減少視覺混淆之互動式網路視覺化系統
Ambiguity-Reducing for Interactive Network Visualization

劉俊良
國立台灣大學
being31@cmlab.csie.ntu.edu.tw

陳炳宇
國立台灣大學
robin@ntu.edu.tw

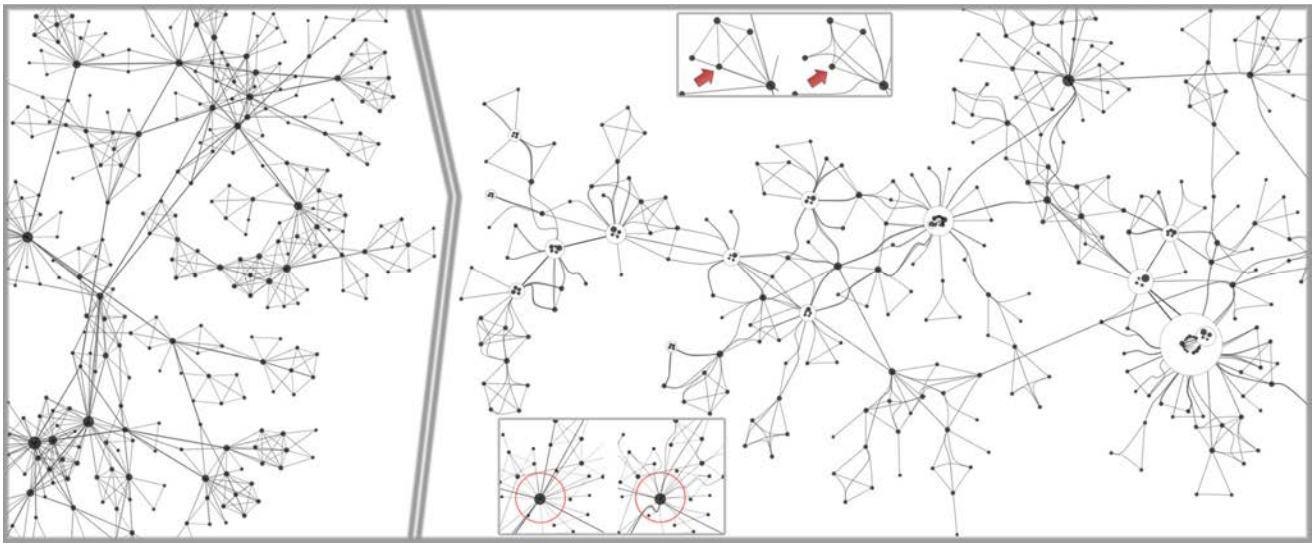


Fig. 1. Ambiguity-Free Edge-Bundling. In the left figure, the graph using straight-line edges has edge ambiguity problem. In the right figure, after deploying our ambiguity-free edge-bundling, the graph becomes clear with less edge ambiguity problem.

ABSTRACT
In this paper, we present an ambiguity-free edge-bundling approach for interactively visualizing graph or network data. To visualize a large graph or network dataset, the edges used to represent the relationship between the nodes are usually very dense. Due to the dense edges and huge amount of nodes, the graph or network is usually hard to be read or used according to the cluttered edge crossing or occlusion. To increase the readability and aesthetical niceness of the edge layout while decreasing the visual ambiguity, we introduce an ambiguity-free edge bundling framework to reduce the visual clutter caused by the edges and also improve the user's perceptual consistency of graph or network visualization and the actual data's relation. In our network visualization system, we use an efficient and generic quadtree structure that can be applied in conjunction with existing graph or network visualization systems. To provide the user an easy-to-use interactive user interface, we also introduce a novel detail-on-demand concept to make the user to be able to control the level-of-details regionally by painting the graph or network.

Keywords
Network visualization, graph visualization, edge bundling, curved edge, visual clutter, edge aggregation, level of detail, detail on demand.

1. Introduction
Visualizing network data as a graph is a common and well-known approach. A graph typically consists of some nodes and some edges connected the nodes to represent the relationship between them. The scalability of a graph is a practical concern when visualizing the network data. To increase the scale or size of the graph implies that more nodes and more edges are added into the graph to represent more information of the network data, so the graph should have more information either explicitly or implicitly. However, due to the limited display space, to increase the scale or size of the graph also makes the density of it become much more dense, which occurs several critical challenges, since a dense graph with a large amount of nodes and edges is usually hard to be recognized and used according to the cluttered edge crossing or occlusion [7].

Drawing the entire network data contained thousands or millions of nodes may provide the user an indication of the overview or a specific location concept within it, but this also makes it much difficult to comprehend further and causes misinterpretation due to the cluttered edge crossing or occlusion of the dense graph or network [20]. However, since most of the network data have small-world characteristics [12, 23, 21], it is possible to cluster some nodes and edges with highly connected relationship to decrease the density of the graph while still preserving the overview of it. Through the level-of-detail approach as that provided by many previous methods, the user may perceive the details of the network data, but without considering the visual clutter, the graph may still be hard to be recognized.

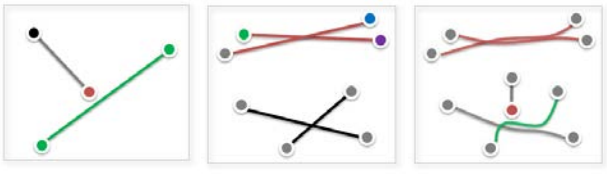
To increase the readability and aesthetical niceness of the edge layout while decreasing the visual ambiguity, we introduce an ambiguity free edge-bundling framework in this paper to reduce

the visual clutter caused by the edges and also improve the user's perceptual consistency of the graph or network visualization and the actual data's relation. Our target is to improve the graph layout for a reasonable graph scale of the network data with small-world characteristics.

Some previous graph layout algorithms focused on the node placement in order to improve the space utilization, but in our observation, most of the visual clutter and relation misinterpretation were caused by inappropriate edge layout style. The main issues of improper edge layout are as the follows:

- Dense edge crossings cause most of the visual clutter and waste the display space.
- Edges passing near to the unrelated nodes cause relation misinterpretation on visual perception, called as edge ambiguity. (Fig. 2(a))
- Edges with both of similar gradient and geometrical position decrease the recognition ability of individual relationships. (Fig. 2(b))

Fig. 2 shows some edge ambiguity cases. In Fig. 2(a), the red node is too close to the green edge that causes incorrect visual perception of additional relationship with the two green nodes. In Fig. 2(b), due to the similar edge gradient, it is hard to recognize if the green node connects to the blue node or the purple one. Although there are some previous edge-bundling methods could help to reduce the visual clutter, without considering the actual relationship of the nodes, the adjusted graph may still has edge ambiguity problem. As shown in Fig. 2(c), although the red and green edges are bundled to prevent the visual clutter, the red edges are still hard to be recognized as Fig. 2(b) and the green edge causes another kind of visual clutter as Fig. 2(a). The visual clutter caused by cluttered edges significantly decreases the readability of the graph, and more importantly, inappropriate edge layout decreases the consistency between the user's visual perception and the actual relation data. In order to make the graph or network visualization to convey the information as effectively and accurately as possible, a good edge representation is needed.



(a) (b) (c)

Fig. 2. Some edge ambiguity cases. (a) The red node is too close to the green edge that causes incorrect visual perception of additional relationship with the two green nodes. (b) Comparing to the black-crossing edge-pair, the red-crossing edge-pair is harder to discriminate which two nodes are connected. (c) Edge bundling without considering the actual relationship of the nodes will cause another kind of visually ambiguity.

Through our ambiguity-free edge-bundling framework, our network visualization system can be used to visualize the middle size, unstructured, and small-world graphs with a more aesthetic edge layout. Our ambiguity-free edge-bundling framework improves the effects that the user can perceive the individual relationships represented by the edges more accurately, since the edges that have the same target or source nodes are merged and an ambiguity-free mechanism is introduced to avoid the edge ambiguity problem. Our framework uses a quadtree structure to overcome the time complexity issue, so the user can easily interact with our graph layout in real-time. To provide the user an easy-to-use interactive user interface, we also introduce a novel detail-on-demand concept to make the user to be able to control the level-of-details regionally by painting the graph or network directly.

2. Related Work

Visual clutter in graph or network visualization has been widely studied. Many methods have been proposed to alleviate the visual clutter when the node and edge densities are increased in the graph visualization, since the visual clutter is particularly troublesome if the users want to incorporate readability considerations in graph layout [16]. A very good survey on general visual clutter reduction techniques can be found in [4].

Existing visual clutter reduction techniques can be categorized into three approaches: node adjustment, edge curving, and distortion oriented methods. Node adjustment approach rearranges the nodes' positions to minimize the edge density, crossing, and occlusion to avoid the content confusion. However, this approach cannot adapt well in practical use, since it is quite difficult to obtain an appropriate rearrangement with dense edges [3, 7]. In addition, it may not be suitable if the nodes' positions have semantic meanings, e.g. the nodes represent the cities on the map. Another attractive approach, distortion-oriented methods, distorts the nodes' positions or sizes while still maintaining the user's mental map model [13]. Through these methods, the user can interactively use a fish-eye-like tool to enlarge some regional areas of the graph or network to check the details of the interesting or cluttering areas. Many extensive methods have been presented in the fields of information visualization [9, 10, 5, 6], human-computer interaction [19, 11], etc.

The edge curving approach can be divided into two categories: edge bundling and edge dispersing. The edge dispersing approach disperses the edges away from one local region, so the underlying pattern can be revealed. EdgeLens [24] provides the user to interactively bend the edges away from one's focus without altering the nodes' positions, and then further opens up a sufficient space to disambiguate the relationship between the nodes and edges. Another promising approach to reduce the visual clutter is to reduce the excessive edges. Based on visually bundling the adjacency edges and preserving more space for distinguishing, edge bundling approach merges the edges that share the specific criteria. FlowMap Layout [15] proposed by Phan et al.has an encouraging result on specific graphs. They use hierarchical binary clustering on a set of nodes, positions, and flow data, and then route the edges which share the common destination into single- or multisource graphs. Hierarchical Edge Bundle [8] proposed by Holten is designed for visualizing the dataset containing both of adjacency relationship and hierarchical structure. To draw the edge linked two leafs, the edge is curved according to the path connecting two nodes on the hierarchical tree structure, and then it bundles the edges together if these edges

share the common path segment on the hierarchical tree. Qu et al. [17] proposed a novel edge-clustering method for node-link diagram. They cluster the edges together based on the intersections with the edges in the Delaunay triangulated mesh of nodes, and hence may produce zigzag edges. Cui et al. [2] proposed a similar edge clustering method based on the nodes' geometrical information. The significant difference from Qu et al. [17] is that they use a uniform grid structure to sample the control points, and further generating the control mesh to guide the edge curving. However, it aggravates another edge ambiguity problem as shown in Fig. 2(c).

3. System Design and Overview

While an edge represents the adjacency relationship, it also introduces incorrect relationship if the edge was inappropriately closed to some unrelated nodes. Fig. 3 illustrates some possible edge ambiguity cases in a three-node graph. If we use straight-line edges to form the graph as shown in Fig. 3(a), it comes out that the middle node seems to have the individual relationships with the upper node and the lower one, respectively. However, if we use curves to represent the edges, several configurations can be revealed as shown in Fig. 3(b) ~ Fig. 3(e). Besides these cases, even though the straight-line green edge shown in Fig. 2(a) does not actually overlap the red node, it is still difficult to discriminate the real connection from the node-edge nearness. Another case is that, if two straight-line edges shared no common node but cross with each other at an acute angle, it is difficult to rapidly interpret which two nodes are connected or not as the red edges shown in Fig. 2(b). Acute crossing angles cause more visual confusion when rapid visual interpretation is needed [22]. This also happens in curved edge layout if two curved-edges were inappropriately bundled as the red curved-edges shown in Fig. 2(c).

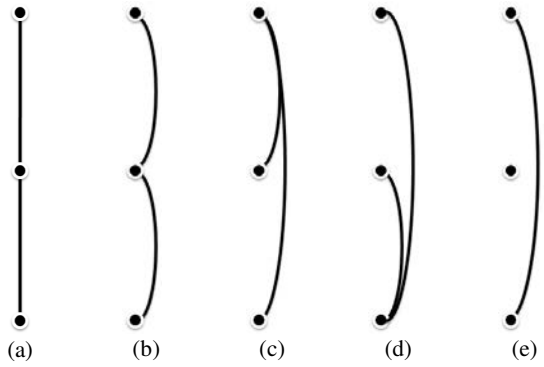


Fig. 3. The ambiguity caused by straight-line edge style. When linking the nodes by straight-lines, the graphs shown in (b) ~ (e) will all become (a).

Our observation for a good curved-edge layout reveals three common characteristics: (1) the curvature of each point on the curved-edge should be as small as possible; (2) the curved-edges could alleviate the edge ambiguity arising from the straight-line edges; (3) merging curved-edges to open up more display space without causing further edge ambiguity should be taken into consideration. Keeping the curvature of the curved-edge small could make the edge smoother and easy to track, and the rest characteristics are aimed to reduce the visual clutter caused by dense and crossing edges while maintaining the consistency between the visual cognition and real data's relationship. Our framework attempts to achieve these characteristics, and also provides the user an efficient and real-time curved-edge layout.

For some applications, the nodes' positions have semantic meanings, such as the geometric information of the cities on a map. Hence, in this case, the geometric attributes should be retained without noticeable displacement. In other cases, the dataset may only contain the adjacency relationship, e.g. social network data. In order to make our framework more generic to adapt to existing graph layouts and relation dataset, we assume that the nodes' positions in the input graph or network have already been decided. If the dataset has no position information in advance, a force-based model [1] is first applied to compute the initial nodes' positions.

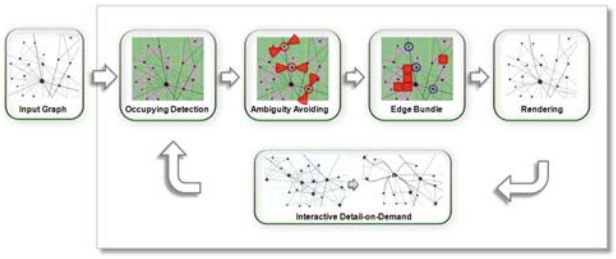


Fig. 4. The system overview.

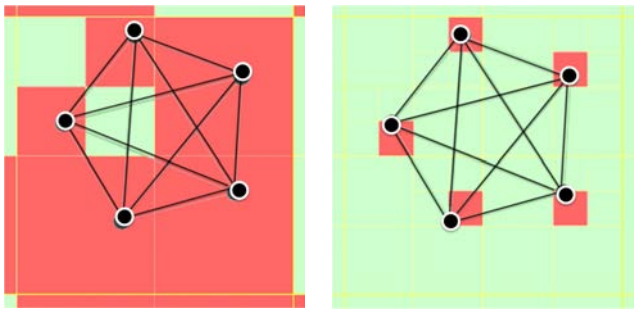
Fig. 4 illustrates the overview of our system. We use a quadtree [18] structure to decompose two dimensional display space according to the nodes' positions in the input graph, and this tree data-structure enhances the efficiency in the following steps. Our system consists of five major steps: (1) quadtree construction and occupying detection (Sec. 4.1), (2) edge ambiguity avoiding (Sec. 4.2), (3) curved edge bundling (Sec. 4.3), (4) rendering (Sec. 5.1), and (5) interaction (Sec. 5.2). The quadtree construction and occupying detection step constructs a quadtree structure by using the nodes in the input graph and detects each edge in the graph that what quadtree cells it passes through. The edge ambiguity avoiding step detects if there exists edges passing nearby one or more unrelated nodes, and routes these edges away from the unrelated nodes in local region. The curved-edge bundling step geometrically bundles the edges together while taking the edge ambiguity into consideration to open up more space, and further enhances the visual discrimination between the related nodes and unrelated ones. Finally, the rendering and interaction steps provide the final visualization and interactive control for the user.

4. Ambiguity-Free Edge Bundling

4.1 Quadtree Construction and Occupying Detection

In order to achieve real-time interaction, an efficient data structure, quadtree, is first deployed. Each node of the input graph is first inserted into a quadtree structure according to its position. Fig.

5(a) simply illustrates the result after inserting five nodes into a quadtree in two dimensional spaces. Here we define two types of the quadtree cells: RedCell and GreenCell. A RedCell means that it contains exactly one node in its local region, and a GreenCell means that it contains nothing and is available for further use. Since the width of the RedCell may be too large to waste too much space, after inserting the nodes into the quadtree, we further subdivide the RedCell to a proper width to release much more GreenCells. Fig. 5(b) illustrates the result after conducting the subdivision, so the nodes can be put into a RedCell with a proper width compared to the original result shown in Fig. 5(a).



(a) (b)

Fig. 5. Occupying detection with a quadtree structure. (a) The original quadtree structure after inserting five nodes. (b) The cells occupied by the nodes are subdivided to be a proper width.

After constructing the quadtree cell map, we need to analyze the occupied state of each edge in the straight-line style on the cell map. If one edge on the cell map occupied only GreenCells, that means it potentially does not have edge ambiguity problem. On the other hand, if one edge occupied one or more RedCells, the edge ambiguity problem may occur in those RedCell regions. The basic idea of our approach is that the curved-edge must route and bypass the RedCells detected in the straight-line mode. Moreover, in each GreenCell on the cell map, we detect what edges pass through it, and if two or more passed edges share the common nodes, they are bundled together in the local region. The bypass and bundle effects are controlled as local as possible, since the shortest path from one node to the other one is a straight-line.

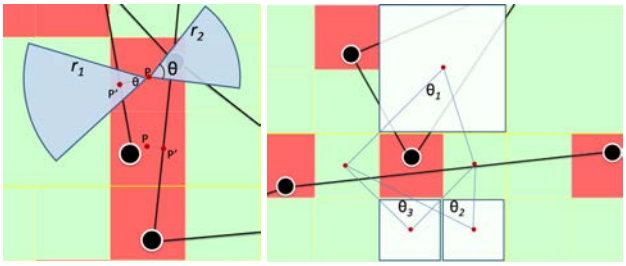
4.2 Edge Ambiguity Avoiding

As mentioned in Sec. 3, an edge passes inappropriately close to some unrelated nodes will cause the edge ambiguity problem and further provides incorrect relationship between the nodes. To avoid the edge ambiguity, we enforce the curved-edge to bypass these nodes. In Sec. 4.1, we have constructed a quadtree cell map and detected the cells occupied by each node and edge. If there is one RedCell which has already been occupied by a node but still passed by one edge, we first calculate the projection point P' of the RedCell's center P on the edge, and then according to the vector $\overrightarrow{PP'}$, we spread out two circular sectors in two directions $\overrightarrow{PP'}$ and $\overrightarrow{P'P}$ for searching the candidates of bypassing GreenCells. The central angle θ and radiuses r_1 and r_2 of the two circular sectors are determined by Eq. (1). Fig. 6(a) illustrates the idea for searching the bypassing GreenCell candidates.

$$\begin{aligned} \theta &= (l / L) * \pi \\ r_1 &= k * L * p \\ r_2 &= k * L * (1 - p) \\ p &= (l + L) / 2L \end{aligned} \qquad (1)$$

where k is a user specified detection range (suggested value: 1, ...,4), $l = \left\| \overrightarrow{PP'} \right\|$ is the distance between P and P', and L is the width of the occupied RedCell.

In Sec. 3, we mentioned that one of the characteristics of good curved-edge layout is keeping the curvature of each point on the curve as small as possible. To achieve this, we calculate all of the turning angles of all GreenCell candidates GC_i. Each turning angle θ_i is formed by the cell center of GC_i, preGreenCell, and postGreenCell, where the preGreenCell and postGreenCell are determined from the list of the GreenCells occupied by the same edge according to the projection points of the cell centers on the edge. After calculating all of the turning angles of all candidates, we select the one which has the maximal turning angle to be our bypassing GreenCell. Fig. 6(b) illustrates the idea for determining the best candidate for curved-edge bypassing.



(a) (b)

Fig. 6. Edge-ambiguity avoiding. (a) Searching bypassing GreenCell candidates. (b) The best bypassing GreenCell is decided by selecting the candidate with the largest turning angle. In this example, θ_3 is the best candidate.

4.3 Curved-Edge Bundling

In Sec. 3, we mentioned that the visual clutter in most graph layouts is caused by dense edges. Hence, we bundle the edges together to open up more space for better visual discrimination while avoiding information losing or visual confusion after edge bundling. To prevent bundled edges losing their information of source or target nodes visually, two or more edges can be bundled together only if they satisfy the condition that each of them shares a common node. This condition guarantees that, even if two edges are geometrically close, they will not be bundled together. Otherwise, it may generate additional incorrect relationship. Fig. 2(c) illustrates the additional incorrect relationship between the node pairs after bundling the edges without common node condition.

We use Catmull-Rom spline which has a well local control property as our curve model. To curve one edge, we add control points to the edge's spline model and it will guarantee that each control point will be hit smoothly. In Sec. 4.2, we have decided the bypassing GreenCells, the control points of the bypassing GreenCells are calculated as:

$$CP = P' + \left(L_r + \left(L_r + L_b\right)\right) * \overrightarrow{PP_b} \qquad (2)$$

where CP is the control point of the spline, P is the center point of the occupied RedCell, P' is the projection point of P on the edge, P_b is the center point of the bypassing GreenCell, and L_r and L_b Lb are the width of the occupied RedCell and bypassing GreenCell, respectively.

To bundle the edges together, we enforce the edges to pass through the specific control points. For every GreenCell that the edge passed, we check whether the GreenCell has other passed edges that share common nodes with the edge or not. If it does, we use Eq. (2) to calculate one control point and add it to the edge's spline model. After calculating all of the control points of one edge, we sort these control points according to their projection points on the straight-line edge, then the control points are aligned in correct sequence for constructing the spline.

5. Rendering and Interaction

5.1 Rendering

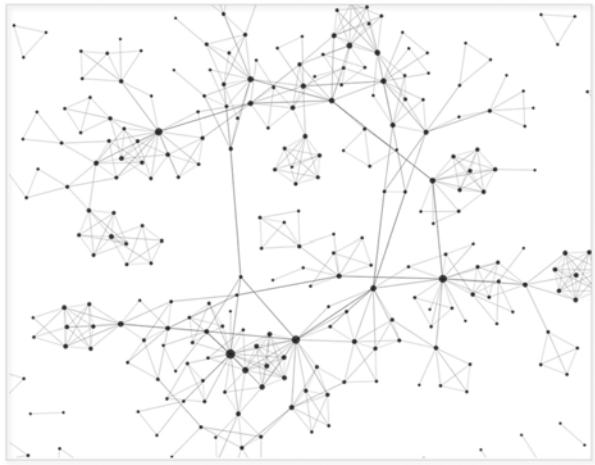
In most cases, we cannot avoid the existence of curves' overlapping. In order to make the dense curves as clear as possible, we draw the curves with different alpha values. In high level of view, it is apparent that long curves occupied larger amounts of screen space. In this case, we should draw the long curves with a lower opacity than the short curves, and this is also a better visualization since the user mostly tries to construct the overview of the whole graph in high level of view, in which lower drawing opacity of the long curves helps to emphasize the short curves and further enlarges the amount of edges that the user perceives. On the contrary, in lower level of view of the graph, the user now is mostly trying to discriminate the distinct relationship between different node pairs. Since a long curve takes more cognitive effort to ascertain which two nodes are connected, in this case, drawing long curves with higher opacity can help to perceive the information quicker.

(a) (b) (c)

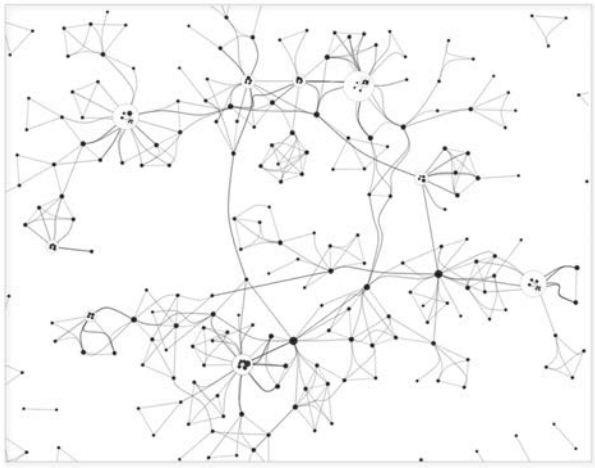
Fig. 7. The level-of-detail control. One region is simplified by merging the underlying connected nodes. (a) The original uniform graph. (b) After modifying the detail map, the connected nodes in one cell are combined if the number of connected nodes is greater than a user-specified value. (c) The result after merging the nodes further which have been combined already.

5.2 Interaction

In our network visualization system, an interactive level-of-detail mechanism is also provided to enable a comprehensible visualization for dealing with a common small-world issue. Many real world relation datasets have the small-world characteristics [12], such as social network data, bibliographic reference data, software structure data, etc. The small-world characteristics come out a high degree of clustering in graph, and make the graph have a small average path length comparing to other random graphs of the same scale or size. The frequent occurrence of high-degree clustering always makes the graph incomprehensible. Hence, we use an interactive level-of-detail mechanism that only modifies the visual representation of the clustered and small average curve length subgraph without losing their underlying relationships while providing the user the control of detail-on-demand.



(a)



(b)

Fig. 8. The comparison between the original graph layout with small-world characteristics and the result of merging high-degree clustering nodes. (a) In traditional graph layout, small-world characteristics make the graph more difficult to comprehend. (b) After the user interactively modifying the level-

of-detail map, it is easier to comprehend the graph within a specific level of view.

In the interaction process, the user can paint a level-of-detail map on display space. For example, if the user wants one local region to be simpler to provide a higher level of view, he or she can just use the mouse cursor to click on that local region to make the local level-of-detail map sparser; on the contrary, he or she can also make the local level-of-detail map denser if the local details are needed. After the level-of-detail map has been modified, our system checks each cell on the level-of-detail map to meet our detail criterion. If one cell contains more than a specific number of connected nodes, we will combine those connected nodes into a larger parent node, apply a simple graph layout to its child nodes, and converge their edges which connect to the nodes outside the cell. The connected nodes are defined as the nodes which are connected to the others in the same cell.

Fig. 7(a) and Fig. 7(b) illustrate the idea of the level-of-detail map, the user can modify the level-of-detail map and our system will further combine the nodes below the same cell. A circular layout is used to simplify the subgraph in one parent node, and the parent node can also be combined by the higher-level parent node if it meets the detail criteria under one cell. Fig. 7(c) illustrates the result that the nodes are combined multiply. Moreover, if the user wants to visualize the details of the combined nodes later, it can be done by modifying the level-of-detail map in the specific region, and our system will check whether there exists a parent node that occupied more than one cell in the level-of- detail map, and then release the combined nodes for one combined level. We can also use this method to alleviate the visual clutter caused by small-world characteristics. Fig.8(b) illustrates that it is easier to comprehend the modified representation of graph comparing with the original one shown in Fig.8(a) with several high-degree clustering nodes.

In each interaction step, we also use a quadtree data-structure to partition our display space and analyze the nodes deployment. It demonstrated high performance with the scale of hundreds of nodes and force-based layout. It can provide real-time interaction by using a desktop PC with a Intel Core-2 1.66 GHz CPU and 2GB memory.

6. Result and Discussion

Fig. 9 shows the visualization result of an academic social network in Taiwan which contains 2,450 professors as the nodes and 2,622 co-advise relationship as the edges. If we use straight-line edges to visualize this graph which has the small-world characteristics, it is hard to comprehend at the first look since there are a lot of edge-node over-passing. After bundling the edges together while considering the edge ambiguity, it is easier to get the real connection between the nodes without too much visual efforts.

In Fig. 10, a co-authorship network of scientists working on network theory and experiment is visualized. The network data was provided by [14] which contains 1,589 nodes and 2,742 edges. The graph has highly local-community property that there exists a lot of near completed subgraphs. Hence, we interactively combine these communities, and without edge gathering at small region, it is easier for learning higher level of view. The user can further drill down for more details, and further release the communities on demand.

In Sec. 4.1, we subdivided the RedCell to make the cell width as small as possible to release much more GreenCells. In our observation, if the cell width is too small, the edge-bundling mechanism will produce a frequent-turning curve layout, as know as zigzag curve. The zigzag curves will cause the visual clutter in which it is difficult to perceive the connection between two nodes. On the contrary, if the cell width is too large, the edge ambiguity avoiding mechanism might take less effect due to the missing of bypassing cell detection. In our experiment, the maximum cell width L' should be decided with consideration of both the node amount on the display and the area of the display space, which can be formulated as the following equation:

$$L' = O\left(\sqrt{A/4N}\right)$$

where A is the area of available display space and N is the number of nodes. In some cases, the average node width would be greater than the suggested maximum cell width, which means the probability of the curved-edges passing near the nodes' bound might be high. In this case, our edge ambiguity avoiding mechanism will not work well due to the insufficiency of available display space. However, in general cases, the suggested maximum cell width is larger than the average node width in which our edge ambiguity avoiding mechanism performs well.

7. Conclusion and Future Work

Graph representation is a common and well-known approach to visualize network data. In this paper, we present a generic and efficient ambiguity-free edge-bundling method to improve the edge layout which plays an important rule to indicate the relationship between the nodes. Our contributions can be listed as the following:

- By applying ambiguity-free edge-bundling, it is easier for the user to perceive the relationship between the nodes with less ambiguity.
- The user can discriminate between distinct edges with less visual loading.
- No interactive effort is needed. The bundled curved-edge layout is automatically generated according to the deployment of nodes and edges.
- A novel detail-on-demand concept is also presented that the user can customize the distribution of level-of-details on display, a.k.a. level-of-detail map. This concept provides more degree of freedom that how deep the user wants to see in a specified local region.

In the future, we will make efforts on improving the proper quadtree cell width to adapt to different density of nodes and edges locally. By considering the local deploying properties, the overall performance of ambiguity-free edge-bundling will be better. In addition, in each curved and bundled edge, we can further improve the routing algorithm to perform more aesthetic curved-edge layout. This might be achieved by extending the detection of ambiguity bypassing and deciding the route path based on both of bundling and bypassing variables simultaneously.

8. REFERENCES

[1] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[2] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008. (Information Visualization 2008 Conference Proceedings).

[3] P. Eades and R. Tamassia. Algorithms for drawing graphs: An annotated bibliography. Technical report, 1988.

[4] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):741–748, 2007. (Information Visualization 2007 Conference Proceedings).

[5] G. W. Furnas. The fisheye view: a new look at structured files. *In Readings in information visualization: using vision to think*, pages 312–330. Morgan Kaufmann Publishers, 1999.

[6] E. R. Gansner, Y. Koren, and S. C. North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[7] I. Herman, G. Melanc , on, and M. Scott Marshall. Visualiation and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24 – 43, 2000.

[8] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006. (Information Visualization 2006 Conference Proceedings).

[9] T. A. Keahey and E. L. Robertson. Techniques for non-linear magnification transformations. *In IEEE Information Visualization 1996 Conference Proceedings*, pages 38–45, 1996.

[10] T. A. Keahey and E. L. Robertson. Nonlinear magnification fields. *In IEEE Information Visualization 1997 Conference Proceedings*, pages 51–58, 1997.

[11] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortionoriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[12] S. Milgram. The small world problem. *Psychology Today*, 1(1):60–67,1967.

[13] K.Misue, P. Eades,W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[14] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[15] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. *In IEEE Information Visualization 2005 Conference Proceedings*, pages 219–224, 2005.

[16] H. C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000.

[17] H. Qu, H. Zhou, and Y.Wu. Controllable and progressive edge clustering for large networks. *In Graph Drawing 2006 Conference Proceedings*, pages 399–404, 2006.

[18] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[19] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. *In ACM CHI 1992 Conference Proceedings*, pages 83–91, 1992.

[20] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.

[21] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. *In IEEE Information Visualization 2004 Conference Proceedings*, pages 199–206, 2004.

[22] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[23] D. J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[24] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. *In IEEE Information Visualization 2003 Conference Proceedings*, pages 51–58, 2003.

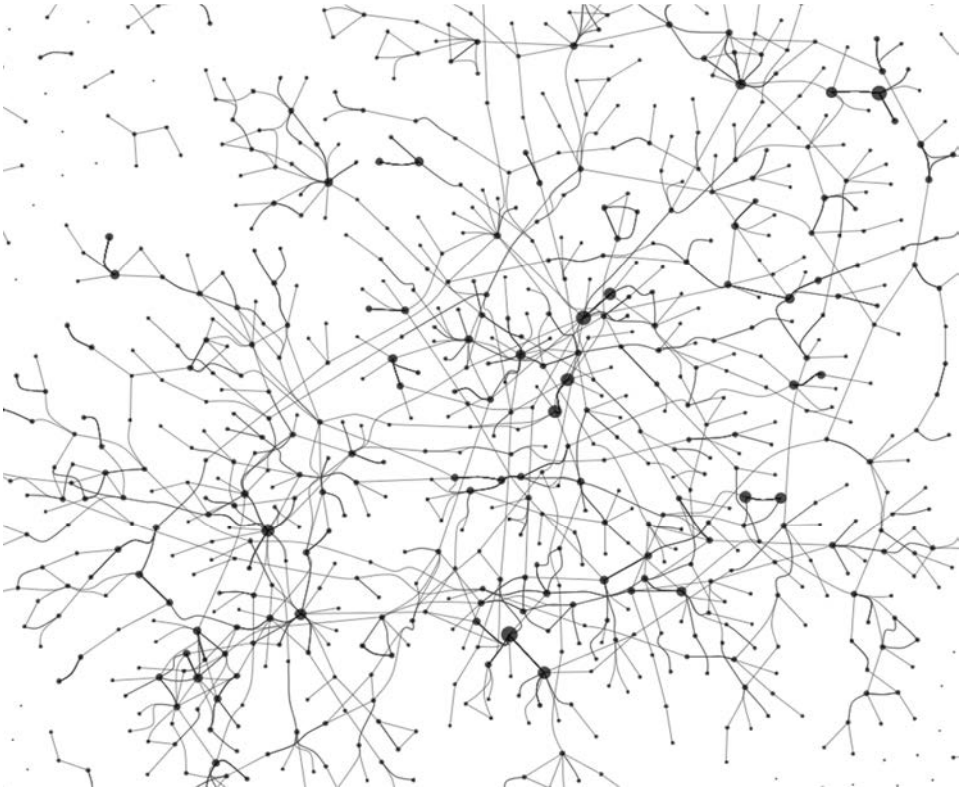


Fig. 9. The result of visualizing an academic social network in Taiwan which contains 2,450 nodes and 2,662 edges.

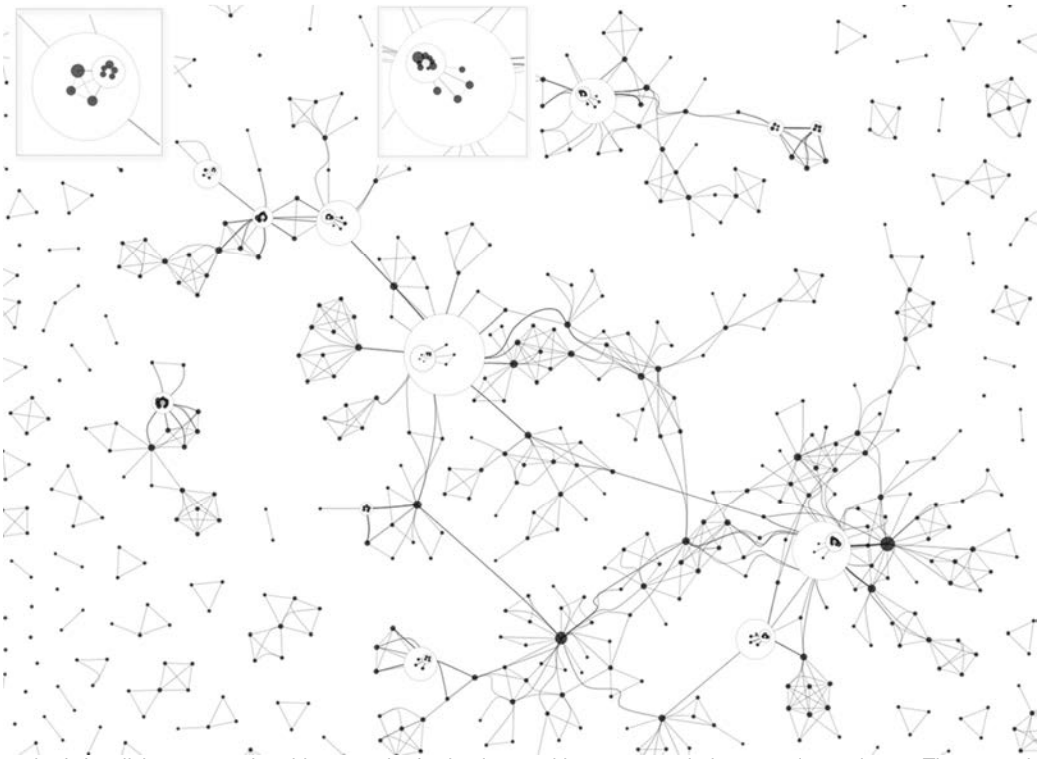


Fig. 10. The result of visualizing a co-authorship network of scientists working on network theory and experiment. The network contains 1,589 nodes and 2,742 edges.