Key-node-Separated Graph Clustering and Layout for Human Relationship Graph Visualization

Takayuki Itoh, Ochanomizu University, Japan
Karsten Klein, Monash University, Australia

Abstract

Many graph drawing methods apply node clustering techniques based on density of edges to find tightly connected subgraphs, and then hierarchically visualize the clustered graphs. On the other hand, users may want to focus on important nodes (called key nodes in this paper) and their connections to groups of other nodes while using some applications. For this requirement, it is effective to separately visualize the key nodes detected based on adjacency and attributes of the nodes. This paper presents a graph visualization technique for attribute-embedded graphs which applies a graph clustering algorithm taking into account the combination of connections and attributes. The graph clustering step divides the nodes according to the commonality of connected nodes and similarity of feature value vectors. It then calculates the distances between arbitrary pairs of clusters according to the number of connecting edges and similarity of feature value vectors, and finally places the clusters based on the distances. Consequently, the technique separates important nodes which have connections to multiple large clusters, and improves the visibility of connections of such nodes. This paper presents examples with human relationship graph datasets, including a co-authorship and a Twitter communication network dataset.

1 Introduction

Graphs are used to model relations between entities in many real-world applications. Common tasks of graph analysis include understanding of correlations between the connections and attributes of the entities, and discovery of key entities which have many connections or many particular attributes. Human relationship is a typical information for which graph visualization can be very useful for analysis purposes. People in social networks have connections, so called friendships, and the relations between subsets of people might form complex subnetworks. Understanding of correlations between the connection structures and attributes of the people, such as jobs, hometowns, and hobbies, may contribute to various social and business analytics. Also, discovery of key persons and analysis of their positions and roles in a network may contribute to understand how information spreads in social networks.

Often, nodes or substructures of a graph are aggregated for analysis and visualization based on either the structure or attributes, for example by hierarchical clustering. Various techniques on hierarchical graph visualization have been developed for such graph analysis tasks, because a hierarchical structure is useful for overview of and quick understanding of the graph structures, and top-down operations for graph exploration and navigation. Graph clustering algorithms are often applied before applying hierarchical graph visualization. Many of them treat tightly connected subregions of the graphs as clusters; however, that does not always work well for discovery of key nodes and tight connections. Figure 1 illustrates the problem of graph clustering. The red nodes in Figure 1(a) are connected to a large number of nodes, and therefore they are certainly important nodes in this graph. However, they may belong to the 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 very visible if the high-level structure of the clustered graph is drawn as Figure 1(b). Also, many edges between blue and red nodes are not visible because they are drawn in a small region corresponding to the cluster (1). Users may miss to find these important connections while looking at the high-level structure. On the other hand, the red nodes will be separated from this cluster if commonality of neighbor nodes or similarity of attributes are applied as criteria of the clustering process. If 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, while two red nodes form another cluster because they are connected to all colors of nodes. Or, if colors of the nodes represent their attributes, the blue and red nodes can form separate clusters. Figure 1(c) illustrates the result. A set of connections between blue and red nodes can be emphatically displayed by a thick bundle connecting the two clusters. In this case, the red node will be still visible when the high-level structure of the graph is drawn as Figure 1(d). This structure is preferable for the following reasons:

- Connectivity with important nodes and multiple clusters will be comprehensive in the visualization results.
- Less edges will be tangled in the small regions of display spaces corresponding to clusters.
- Edge bundling algorithms can be easily applied

to large number of edges connecting to important nodes such as the red node in Figure 1(d).

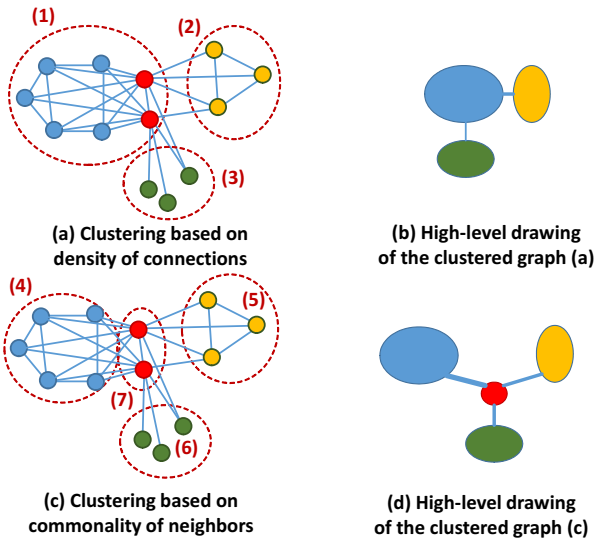


Figure 1: Clustering based on density of connections or commonality of neighbors.

Key node finding is useful in many real world applications. Development of key-node-conscious graph visualization techniques is therefore expected to be an important contribution in many academic and business fields. Moreover, additional information such as attributes of nodes have to be taken into account in many applications. In case of human relationship graphs where nodes correspond to humans, we often need to take into account preferences or characters of humans as attributes of nodes. Their preferences or characters can be often converted to numeric values by recent analysis techniques, including natural language processing with the documents written by the humans. Graph visualization featuring key-node-identification combining 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 paper 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 similarity of feature vectors as well as commonality of neighbor nodes, and 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 by triangular mesh smoothing algorithm, where the mesh is generated by connecting centers of clusters by 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.

2 Related Work

Graph clustering partitions the nodes of a graph into clusters, i.e. node sets that are usually disjoint, with the goal of having many intra-cluster edges and few inter-cluster edges. Graph clustering is useful not only for visualization but also various analytic purposes, and actually large number of algorithms have been presented, as introduced by Schaeffer [10] or several other survey papers.

The most popular approach for graph clustering is extraction of tightly connected subgroups of nodes. Many graph clustering techniques using this approach are based on agglomerative clustering algorithms. Meanwhile, Schaeffer's survey [10] also points that there are various criteria to measure the distances among the nodes of graphs, including similarity of feature value vectors in the multidimensional Euclidean spaces, adjacency-based node similarity measurements, in addition to the tightness of the connections which can be measured as number of paths between two nodes. The presented technique calculates distances among nodes as a combination of feature value vector distances and adjacency-based dissimilarities.

Hierarchically clustered graphs are an effective data structure for information visualization because they are suitable for overview, zoom, and filtering operations. Node layout is definitely 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 Eades proposed his seminal spring embedder model [4]. Better alternatives include multilevel approaches and recent advances in stress-based layout computation [5], which basically apply a multidimensional scaling. Meanwhile, several techniques [3] [8] efficiently visualize hierarchical graphs by applying space-filling layout techniques such as Treemaps.

Edge bundling has been an active topic since Holten [6] presented a technique which hierarchically bundles edges connecting nodes of tree structure level-by-level, and draws them as Spline curves. Edge bundling is especially effective coupling with the presented technique, because it clusters nodes based on

the commonality of adjacent nodes, and consequently it constructs hierarchy of the nodes so that large number of edges can be bundled.

It is often interesting to emphasize important nodes in graphs. Ohsawa et al. [9] presented the KeyGraph concept which features nodes bridging multiple subgraphs. Correa et al. [2] applied various schemes to calculate centrality sensitivity to effectively visualize social networks. Our study presented in this paper takes into account the combination of neighborhoods and attributes of nodes differently from the above techniques. Meanwhile, several graph drawing techniques are suitable for key-node conscious drawing: for example, bipartite graphs. Hong et al. [7] visualized citation networks by dividing most cited papers and others as different types of nodes and applying a bipartite graph drawing technique, to focus on the connections between important nodes and the others. Our technique presented in this paper also aims to focus on such connections; however, the technique can be combined with a variety of graph layout algorithms.

Multivariate node attribute is also often taken into account during the graph layout processes. Shneiderman et al. [12] presented an implementation and a case study of graph visualization applying a metaphor of substrates, which specifies the positions of nodes based on their attribute values. Shi et al. [11] presented a graph clustering scheme which applies attribute-based partitioning for higher-level nodes and topology-based partitioning for lower-level nodes. Our technique also takes into account both attributes and topology; however, our technique is different from it since ours applies attribute-based and adjacency-based metrics simultaneously in a single clustering process.

3 Presented Technique

This section proposes the presented graph visualization technique and its components. This includes the clustering and layout methods, rendering, as well as the interaction concept.

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

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

Here, n_i denotes the i-th node, e_i denotes the i-th edges, a_{ij} denotes the value of the j-th dimension of the i-th node, N_n denotes the number of nodes, N_e denotes the number of edges, and N_a denotes the number of dimensions.

The presented 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 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 applying a Bézier-curve-based edge bundling. Our current implementation specifies the color of a node based on the dimension which 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 colors of the nodes.

The following sections describe technical components of the presented technique.

3.2 Graph Clustering

The presented technique calculates two types of distances between all pairs of nodes: dissimilarity of feature vectors d_{vec}, and discommonality of the adjacent nodes d_{adj}. It defines the distance between a pair of nodes as

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

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.

Dissimilarity of feature vectors

The 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}, by the following equation.

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

Discommonality of adjacent nodes

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, the technique counts the number of nodes which are connected to both n_i and n_j. It simply calculates the distance as follows.

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

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

3.3 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

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

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.

As our approach is based on the calculation of distances between nodes, a distance-based layout method is a natural choice to calculate the layout for the cluster graph. In our implementation, we apply the stress minimization layout algorithm from the Open Graph Drawing Framework OGDF [1] 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 as described in [1], the algorithm iteratively minimizes a stress function on the nodes' positions. Stress in the layout is defined here as

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

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 to long range errors. p_u is the position of node u in the layout, and d_{uv} given ideal distances between u and v. In graph layout algorithms, these distances are usually set to the graph theoretic distance between the nodes. In order 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 would also like to take into account the cluster graph topology, i.e. the connectivity between clusters. To this end, our current implementation defines a weight w for each bundle as

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

where w_{bun} is proportional to the number of edges belonging to the bundle. w_{vec} is calculated from inner products of the average vector values of the two clusters, similar to the 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, i.e. 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.

It may happen that the layout algorithm places vertices too closely especially due to the dense connections. To relax such layout results, the technique then forms a triangular mesh connecting the set of vertices by applying the incremental Delaunay triangulation algorithm, and moves the vertices by applying a mesh smoothing algorithm. The movement of the i-th vertex, $\mathbf{mov}_{v_i}$, is calculated by the following equation while repeating the mesh smoothing algorithm.

$$\mathbf{mov}_{v_i} = len_1 \frac{\mathbf{sum}_{v_i}}{|\mathbf{sum}_{v_i}|}$$
$$\mathbf{sum}_{v_i} = \sum_k \mathbf{dist}_{ik}$$
$$\mathbf{dist}_{ik} = 0 \ (if |\mathbf{v}_i - \mathbf{v}_k| \geq len_2)$$
$$\mathbf{dist}_{ik} = len_2 \ \frac{\mathbf{v}_i - \mathbf{v}_k}{|\mathbf{v}_i - \mathbf{v}_k|} (if |\mathbf{v}_i - \mathbf{v}_k| < len_2) \quad (8)$$

Here, len_1 is a constant value to keep the length of the movement vector constant. $\mathbf{v}_i$ is the position of the

[1] http://www.ogdf.net

i-th vertex. $\mathbf{v}_k$ is the position of a vertex connected to $\mathbf{v}_i$ by a bundle. len_2 is the preferable distance between $\mathbf{v}_i$ and $\mathbf{v}_k$. This algorithm attempts to keep constant distances between adjacent vertices to avoid a too dense layout, while it does not attract far vertices.

3.4 Graph Rendering

After calculating the positions of vertices corresponding to the clusters, the technique assigns positions to nodes in each of the clusters. Our current implementation just places nodes 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 unfixed. It selects an unfixed node, and calculates the sum of edges 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 process swaps the position of the current node as the best position and fixes it. The process repeats one-by-one 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 [6] that applied B-Spline curves. 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 either of the following procedure.

Distribution of feature vector values. The technique assigns node colors according to the feature vector values, to represent the relevancy among the connections and features. One could assign either a color depending 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 dimensions of the feature vector, and selects the color of a node as the color of the dimension which has the largest value in the feature vector of the node.

Centrality. The technique also assigns node colors according to their centrality [2]. Our current implementation just calculates colors based on their degree: it assigns gray to low-degree nodes, and red to high-degree nodes. We would like to apply a variety of centrality metrics to calculate colors of nodes in later versions.

4 Experiments

This section introduces our experiments with two datasets. We set α in equation (2) to 0.5 for all examples shown in the figures.

4.1 Data

We applied our technique to the following datasets. Our motivation to apply our method is to visually find key persons, and explore the relationships between the key persons and their adjacent persons. This section demonstrates that our technique is able to show the edges between key nodes and groups of their adjacent nodes.

Set A: Co-authorship data. This dataset was created based on a publication bibliography from the NERC Biomolecular Analysis Facility (NBAF) [2]. We downloaded the full NBAF bibliography for the years 1998-2013, which contains 1821 authors and 564 paper titles. We constructed a graph that models the paper co-authorship, consisting of 1821 nodes and 11097 co-authorship edges, where each node has a 12-dimensional feature vector.

To construct the feature vectors, we firstly counted the frequency of terms in all the paper titles, and selected 20 terms from them. We then counted the frequency of the selected terms in the paper titles for each author, and specified the top one term as the corresponding term of 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 the 12-dimensional feature vector of an author. The selected terms were as follows: *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).

Set B: Twitter communication data. This dataset was published on the NodeXL Graph Gallery, which records Twitter users whose tweets are related to food security, or who were replied to or mentioned in those tweets, on 17 September 2014 [3]. We constructed a graph that models the communication among Twitter users, consisting of 4973 nodes and 4223 communication edges.

[2] `http://nbaf.nerc.ac.uk/support/publications`

[3] `http://www.nodexlgraphgallery.org/Pages/Graph.aspx?graphID=28286`

Similar to the set A, we firstly counted the frequency of terms in their tweets, and subjectively selected the 8 important keywords, *Somalia*, *Africa*, *Ebola*, *India*, *Mubasher_lucman*, *PTI*, and *AMP*, from the top 30 terms. We counted the frequency of the selected 8 terms again, and treated the frequency values as the 8-dimensional feature vector of a Twitter user.

4.2 Examples of Clustering and Layout Results

This section first presents several clustering and layout results using the dataset A. Two of the nodes in the graph have an extremely large number of connections; they are connected to 527 and 412 edges, respectively. We will call these nodes *key nodes* in the following.

Figure 2 shows an example of cluster layout and edge bundling. Nodes are colored according to their feature vector values. Edge bundling algorithm in our implementation effectively reduces the cluttering among the nodes and edges. Edges of the specified nodes can be highlighted by click operations to focus on the connectivity of particular important nodes. Similarly colored nodes are more concentrated in the same clusters if we use a larger α value, while 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, respectively.

Figure 3 shows the layout of clusters, where nodes are colored based on centrality, while adjusting the threshold for sizes of clustering. The key nodes were not swallowed by large clusters, but belonged to the small clusters enclosed by red circles in this figure.

Figure 4 compares visualization results between our and common techniques applying the same dataset and dividing the nodes into approximately same numbers of clusters. Figure 4(a) is a result by our technique, where the two key nodes belonged to the same small cluster indicated by a red circle. Figure 4(b) shows the result 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 4(c) and 4(d) compare the visibility of edges connected to the specified node. In Figure (c), most of edges are connected to nodes in other clusters and therefore the edges are more comprehensive. On the other hand, most of edges shown in Figure (d) are connected to nodes in the same cluster and therefore it is difficult to observe the structure around the specified key node.

Table 1 shows the statistics of the clustering results, where $NumC$ denotes the number of clusters, $NumN$ denotes the number of nodes in the clusters which the key nodes belong to, and $NumIE$ denotes the number of edges inside the clusters (in other words, edges connecting the nodes belonging to the same cluster). This result demonstrates our technique successfully divided such key nodes from large clusters As a result, 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 from the research community information.

Next, we show the clustering and layout results using the dataset B. Figure 5 shows the comparison of graph clustering and layout results between our technique and the common comparison technique, where nodes are colored based on centrality. The dataset contains five users which have extremely large number of communications with other Twitter users, as indicated by red circles in the results. Figure 5(a) 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. On the other hand, Figure 5(b) shows that the key nodes were swallowed by large clusters when we applied a common clustering algorithm. Table 2 shows the statistics of the clustering results with the dataset B. Our technique drastically reduced the number of edges inside the clusters, where the result by our technique just contained 296 inside edges while the result by common technique contained 6668 inside edges.

4.3 Case Study: Paper Co-authorship Network

This section introduces the knowledge discovered from the visualization of the dataset A.

Figure 6 (Upper left) shows that many authors which have common co-authors are associated to the common technical terms, and consequently formed large clusters consisting of single colors. Then, we turned the colors of nodes to their centrality, as shown in Figure 6 (Upper right). We clicked several key nodes placed in the small clusters drawn in bright red, and observed the co-authorship of such important authors. Other pictures in Figure 6 show the connections of four important authors.

Here, the first and second examples shown in Figure 6 (Center left) and 6 (Center right) represent the con-

nections of important two authors, *T. Burke* and *D. A. Dawson*, belonging to the same cluster and having very similar co-authorships. We compared the small differences of 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. She had connections with the clusters of terms *Loci*, *Inbreeding*, and *Isolation*, which the first one did not have strong connections to. Our technique reduces the edges inside the clusters as demonstrated in Tables 1 and 2, and therefore the edges connecting to the specified nodes get more visible in these visualization results rather than the results by the common technique.

The third and fourth examples shown in Figure 6 (Lower left) and 6 (Lower right) represent two other important authors, *A. R. Cossins* and *J. K. Chipman*. The former author had connections with clusters of terms *Molecular*, *Expression*, and *Bacterial*, with which the first and second authors did not have strong connections. We found he was a certainly another key person who has different specialty comparing with the first and second persons. The latter author 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 the specialty of the fourth important person and his related persons. Again, the edges connecting to the specified nodes are visible thanks to the property of our technique.

4.4 User Evaluation

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

Q1: Which result do you feel is easier to find key nodes?

Q2: Which result are you interested in the key nodes and want to click them?

Q3: Which result do you feel is better to find the number of clusters connected to the key nodes?

Q4: Which result do you feel is better to understand how many nodes are connected to the key nodes?

Table 3 shows the statistic of the answers of the participants. The result denotes that many participants rated our technique to be better to explore the connection of key nodes, even though many of them felt it was easier to find key nodes by using the common technique. Several participants mentioned that it was easier to focus on larger clusters in the visualization results, and therefore it was also easier to find key nodes in the large clusters. Meanwhile, they also mentioned that the key nodes visualized by our technique were more interesting because it is easier to find how they are connected to other clusters. Several participants also mentioned that it was difficult to understand how many nodes are connected to key nodes by the common technique, because many of the edges connected the key nodes were inside the large cluster. The result suggests that our technique is better to observe the connection of the key nodes compared to the common technique.

5 Conclusions and future work

This paper presented graph clustering and layout techniques for visualization of feature vector embedded graphs, which improves the visibility of important nodes by separating them from large clusters. The paper showed the experimental result that the presented technique well separated the important nodes which have extremely large number of edges from large clusters, and reduced the number of inner cluster edges, comparing with more common clustering algorithm. The paper also introduced a case study with a co-authorship network dataset.

The way the attribute values are used, for example regarding the coloring, is biased towards the largest value in the feature vector, even when it is not dominating the others. For two vectors where the entries are quite similar and largely differing, respectively, the same value might be picked, which is not always desirable. We will investigate further approaches to use the attribute values, tailored to the application area and task at hand. In addition, it will be interesting to explore the impact of using different measures for distances and importance, e.g. taking into account neighborhood structures. We also started to investigate the value of different layout methods in our approach.

Acknowledgments This research was partly supported by the Australian Research Council through Discovery Project grant DP140100077.

References

[1] U. Brandes, C. Pich, Eigensolver Methods for Progressive Multidimensional Scaling of Large Data, *The 14th International Symposium on Graph Drawing GD*, 42–53, 2006.

[2] C. Correa, T. Crnovrsanin, K.-L. Ma, Visual Reasoning about Social Networks Using Centrality Sensitivity, *IEEE Transactions on Visualization and Computer Graphics*, 18(1), 106–120, 2012.

[3] W. Didimo, F. Montecchiani, Fast layout computation of clustered networks: Algorithmic advances and experimental analysis, *Information Sciences*, 280, 185–199, 2014.

[4] P. Eades, A Heuristics for Graph Drawing, *Congressus numerantium*, 42, 146–160, 1984.

[5] E. R. Gansner, Y, Hu, S. North, A Maxent-Stress Model for Graph Layout, *IEEE Transactions on Visualization and Computer Graphics*, 19(6), 927–940, 2013.

[6] D. Holten, Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, *IEEE Transactions On Visualization And Computer Graphics*, 12(5), 741–748, 2006.

[7] S. Hong, W. Huang, K. Misue, W. Quan, A Framework for Visual Analytics of Massive Complex Networks, *International Conference on Big Data and Smart Computing (BIGCOMP)*, 15–17, 2014.

[8] T. Itoh, C. Muelder, K.-L. Ma, J. Sese, A Hybrid Space-Filling and Force-Directed Layout Method for Visualizing Multiple-Category Graphs, *IEEE Pacific Visualization Symposium*, 121–128, 2009.

[9] Y. Ohsawa, N. E. Benson, M. Yachiba, KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor, *IEEE International Forum on Research and Technology Advances in Digital Libraries*, 1998.

[10] S. E. Schaeffer, Graph Clustering, *Comuter Schience Review*, 1(1), 27–64, 2007.

[11] L. Shi, Q. Liao, H. Tong, Y. Hu, Y. Zhao, C. Lin, Hierarchical Focus+Context Heterogeneous Network Visualization, IEEE Pacific Visualization Symposium, 89-96, 2014.

[12] B. Shneiderman, A. Aris, Network Visualization by Semantic Substrates, *IEEE Transactions on Visualization and Computer Graphics*, 12(5), 733-740, 2006.

Table 1: Clustering result while adjusting the threshold for sizes of clusters using the dataset A.

	$NumC$	$NumN$	$NumIE$
Our (Figure 3 (a))	813	4,4	5946
Our (Figure 6 (Upper right))	354	4,4	5421
Our (Figure 3 (b))	264	4,4	5868
Our (Figure 4 (a))	170	9,9	6141
Common (Figure 4 (b))	159	33,54	8214

Table 2: Clustering result while adjusting the threshold for sizes of clusters using the dataset B.

	$NumC$	$NumIE$
Our (Figure 5 (a))	2117	296
Common (Figure 5 (b))	2076	6668

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

	dataset A		dataset B	
Question	Our	Common	Our	Common
Q1	6	7	2	11
Q2	9	4	10	3
Q3	9	4	7	6
Q4	10	3	8	5

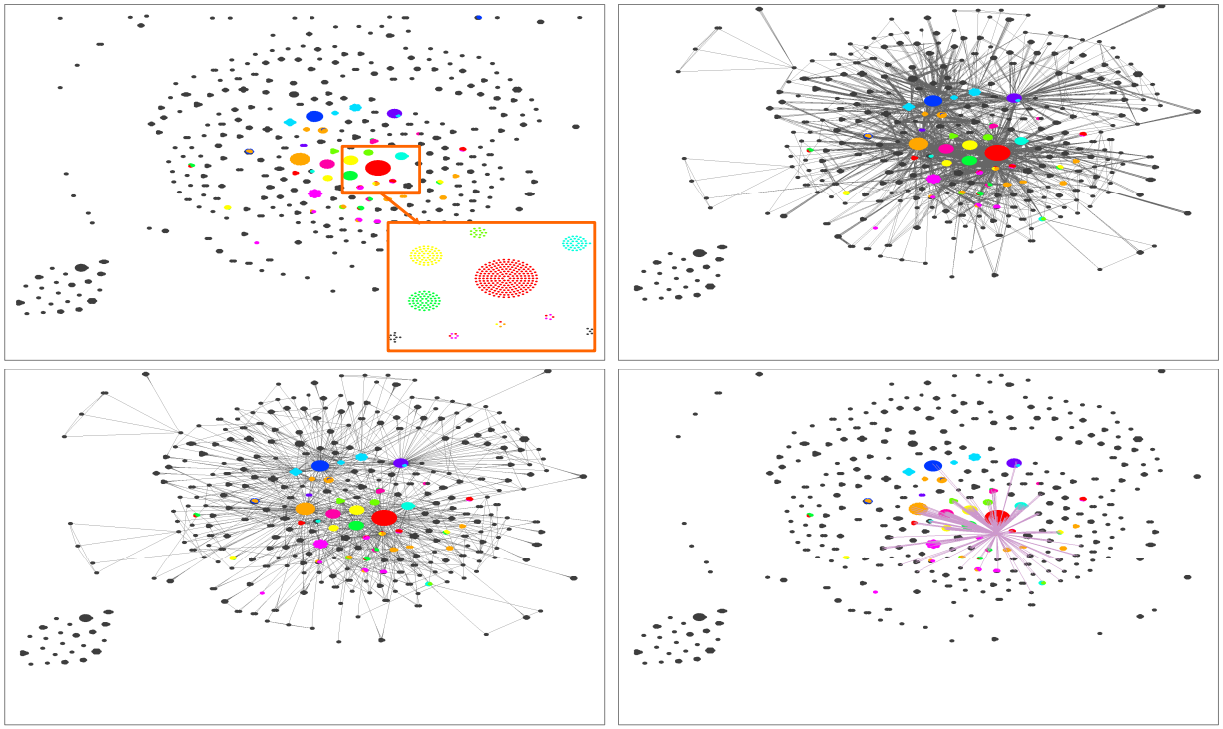
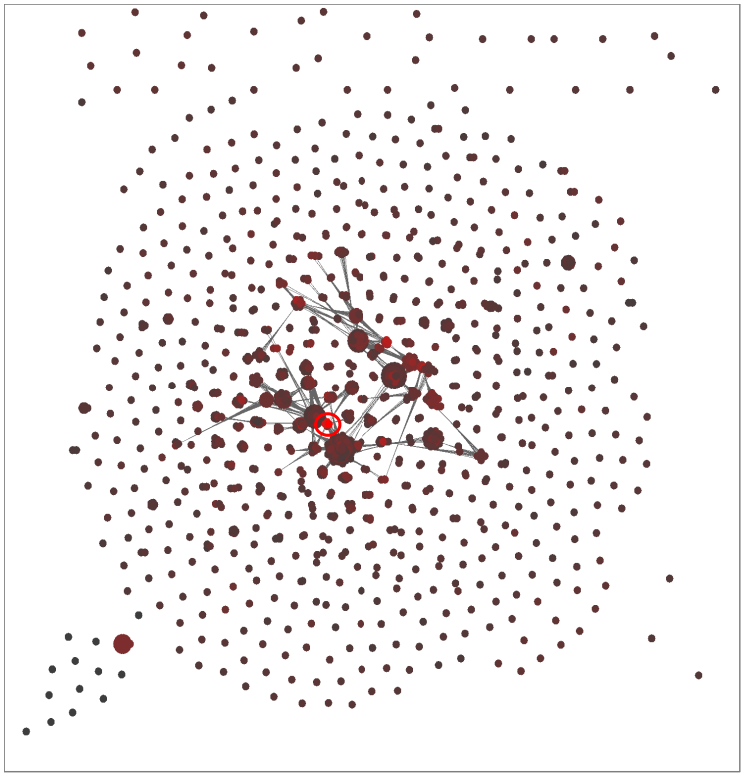
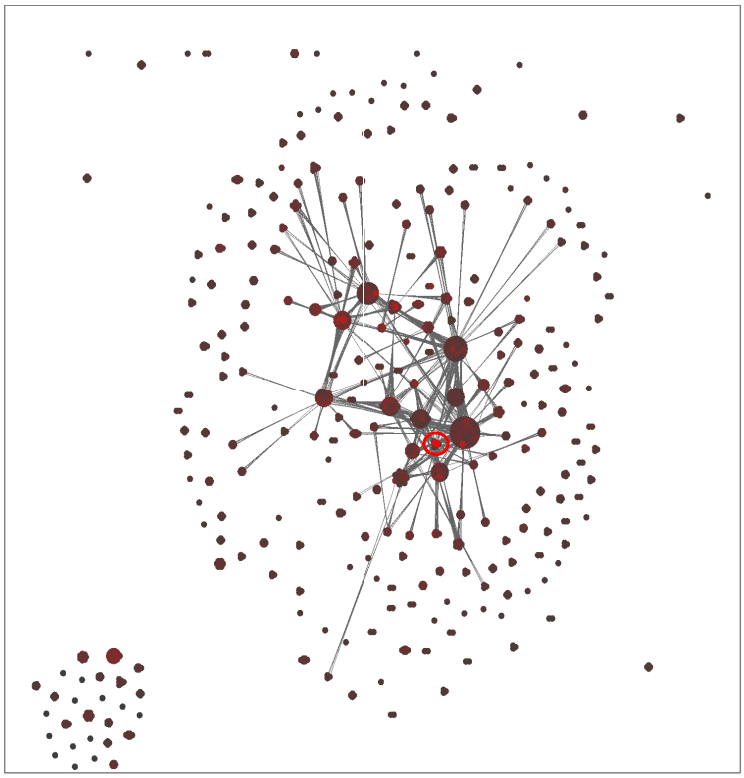


Figure 2: An example. (Upper-left) Cluster layout. Rectangular regions are manually overlaid to show zooming views. (Upper-right) Drawing all the edges without bundling. (Lower-left) Drawing all the edges with bundling. (Lower-right) Highlight the edges connected to the particular node clicked by a user.



(a)



(b)

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

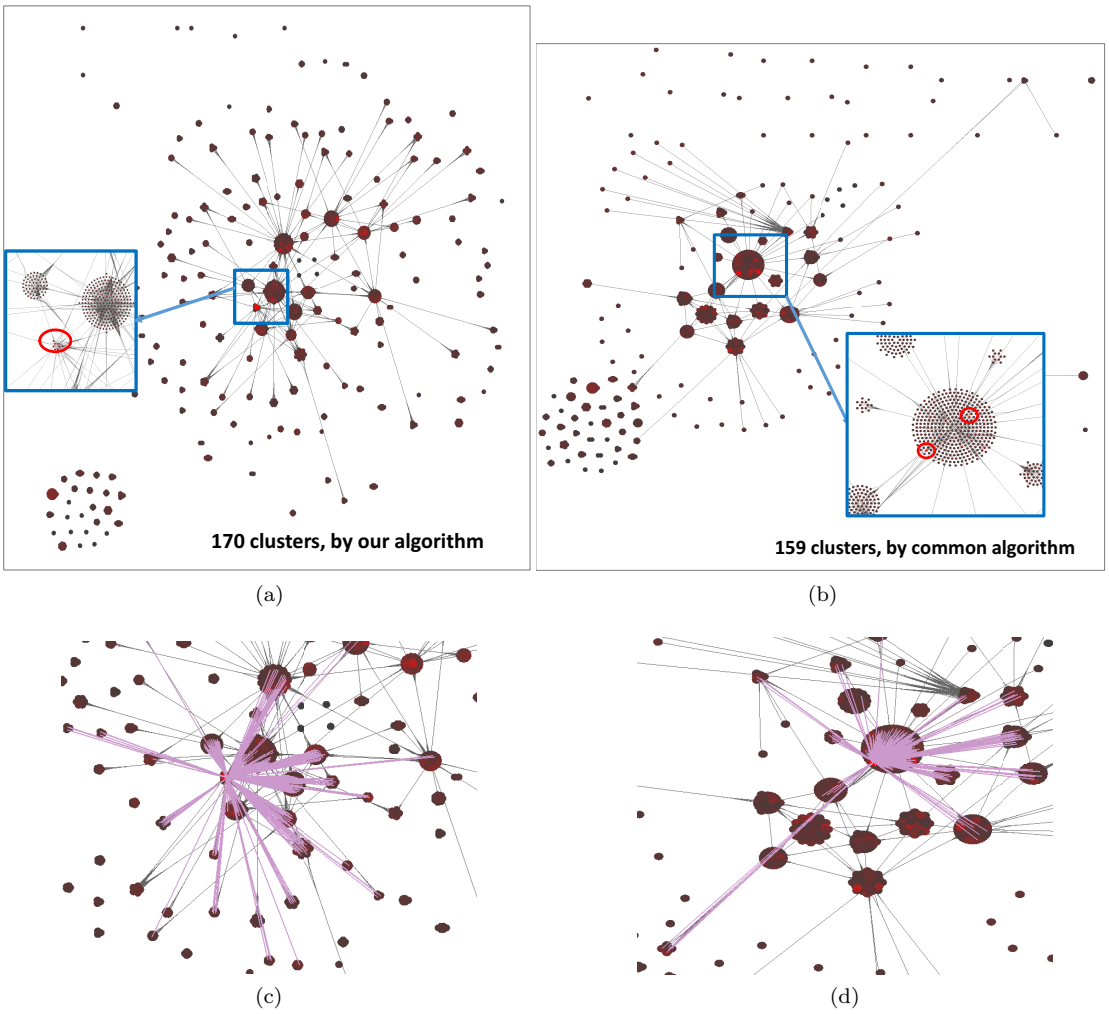
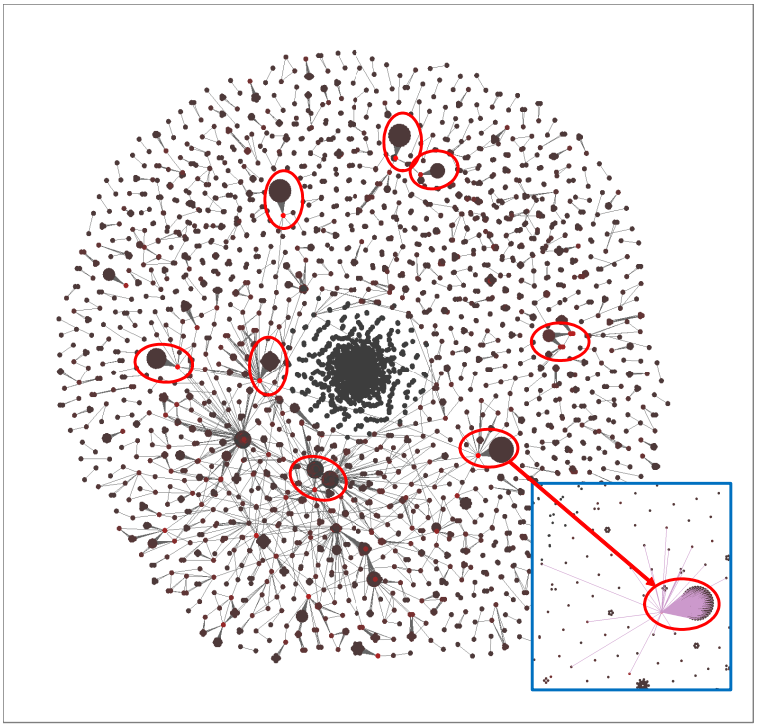
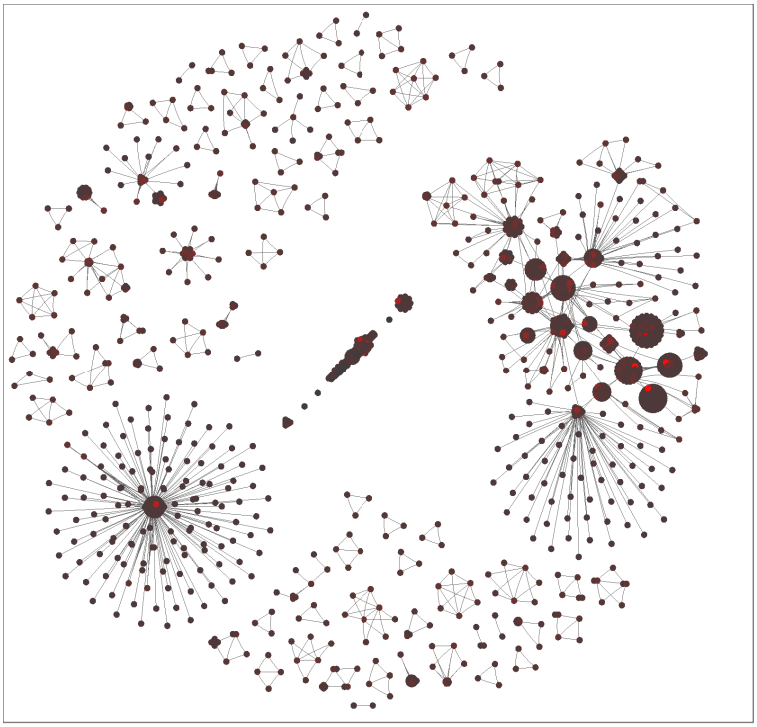


Figure 4: Comparison of visualization results between our and common techniques. Rectangular regions are manually overlaid to show zooming views. Red circles are manually drawn to indicate key nodes. (a) Result by our technique. Two key nodes connected to extremely large number of other nodes remain in the small cluster. (b) Result applying a common clustering algorithm. The key nodes are enclosed to a large cluster as indicated by red circles.. (c) One of the key nodes visualized by our technique is clicked. Most of edges are connected to nodes in other clusters and therefore the edges are more comprehensive. (d) The key node visualized by a common technique is clicked. Most of the edges are connected to nodes in the same cluster and therefore it is difficult to observe the structure around the clicked key node.



(a)



(b)

Figure 5: Comparison of graph clustering and layout of Twitter communication data. Rectangular regions are manually overlaid to show zooming views. Red circles are manually drawn to indicate key nodes. (a) Our technique well 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) Clustering by a common technique. The key nodes are hidden in large clusters. It is difficult to observe the connection between the key nodes and others.

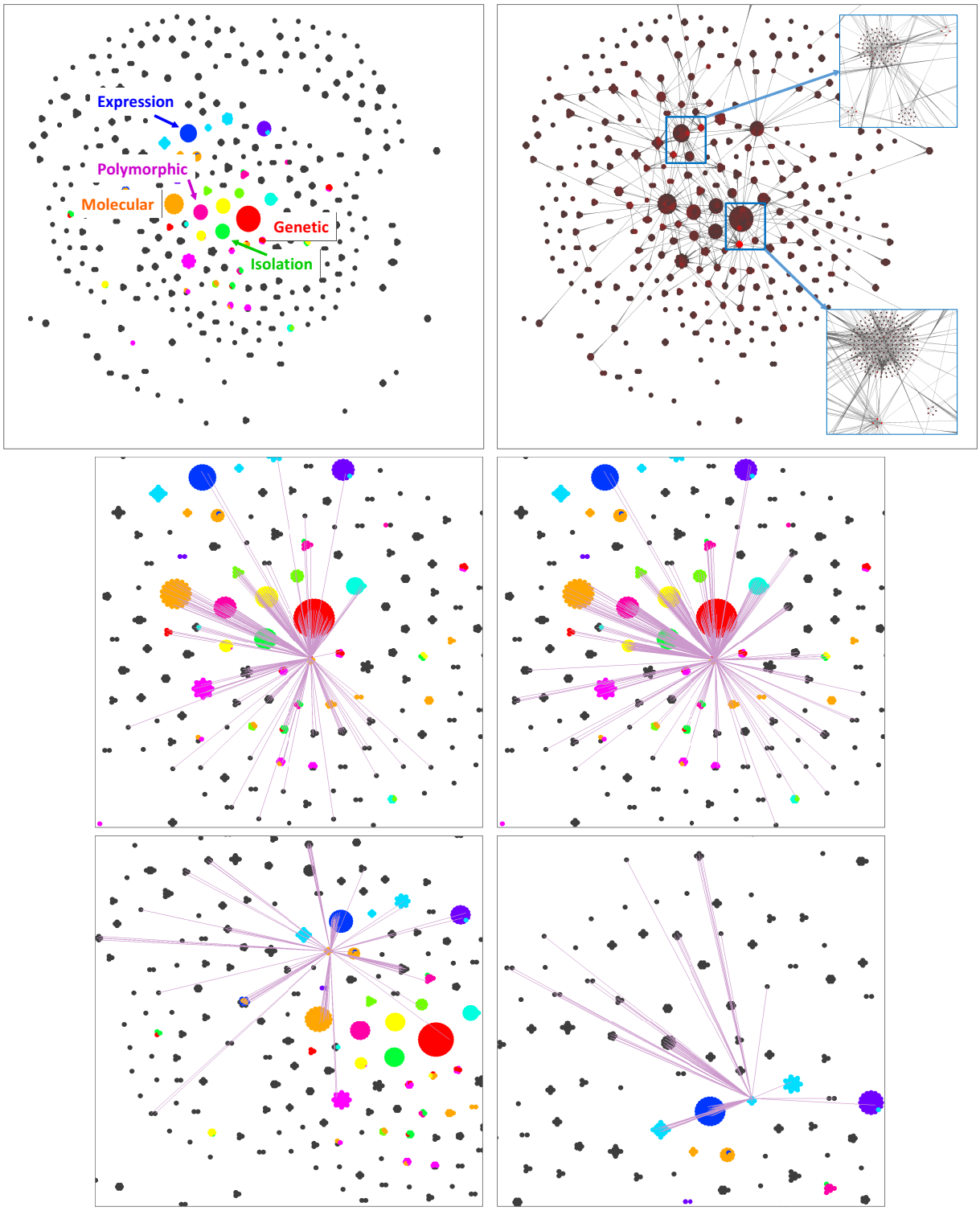


Figure 6: Case study with a co-authorship network.Rectangular regions are manually overlaid to show zooming views. (Upper left) Cluster layout. (Upper right) Importance representation. Important nodes are placed in the small clusters indicated by red circles. (Others) Co-authorship of important persons.