Controllable and Progressive Edge Clustering for Large Networks

Huamin Qu, Hong Zhou, and Yingcai Wu

Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
{huamin,zhouhong,wuyc}@cse.ust.hk

Abstract. Node-link diagrams are widely used in information visualization to show relationships among data. However, when the size of data becomes very large, node-link diagrams will become cluttered and visually confusing for users. In this paper, we propose a novel controllable edge clustering method based on Delaunay triangulation to reduce visual clutter for node-link diagrams. Our method uses curves instead of straight lines to represent links and these curves can be grouped together according to their relative positions and directions. We further introduce progressive edge clustering to achieve continuous level-of-details for large networks.

1 Introduction

Many visualization problems can be modeled using node-link diagrams or networks, where nodes represent data elements and links their relationships. For example, hyperlinks among Internet webpages, citations in scientific papers, traffics between telecommunication switches, and airline routes can all be represented by node-link diagrams. As the amount of data from real world keeps increasing, visual clutter becomes a very serious problem for large networks and greatly affects the effectiveness of networks for conveying information.

Visual clutter is usually caused by an excessive number of nodes and links. Visual clutter for edges caused by too many edge crossings is also called edge congestion [1] [2]. Too many crossings of links will obscure some nodes and links in the graph. Various filtering and clustering methods [3] can be used to effectively reduce the number of nodes and thus the number of links. However, simply reducing the number of nodes is not a practical solution for some applications. For example, in typical airline routes, removing a node will cause the lost of route information for an airport. Clustering of nodes may not be a good solution either. Users may have problems to relate the links coming in or out of virtual clustered nodes to real links.

Edge congestion is a challenging problem and many approaches have been proposed. Edge crossings can be reduced by rearranging the nodes and edges. Various force-based or energy-based node layout algorithms for graphs [4] can generate good layouts for small-size graphs according to some aesthetic criteria including minimum edge crossings. However, for large graphs, edge crossings usually cannot be reduced to a satisfying level. Some researchers try to totally avoid edge crossings by making the graph planar [5]. However, large graphs usually cannot be drawn in a planar way. For some applications such as communication and transportation networks, the semantic meanings of the node positions limit the room of adjustment for nodes.

Edge congestion can be alleviated by merging edges and drawing edges as curves. Phan et al. [6] proposed a *flow map layout* to reduce visual clutter by merging edges. The flow map layout is generated by hierarchical clustering. The node positions are distorted but relative positions are maintained in their algorithm. Their method is mainly designed for single-source node-link diagrams although the authors mentioned that their methods can be extended for multiple-source networks by overlapping multiple flow maps. Carpendale and Rong described an interesting technique to examine edge congestion around node areas by adjusting the edge position if the edge passing through the node [1]. They used an edge-displacement algorithm to curve away all edges from the region of interest. Wong et al. [2] introduced *EdgeLens* to manage edge congestion in graphs. They pointed out that the relation between vertices might be displayed ambiguously if their edges overlap. Their solution is to introduce a lens which displaces edges in a local area with dense edge overlapping and reveals hidden information in that area and clarifies graph structures. While the *EdgeLens* is quite effective to make the edges in a small area more discernable, the visual clutter of the overall graphs is not reduced and the large-scale patterns cannot be effectively revealed by this method. Wong and Carpendale further introduced *Edge Plucking* [7], an interesting interactive technique which allows users to temporarily pluck edges apart to clarify node-edge relationships.

In this paper, we describe a controllable and progressive edge clustering method to address edge congestion for large network visualization. Our research follows the same direction of some previous work [1, 6]. Our method first connects the nodes by Delaunay triangulation and then sets some control points on the Delaunay edges of these triangles. After that, we convert all links into a series of paths consisting of these control points. By adjusting the number and positions of these control points, different levels of edge clustering can be achieved. By setting a minimum distance between these control points and the original nodes, the ambiguity cases in the traditional node-link diagrams can be avoided. By grouping links together, the high level linkage patterns related to the whole node-link diagram can be revealed. Compared with node-clustering methods, our method can reveal the linkage information for real nodes instead of clustered virtual nodes. Compared with force-based or energy-based approaches, our method is geometry based and all computations (e.g., Delaunay triangulation, ray/triangle intersection, and K-Means clustering) can be done very efficiently and can be accelerated by graphics hardware. Our method is easy to implement and the generated layouts are visually appealing. In addition, our approach gives users great flexibility for layout generation. Users can easily and dynamically change the size of “protected” areas around nodes and the levels of clustering for links. We further introduce progressive edge clustering to achieve continuous level-of-details for large networks.

2 Controllable Edge Clustering

In this section, we introduce our method for edge clustering given a set of nodes and links. We assume that the positions of nodes have been computed by some other methods such as force-based models [4] and a relatively good initial layout has been obtained. We do not further distort node positions. Therefore, their original layout is preserved. Actually, for certain applications, node positions encode geographic information and any adjustment of node positions may cause confusion for users.

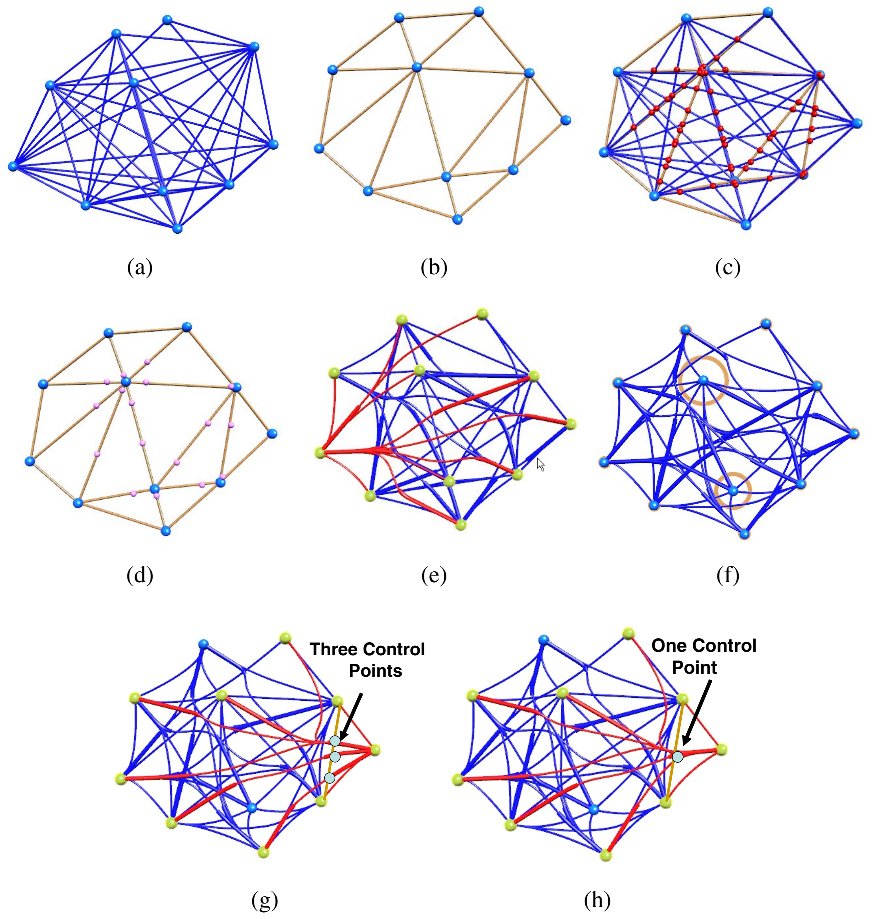

Fig. 1. Framework of our method: (a) Original node-link diagram; (b) Delaunay triangulation of the original nodes; (c) The intersection points of all links and Delaunay edges indicated by the red dots; (d) The control points computed by clustering the intersection points; (e) All links are forced to pass through the control points; (f) Protected areas for nodes; (g) Three control points on a Delaunay edge; (h) One control point on a Delaunay edge

Figure 1 shows the framework of our method. Figure 1 (a) shows the original node-link diagram. We first compute a Delaunay triangulation of the points given by the positions of the vertices (See Fig. 1 (b)). Then, for each link, we compute the intersection points of this link with the Delaunay edges (See Fig. 1 (c)). For each Delaunay edge, we cache all the links passing through this Delaunay edge and their intersection points with this edge. Then we assign one or more control points on each Delaunay edge and use these control points to cluster links (See Fig. 1 (d)). The control points can be clustered using the K-Means algorithm. Later, the number and positions of these control points can be adjusted by users to achieve different levels of clustering. After that, we force all the links to pass through these control points and compute a path consisting of these control points for each link (See Fig. 1 (e)). Some overlapping paths will be then grouped together by the system or by users. In this paper, all the paths are drawn as Nurbs curves.

By forcing all links to pass through control points on the Delaunay edges, we can easily solve the ambiguity cases for node-link diagrams. We can set up a **"protected area"** for each node by forcing the control points to be at lease certain distance away from the node. If only one control point is used on each Delaunay edge, the protected area for each node can be as large as half of the minimum distance from this node to its neighboring nodes. If two or more control points are used, then we can easily control the size of the protected areas for nodes. In practice, we can compute this size based on the importance of the nodes if any criterion for importance is given by users. The size of protected areas can also be adjusted by users during the visualization process. Figure 1f shows the protected areas for two nodes.

Because all links have to pass through the control points on Delaunay edges, then some link segments will be automatically clustered and some flow-map-style effect can be achieved. In practice, we can start from the coarse level and gradually refine edge clustering. First, we assign only one control point on each Delaunay edge and then we compute the shortest paths consisting of these control points for all links. Then we examine all incoming and outgoing links for each node. For all paths passing through the same set of control points, we group that parts of links together. After finding all the overlapping path segments for all links, we put more control points on the Delaunay edges so the directions of links will be closer to their original directions.

The level of clustering can be controlled by users. We provide an interface which allows users to dynamically change these parameters so the node-link diagrams can be examined at different level-of-details. There are two major parameters users can adjust: the number of control points on Delaunay edges and their positions. We use the following principles to position control points: For short or unimportant Delaunay edges (i.e., edges with only a few or even zero link passing through), there are fewer control points; The positions of control points will be close to the real intersection points of the links and the Delaunay edges. Figure 1 (g) and (h) show different levels of edge clustering by adjusting the number and positions of control points on a Delaunay edge. From the figure we can see that our layout is visually appealing and our method gives users great flexibility to control the final layout.

3 Progressive Edge Clustering

If there are too many nodes and links, even clustering all links on all Delaunay edges cannot solve the visual clutter problem. Inspired by progressive mesh simplification [8], we propose progress edge clustering by collapsing Delaunay edges. As illustrated in

Figure 2, if we collapse one Delaunay edge (or two nodes), then the total number of Delaunay edges will be reduced and the links will become smoother and more space will be available for positioning nodes and links. Progressive edge clustering is based on collapsing Delaunay edges one by one so that at each step only one Delaunay edge disappears. There are two major issues for progressive edge clustering: the order of collapsing for Delaunay edges and the new position of the node(s) after collapsing one Delaunay edge. We use the following criteria to determine the order in which vertices collapse: The length of the Delaunay edges; The number of links passing through the Delaunay edges; After the collapsing, the maximum distance of the newly positioned paths to the original links and the curvatures of these paths. The short Delaunay edges with less links passing through will be clustered first.

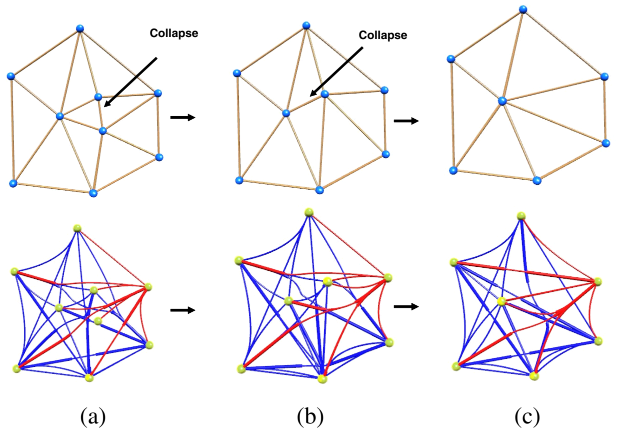

Fig. 2. Progressive edge clustering: (a) Traditional node-link diagram; (b) Collapse one Delaunay edge; (c) Collapse another Delaunay edge

For the new position of the collapsed nodes, we consider two choices. First, the nodes are still at their original positions. Only the “road” becomes unavailable so all links passing through this Delaunay edge have to go through the neighboring control points. Second, these two nodes overlap and share one position. For example, these two nodes can assume the position of one of the original nodes or are moved into a new position such as the middle point of the Delaunay edge. We provide an interface to allow users to decide the new position(s) of the nodes associated with the collapsed Delaunay edge. Please notice that our focus is not to cluster nodes, as there are many excellent papers on this topic. Therefore, these nodes only overlap in 2D space and links are still associated to their original nodes instead of one abstract virtual node. We want to group more links together by progressive edge collapsing so that different level-of-details for large networks can be achieved and the visual clutter problem can be alleviated. Figure 2 illustrates progressive edge clustering.

4 Conclusions and Future Work

In this paper, we proposed a geometry-based edge clustering method for traditional node-link diagrams. Our method is easy to implement and can generate visually appealing layouts for large networks in real time. By setting control points on Delaunay edges to control the flow of links we can easily obtain different levels of edge clustering. The classic ambiguity cases for node-link diagrams can be easily solved by setting a protected area for each node. We also proposed progressive edge clustering so that continuous level-of-details can be generated for large graphs.

In the future, we plan to further analyze the intersection points cached on Delaunay edges and automatically generate some road-map style layout for large graphs. Our method can be further improved if combined with some node layout adjustment algorithm to make edge merging more effective. We plan to use polylines with very small round corners to display clustered links. More sophisticated progressive edge clustering techniques which take the graph's topology into consideration will also be developed.

Acknowledgments

This work is partially supported by HKUST grant DAG 04/05 EG02. We would like to thank the anonymous reviewers for their valuable comments.

References

1. Carpendale, M., Rong, X.: Examining edge congestion. CHI '01 extended abstracts on Human factors in computing systems (2001) 115–116
2. Wong, N., Carpendale, M., Greenberg, S.: Edgelens: An interactive method for managing edge congestion in graphs. IEEE Symposium on Information Visualization 2003 (2003) 51–58
3. van Ham, F., van Wijk, J.J.: Interactive visualization of small world graphs. (2004) 199–206
4. Noack, A.: Energy-based clustering of graphs with nonuniform degrees. Graph Drawing (2005) 309–320
5. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: Visualizing non-planar diagrams in a planar way. J. Graph Algorithms Appl. **9**(1) (2005) 31–52
6. Phan, D., Xiao, L., Yeh, R., Hanrahan, P., Winograd, T.: Flow map layout. IEEE Symposium on Information Visualization 2005 (2005) 219–224
7. Wong, N., Carpendale, S.: Interactive poster: Using edge plucking for interactive graph exploration. Poster in the IEEE Symposium on Information Visualization (2005)
8. Hoppe, H.: Progressive meshes. Proceedings of ACM SIGGRAPH '96 (1996) 99–108