# Convergent Drawing for Mutually Connected Directed Graphs

Naoko Toeda[a], Rina Nakazawa[a], Takayuki Itoh[a], Takafumi Saito[b], Daniel Archambault[c]

[a] *Ochanomizu University.*
[b] *Tokyo University of Agriculture Technology.*
[c] *Swansea University.*

**Abstract**

Directed graphs are used to represent a variety of datasets, including friendship on social networking services (SNS), pathways of genes, and citations of research papers. Graph drawing is useful in representing such datasets. At the international conference on Information Visualisation (IV), we have presented a convergent edge drawing and a node layout technique for tightly and mutually connected directed graphs. The edge drawing technique in the IV paper includes three features: ordinary bundling of edges connecting pairs of node clusters, convergence of multiple bundles that connect to the same node cluster, and shape adjustment of two bundles connecting the same pair of node clusters. In this paper, we present improved node layout and edge drawing techniques, which make our edge bundling more effective. This paper also introduces a case study with a directed paper citation graph dataset.

*Keywords:* Directed graph, Edge bundling, Node layout, Graph clustering.

## 1. Introduction

A variety of datasets in academia and industry can be represented as directed graphs. For example, friendships on social networking services (SNS) can be

*Email addresses:* toeda@itolab.is.ocha.ac.jp (Naoko Toeda), leena@itolab.is.ocha.ac.jp (Rina Nakazawa), itot@itolab.is.ocha.ac.jp (Takayuki Itoh), txsaito@cc.tuat.ac.jp (Takafumi Saito), d.w.archambault@swansea.ac.uk (Daniel Archambault)

collected via Web APIs and transformed into directed graphs. Genetic pathways are available through open databases and can be considered as directed graphs. Paper citation networks are also directed graphs, which can be constructed from open information in digital libraries of academic societies.

Graph drawing techniques are useful in visually representing directed graphs. Software packages exist to visualize friendships on social networking services or genetic pathways. From our observation, many of these software packages feature general graph drawing algorithms, which can be applied to undirected graphs. The development of directed graph drawing algorithms is, therefore, still an interesting problem.

Edge bundling has been an active area of research in graph drawing for a number of years [3] - [7]. Most of the algorithms in this area are targeted at undirected or hierarchically clustered graphs and do not consider mutually connected directed graphs. Most of these approaches cluster edges at their centers, but it is also effective to bundle edges at their end points [10] - [11]. More effective techniques may be possible by effectively combining these two methods.

At the Information Visualization conference, we presented edge drawing and node layout techniques for tightly and mutually connected directed graphs [1]. Our approach constructs a hierarchy of nodes by recursively applying a graph clustering algorithm. Then, it applies a layout algorithm to draw the graph. Edges are bundled using an approach designed specifically for directed graphs, effectively revealing connectivity between node clusters. This algorithm bundles edges between pairs of clusters. Two bundles may be generated if the edges between the clusters are in opposite directions. The algorithm then combines multiple bundles departing from or arriving at the same cluster. Finally, it adjusts the shape of the bundles between each pair of node clusters. Our clustering and layout algorithms are designed so that our edge bundling algorithms work effectively. The node clustering algorithm groups nodes that have many edges connected to the same node, because this strategy bundles more edges. The node layout algorithm calculates weights of connections between a pair of

node clusters according to the smaller number of edges of up to two bundles connecting the clusters, so that tightly connected clusters are placed nearby.

This paper presents an improved layout algorithm and edge drawing technique. This technique inserts additional dummy nodes and edges into the graph before applying a force-directed algorithm so that nodes referred to or referred by the same node and placed closer to each other. Our edge bundling technique has also been improved to cluster more edges, reducing visual clutter.

This paper presents a case study using a paper citation network dataset retrieved from ACM Digital Library, and discusses the effectiveness of the presented technique.

## 2. Related Work

### *2.1. Edge Bundling*

Edge bundling has been widely studied. Zhou et al. [2] survey bundling techniques for information visualization. The purpose of edge bundling is to improve the comprehensibility of drawing complex graphs, which cluster similar edges together and draws them as bundles. Holten [3] presented a hierarchical edge bundling technique, which groups edges connected nodes belonging to the same pairs of clusters following a hierarchy. This technique is designed for hierarchical graphs, but does not implement processes specified for mutually connected directed graphs.

Selassie et al. [4], Holten et al. [5] presented techniques that group nearby, directionally similar edges using a spring model. Gansner and Koren [19], Pupyrev et al. [20] bundled edges to minimize the total ink used in the drawing. These techniques bundle edges optimizing an ink or energy function.

There are also many bundling techniques that use a grid superimposed on free space as a scaffold for bundling. Qu et al. [21] proposed an edge clustering method based on Delaunay triangulation. Lambert et al. [23] uses a hybrid grid. Luo et al. [10] uses a quadtree to guide the edge-bundling process.

These techniques reduce visual clutter and support the understanding of graph structure. However, most of these techniques may create a visual ambiguity that causes the perception of false connections between unconnected nodes as edges are gathered at their midpoints.

Several other techniques presented by Telea et al. [6] and Ersoy et al. [7] compute skeletons of edge clusters and enhance the visibility of overlapping bundles, often using image-based techniques. We would like to apply these techniques to our implementation as future work.

### *2.2. Edge Convergence*

In this paper, we define "bundling" as techniques that gather edges at their midpoints, and "convergence" as techniques which gather edges at their end points.

Luo et al. [10] named the problem of false connectivity caused by bundling "edge ambiguity" and presented a technique to draw edges as curves that converge at their endpoints to clearly represent connections. Bach et al. [11] presented a technique to bundle edges based on common sources or targets and generate confluent bundles indicating precise connectivity. This technique inserts nodes for confluent edges in addition to the original nodes before calculating the graph layout. These techniques successfully reduce clutter and ambiguity around end points of edges. However, these techniques do not support hierarchical and large graphs. In contrast to these techniques, this paper presents a directed graph visualization technique that cluster nodes, bundle edges at the end points and avoids edge ambiguity.

Edge compression using power graph [8] can also reduce cluttering and ambiguity around end points of edges. Power graphs group nodes hierarchically and surround the clusters with a closed curve. If all nodes in a module connect to the same node, a single edge connects the module to the node and can represent the relationship. Dwyer et al. [9] demonstrated that drawings that use power graphs leads to quicker understanding than standard network drawing. However, power graph drawing cannot cluster all of the important edges. We

solve this problem by supporting hierarchical graphs as input.

### *2.3. Node Clustering*

Graph clustering is a key technique that makes graph drawing more effective and comprehensive. There have been a large number of clustering algorithms introduced by several survey papers [12]. Though finding densely connected components is definitely the most popular approach for graph clustering, other approaches are also effective if we want to maximize the number of edges to be bundled. Itoh et al. [13] applied edge-adjacency and node-similarity based graph clustering to make large number of edges bundled and separate key nodes from large clusters. The technique presented in this paper also applies the same clustering algorithm.

### *2.4. Other Graph Visualization Techniques*

There have been many tree-like visualization techniques that operate on directed acyclic graphs [15][16], which place nodes based on their distance from a starting node. These techniques do not often work well for dense graphs as tree-like graphs are expected as input and their edges are expected to flow along common directions. Therefore, we did not apply this approach. Instead, we applied general algorithms for graph clustering and node layout, and extended edge bundling techniques for mutually connected directed graphs.

Incidentally, there have been many 3D graph visualization techniques [14]. These approaches are also effective if the graph is dense and large. However, 3D visualizations are not always suitable for an overview, as they require extensive training. Therefore, we developed a 2D graph visualization technique to provide overview of the whole graph.

## 3. Presented Visualization Tool

This section presents the data structure and processing algorithms of our technique. Our algorithm has the following steps:

1. Cluster nodes based on distances calculated from dissimilarity of feature vectors and discommonality of the adjacent nodes.
2. Calculate positions of nodes by applying force-directed node layout algorithm to a clustered graph treating the clusters as nodes.
3. Bundle and converge edges and draw as Bezier curves.

Our implementation calculates the positions of nodes and node clusters once while providing a mechanism for users to adjust edge drawing interactively. Therefore, our edge bundling and drawing algorithms need to calculate the bundling quickly in response to user interaction. However, our clustering and layout algorithms can be run as a preprocessing.

### *3.1. Data Structure*

A graph $G$ consists of a set of nodes $N$ and edges $E$ as follows:

$$G = \{N, E\}$$
$$N = \{n_1, ..., n_{N_n}\}$$
$$E = \{e_1, ..., e_{N_e}\}$$
$$n_i = \{a_{i1}, ..., a_{iNa}\}$$
$$e_i = \{n_i, n_k\}$$

Here, $n_i$ denotes the $i$-th node. We suppose that a node has a multi-dimensional attribute value, where $a_{ij}$ denotes the value of the $j$-th dimension of the $i$-th node. $e_i$ denotes the $i$-th edges, where the edge is directed from $n_i$ to $n_k$. $N_n$ denotes the number of nodes. $N_e$ denotes the number of edges. $N_a$ denotes the number of dimensions.

### *3.2. Node Clustering*

The technique calculates high dimensional distances between arbitrary pairs of nodes, and then applies a clustering algorithm to construct the hierarchy of the nodes. This section describes the definition of node distance, and processing flow of the node clustering algorithm.

*3.2.1. Node Distance Calculation*

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

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

Here, $\alpha$ is a user-specified value satisfying ( $0.0 \leq \alpha \leq 1.0$ ) .

*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.

$$d_{vec} = 1.0 - inner$$
$$inner = \frac{n_i \cdot n_j}{|n_i||n_j|}$$

*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 calculates the distance as follows.

$$d_{adj} = \frac{1.0}{1 + n_{adj}}$$

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

*3.2.2. Hierarchical Clustering*

Our implementation then constructs hierarchy of nodes based on the above mentioned distances. Currently, we generate a two-level hierarchy of clusters by applying an agglomerative approach that uses the furthest neighbor method.

We limit the number of levels in the hierarchy to two to simplify the user-controlled parameters.

This implementation requires to adjust two user-specified thresholds $\alpha$ and $\beta$ ($\alpha < \beta$) for each level of clusters. We limit the number of level of hierarchies up to two, because we would not like to increase the number of user-adjusted parameters. The process begins by grouping nodes while satisfying the maximum distance $\alpha$. Clusters produced in this stage are called "cluster A". Then, the process groups nodes while satisfying the maximum distance $\beta$. Clusters produced in this stage are called "cluster B".

### *3.3. Node Layout*

Our implementation calculates positions of nodes by the following a bottom-up algorithm. First, it calculates the positions of level A clusters that belong to a level B cluster. Then, it calculates the positions of level B clusters. The second part of this algorithm is similar to the implementation presented in Itoh and Klein [13].

#### *3.3.1. Positions of level A clusters*

1. Generate a graph treating the level A clusters as nodes. Weight the edges according to the number of edges of the input graph connecting pairs of clusters, so that tightly connected clusters are placed closely.
2. Apply the later mentioned algorithm (Layout for edge convergence) or a force-directed node layout algorithm to the above mentioned graph to calculate the positions of level A clusters.
3. Calculate the radii of circles enclosing each of level A clusters based on number of belonging nodes.
4. Adjust distances between pairs of level A clusters closer to the sum of radii by applying Laplacian smoothing algorithm.
5. Calculate positions of nodes belonging to the level A clusters.

*Layout for edge convergence*

In our previous implementation, we applied a force-directed algorithm in the second step of our approach. However, applying a force-directed algorithm

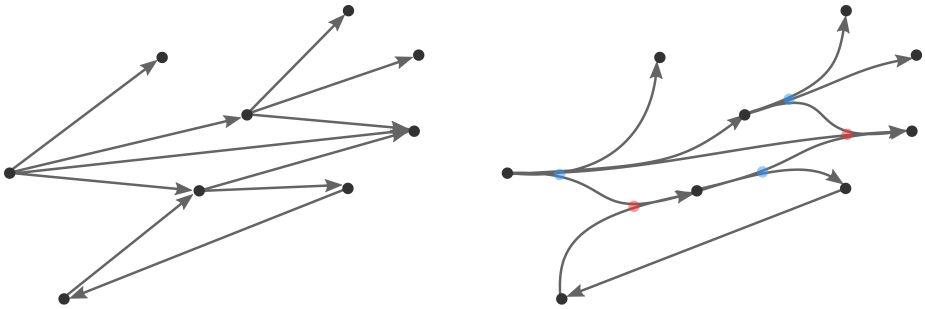

Figure 1: Example of node layout. (Left) Input graph with straight line drawing. (Right) The good layout and edge bundling for left graph. Blue and red nodes can be control points for curve drawing.

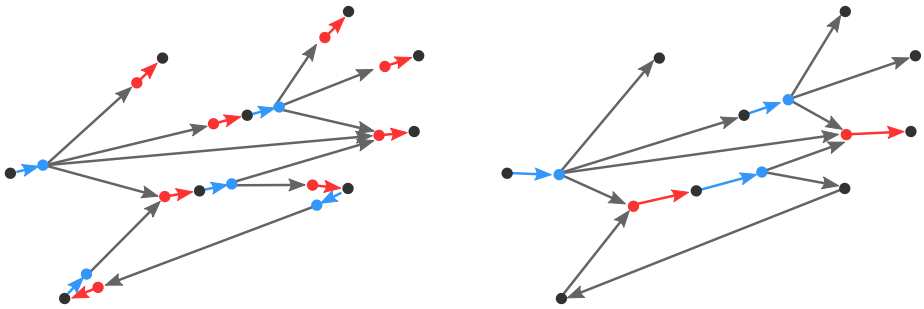

Figure 2: Layout for edge convergence. (Left) Generating additional nodes for each nodes referring of referred one or more nodes. (Right) Generating additional nodes for each nodes referring of referred two or more nodes.

treats the clusters as nodes and does not allow for edge convergence. If two nodes connected by an edge are close, their bundles can be converged. This technique inserts additional dummy nodes and edges into the graph before applying a force-directed algorithm so that these nodes are placed closer to each other (Figure 1).

1. Apply a force-directed node layout algorithm.
2. Apply the following steps (Figure 2(Left)):
   (a) Select a node ($N_x$ in Figure 3).
   (b) If $N_x$ refers to one or more nodes
      i. Generate a blue node ($N_b$).

ii. Generate edges connecting $N_b$ to the nodes connected $N_x$ ($N_bN_y$ and $N_bN_z$).

iii. Remove the edges connecting from $N_x$ ($N_xN_y$and$N_xN_z$).

iv. Generate an edge connecting $N_x$ to $N_b$.

(c) Apply the following steps if $N_x$ is connected one or more nodes:

i. Generate a red node ($N_r$).

ii. Generate edges connecting the nodes connected $N_a$ to $N_r$ ($N_{y'}N_r$ and $N_{z'}N_r$).

iii. Remove the edges connecting to $N_x$ ($N_{y'}N_x$ and $N_{z'}N_x$).

iv. Generate an edge connecting $N_r$ to $N_a$.

(d) Repeat the above steps to all the nodes.

(e) Apply a force-directed node layout algorithm to the above mentioned graph.

(f) Remove red and blue nodes and edges.

(g) Generate edges.

3. Apply the similar algorithm while generating red and blue nodes for each nodes referring of referred two or more nodes (Figure 2(Right)).

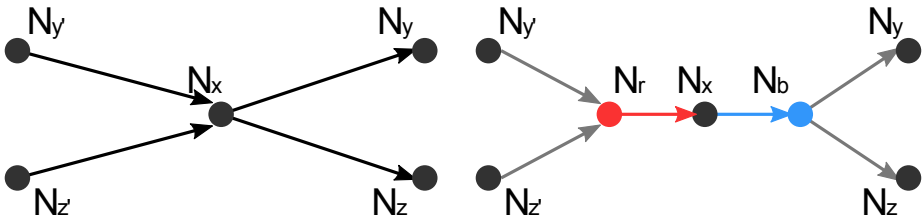


Figure 3: Algorithm to generate graph before applying a force-directed node layout algorithm. (Left) Input graph. (Right) Output graph.

*3.3.2. Positions of level B clusters*

1. Generate a layout treading the level B clusters as nodes. Weight the edges according to the number of edges of the input graph connecting pairs of clusters, so that tightly connected clusters are placed close to each other.

2. Apply the above mentioned node layout algorithm for level A clusters belonging to each of level B clusters.
3. Calculate radii of circles enclosing each of level B clusters based on the node layout results of level A clusters.
4. Adjust distances between pairs of level B clusters closer to the sum of radii by applying Laplacian smoothing algorithm.
5. Calculate positions of level A clusters belonging to the level B clusters.

### *3.4. Edge Bundling*

The presented edge bundling technique aims to satisfy the three conditions illustrated in Figure 4. First of all, our technique specifies sets of bundles to be clustered according to Condition C and rotation direction of bidirectional bundles. Then, it calculates the shape and thickness of the bundles, taking all conditions into account. This section describes how bundles are selected and how their final shapes are calculated.

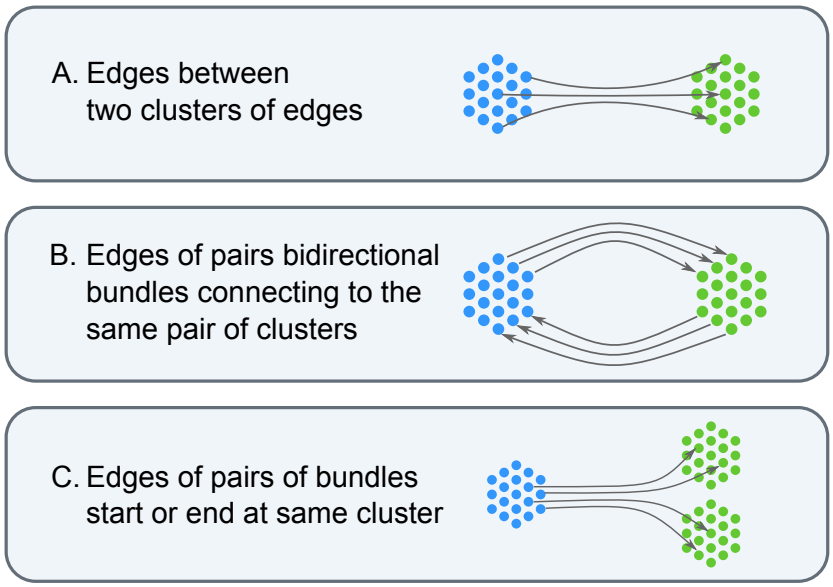


Figure 4: Conditions for edge bundling

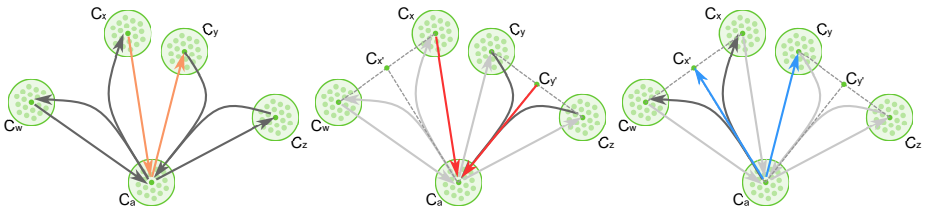


Figure 5: Selection of pair of vectors for score of convergence. (Left) Pair for the first calculation. (Center) Pair for bundles connecting to $C_a$. (Right) Pair for bundles connecting from $C_a$.

### *3.4.1. Convergence of Bundles*

The technique groups adjacent bundles connected to the same cluster using the following algorithm (Figure 5).

1. Select a cluster ($C_a$).
2. Select the next bundle connected to the cluster in clockwise order ($C_y$).
3. Calculate the later mentioned score (3.4.1.1) of the selected bundle and the bundle on the left ($C_x$), if they have one or more similarly directed edges. This calculation uses two vectors, $C_aC_y$ and $C_aC_x$ (Figure 5 (Left)).
4. If the score is larger than a user-specified threshold,
   (a) Calculate the position of the gravity of the all converging clusters ($C_{x'}$ and $C_{y'}$) if these two bundles have grouped with other bundles.
   (b) Calculate the score of the two bundles connecting to $C_a$ ($C_xC_a$ and $C_{y'}C_a$) if both of them have one or more edges connecting to $C_a$ (Figure 5 (Center)). Let this score $S_a$.
   (c) Calculate the score of the two bundles connecting from $C_a$ ($C_aC_{x'}$ and $C_aC_y$) if both of them have one or more edges connecting from $C_a$ (Figure 5 (Right)). Let this score $S_b$.
   (d) Group the two bundles connecting to $C_a$ if the score is larger than a user-specified threshold and $S_a > S_b$.
   (e) Group the two bundles connecting from $C_a$ if the score is larger than a user-specified threshold and $S_b > S_a$.
5. Repeat the above process to all the clusters.

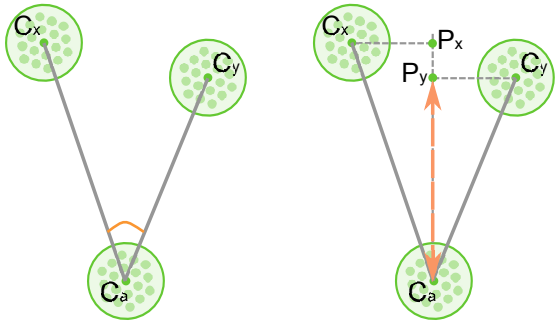


Figure 6: Conditions for convergence of bundles. (Left) Orange part is angle between two bundles. (Right) Orange segment is the convergable part of the two bundles.

Our implementation calculates the score of two adjacent bundles as described in the next sections.

*3.4.1.1. Score for Convergence of Bundles.*

**Angle between the two bundles** Let us assume nodes in cluster $C_a$ are connected to nodes in clusters $C_x$ and $C_y$, and call centers of these clusters $C_a$, $C_x$, and $C_y$. Also, let us call the segments connecting two clusters $C_aC_x$ and $C_aC_y$. The technique treats the score as zero if the angle between $C_aCxz$ and $C_aC_y$ is larger than a user-specified threshold.

**Length of convergable parts** The technique generates the bisector of $C_aC_x$ and $C_aC_y$, and then generates perpendicular segments from centers of two clusters ($C_x$ and $C_y$) connected to the bisector. Let us call the intersections between the bisector and perpendicular segments $P_x$ and $P_y$. The technique then calculates the lengths of $C_aP_x$ and $C_aP_y$. We treat the smaller length as the score (i) of $C_aC_x$ and $C_aC_y$, because the distance denotes the convergable part of the two bundles. In Figure 6, $C_aP_y$ is the length of convergable parts.

*3.4.2. Rotation Direction of Bidirectional Bundles*

This technique specifies the rotation direction of bidirectional bundles converging without bundles by the following algorithm (Figure 7).

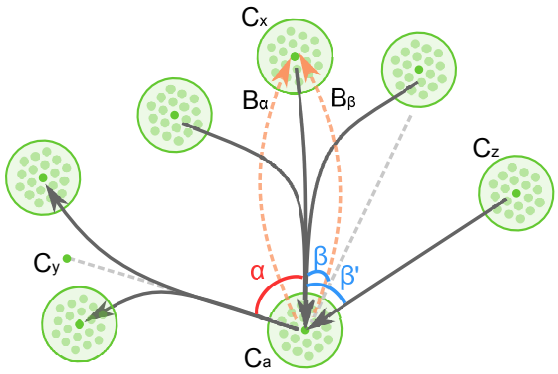


Figure 7: Selection of rotation direction of bidirectional bundle. The technique compares $\alpha$ and $\beta$.

1. Select a bundle converging without bundles. See the bundle connecting $C_a$ to $C_x$ in Figure 7.
2. Calculate the angle $\beta$' between $C_aC_x$ and the bundle on the right $C_aC_z$. If the left bundle has the reverse direction, bisect the angle $\beta$.
3. Calculate the angle $\alpha$ between $C_aC_x$ and the bundle on the left $C_aC_y$ by the similar calculation.
4. Specify the rotation direction of the bundle in a clockwise direction if $\alpha$ is larger than $\beta$. If not, specify the rotation direction in a counterclockwise direction.
5. Repeat the above process for all the bidirectional bundles.

*3.4.3. Edge Drawing*

The technique categorizes the edges of the input graph into the following three cases. Bezier curves are used to draw these cases. Edges fade from blue (source) to red (target) to indicate their directions because arrowheads are a poor choice for depicting directionality unless a graph is very simple [18].

*Edges of bundles converged to other bundles*

The technique places the control points of the Bezier curves on the bisectors of the adjacent two bundles, as illustrated in Figure 8(Left). The position of

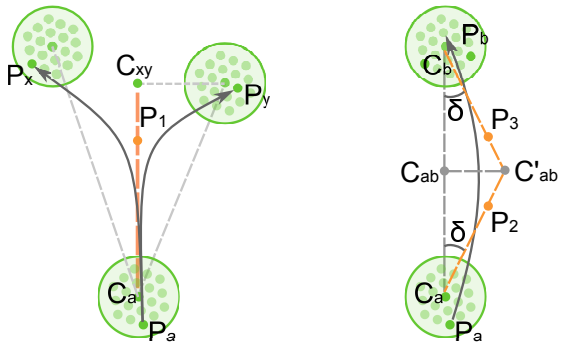


Figure 8: Positions of control points of Bezier curves for edge bundling. (Left) For edges of bundles converged to other bundles. (Right) For edges of pairs of bidirectional bundles connecting to the same pair of bundles.

the control point $P_1$ is calculated by the following equation.

$$P_1 = C_a(1-\gamma) + P_{xy}\gamma$$

Here, $\gamma$ is a user-specified value satisfying ( $0.0 \leq \gamma \leq 1.0$ ) . If more than three bundles can be converged, we specify the position of $E$ on the line $A$ to center of gravity of the all clusters ($B$, $C$, $\cdots$).

*Edges of pairs of bidirectional bundles connecting to the same pair of clusters.*

As illustrated in Figure 3(Right), the technique places the control points of the Bezier curves on $C_aC'_{ab}$ and $C_bC'_{ab}$, which is on the side of $C_aC_b$. Here, $C_{ab}C'_{ab}$ is the bisecting perpendicular segment of another segment connecting the centers of clusters $C_a$ and $C_b$. Additionally, the angle between $C_aC_{ab}$ and $C_aC'_{ab}$, $\delta$, is calculated by the following equation (Figure 7):

$$\delta = \min\{\alpha, \beta, \frac{\pi}{4}\}$$

Positions of the control points $P_2$ and $P_3$ are calculated by the following equations:

$$P_2 = C_a(1-\gamma) + C'_{ab}\gamma$$
$$P_3 = C_b(1-\gamma) + C'_{ab}\gamma$$

Here, $\gamma$ is a user-specified value satisfying $0.0 \leq \gamma \leq 1.0$.

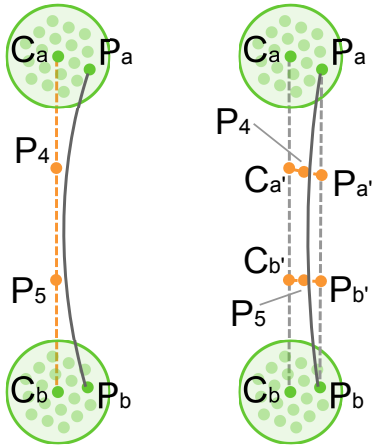


Figure 9: Positions of control points of Bezier curves for edge bundling.

*Other Cases*

In all other cases, the following algorithm is applied if the above-mentioned cases are not applicable. Our implementation places two of the control points on the segment connecting the centers of clusters, as shown in Figure 9(Left), when we would like to tightly bundle the edges. Otherwise, it first calculates the positions that divide the two segments (one connects two nodes, and the other connects centers of the clusters) into three equal-sized parts. It then places the control points on the segments connecting them. Figure 9(Right) illustrates this process. Positions of the control points $P_4$ and $P_5$ are calculated by the following equations.

$$
\begin{aligned}
if\ \ \gamma > 0.5 \quad then \quad & r = (\gamma + 0.5) \times \frac{2}{3} \\
& P_4 = C_a r + C_b (1 - r) \\
else \quad & r = \gamma \times 2 \\
& P_5 = \frac{C_a \times 2 + C_b}{3} \times r + \frac{P_a \times 2 + P_b}{3} \times (1 - r)
\end{aligned}
$$

Here, $\gamma$ is a user-specified value satisfying ( $0.0 \leq \gamma \leq 1.0$ ).

| layout | converging edges | bidirected edges | other edges |
|---|---|---|---|
| 1. simple layout [1] | 59 | 736 | 540 |
| 2. layout in Figure 2(Left) | 247 | 674 | 414 |
| 3. layout in Figure 2(Right) | 223 | 694 | 418 |

| layout | converging edges | bidirected edges | other edges |
|---|---|---|---|
| 1. simple layout [1] | 533 | 402 | 400 |
| 2. layout in Figure 2(Left) | 699 | 347 | 289 |
| 3. layout in Figure 2(Right) | 785 | 292 | 258 |

Table 1: Number of converging edges for each layout algorithm.

| layout | number of crossing pairs of edges | percentage of them |
|---|---|---|
| 1. simple layout [1] | 184 | 1.37% |
| 2. layout in Figure 2(Left) | 200 | 1.49% |
| 3. layout in Figure 2(Right) | 212 | 1.57% |

| layout | number of crossing pairs of edges | percentage of them |
|---|---|---|
| 1. simple layout [1] | 670 | 2.27% |
| 2. layout in Figure 2(Left) | 668 | 2.26% |
| 3. layout in Figure 2(Right) | 786 | 2.66% |

Table 2: Number and percentage of edge crossing.

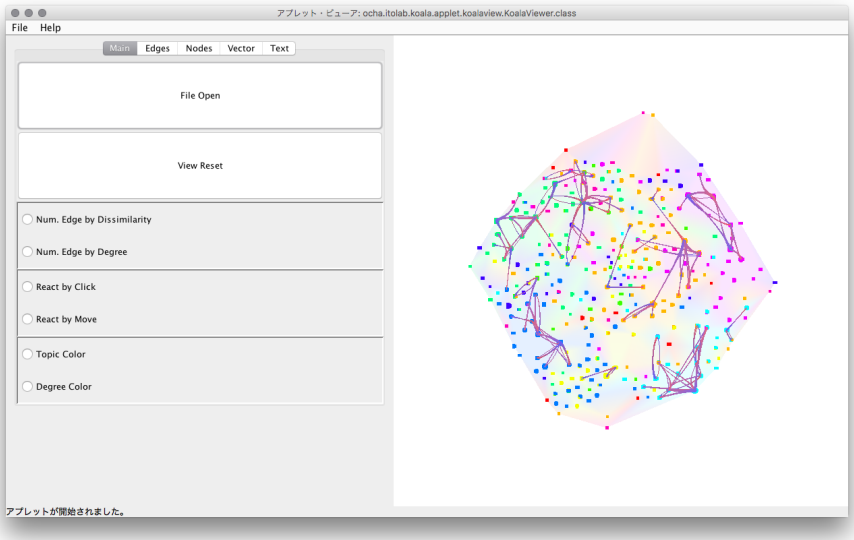


Figure 10: User interface

### *3.5. User Interface*

Figure 10 is a snapshot of the user interface we implemented. The right side of the window features the drawing space while the left side features five tabs. Users can scale and pan the view so that all nodes can be made visible (Figure 10).

Users can adjust the bundling and layout parameters. In accordance with user operation by the sliders, our implementation regroups and redraws edges. Users can change node clustering and layout by pressing the "replace" button after adjusting the sliders.

We assign individual colors to each of the dimension of the feature vectors. Node color is specified by the dimension with the largest difference in these feature values. The implementation generates a triangular mesh connecting the nodes and linear interpolation is used to color areas between nodes. As a result, the background color represents the largest difference in feature values for peripheral nodes.

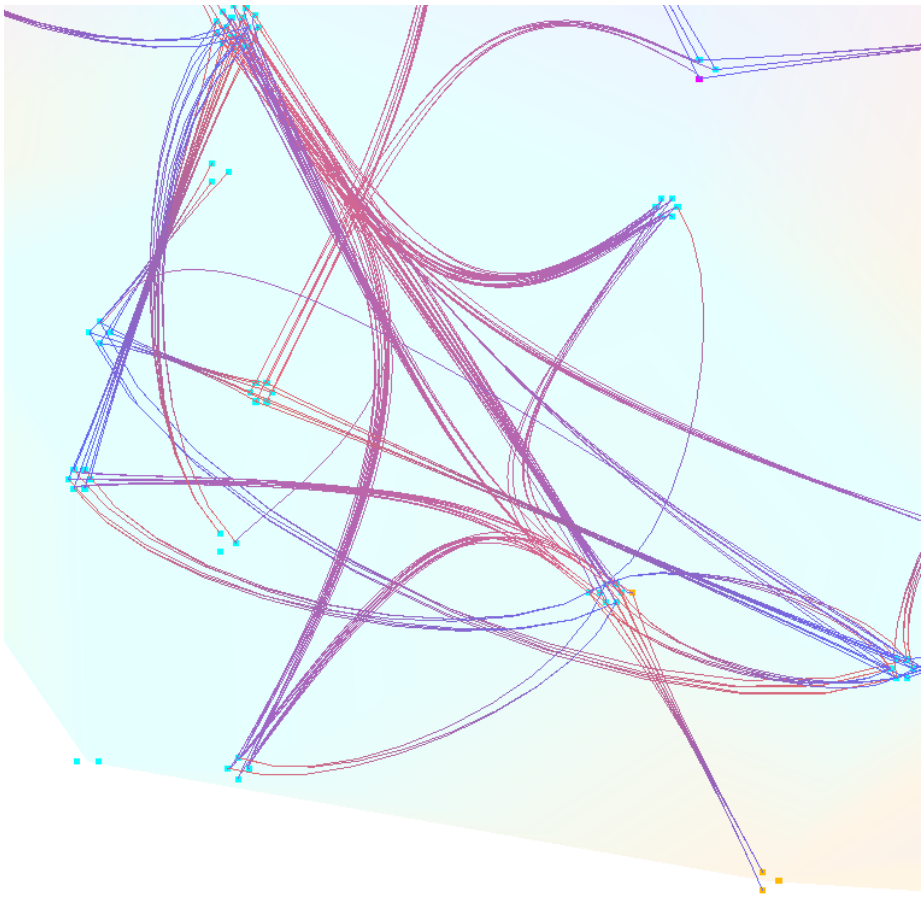

Figure 11: Visualization result zooming in some cluster.

## 4. Case Study with Paper Citation Network

We applied our technique to a citation network dataset [17], consisting of 1072 full papers presented at ACM SIGGRAPH conferences. The papers were published between 1990 - 1994 and 2000 - 2010 and were collected from the ACM Digital Library. We extracted the title, publication year, abstract, references, and authors from html files of the papers. This dataset consists of 1072 papers (nodes) and 5498 references (edges). We created a 10 dimensional attribute vector for each paper from their abstracts by applying generative topic model LDA (Latent Dirichlet Allocation).

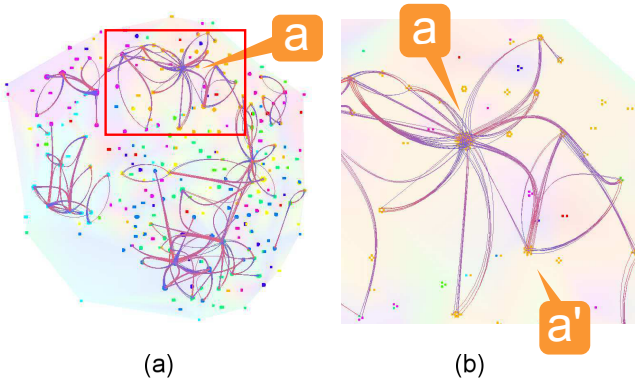


Figure 12: Visualization where edges are bundled based on level B clusters. (a) Overview. (b) Zooming to the cluster `a`.

### *4.1. Count of converge part of each layout*

Table 1 shows the number of converging edges for each layout algorithm described in section 3.3.1. Our technique, which generates additional nodes and edges during the layout process, is more effective at converging edges when compared to our previous technique [1]. However, Table 2 shows that our layout technique does so at the expense of edge crossings. In future work, we would like to conduct user evaluations to determine the impact of edge crossings on the effectiveness of our visualizations.

### *4.2. Example of level A clusters*

Figure 12 shows a result when the bundling is based on level A clusters. Many of the edges connected to nodes in the cluster labeled `a` in Figure 12 are unexpectedly bundled. This cluster likely has several important papers related to the topic colored blue. We checked the titles of papers in this cluster and found many of them are related to fundamental illumination methods: light transport, precomputed radiance transfer, and tree structures for shadow computation. The papers of this cluster have mutual connections with many other clusters because they are fundamental methods and are likely to influence other research topics. Our visualization result highlights this important cluster clearly.

On average, the edge redrawing required 0.0148 seconds after changing the bundle shape parameter and 0.031 seconds after changing the converge parameter. This result demonstrates that users can adjust edge drawing interactively by adjusting the sliders.

### *4.3. Example of level B clusters*

Figure 13 shows a visualization result where edges are bundled based on level B clusters. In this figure, many edges connect pairs of nodes with the same colors, meaning that very similar topics are referred. On the other hand, nodes painted in yellow-green in the cluster `b` are connected to the clusters `c`, `d`, and `e`, painted in blue, green, or orange. These relationships are interesting because papers `b` are connected to other research topics.

Cluster `b` includes papers presenting automatic texture generating and retouching techniques. In cluster `c`, papers related to 3D rendering techniques are connected to papers in cluster `b` because they need textures to map on 3D models for realistic rendering. Furthermore, papers related to image processing that belong to `d` and 3D modeling that belong to `e`, are connected to papers in `b` as texture mapping is applied to complex 3D models. These citation relationship indicates that the topic of `b` is fundamental and has been applied to many studies.

On the other hand, it was somewhat time-consuming for us to discover the above-mentioned relationships, as the clusters, especially `d` and `e`, contain many different research topics. Therefore, we would like to further refine our clustering methods and the definition of both our multi-dimensional attribute values and node distances.

### *4.4. Example of edge drawing*

Figure 14 shows a part of our visualization result. If only bidirectional bundles are enabled (Figure 14(a)), the bundle connecting $D$ to $A$ overlaps with the bundle connecting $A$ to $C$. If only converging bundles are enabled (Figure 14(c)), the bundle connecting $A$ to $D$ overlaps with the bundle connecting $D$ to

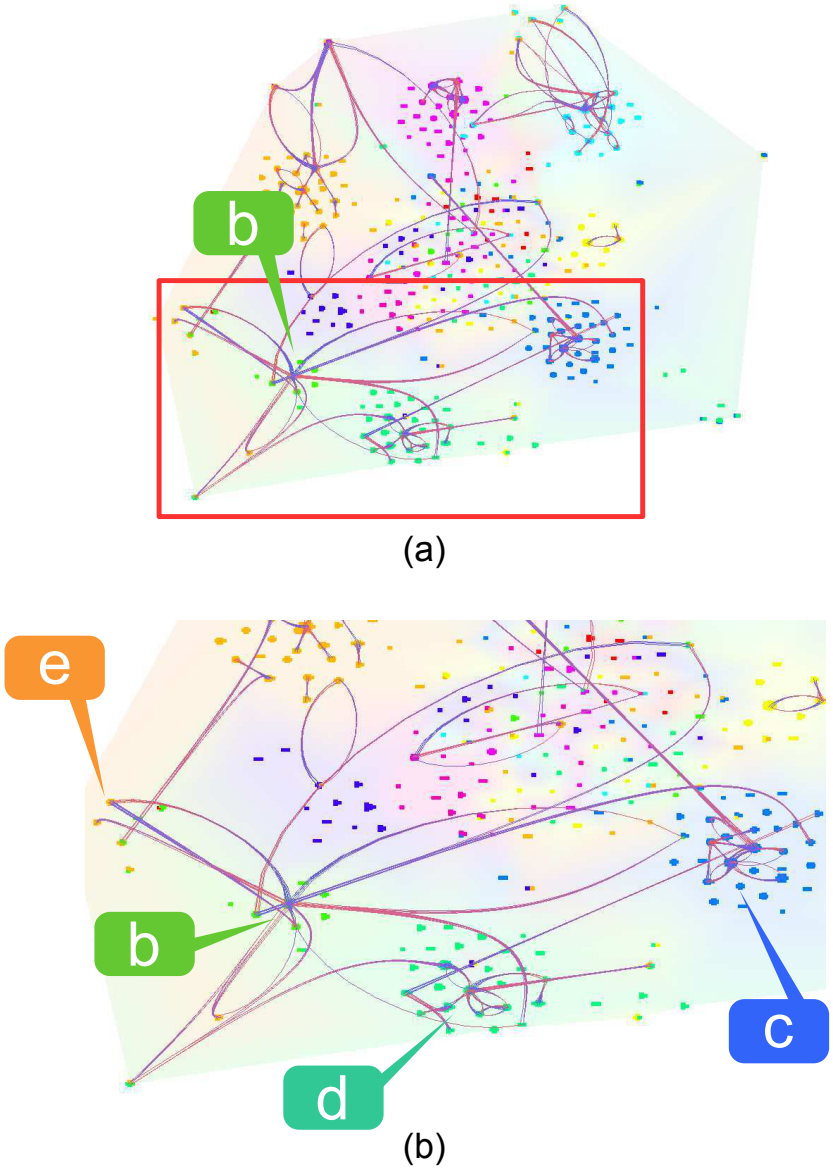


Figure 13: Visualization where edges are bundled based on level B clusters. (a) Overview. (b) Zooming to the cluster `b`.

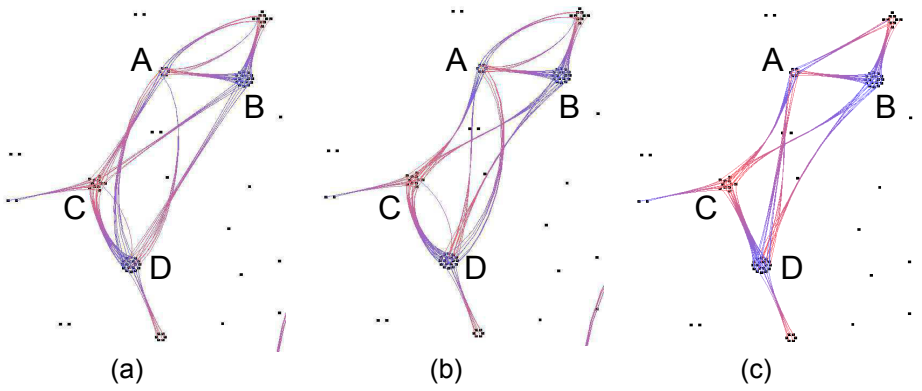


Figure 14: Visualization result zooming in some cluster.

*A*. However, if both are enabled, all bundles are made clear as shown in Figure 14(b).

## 5. Conclusion

This paper presented an edge bundling technique for directed graphs with many mutually connected edges. This technique deforms edges based on the following three points: 1) bundling edges connecting two nodes belonging to same pairs of clusters as ordinary edge bundling techniques, 2) preserving proper distances between bidirectional bundles between the same pairs of clusters, and 3) converging adjacent bundles connecting to the same cluster. This technique is especially effective for directed graphs, which have many mutually, connected edges. We presented a case study using a paper citation network dataset and discussed what our technique can visualize.

In future work, we would like to improve our algorithm for inserting temporary nodes used in generating the layout. We would like to develop rotation of node clusters [22] so that the total lengths of edges is minimized. Edge drawing can be also improved. We would like to evaluate edge pairs to be bundled so that we can better cluster edges.

After these improvements, further case studies with real-world directed graphs

would help. Also, formal user experiments would provide evidence of effectiveness of our technique in terms of human performance.

## References


[1] Naoko Toeda, Rina Nakazawa, Takayuki Itoh, Takafumi Saito, Daniel W. Archambault On Edge Bundling and Node Layout for Mutually Connected Directed Graphs *20th International Conference on Information Visualisation (IV2016)*, 94-99, 2016.

[2] Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu Edge Bundling in Information Visualization *singhua Science and Technology* , 18(2), 145-156, 2013.

[3] D. Holten Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, *IEEE Trans. on Visualization and Computer Graphics*, 12(5), 741–748, 2006.

[4] D. Selassie, B. Heller, J. Heer, Divided Edge Bundling for Directional Network Data, *IEEE Trans. on Visualization and Computer Graphics*, 17(12), 2354–2363, 2011.

[5] D. Holten, J. J. van Wijk, Force-Directed Edge Bundling for Graph Visualization, *Computer Graphics Forum*, 28, 3, 2009.

[6] A. Telea and O. Ersoy, Image-based edge bundles: Simplified visualization of large graphs, *Computer Graphics Forum*, 29(3), 843–852, 2010.

[7] O. Ersoy, C. Hurter, F. V. Paulovich, G. Cantareira, A. Telea, Skeleton-based edge bundling for graph visualization, *IEEE Trans. on Visualization and Computer Graphics*, 17(12), 2364–2373, 2011.

[8] T. Dwyer, C. Mears, K. Morgan, T. Niven, K. Marriott, M. Wallace, M, Improved Optimal and Approximate Power Graph Compression for Clearer Visualisation of Dense Graphs, *IEEE Pacific Visualization Symposium*, 105–112, 2014.

[9] T. Dwyer, N. H. Riche, K. Marriott, C. Mears, Edge compression techniques for visualization of dense directed graphs, *IEEE Trans. on Visualization and Computer Graphics*, 19(12), 2596–2605, 2013.

[10] S.-J. Luo, C.-L. Liu, B.-Y. Chen, K.-L. Ma, Ambiguity-Free Edge-Bundling for Interactive Graph Visualization, *IEEE Trans. on Visualization and Computer Graphics*, 18(5), 810–821, 2011.

[11] B. Bach, N. H. Riche, C. Hurter, K. Marriott, T. Dwyer, Towards Unambiguous Edge Bundling: Investigating Confluent Drawings for Network Visualization, *IEEE Trans. on Visualization and Computer Graphics*, 2016.

[12] S. E. Schaeffer, Graph Clustering, *Computer Science Review*, 1(1), 27–64, 2007.

[13] T. Itoh, K. Klein, Key-node-Separated Graph Clustering and Layout for Human Relationship Graph Visualization, *IEEE Computer Graphics and Applications*, 35(6), 30–40, 2015.

[14] T. Munzner, H3: Laying out large directed graphs in 3D hyperbolic space., *IEEE Symposium on Information Visualization*, 2–10, 1997.

[15] K. Sugiyama, S. Tagawa, M. Toda, Method for Visual Understanding of Hierarchical System Structures, *IEEE Trans. on Systems, Man, and Cybermetics*, 11(2), 109–125, 1981.

[16] T. Dwyer, Y. Koren, Dig-CoLa: Directed Graph Layout through Constrained Energy Minimization, *IEEE Symposium on Information Visualization*, 65–72, 2005.

[17] R. Nakazawa, T. Itoh, T. Saito, A Visualization of Research Papers Based on the Topics and Citation Network, *18th International Conference on Information Visualisation (IV2015)*, 283–289, 2015.

[18] D. Holten, J. J. van Wijk, A User Study on Visualizing Directed Edges in Graphs, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2299–2308, 2009.

[19] E. R. Gansner and Y. Koren, Improved circular layouts, *Proceedings of International Symposium on Graph Drawing*, 386–398, 2006.

[20] S. Pupyrev, L. Nachmanson, and M. Kaufmann, Improving layered graph layouts with edge bundling, *Proceedings of International Symposium on Graph Drawing*, 329–340, 2010.

[21] H. Qu, H. Zhou, and Y. Wu, Controllable and progressive edge clustering for large networks, *Proceedings of International Symposium on Graph Drawing*, 399–404, 2006.

[22] D. Archambault, T. Munzner, D. Auber, TopoLayout: Multilevel Graph Layout by Topological Features, *IEEE Trans. on Visualization and Computer Graphics*, 13(2), 305–317, 2007.

[23] A. Lambert, R. Bourqui, and D. Auber, Winding roads: Routing edges into bundles, *Computer Graphics Forum*, 29(3), 853–862, 2010.