout and space-filling hierarchical data visualization techniques. First of all, the technique generates hierarchical clusters of nodes based on their categories and connectivity. It then hierarchically calculates their positions by applying force-directed and space-filling techniques.

This section describes the supposed data structure, clustering and graph layout techniques, and functions for drawing and interaction.

### 3.1 Data Structure

In this paper we consider input graphs described as  $G = \{N, L\}$ .  $N = \{n_1, \dots, n_{n_N}\}$  is a set of nodes, and  $n_N$  is the total number of nodes. Each of nodes  $n_i$  has  $m$ -dimensional Boolean values,  $n_i = \{b_1, \dots, b_m\}$ , where  $m$  is the number of categories, and  $b_j$  is *true* if  $n_i$  belongs to the  $j$ -th category.  $L = \{l_1, \dots, l_{n_L}\}$  is a set of links

that connect pairs of nodes, where  $n_L$  is the total number of links. Nodes and links are not weighted.

Our method generates hierarchical graphs described as  $G' = \{C, E\}$ .  $C = \{c_1, \dots, c_{n_C}\}$  is a set of hierarchical clusters, consisting of a set of children clusters, and  $n_C$  is the total number of clusters. Each of clusters  $c_i$  has  $m$ -dimensional Boolean values,  $c_i = \{b_{i1}, \dots, b_{im}\}$ , as well as those of nodes. The Boolean values must be completely equal between a cluster and its children nodes or clusters.  $E = \{e_1, \dots, e_{n_E}\}$  is a set of edges that connect pairs of clusters, where  $n_E$  is the total number of edges. Each of edges  $e_i$  has a positive weight  $w_i$ .

### 3.2 Hierarchical Clustering

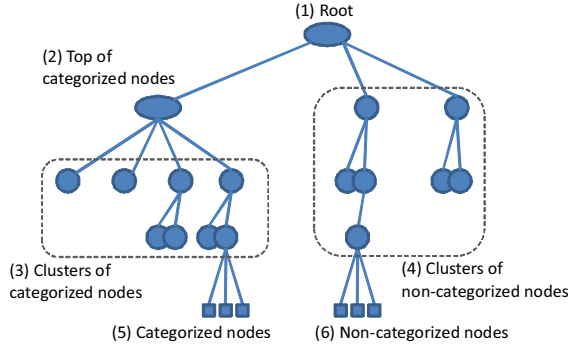


Figure 1: Hierarchical structure for the presented technique. Categorized nodes are clustered based on both connectivity and categories. Non-categorized nodes are clustered based on connectivity.

Figure 1 shows the hierarchical data structure generated by the following clustering algorithm. First of all, the algorithm divides all the nodes into *categorized nodes* and *non-categorized nodes*. Here, a categorized node  $n_i$  means a node that at least one of its Boolean values,  $b_1$  to  $b_m$ , is true. Otherwise  $n_i$  is a non-categorized node.

Then the algorithm applies connectivity-based graph clustering to the sets of categorized and non-categorized nodes, respectively. Current our implementation applies Fast Modularity community structure inference algorithm [5], which joins nodes to groups of densely connected nodes one-by-one, and consequently constructs a dendrogram of the nodes. Our implementation then gathers the nodes by a bottom-up traversal on the dendrogram, and generates clusters that consist of densely connected nodes and/or lower-level clusters. Recursively repeating the process, it finally constructs hierarchical clusters of categorized and non-categorized node, shown as (3) and (4), respectively, in Figure 1.

Finally, the technique gathers top-level clusters of categorized nodes and connects them to a branch of a tree, shown as (2) in Figure 1. Similarly, it gathers top-level clusters of non-categorized nodes and directly connects them to the root, shown as (1) in Figure 1. This structure is useful to place all categorized nodes at the center, and non-categorized nodes around the categorized nodes.

The graph clustering process consumes the largest part of the computation time of the technique; however, if users deal with the same data many times, they can store the clustering result so that they invoke the clustering process only once.

### 3.3 Hierarchical Data Layout

The technique applies a space-filling hierarchical data layout technique, because the space-filling technique satisfies Requirements 3, 4, and 5, described in Section 1. Figure 2 shows an illustration of space-filling layout applied in the technique. It applies our Treemap-like space-filling technique [8] [9], which represents the hierarchy as nested rectangular regions. It represents the root of a

tree as entire layout space, and children nodes as interior rectangular regions.

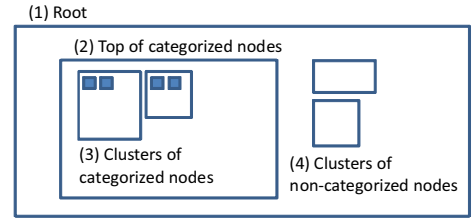


Figure 2: Illustration of space-filling hierarchy layout.

Since our Treemap-like layout is a bottom-up algorithm, it calculates the positions of nodes and clusters in the following steps:

**Step 1:** Calculate positions of lower-level clusters, placed inside the rectangular regions, shown as (3) and (4) in Figure 2.

**Step 2:** Calculate positions of clusters under the top branch of categorized nodes, placed inside the rectangular region shown as (2) in Figure 2.

**Step 3:** Calculates positions of clusters under the root node, placed inside the rectangular region shown as (1) in Figure 2.

The presented technique applies our Treemap-like layout in Step 1, and the hybrid layout introduced in Section 3.4 in Steps 2 and 3.

### 3.4 Hybrid Force-Directed and Space-Filling Layout

The technique places clusters in a hierarchy by applying a new hybrid force-directed and space-filling layout, for Steps 2 and 3 described in Section 3.3. It firstly calculates the positions by a force-directed algorithm, and then places the rectangles corresponding to the clusters by our Treemap-like algorithm, referring the results of the force-directed algorithm. Figure 3 shows the processing flow of the layout algorithm.

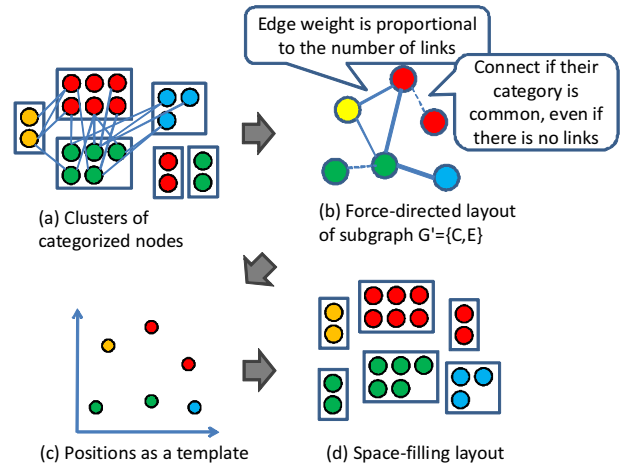


Figure 3: Processing flow of the hybrid layout technique.

The technique firstly constructs a subgraph  $G' = \{C, E\}$ , consisting of clusters  $C$  which are the children of the branch-nodes shown as (1) or (2) in Figure 1, and edges  $E$ . The technique generates an edge  $e_k$  connecting a pair of cluster  $c_i$  and  $c_j$ , if the pair satisfies at least one of the following conditions:

**Condition 1:** At least one pair of nodes  $n_i$  and  $n_j$  is connected by a link, where  $n_i$  is under  $c_i$ , and  $n_j$  is under  $c_j$ .

**Condition 2:** One or more pairs of both  $b_{ik}$  and  $b_{jk}$  are true.

The generated edges makes pairs of clusters closer, if the pairs are connected by links or belong to common categories. The technique calculates the weight  $w_i$  of the edge  $e_i$  as follows:

$$w_i = an_i + (1.0 - a)r \quad (1)$$

$$r = n_{AND}/n_{OR} \quad (n_{OR} > 0)$$

$$r = 0.0 \quad (n_{OR} = 0) \quad (2)$$

where

- $a$  is a constant value satisfying  $0 < a < 1$ ,
- $n_i$  is the number of links connecting nodes under  $c_i$  and nodes under  $c_j$ ,
- $n_{AND}$  is the number of dimensions of Boolean values that both  $b_{ik}$  and  $b_{jk}$  are true, and
- $n_{OR}$  is the number of dimensions of Boolean values that at least  $b_{ik}$  or  $b_{jk}$  are true.

Figure 3(a) is an example of a set of clusters of categorized nodes, where four clusters (red, yellow, green, and blue nodes inside) are connected by links, but other two clusters (red and green nodes inside) are isolated. Figure 3(b) is an example of the subgraph, where edges of the subgraph connect clusters which satisfy at least one of the above conditions. Equation (1) works so that the edges are more weighted according to the number of links connecting the clusters, or similarity of their Boolean values.

The technique then calculates the position of the clusters by applying a force-directed layout technique that can deal with weighted edges. Currently, our implementation applies the LinLog algorithm [12]. It then applies our Treemap-like rectangular packing algorithm [8] to realize space-filling layout. The algorithm refers the positions of clusters calculated by the force-directed layout as a template, as shown in Figure 3(c). It attempts to minimize the sum of the three penalties, which are calculated from:

- distances of corresponding rectangles from the positions described in the template,
- aspect ratio of the layout space, and
- area of the layout space,

as described in Section 5 of the reference [8]. Figure 3(d) illustrates the result of the space-filling layout process.

The hybrid layout brings the features of both layout techniques: smaller edge lengths by force-directed layout, no cluttering and better screen space utilization by space-filling layout. Moreover, space-filling layout algorithm does not occupy large part of computation time, at least in our test cases, because numbers of clusters connected by parent nodes ((1) or (2) in Figure 1) are much smaller than the total number of nodes.

### 3.5 Drawing and Adjustment

Currently, our implementation supports the following features for effectively drawing the multiple-category graphs.

#### Node Swapping

When an edge connects two nodes that are in adjacent clusters, it may have to travel through the two cluster regions. We can improve the visualization by swapping the position of the node with that of another node in the same cluster. Based on this idea, our implementation swaps positions of nodes in a cluster after completing the space-filling hierarchy layout. This process firstly sorts the nodes in a cluster in descending order based on their degree. Then, as the

current node traverses the positions of all unfixed nodes in the cluster, it calculates the sum of link lengths for all links connecting to the node. Once the best position is found, the process swaps node positions and fixes the current node. The process repeats one-by-one in sorted order.

#### Randomization

The collinear node problem found in the Treemap-based technique [11] is also inherited by our approach. Whenever nodes are placed collinearly, edges between these nodes may overlap, making it difficult to understand the connections. To help correct this problem, our implementation adds small random positive or negative values to node positions.

#### Node Concentration

We assume that users are often interested in connectivity between pairs of categorized nodes or pairs of categorized and non-categorized nodes but not in pairs of non-categorized nodes. In this case, we do not need to show details of clusters among non-categorized nodes, and focus and context techniques are useful for providing this type of support.

Currently, our implementation concentrates non-categorized nodes to the center of their rectangular regions. This feature hides the detailed connectivity among non-categorized nodes and clarifies inter-cluster relationship between pairs of categorized and non-categorized clusters.

Also, our implementation features the ability to distort regions where non-categorized nodes are placed by moving the clusters of non-categorized nodes closer to the clusters of categorized nodes. It simply calculates the new position of a non-categorized node  $(x', y')$  from its original position  $(x, y)$  by the following equations:

$$\begin{aligned} x' &= (1.0 - d)x + dx_{c0} & (x < x_{c0}) \\ x' &= x & (x_{c0} \leq x \leq x_{c1}) \\ x' &= (1.0 - d)x + dx_{c1} & (x > x_{c1}) \\ y' &= (1.0 - d)y + dy_{c0} & (y < y_{c0}) \\ y' &= y & (y_{c0} \leq y \leq y_{c1}) \\ y' &= (1.0 - d)y + dy_{c1} & (y > y_{c1}) \end{aligned} \quad (3)$$

Here,  $d$  is a constant value satisfying  $0 < d \leq 1$ ,  $x_{c0}$  and  $x_{c1}$  are the x-coordinates of the left/right edge of the top cluster of categorized nodes, and  $y_{c0}$  and  $y_{c1}$  are the y-coordinates of the lower/upper edge of the top cluster.

By using this feature, users can view overall context of the graph while zooming in on the categorized nodes. An example in Section 4 demonstrates the effectiveness of the feature.

#### Drawing

Figure 4 shows how to draw nodes and links with our current implementation.

Our implementation assigns independent colors to the categories and draws the categorized nodes as colored circles. When a node belongs to multiple categories, the implementation divides the circle into multiple fans, similar to a pie chart. The colors are assigned so that each fan is closest to the adjacent clusters of the same color. Our approach calculates radii of nodes from the number of connected links because we would like to emphatically represent hub nodes. For non-categorized nodes, however, our implementation simply draws smaller gray dots. Also, current our implementation allows for category selection via a graphical user interface. If a user selects a category, it highlights the nodes which belong to the selected category and draws other categorized-nodes darker.

At the same time, the implementation draws each link in one of following three levels:

**Level 1:** The line is relatively thick and bright. Our implementation usually draws links connecting two categorized nodes as Level 1. If a user selects a category on the GUI, it applies Level 1 only to the links connecting one or two nodes belonging to the selected category.

**Level 2:** The line is moderately thick and bright. Our implementation draws links connecting pairs of categorized and non-categorized nodes as Level 2.

**Level 3:** The line is relatively thin and transparent. Our implementation draws links connecting two non-categorized nodes as Level 3.

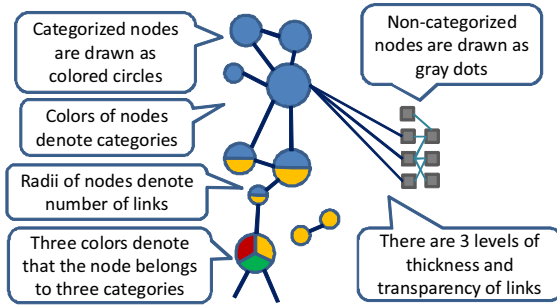


Figure 4: Drawing style of our implementation. It distinguishes categorized and non-categorized nodes by the colors of nodes and the thickness/transparency of links.

### 3.6 Interactive Layout Modification

We assume that users are more often interested in connectivity of nodes that belong to a specific category. Our hybrid layout algorithm attempts to place clusters that belong to the same categories close together. However, this is a complicated procedure because the layout takes both connectivity and categories into consideration, and therefore, it is often difficult to obtain the best placement for every cluster. One solution to this problem is interactive selection of categories of interest and layout modification so that these important clusters are placed in close proximity.

Figure 5 shows an illustration of the layout modification. When selecting a category, the process firstly extracts clusters that belong to the category. It fixes the cluster  $c_F$  that contains the largest number of nodes in the extracted clusters. It then moves other clusters around the fixed cluster, as shown in Figure 5(right). Digits in Figure 5 denote the categories of the clusters, and the rectangle painted in sky-blue denote the fixed cluster  $c_F$ . An example in Section 4 shows the effectiveness of this feature.

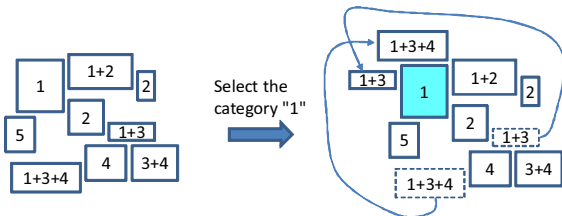


Figure 5: Local modification of clusters to closely place the clusters belonging to the selected category.

When calculating the position of moving clusters, our system firstly connects  $c_F$  and other extracted clusters by temporary edges

and calculates their weights with equation (1). The process then sorts the extracted clusters in descending order of weights of the temporary edges. It then calculates their positions one-by-one in the sorted order. Afterwards, it extracts multiple candidate positions, as described in Section 3 of the reference [9]. Finally, it selects the position closest to  $c_F$ , to place the current cluster there. By repeating this process in the order of the temporary edge weights, tightly connected clusters are placed closer together.

## 4 EXAMPLES AND DISCUSSION

This section shows some examples of the visualization results, and discusses the effectiveness of the presented technique. We developed the technique on JDK (Java Development Kit) 1.5, and executed on MacBook Pro (CPU 2.2GHz, RAM 2.0GB) with Windows XP Service Pack 2. We used a test data containing 6,512 nodes, 7,564 links, and 10 categories. Computation time was 21.3 seconds for hierarchical clustering, and 4.7 seconds for data layout. Though hierarchical clustering process occupied a large part of computation time, we executed the process and store the result only once.

Figure 6 shows an example of results, zooming in the clusters of categorized nodes. In Figure 6(left) only links connecting pairs of categorized nodes are highlighted. The layout result looks totally good, because many nodes connected by links or painted in the same colors get closer, and the nodes are compactly placed. For example, bright blue and red nodes are concentrated in the lower-left region indicated as [A] in Figure 6(left), and a hub node is properly placed in a cluster where is closer to the adjacent two clusters. Similarly, dark blue and brown nodes are concentrated in the lower-right region indicated as [B] in Figure 6(left). Yellow nodes are concentrated in the central region indicated as [C] in Figure 6(left). Meanwhile, almost isolated two sets of nodes, one is the set of dark green and orange nodes, and the other is the set of bright green and blue nodes, are separately placed in the upper-right region indicated as [D] in Figure 6(left).

Figure 6(right) shows rectangular borders of the clusters, where it denotes that the technique did not cause any cluttering.

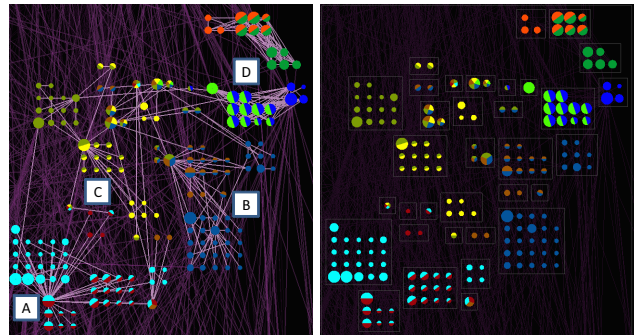


Figure 6: Result(1). Zooming in the categorized nodes. (left) Nodes connected by links or painted in the same colors properly get closer, indicated as [A], [B], and [C]. Almost isolated two sets of nodes are properly separated, indicated as [D]. (right) Rectangular regions of the clusters. No nodes or clusters are cluttered.

Next, let us discuss about the advantage of the presented layout against simply applying force-directed or space-filling layout.

We simply applied hierarchical clustering and space-filling layout to the same data, without applying force-directed layout, for the comparison with the presented technique. We generated the following two types of hierarchical graphs for this test, so that space-filling layout can reflect both categories and connectivity.

- *Hierarchical graph 1:* We divided the nodes into categorized and non-categorized nodes, and divided the categorized nodes

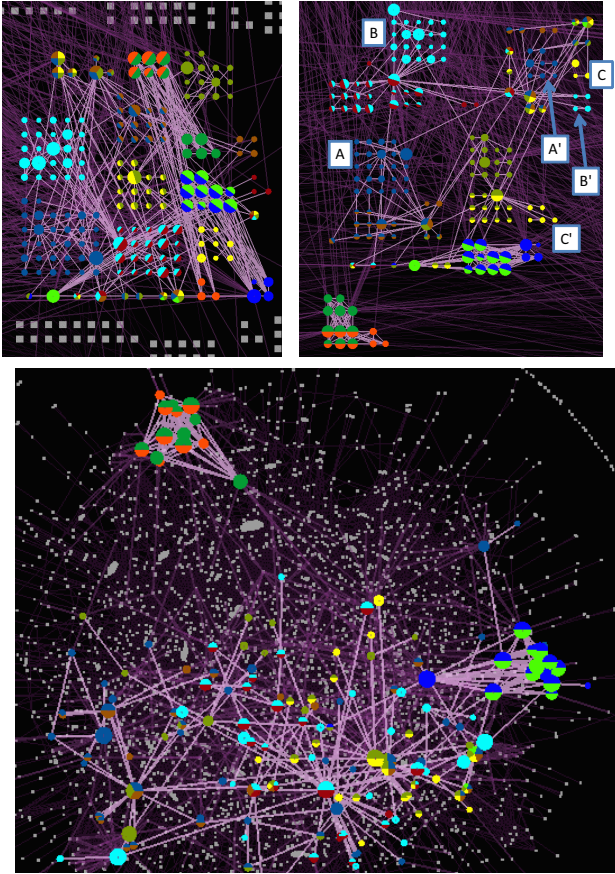


Figure 7: Comparison with existing techniques. (upper left) Result with hierarchical clustering 1. There are too many intersections among edges. (upper right) Result with hierarchical clustering 2. Two clusters belonging to the common categories separate each other, indicated as [A] and [A'], [B] and [B'], and [C] and [C']. (lower) Result by simply applying force-directed layout. Some clusters are not clearly represented. Moreover, it takes more than 4 minutes.

based on their categories. Finally, we divided the nodes of each cluster based on connectivity.

- *Hierarchical graph 2*: We divided the nodes into categorized and non-categorized nodes, and divided the categorized nodes based on their connectivity. Finally, we divided the nodes of each cluster based on their categories.

Figure 7(upper) shows the result of displaying the above two hierarchical graphs by simply applying space-filling data layout. Computation times were smaller than the presented technique: 1.3 seconds for hierarchical data 1, and 1.2 seconds for hierarchical data 2. Figure 7(upper left) shows that the space-filling layout could compactly represent hierarchical graph 1. Since the nodes are firstly clustered based on their categories, it could closely place the nodes painted in the same colors. However, it is quite difficult to understand the connectivity, because many pairs of far nodes are connected and therefore many links intersect with one another.

On the other hand, Figure 7(upper right) shows that space-filling layout could preferably represent hierarchical data 2. Since the nodes are firstly clustered based on their connectivity, it could comprehensively draw the links. However, clusters painted in the same colors are often separately placed, such as dark blue nodes indicated as [A] and [A'], bright blue nodes indicated as [B] and [B'],

and yellow nodes indicated as [C] and [C'].

We numerically evaluated the layout results by the hybrid layout and space-filling layouts using hierarchical graphs 1 and 2. For the evaluation, we firstly normalized the positions of all categorized nodes, where positions of corners of the rectangular region encloses the all categorized nodes are  $(-1, -1)$ ,  $(-1, 1)$ ,  $(1, -1)$ , and  $(1, 1)$ . And then we calculated the five metrics: maximum and average distances between pairs of nodes which belong to completely common categories, maximum and average lengths of links connecting two categorized nodes, and total number of intersections of the links connecting two categorized nodes. Table 1 denotes the results of the numerical evaluations, where HG1 and HG2 are hierarchical graphs 1 and 2, respectively. It shows that maximum and average lengths of links and number of intersections were much larger when we used HG1. Also, it shows that maximum and average distances of nodes were relatively large when we used HG2.

Above results suggest that it is a difficult problem to satisfy both category-based and connectivity-based requirements by simply drawing hierarchically clustered graphs. We think the hybrid layout is a good solution to satisfy the both two requirements.

Table 1: Evaluation of our hybrid method with other methods.

	HG1	HG2	Hybrid
Distance of nodes (max.)	2.133	2.374	2.155
Distance of nodes (ave.)	0.827	0.879	0.776
Length of links (max.)	2.089	1.389	1.326
Length of links (ave.)	0.567	0.254	0.255
Number of intersections	5123	1746	1489

Next, we used the LinLog force-directed algorithm [12] to the same data, without applying hierarchical clustering. Figure 7(lower) shows the result. It succeeded to clearly represent some of clusters of categorized nodes, but still other clusters are not clear. It is similar to the result shown in [11], which hierarchical graph is useful to represent clusters of the graphs, rather than force-directed layout. In addition, the computation time was much larger by simply applying force-directed layout. The experiment required 267.5 seconds for 100 times of iterative calculation, even though LinLog implementation applies a quadtree for acceleration. On the other hand, our hybrid layout algorithm required 4.7 seconds with the above test data, as above mentioned. Of them, 2.6 seconds was spent for force-directed layout, 1.7 seconds was for template-based space-filling layout. 0.4 seconds was for the rest of processes, such as lower-level data layout (without using the template), and node swapping. We think that hybrid technique is therefore preferable in performance rather than traditional force-directed techniques.

Figure 8 shows an example of results, zooming out to display the whole graph. Figure 8(upper left) shows the original layout, and Figure 8(upper right) figure shows that non-categorized nodes are concentrated to the centers of the clusters. The result shows that the node concentration process improves the comprehensibility of inter-cluster connectivity. Figure 8(lower) shows that the clusters of non-categorized nodes are not only concentrated but also distorted to get closer to the clusters of categorized nodes. The result shows the focus and context effect, which makes easier to understand the connectivity of non-categorized nodes while zooming in the categorized nodes.

Figure 9 shows an example of results, where a category (painted in ocher) is selected. Both left and right images highlight links connected to the ocher nodes. Figure 9(right) is an example that applies local modification so that the ocher nodes get closer, where Figure 9(left) is an example that does not apply. If users would like to focus on a specific category and zoom in the corresponding clusters, the local modification should be useful.

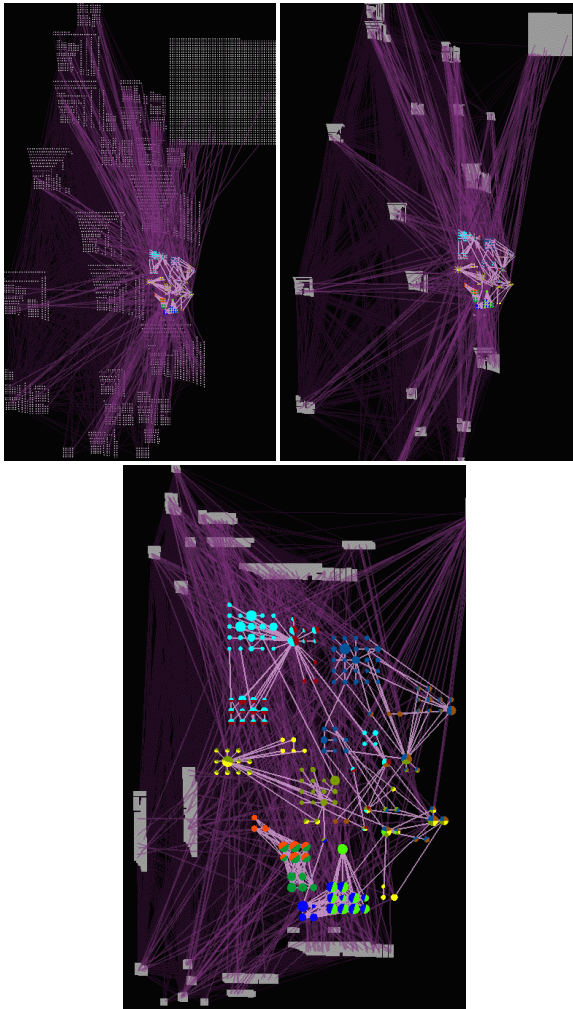


Figure 8: Result(2). (upper left) Original layout. (upper right) Non-categorized nodes get closer to centers of the clusters. It helps to understand inter-cluster connections. (lower) Non-categorized nodes are distorted and get closer to categorized nodes, to realize focus and context. It helps to understand which clusters of non-categorized nodes are tightly connected to specific categorized nodes.

## 5 A CASE STUDY: VISUALIZATION OF ACTIVE BIOLOGICAL NETWORK

The test data introduced in Section 4 was originally an active biological network. The data was created by one of the authors working on bioinformatics. It was generated by d-COPINE algorithm [14], from a protein-protein interaction network consisting of 6,152 genes, and 7,564 interactions between pairs of genes. Also, the genes had expressions of 173 types of stress conditions. In other words, the original graph data consisted of 6,152 nodes with 173-dimensional values, and 7,564 links.

Goal of the data construction was simultaneous identification of the active biological networks and conditions from protein-protein interactions and gene expression data. To archive the goal, d-COPINE extracted interaction subnetworks, called common pattern graph (CPG) in [14], whose genes satisfy common expression or repression conditions. Here, the expression or repression conditions can be described as a logical multiplication of equation/inequation of the values in the specific dimensions.

As a result, d-COPINE defined 10 expression or repression con-

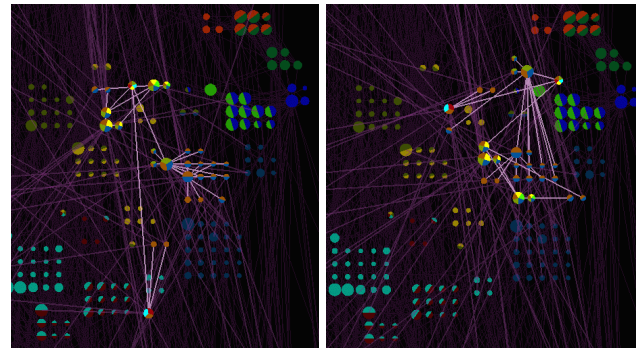


Figure 9: Result(3). (left) Highlighting links connecting nodes that belong to the specific category, without layout modification. (right) Highlighting links connecting nodes that belong to the specific category, with layout modification. The layout modification is useful to zoom in the specific category.

ditions to adequately extract subnetworks, where it allowed genes to satisfy multiple conditions. These conditions were useful to identify specific interactions of genes. We used the network as the test data, treating the 10 conditions as categories. We discussed with a bioinformatics scientist how the visualization result shown in Figure 10 brought knowledge for the active biological network. This section introduces the discussion as below.

Here we can clearly observe some hub nodes, indicated as [A] in Figure 10, where such hub genes are often responsible genes. Observation of such hub nodes and their connectivity is often helpful for clarification of the cause of diseases and development of drugs.

Several nodes indicated as [B] in Figure 10 belong to three or more categories, and connect to nodes in large clusters. The result means that these genes are sensitive with many conditions, and have interactions with many other genes. Such multifunctional genes often locate on important portions of living bodies. However, it is difficult to clarify the behavior and functionality of such important genes, because experiments for the genes may often kill the cells that the genes belong to. Therefore, it is important to alternatively observe which nodes/clusters such important genes interact with, because this knowledge is helpful to understand their behavior. The result shows that the presented technique can contribute to observe such important multifunctional genes.

Figure 10 [C] shows that a subgraph, consists of nodes painted in orange or dark green, is disconnected from other categorized nodes. This result shows that d-COPINE found good two conditions, because we can separately observe the genes in the subgraph from other categorized genes. It means that the presented technique can be used to determine the quality of categorization results.

Figure 10(lower) shows an interesting feature of multiple categorizations by d-COPINE. There are three pairs of clusters [D] and [D'], [E] and [E'], and [F] and [F'], which consist of nodes painted in the same color. It shows that d-COPINE defined conditions that contribute to extract multiple subgraphs. This kind of clustering result is novel in active biological network analysis, and we expect such result may bring new knowledge in this field.

Observation of inter-cluster connectivity between categorized and non-categorized nodes is also an interesting issue. Figure 8 shows that the presented technique clearly represents connections between clusters of nodes belong to specific categories and clusters of non-categorized nodes. It is possible to discover interesting clusters of non-categorized nodes that have many interactions with specific categorized nodes, by using the presented technique. Or, it is also possible to define more categories from the result, if such clusters consist of non-categorized nodes which have common ex-

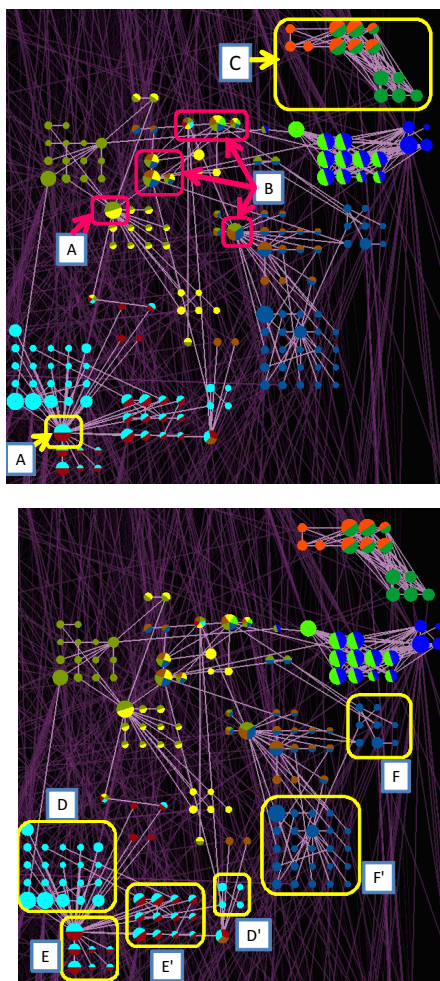


Figure 10: Visualization of active biological networks. We discussed the usefulness of the result with a bioinformatics scientist as follows. It helps to find hub genes indicated as [A], and important genes that belong to many categories indicated as [B]. It also helps to find clusters successfully isolated indicated as [C], and sets of clusters belong to completely common categories but separated, indicated as [D] and [D'], [E] and [E'], and [F] and [F'].

pression/repression conditions.

## 6 CONCLUSION

This paper addresses the need of understanding multiple-category graphs by introducing a hybrid space-filling and force-directed layout method. The case study shows that the hybrid layout method helps reveal important features of an active biological network.

Our method has two potential limitations. Firstly, the cost of our method grows with the number of categories. Fortunately, in practice, most data sets contain a moderate number of categories. Secondly, since we color nodes according to their categories, there is a limit on the number of distinguishable colors. One possible solution is to incorporate glyph-based multi-dimensional visualization techniques [19], and the other is to display the graph at multiple abstraction layers based on a hierarchy for categories.

Without losing generality, our study to date is based on one particular multiple-category graph data set. We anticipate obtaining additional data sets to expand our case study and perform more comprehensive evaluations. Multiple-category graphs are com-

monly found in other applications. We will also apply our techniques to these other applications.

## ACKNOWLEDGEMENT

This work is supported in part by the US National Science Foundation and the US Department of Energy. Also, we appreciate Mr. Chad Jones for the proofread of this paper.

## REFERENCES

- [1] B. Bederson, B. Schneiderman, Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, *ACM Transactions on Graphics*, 21(4), 833-854, 2002.
- [2] D. Bolser, P. Dafas, R. Harrington, J. Park, M. Schroeder, Visualisation and Graph-Theoretic Analysis of a Large-scale Protein Structural Interactome, *BMC Bioinformatics*, 4, 45, 2003.
- [3] R. Bourqui, D. Auber, P. Mary, How to Draw Clustered Weight Graphs Using a Multilevel Force-Directed Graph Drawing Algorithm, *11th International Conference on Information Visualization (IV07)*, 757-764, 2007.
- [4] B.-J. Breitkreutz, C. Stark, M. Tyers, Osprey: a Network Visualization System, *Genome Biology*, 4, R22, 2003.
- [5] A. Clauset, M. E. J. Newman, C. Moore, Finding Community Structure in Very Large Networks, *Physical Review*, E70, 066111, 2004.
- [6] S. Hachul, M. Junger, An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs, *Proceedings of Graph Drawing 2005*, 235-240, 2005.
- [7] D. Holten, Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Information Visualization 2006)*, 12(5), 741-748, 2006.
- [8] T. Itoh, Y. Yamaguchi, Y. Ikehata, Y. Kajinaga, Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm, *IEEE Transactions on Visualization and Computer Graphics*, 10(3), 302-313, 2004.
- [9] T. Itoh, H. Takakura, A. Sawada, K. Koyamada, Hierarchical Visualization of Network Intrusion Detection Data in the IP Address Space, *IEEE Computer Graphics and Applications*, 26(2), 40-47, 2006.
- [10] B. Johnson, B. Shneiderman, Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, *IEEE Visualization '91*, 275-282, 1991.
- [11] C. Muelder, K.-L. Ma, A Treemap Based Method for Rapid Layout of Large Graphs, *Proceedings of IEEE Pacific Visualization Symposium 2008*, 231-238, 2008.
- [12] A. Noack, Energy Models for Graph Clustering, *Journal of Graph Algorithms and Applications*, 11(2), 453-480, 2007.
- [13] D. Schaffer, Z. Zuo., S. Greenberg, L. Bartram, J. Dill, S. Dubs, M. Roseman, Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods, *ACM Transactions on Computer-Human Interaction*, 3(2), 162-188, 1996.
- [14] M. Seki, J. Sese, Identification of Active Biological Networks and Common Expression Conditions, *8th IEEE International Conference on Bioinformatics and BioEngineering*, 2008.
- [15] K. Sugiyama, K. Misue, Graph Drawing by the Magnetic Spring Model, *Journal of Visual Languages and Computing*, 6(3), 217-231, 1995.
- [16] Y. Tao, Y. Liu, C. Friedman, Y. A. Lussier, Information Visualization Techniques in Bioinformatics during the Postgenomic Era, *Drug Discovery Today: BIOSILICO*, 2(6), 237-245, 2004.
- [17] Y. Tian, R. A. Hankins, J. M. Patel, Efficient Aggregation for Graph Summarization, *Proceedings of the 28th ACM SIGMOD International Conference on Management of Data*, 2008.
- [18] V. Tsiaras, S. Triantafilou, I. G. Tollis, Treemaps for Directed Acyclic Graphs, *Lecture Notes in Computer Science (Proceedings of Graph Drawing 2007)*, 377-388, 2007.
- [19] M. O. Ward, A Taxonomy of Glyph Placement Strategies for Multi-dimensional Data Visualization, *Information Visualization*, 1(3), 194-210, 2002.
- [20] S. Zhao, M. J. McGuffin, M. H. Chignell., Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams, *Proceedings of IEEE Information Visualization 2005*, 57-64, 2005.