Key-Node-Separated Graph Clustering and Layouts for Human Relationship Graph Visualization

Takayuki Itoh ■ *Ochanomizu University, Japan*

Karsten Klein ■ *Monash University, Australia*

Many graph-drawing methods apply node-clustering techniques based on edge density to find tightly connected subgraphs and then hierarchically visualize the clustered graphs. The proposed graph visualization technique for attribute-embedded graphs separately visualizes the key nodes by accounting for a combination of connections and attributes.

Many real-world applications use graphs to model relations between entities. Common graph analysis tasks include understanding correlations between the entities' connections and attributes and discovering key entities with many connections or particular attributes. For example, graph visualization can be useful for analyzing human relationships. People in social networks have connections (or friendships), and the relations between subsets of people might form complex subnetworks. Understanding of correlations between people's connection structures and attributes, such as jobs, hometowns, and hobbies, may contribute to various social and business analytics. Also, discovering key people and analyzing their positions and roles in a network may help us understand how information spreads in social networks.

Development of key-node-conscious graph visualization techniques is therefore expected to be an important contribution in many academic and business fields. Moreover, many applications must take into account additional information, such as node attributes. In the case of human relationship graphs, where nodes correspond to people, we often need to take into account people's preferences as node attributes. Their preferences can be converted to numeric values using analysis techniques such as natural language processing of the documents written by each person. Graph visualizations featuring key-node-identification that combines topology and attribute-based clustering is therefore useful for these applications. Also, this clustering will contribute to visible representation of the tight connections between key nodes and their adjacent clusters.

This article presents a graph visualization technique that features graph clustering based on a combination of structural neighborhood and attribute similarity. The technique supposes that each node of the graph has a constant-sized feature vector representing its attributes. It calculates distances between pairs of nodes according to the similarity of feature vectors and the commonality of neighbor nodes, and it generates clusters of the nodes according to their distances. It then calculates the initial positions of clusters applying a graph layout algorithm and adjusts the positions using a triangular mesh-smoothing algorithm, in which the mesh is generated by connecting centers of clusters with a Delaunay triangulation algorithm. Finally, it calculates the positions of nodes in each of the clusters, applies an edge bundling technique to design the shapes of edges, and draws all the nodes and edges.

 Published by the IEEE Computer Society

Graph-Clustering Problem

Graph clustering partitions a graph's node set into clusters (that is, disjoint node sets); the goal is to have many intracluster edges and a few intercluster edges. Graph clustering is useful for both visualization and various analytic purposes.

The most popular graph-clustering approach is to extract tightly connected subgroups of nodes. Many graph-clustering techniques using this approach are based on agglomerative-clustering algorithms. Various criteria exist for measuring the distances among graph nodes,[1] including the similarity of feature value vectors in multidimensional Euclidean spaces, adjacency-based node similarity measurements, and measuring the number of paths between two nodes to determine the tightness of the connections.

Often, nodes or substructures of a graph are aggregated for analysis and visualization based on either the structure or attributes, for example, by using hierarchical clustering. Various hierarchical graph visualization techniques have been developed because a hierarchical structure can provide an overview and quick understanding of the graph structures and facilitates top-down operations for graph exploration and navigation. When no hierarchy on the nodes is given as input, graph-clustering algorithms are often applied to allow hierarchical graph visualization. Many of them treat tightly connected subregions of the graphs as clusters, but that does not always work well for the discovery of key nodes and tight connections.

Figure 1 illustrates the graph-clustering problem. The red nodes in Figure 1a have numerous connections, making them important nodes in this graph. However, they may belong to cluster 1 because they form a tightly connected subgraph with the five blue nodes, even though they also connect to many other nodes. In this case, the red nodes are not as visible in the high-level structure of the clustered graph shown in Figure 1b. Also, many edges between the blue and red nodes are not visible because they are drawn in a small region corresponding to cluster 1. Users may miss these important connections while looking at the high-level structure. If we use commonality of neighbor nodes or similarity of attributes as criteria for the clustering process, however, the red nodes will be separated from this cluster. When the commonality of neighbor nodes is applied to the clustering, five blue nodes form a cluster because they are connected to just blue and red nodes, and two red nodes form another cluster because they are connected to all the colored nodes. Then, if the nodes' colors represent their attributes, the blue and red nodes can form separate clusters, as Figure 1c illustrates. A thick bundle connecting the two clusters can highlight a set of connections between blue and red nodes. In this case, the red node is still visible in the high-level structure of the graph in Figure 1d. This structure is preferable for the following reasons:

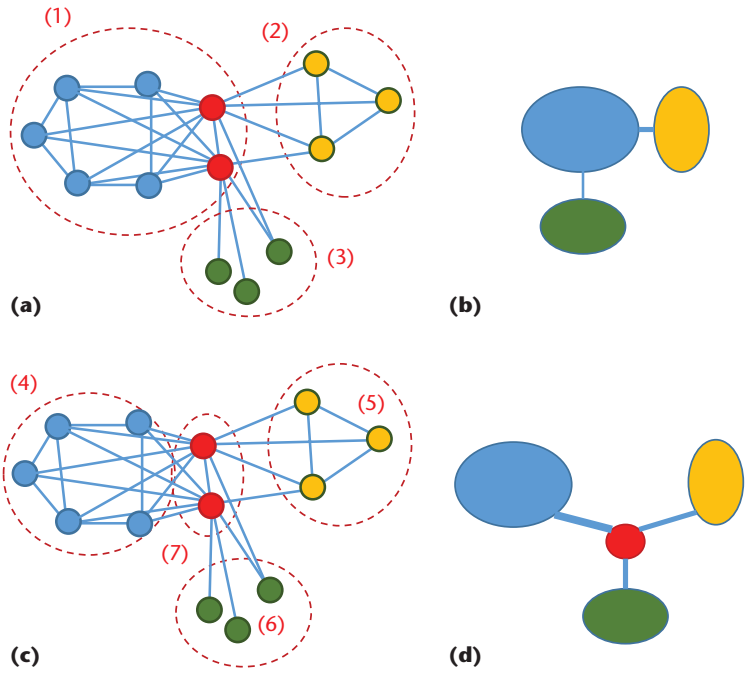


Figure 1. Graph clustering. Clustering based on (a) the density of connections and (b) a high-level drawing of the corresponding clustered graph. Clustering based on (c) the commonality of neighbors and (d) a high-level drawing of the corresponding clustered graph.

- Connectivity with important nodes and multiple clusters will be comprehensive in the visualization results.
- Fewer edges will be tangled in the small regions of the display spaces corresponding to clusters.
- Edge-bundling algorithms can be easily applied to a large number of edges connecting to important nodes, such as the red node in Figure 1d.

See the "Related Work in Graph Visualization" sidebar for a description of related research in graph-based information visualization.

Presented Technique

We begin by presenting our graph visualization technique and its components. This includes the clustering and layout methods, rendering, and the interaction concept.

Processing flow

The presented technique supposes that all nodes have the same constant number for the dimension of the feature vector. The following is the supposed

Related Work in Graph Visualization

Hierarchically clustered graphs are an effective data structure for information visualization because they are suitable for overview, zoom, and filtering operations. The node layout is an important issue for comprehensive graph visualization. Many improvements in quality and computational efficiency have been achieved in the development of force-directed methods since Peter Eades proposed his seminal spring embedder model.[1] Better alternatives include multilevel approaches and recent advances in stress-based layout computation[2] that basically apply a multidimensional scaling. Several techniques efficiently visualize hierarchical graphs by applying space-filling layout techniques such as tree maps.[3,4]

Edge bundling has been an active topic since Danny Holten presented a technique that hierarchically bundles edges connecting the nodes of a tree structure level-by-level and draws them as spline curves.[5] Edge bundling is especially effective when coupled with our proposed technique because it clusters nodes based on the commonality of adjacent nodes, and consequently constructs a node hierarchy so that a large number of edges can be bundled.

It is often interesting to emphasize important nodes in graphs. Yukio Ohsawa and his colleagues presented the KeyGraph concept, which features nodes bridging multiple subgraphs.[6] Carlos Correa and his colleagues applied various schemes to calculate centrality sensitivity to effectively visualize social networks.[7] Our work takes into account the combination of neighborhoods and attributes of nodes in a different way than these techniques. Meanwhile, several graph-drawing techniques are suitable for key-node conscious drawing, such as bipartite graphs. Seok-Hee Hong and her colleagues visualized citation networks by dividing the most cited papers and other papers into different types of nodes and applying a bipartite graph-drawing technique to focus on the connections between important nodes and the others.[8] Our technique also focuses on such connections, but we can combine it with a variety of graph layout algorithms.

Multivariate node attributes are also often taken into account during the graph layout processes. Ben Shneiderman and Aleks Aris presented an implementation and a case study of graph visualization that applied a substrate metaphor, which specifies the positions of nodes based on their attribute values.[9] Lei Shi and his colleagues presented a graph-clustering scheme that applies attribute-based partitioning for higher-level nodes and topology-based partitioning for lower-level nodes.[10] The technique proposed here also takes into account both attributes and topology, but it applies attribute- and adjacency-based metrics simultaneously in a single clustering process.

References

1. P. Eades, "A Heuristics for Graph Drawing," *Congressus Numerantium*, vol. 42, 1984, pp. 146–160.
2. E.R. Gansner, Y, Hu, and S. North, "A Maxent-Stress Model for Graph Layout," *IEEE Trans. Visualization and Computer Graphics*, vol. 19, no. 6, 2013, pp. 927–940.
3. W. Didimo and F. Montecchiani, "Fast Layout Computation of Clustered Networks: Algorithmic Advances and Experimental Analysis," *Information Sciences*, vol. 280, 2014, pp. 185–199.
4. T. Itoh et al., "A Hybrid Space-Filling and Force-Directed Layout Method for Visualizing Multiple-Category Graphs," *Proc. IEEE Pacific Visualization Symp.*, 2009, pp. 121–128.
5. D. Holten, "Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, 2006, pp. 741–748.
6. Y. Ohsawa, N.E. Benson, and M. Yachiba, "KeyGraph: Automatic Indexing by Co-occurrence Graph Based on Building Construction Metaphor," *Proc. IEEE Int'l Forum on Research and Technology Advances in Digital Libraries* (ADL), 1998, pp. 12–18.
7. C. Correa, T. Crnovrsanin, and K.-L. Ma, "Visual Reasoning about Social Networks Using Centrality Sensitivity," *IEEE Trans. Visualization and Computer Graphics*, vol. 18, no. 1, 2012, pp. 106–120.
8. S. Hong et al., "A Framework for Visual Analytics of Massive Complex Networks," *Proc. Int'l Conf. Big Data and Smart Computing* (BIGCOMP), 2014, pp. 15–17.
9. B. Shneiderman and A. Aris, "Network Visualization by Semantic Substrates," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, 2006, pp. 733–740.
10. L. Shi et al., "Hierarchical Focus+Context Heterogeneous Network Visualization," *Proc. IEEE Pacific Visualization Symp.*, 2014, pp. 89–96.

data structure of an input graph G consisting of a set of nodes N and edges E:

$$
\begin{aligned}
G &= \{N, E\} \\
N &= \{n_1, \ldots, n_{N_n}\} \\
E &= \{e_1, \ldots, e_{N_e}\} \\
n_i &= \{a_{i1}, \ldots, a_{iN_a}\} \\
e_i &= \{n_j, n_k\}
\end{aligned} \tag{1}
$$

Here, n_i denotes the ith node, e_i denotes the ith edges, a_{ij} denotes the value of the jth dimension of the ith node, N_n denotes the number of nodes, N_e denotes the number of edges, and N_a denotes the number of dimensions.

The proposed technique first generates clusters of nodes according to the similarity of feature vectors and commonality of adjacent nodes. It then applies a stress-minimizing graph layout algorithm to the

1 生成簇类
2 计算初始位置
3 边绑定

节点颜色由最大的特征值决定

set of clusters to calculate these initial positions. In the next step, these positions are connected by a Delaunay triangulation algorithm, and a mesh-smoothing algorithm is applied to set apart the dense clusters and maintain the preferable distances between them. Finally, the technique draws the graph by applying a Bézier-curve-based edge bundling. Our current implementation specifies a node's color based on the dimension that has the largest value in a_{i1} to a_{iN_a}. Also, it optionally paints the triangular mesh used for the cluster position arrangement as the background color by interpolating the nodes' colors.

Graph Clustering

We calculate two types of distances between all pairs of nodes: dissimilarity of feature vectors d_{vec} and discommonality of the adjacent nodes d_{adj}. We define the distance between a pair of nodes as

$$d = \alpha d_{vec} + (1.0 - \alpha) d_{adj} , \quad (2)$$

where α is a user-specified value satisfying ($0.0 \leq \alpha \leq 1.0$). Our implementation features a GUI slider widget so that users can freely adjust α and interactively look at various visualization results. The technique then generates clusters of nodes by an agglomerative-clustering algorithm with the furthest-neighbor method. The algorithm starts generating clusters consisting of one node and repeats the merge of clusters until the minimum distance between two clusters exceeds the user-defined threshold. The two types of distances are calculated by the following process.

The dissimilarity of feature vectors technique calculates the similarity between the two nodes as the inner product of the feature values. We define the dissimilarity calculated from the inner product as the distance between the two nodes d_{vec} as follows:

$$d_{vec} = 1.0 - \text{inner}$$
$$\text{inner} = n_i \cdot n_j \,/\, |\, n_i \,||\, n_j \,| \,. \quad (3)$$

We define the discommonality of adjacent nodes by the number of commonly connected nodes. To specify the distance d_{adj} between two nodes n_i and n_j, we count the number of nodes that are connected to both n_i and n_j, simply calculating the distance as follows:

$$d_{adj} = 1.0/(1 + n_{adj}) , \quad (4)$$

where n_{adj} is the number of nodes connected to both the nodes.

Cluster Layout

The presented technique then generates a cluster graph, consisting of vertices corresponding to the clusters of nodes, and bundles corresponding to sets of edges connecting two nodes belonging to two different clusters. Bundles are weighted according to the number of edges.

We denote the cluster graph as

$$CG = \{V, B\}$$
$$V = \{v_1, \ldots\}$$
$$B = \{b_1, \ldots\}$$
$$v_i = \{n_{i1}, \ldots\}$$
$$b_i = \{e_{i1}, \ldots\} , \quad (5)$$

where CG is the cluster graph, V is the set of vertices, and B is the set of bundles. A vertex v_i consists of a set of nodes n_{ij} belonging to the cluster corresponding to v_i, and a bundle b_i consists of a set of edges e_{ij} connecting the pairs of nodes belonging to the corresponding same pair of clusters.

Because our approach is based on the calculation of distances between nodes, a distance-based layout method is a natural choice to calculate the cluster graph's layout. In our implementation, we apply the stress minimization layout algorithm from the Open Graph Drawing Framework (OGDF, www.ogdf.net) to calculate initial positions for the vertices. This algorithm tries to minimize the error between the geometric distances between pairs of nodes in the drawing and given input distances. Starting from an initial drawing using the PivotMDS method,[2] the algorithm iteratively minimizes a stress function on the nodes' positions. Stress in the layout is defined as

$$\sum_{u,v} w_{uv} \left(\|p_u - p_v\| - d_{uv} \right)^2 , \quad (6)$$

where w_{uv} is a normalization constant set to $1/sp(u, v)^2$, the inverse of the square of the shortest path between two nodes. This gives priority to the error between nodes that have a small graph theoretic distance compared with long-range errors. The position of node u in the layout is p_u, and d_{uv} is the ideal distance between u and v. In graph layout algorithms, these distances are usually set to the graph theoretic distance between the nodes. To apply the distance-based method to the cluster graph, we need to derive distances for the clusters similar to the distance calculation for the nodes. In addition to the distances, we also want to account for the cluster graph topology—that is, the connectivity between clusters. To this

end, our current implementation defines a weight w for each bundle as

$$w = \beta w_{bun} + (1.0 - \beta) w_{\text{vec}}, \quad (7)$$

where w_{bun} is proportional to the number of edges belonging to the bundle. We calculate w_{vec} from the inner products of the average vector values of the two clusters, similar to Equation 3. β is a user-specified value satisfying $(0.0 \leq \beta \leq 1.0)$. The obtained weights are reversed to obtain distances for the layout algorithm—that is, the larger the weight, the smaller the ideal distance between the connected vertices. Based on these distances between adjacent vertices, all pairwise distances between clusters are calculated using all pairs shortest path computation.

The layout algorithm may place vertices too close together, especially in cases with dense connections. To relax such layout results, our technique forms a triangular mesh connecting the set of vertices by applying the incremental Delaunay triangulation algorithm, and it moves the vertices by applying a mesh-smoothing algorithm. The movement of the ith vertex, $\mathbf{mov}_{v_i}$, is calculated by the following equation while repeating the mesh-smoothing algorithm:

$$\begin{aligned}
\mathbf{mov}_{v_i} &= len_1 \frac{\mathbf{sum}_{v_i}}{\left|\mathbf{sum}_{v_i}\right|} \\
\mathbf{sum}_{v_i} &= \sum_k \mathbf{dist}_{ik} \\
\mathbf{dist}_{ik} &= 0 \left(\text{if } \left|\mathbf{v}_i - \mathbf{v}_k\right| \geq len_2\right) \\
\mathbf{dist}_{ik} &= len_2 \frac{\mathbf{v}_i - \mathbf{v}_k}{\left|\mathbf{v}_i - \mathbf{v}_k\right|} \left(\text{if } \left|\mathbf{v}_i - \mathbf{v}_k\right| < len_2\right). \quad (8)
\end{aligned}$$

Here, len_1 is a constant value that keeps the length of the movement vector constant, $\boldsymbol{v}_i$ is the position of the ith vertex, $\boldsymbol{v}_k$ is the position of a vertex connected to $\boldsymbol{v}_i$ by a bundle, and len_2 is the preferable distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_k$. This algorithm attempts to keep constant distances between adjacent vertices to avoid a too-dense layout, while it does not attract far vertices.

Graph Rendering

After calculating the positions of vertices corresponding to the clusters, the proposed technique assigns positions to nodes in each cluster. Our current implementation places nodes simply on the adequate number of concentric circles around the position of the cluster.

It then applies a node swapping algorithm to reduce the sum of edge lengths. The algorithm initially lets all the nodes in a cluster be unfixed. It selects an unfixed node and calculates the sum of edge lengths for all edges connecting to the node, while traversing the positions of all unfixed nodes in the cluster. Once the best position is found, the node's current position is swapped with the best position. The process repeats for all the nodes in the cluster.

After the node placement, we apply an edge-bundling algorithm between pairs of clusters. Our implementation applies Bézier curves between pairs of nodes setting four control points, where two of them are placed at the position of two nodes and others are placed between them. This design is close to Holten's hierarchical edge bundling that applied B-Spline curves.[3] However, our design is simpler because we do not need to consider the variety of depths of hierarchy.

Our implementation assigns colors of nodes based on one of the following procedures.

For the distribution of feature vector values, we assign node colors according to the feature vector values to represent the relevancy among the connections and features. We can assign a color based on all individual values per dimension, the variation of values, or just on the dominating dimension. Our current implementation simply assigns colors to each of the feature vector's dimensions and selects the node color based on the color of the dimension that has the largest value in the node's feature vector.

We also assign node colors according to their centrality.[4] Our current implementation simply calculates colors based on their degree, assigning gray to low-degree nodes and red to high-degree nodes. In later versions, we would like to apply a variety of centrality metrics to calculate node colors.

Experiments

To test the proposed approach, we performed experiments using two datasets. The two datasets we used to apply our technique contain coauthorship data (dataset A) and Twitter communication data (dataset B). Our motivation was to explore the relationships between the key people and their adjacent relations. Our results demonstrate that our technique can show the edges between key nodes and groups of their adjacent nodes.

Dataset A was created based on a publication bibliography from the NERC Biomolecular Analysis Facility (NBAF, http://nbaf.nerc.ac.uk). We downloaded the full NBAF bibliography for the years 1998 to 2013, for a total of 1,821 authors and 564 paper titles. We constructed a graph that models the paper coauthorship that consists of

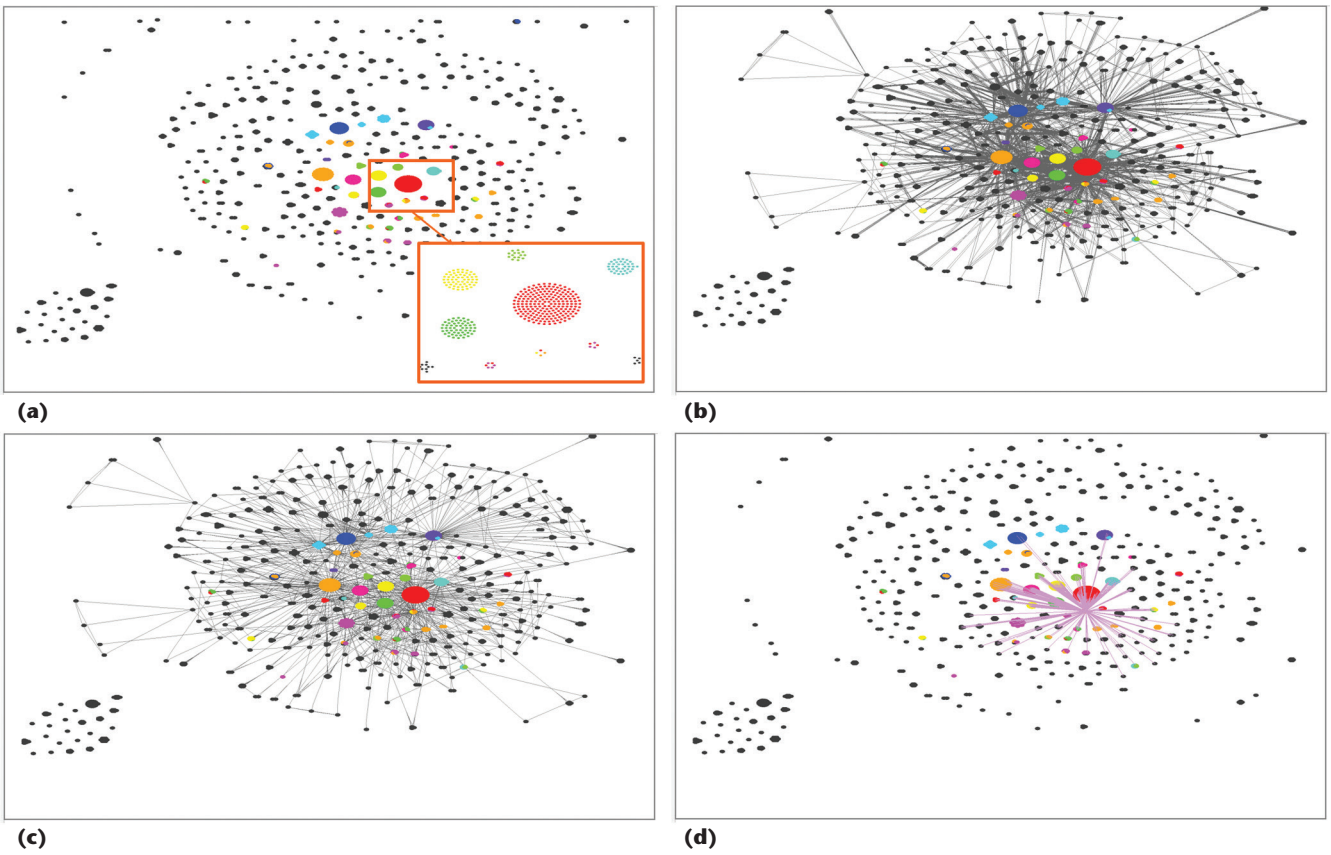


Figure 2. Cluster layout and edge-bundling examples. (a) In the cluster layout, the rectangular regions are manually overlaid to show zooming views. (b) All the edges without bundling. (c) All the edges with bundling. (d) The user can highlight the edges connected to the particular node by clicking on it.

1,821 nodes and 11,097 coauthorship edges, where each node has a 12-dimensional feature vector.

To construct the feature vectors, we first counted the frequency of terms in all the paper titles and selected 20 terms. We then counted the frequency of the selected terms in the paper titles for each author and specified the top term as the corresponding term for an author. We counted the number of authors for each term and finally selected the 12 terms covering a large number of corresponding authors. We counted the frequency of the selected 12 terms in the paper titles again and treated the frequency values as an author's 12-dimensional feature vector. The selected terms were *genetic* (red), *molecular* (orange), *loci* (yellow), *microsatellites* (yellowish green), *isolation* (green), *inbreeding* (bluish green), *transcriptomics* (sky blue), *expression* (blue), *bacterial* (indigo), *breeding* (purple), and *polymorphic* (pink).

Dataset B was published on the NodeXL Graph Gallery, which records Twitter users with tweets that are related to food security or who were replied to or mentioned in those tweets on 17 September 2014 (www.nodexlgraphgallery.org/Pages/Graph.aspx?graphID=28286). We constructed a graph that models the communication among Twitter users, consisting of 4,973 nodes and 4,223 communication edges.

Similar to dataset A, we first counted the frequency of terms in the tweets and, from the top 30 terms, subjectively selected eight important keywords: Somalia, Africa, Ebola, India, Mubasher Lucman, PTI, and AMP. We counted the frequency of the selected eight terms again and treated the frequency values as a Twitter user's eight-dimensional feature vector.

For all the examples in the figures, we set α in Equation 2 to 0.5.

Clustering and Layout Result Examples

We first show the clustering and layout results for dataset A. Two of the nodes in the graph have an extremely large number of connections, connecting to 527 and 412 edges, respectively. We identified these as the key nodes.

Figure 2 shows a cluster layout and edge-bundling example. The nodes are colored according to their feature vector values. The edge-bundling algorithm in our implementation effectively reduces the cluttering among the nodes and edges. Users can click on the edges of the specified nodes to focus on the connectivity of particular important nodes.

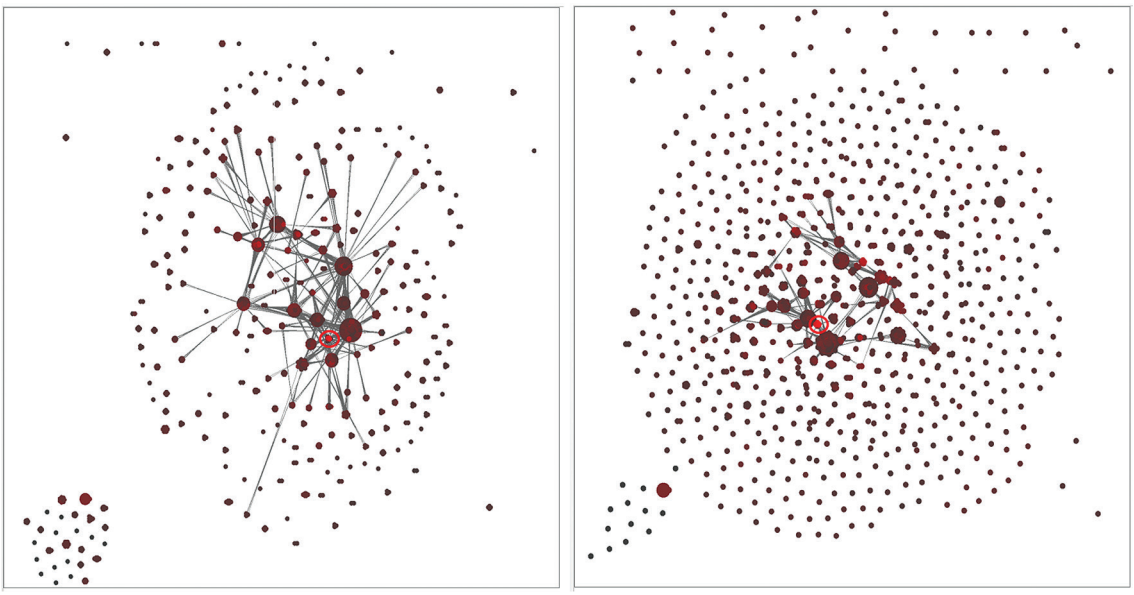


Figure 3. Clustering and layout result while adjusting the threshold for cluster sizes. The red circles are manually drawn to show that key nodes connected to an extremely large number of other nodes are contained in the small clusters. (a) Divided into 813 clusters. (b) Divided into 264 clusters.

Similarly colored nodes are more concentrated in the same clusters if we use a larger α value, whereas more key nodes tend to be outside the large clusters if we use a smaller α value. We repeated the same processes 10 times with the same dataset to measure the computation time. On average, the technique required 2.496 seconds for graph clustering, 7.343 seconds for stress-minimizing layout, and 1.464 seconds for mesh generation and smoothing.

Figure 3 shows the layout of clusters, where nodes are colored based on centrality, while adjusting the threshold for cluster sizes. In this example, the key nodes were not swallowed by large clusters, but the small clusters are enclosed by red circles.

Figure 4 compares visualization results from our technique with those of other common techniques using the same dataset and dividing the nodes into approximately the same number of clusters. Figure 4a is a result of our technique, where the two key nodes belonged to the same small cluster indicated by a red circle. Figure 4b shows the result of applying a common clustering algorithm for comparison (modularity clustering in this example), where the two key nodes belonged to larger clusters indicated by red circles. Figures 4c and 4d compare the visibility of edges connected to the specified node. In Figure 4c, most of edges are connected to nodes in other clusters so the edges are more comprehensive. Most of edges in Figure 4d, on the other hand, are connected to nodes in the same cluster so it is difficult to observe the structure around the specified key node.

Table 1 shows the clustering results, where the number of edges inside the clusters indicates the edges connecting the nodes belonging to the same cluster. These results demonstrate that our technique successfully divided key nodes from large clusters. We reduced the number of edges tangled inside the cluster and therefore effectively showed the connectivity between the key nodes and others. This visualization represents the publication situation in the area well, judging by the research community information.

Figure 5 shows the graph-clustering and layout results of our technique and the common technique for dataset B. The nodes are colored based on centrality in the figure. The dataset contains five users that have an extremely large number of communications with other Twitter users, as indicated by the red circles in the results. Figure 5a demonstrates that our technique separated such key nodes from large clusters. Also, the result effectively shows that the key nodes connected the large clusters and other portions of the graph. When we applied a common clustering algorithm, on the other hand, the key nodes were swallowed by large clusters (see Figure 5b).

Table 2 shows the clustering results for dataset B. Our technique drastically reduced the number of edges inside the clusters; where the results of our technique contained 296 inside edges, the results of using the common technique contained 6,668 inside edges.

Case Study: Coauthorship Network

Figure 6a shows that many authors who have common coauthors are associated with the common

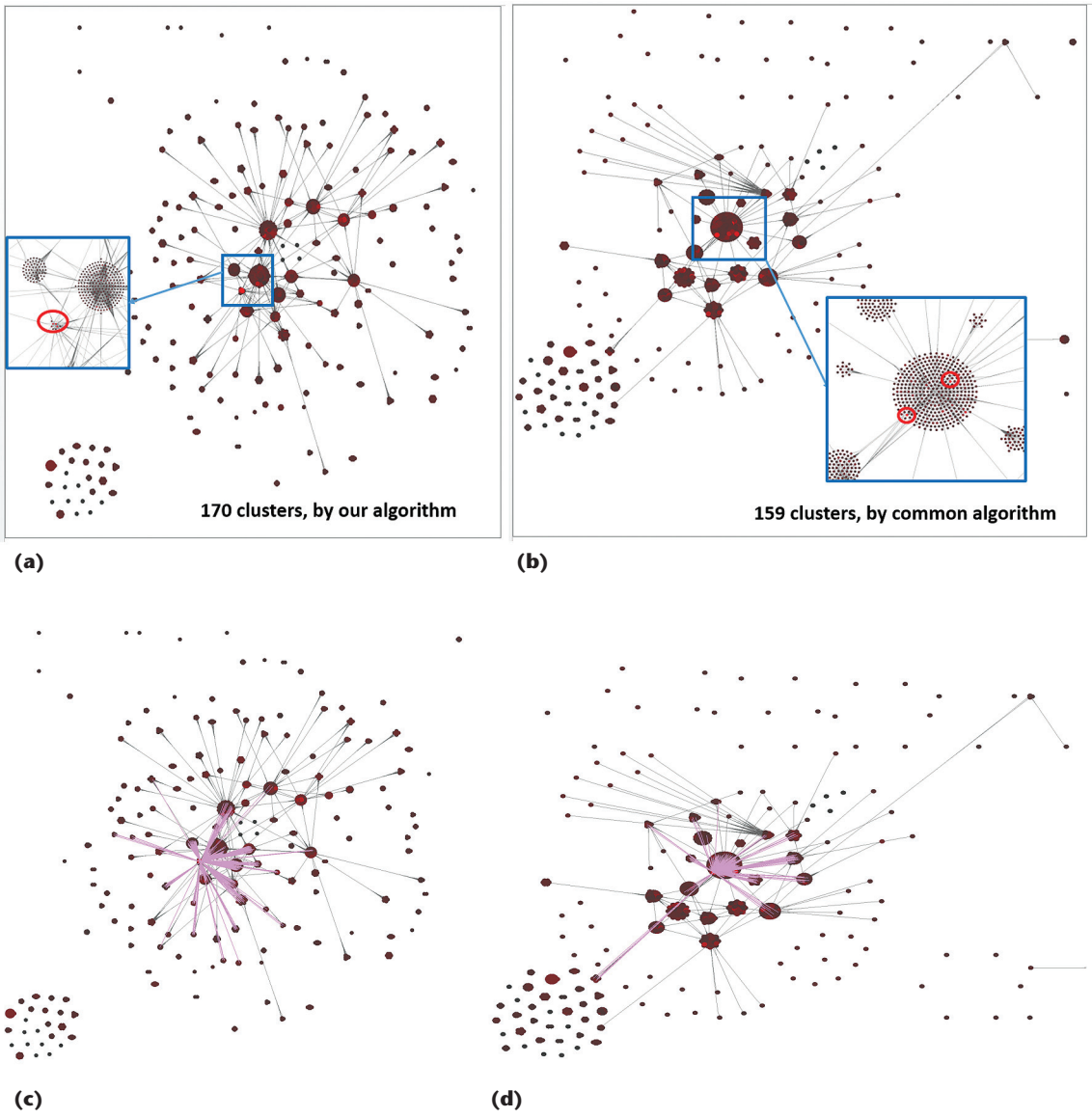


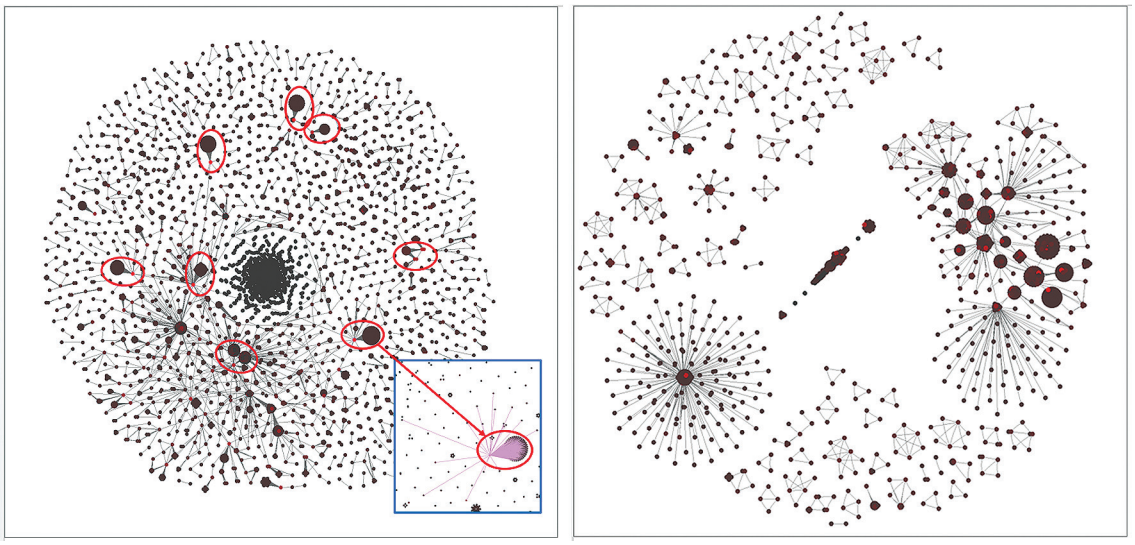
Figure 4. Visualization results for the coauthorship dataset. The rectangular regions are manually overlaid to show zooming views. Red circles are manually drawn to indicate key nodes. (a) Result of our technique. Two key nodes connected to an extremely large number of other nodes remain in the small cluster. (b) Result of applying a common clustering algorithm. The key nodes are enclosed by a large cluster as indicated by red circles. (c) One of the key nodes visualized by our technique is clicked. Most of the edges are connected to nodes in other clusters so the edges are more comprehensible. (d) The key node visualized by a common technique is clicked. Most of the edges are connected to nodes in the same cluster so it is difficult to observe the structure around the clicked key node.

technical terms, and consequently they formed large clusters consisting of a single color. We then changed the nodes' colors to indicate their centrality, as shown in Figure 6b. We clicked several key nodes placed in the small clusters in bright red and observed the coauthorship of such important authors. The clusters in Figures 6c through 6f show the connections of four important authors.

The examples in Figures 6c and 6d represent the connections of important authors T. Burke and D.A. Dawson, respectively, who belong to

Table 1. Clustering results while adjusting the threshold for cluster sizes using dataset A (coauthorship).

Method	Number of clusters	Number of nodes	Number of edges inside the clusters
Our technique (Figure 3a)	813	4.4	5,946
Our technique (Figure 6a)	354	4.4	5,421
Our technique (Figure 3b)	264	4.4	5,868
Our technique (Figure 4a)	170	9.9	6,141
Common technique (Figure 4b)	159	33.54	8,214



(a) (b)

Figure 5. Graph clustering and layout of the Twitter communication dataset. The rectangular regions are manually overlaid to show zooming views. The red circles are manually drawn to indicate key nodes. (a) Our technique separates the key nodes, painted in red, from large clusters. We can observe that key nodes connect to large clusters or bridge many clusters. (b) In the clustering from the common technique, the key nodes are hidden in large clusters, making it difficult to see the connections between the key nodes and other nodes.

Table 2. Clustering results while adjusting the threshold for cluster sizes using dataset B (Twitter communication).

Method	Number of clusters	Number of edges inside the clusters
Our technique (Figure 5a)	2,117	296
Common technique (Figure 5b)	2,076	6,668

the same cluster and have similar coauthorships. We compared the small differences between the connections of these two authors. The first one had stronger connections with a few clusters of particular terms, such as *genetic*. The second one had relatively wide connections and had connections with the clusters of terms *loci*, *inbreeding*, and *isolation*, which the first author did not. Our technique reduces the edges inside the clusters, as Tables 1 and 2 demonstrate. Thus, the edges connecting to the specified nodes are more visible in these visualization results than in the results from the common technique.

The examples in Figures 6e and 6f represent the important authors A.R. Cossins and J.K. Chipman, respectively. The former author had connections with the clusters of terms *molecular*, *expression*, and *bacterial*, with which the first and second authors did not have strong connections. We found that A.R. Cossins was a key person, but one with a different specialty than T. Burke and D.A. Dawson. J.K. Chipman had connections with clusters of terms *transcriptomics* and *expression* and many other clusters consisting of uncolored nodes. This visualization suggests that we may need to focus on several other terms to understand J.K. Chipman's specialty and connections. Nevertheless, our technique makes the edges connecting to the specified nodes more visible.

User Evaluation

We conducted a subjective user evaluation with 13 university student participants majoring in computer science, not limited to graph drawing or information visualization. We showed the participants the two sets of visualization results in Figures 4 and 5 and asked them to compare our results with those of the common techniques. We also asked them to answer the following questions.

1. Which result do you feel makes it is easier to find key nodes?
2. With which results are you interested in the key nodes and want to click them?
3. Which result better helps you to find the number of clusters connected to the key nodes?
4. Which result better helps you to understand how many nodes are connected to the key nodes?

Table 3 shows the participants' answers when we asked them to select the technique with the better result. The results indicate that many participants rated our technique better when attempting to explore the connections between key nodes, even though many of them felt it was easier to find key nodes when using the common technique. Several participants mentioned that it was easier to focus

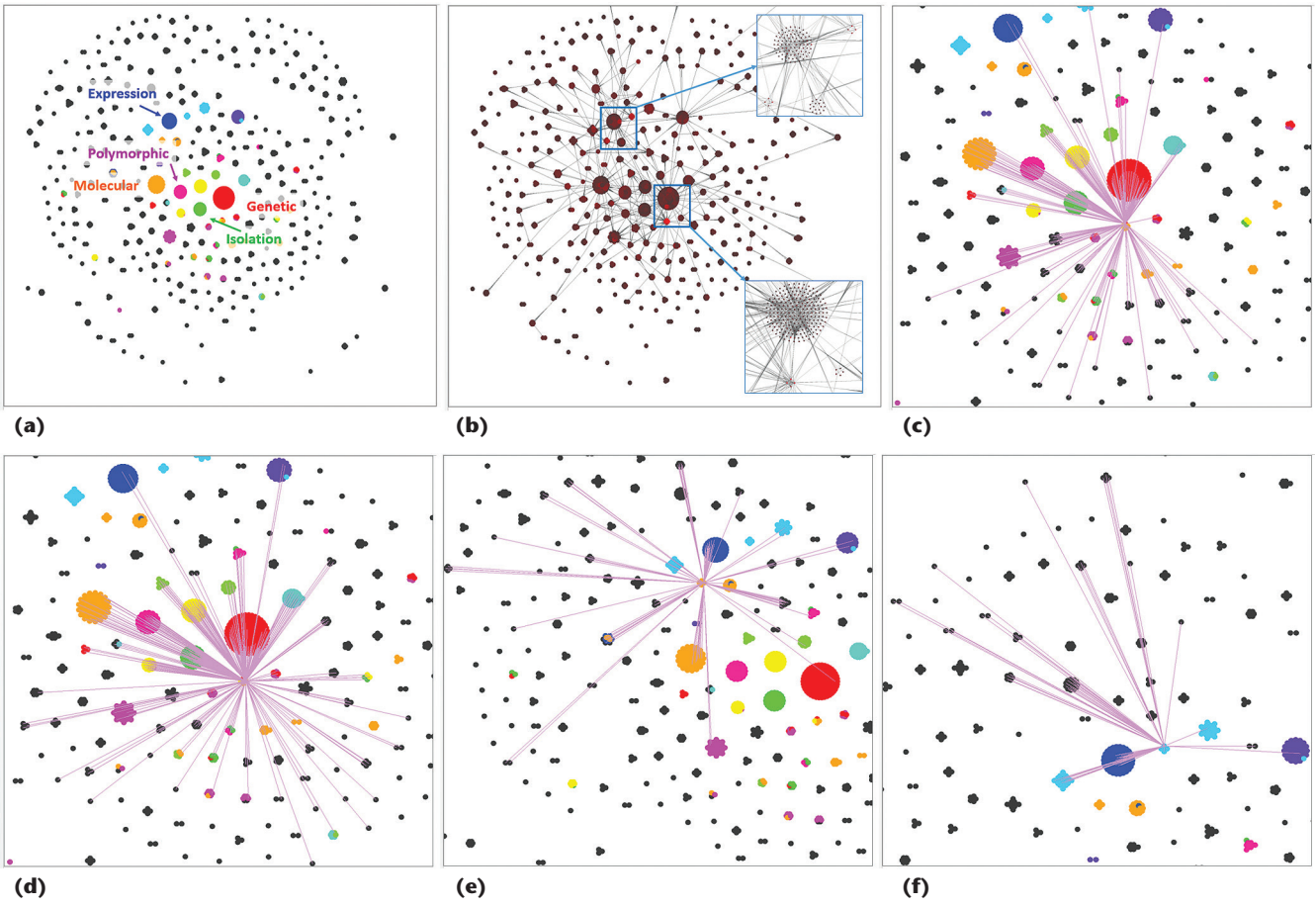


Figure 6. Case study with a coauthorship network dataset. The rectangular regions are manually overlaid to show zooming views. (a) Cluster layout. (b) Importance representation. Important nodes are placed in the small clusters indicated by red circles. Coauthorship of four important people: (c) T. Burke, (d) D.A. Dawson, (e) A.R. Cossins, and (f) J.K. Chipman.

on larger clusters in the visualization results, so it was also easier to find key nodes in the large clusters. They also mentioned that the key nodes visualized by our technique were more interesting because it was easier to determine how they are connected to other clusters. Several participants also mentioned that it was difficult to understand how many nodes were connected to key nodes using the common technique because many of the edges connecting the key nodes were inside a large cluster. These results suggest that our technique can better highlight the key node connections than the common technique.

Table 3. Subjective evaluation results with datasets A and B.*

	Dataset A		Dataset B	
Question	Our technique	Common technique	Our technique	Common technique
1	6	7	2	11
2	9	4	10	3
3	9	4	7	6
4	10	3	8	5

**Numbers in this table denote the number of participants who selected the corresponding technique in the response to our four questions.*

The way we use attribute values in this work (with coloring, for example) is biased toward the largest value in the feature vector, even when it does not dominate the others. Two vectors may have entries that are similar or largely differing, yet the same value might be picked, which is not always desirable. In future work, we will investigate additional ways to use the attribute values, tailored to the application area and the task at hand. In addition, it will be interesting to explore the impact of using different measures for distances and importance, such as taking into account neighborhood structures. We have also started to investigate the value of different layout methods in our approach.

Acknowledgments

This research was partially supported by the Australian Research Council through its Discovery Project grant DP140100077.

References

1. S.E. Schaeffer, "Graph Clustering," *Computer Science Rev.*, vol. 1, no. 1, 2007, pp. 27–64.
2. U. Brandes and C. Pich, "Eigensolver Methods for Progressive Multidimensional Scaling of Large Data," *Proc. 14th Int'l Symp. Graph Drawing* (GD), 2006, pp. 42–53.
3. D. Holten, "Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, 2006, pp. 741–748.
4. C. Correa, T. Crnovrsanin, and K.-L. Ma, "Visual Reasoning about Social Networks Using Centrality Sensitivity," *IEEE Trans. Visualization and Computer Graphics*, vol. 18, no. 1, 2012, pp. 106–120.

Takayuki Itoh *is a professor in the Department of Information Sciences at Ochanomizu University, Japan. His research interests include visualization, computer graphics, and multimedia. Itoh has a PhD in engineering from Waseda University. Contact him at itot@is.ocha.ac.jp.*

Karsten Klein *is a research fellow in computer science at Monash University, where he is part of the Monash Adaptive Visualization Lab. His research interests include visual analytics, information visualization, graph drawing, and algorithm engineering. Klein has a PhD in computer science from the Technical University of Dortmund, Germany. Contact him at karsten.klein@udo.edu.*