Energy-Based Hierarchical Edge Clustering of Graphs

Hong Zhou† Xiaoru Yuan‡ Weiwei Cui† Huamin Qu† Baoquan Chen§

The Hong Kong University of Science and Technology, Kowloon, Hong Kong†*
Peking University, Beijing, China‡
The University of Minnesota at Twin Cities, MN, USA§

Abstract

Effectively visualizing complex node-link graphs which depict relationships among data nodes is a challenging task due to the clutter and occlusion resulting from an excessive amount of edges.

In this paper, we propose a novel energy-based hierarchical edge clustering method for node-link graphs. Taking into the consideration of the graph topology, our method first samples graph edges into segments using Delaunay triangulation to generate the control points, which are then hierarchically clustered by energy-based optimization. The edges are grouped according to their positions and directions to improve comprehensibility through abstraction and thus reduce visual clutter. The experimental results demonstrate the effectiveness of our proposed method in clustering edges and providing good high level abstractions of complex graphs.

Keywords: Graph visualization, Edge clustering, Delaunay triangulation, Node-link diagrams, Hierarchies.

Index Terms: E.1 [Data Structures]: Graphs and networks; G.1.6 [Numerical Analysis]: Optimization—Constrained Optimization; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; H.5.0 [Information Interfaces and Presentation]: General.

1 Introduction

Graph is a popular tool to represent complex relationships. Information such as trade relationships among countries or cities, hyperlinks among Internet webpages, citations between literatures, dynamic biological reactions within a complex system, power grids, airline routes, road maps, and interpersonal relationships can all be modeled by node-link diagrams in which nodes represent data elements and links indicate their relationships.

However, as the amount of information increases, visual clutter becomes a major challenge to achieve effective visualization. Edge congestion [3, 16, 19], where excessive edge density in a region leads to edge-crossings and edge-overlappings, obscures nodes in these regions, and thus causes difficulties in understanding the information represented by the graph. One typical example is the maps of the flight routes of airline companies. Severe edge congestion can often be observed near major hub cities from which many flights originate. The edge density is so high that the entire region surrounding the hub cities can be fully covered by lines. It would be very difficult to trace individual flight route and extract useful information from the underlying graph in such cases.

The graph edge density can be alleviated by magnifying the congestion regions [2,5,6,10], merging and rearranging edges [3,4,12]. More recently, hierarchical edge clustering methods have been applied to avoid visual clutter in single-source graphs [15] and graphs with known hierarchical structures [9]. Through grouping the edges, visual clutter can be dramatically reduced without moving the node positions which may have semantic or geographic meanings in graphs such as road maps. However, the research on edge clustering, especially hierarchical edge clustering for general node-link graphs is scarce. In this paper, we propose a novel energy-based hierarchical edge clustering method for general node-link diagrams. Our method first performs Delaunay triangulation on graph nodes and segments original graph edges based on the computed Delaunay edges. Then adaptive sampling which reflects both node and edge densities in graphs are achieved. Clustering edges based on the adaptively sampled segments instead of uniformly sampled segments can generate more desirable results which maintain the topology information of the graph. The sampled segments are then filtered and hierarchically clustered based on our energy-based optimization method. The proposed energy function takes both locations and directions of edges into consideration during the edge clustering stage, in order to make spatially close and visually parallel edges cluster together. Finally, the clustered edges are dynamically grouped according to their positions in the hierarchy to improve comprehensibility through abstraction and thus reduce visual clutter. Users can slide our provided scrolling bar to dynamically explore the edge clustering results at continuous level of details in real time.

The rest of the paper is organized as follows. We briefly discuss related work in Section 2. In Section 3, details of our proposed hierarchical edge clustering algorithm are given, followed by experimental results and performance discussions in Section 4. Finally, we present our conclusions and future work in Section 5.

2 Related Work

Much efforts have been devoted to relieving edge congestion since over a decade ago. Without modifying the layout of the graph itself, directly magnifying the congestion regions with techniques such as Magic Lens [2] and fisheye views [5, 6, 10] can make the central part larger and provide a focus+context view, but cannot relieve the edge confusion and occlusion. Edge crossings can also be reduced by rearranging the nodes and edges. One possible way to alleviate edge congestion is to merge edges and draw them as curves. Force-based or energy-based node layout algorithms [13, 14] have been investigated to generate good graph layouts for medium size graphs, based on some aesthetic criteria. However, for graphs with dense edges, edge crossings usually cannot be reduced to a satisfactory level. Ware et al. [17] discussed the perception of continuity in edge paths and showed that the three most important factors of the cognitive criteria of graph aesthetics are the length of the path, continuity, and edge crossings, in a decreasing order of importance. It can be observed that the most severe congestion occurs in the areas where edges heavily cross each other. NewBery [12] proposed the Edge Concentration method which reduces the number of edges while retaining the graph structural information. The proposed method eliminates edges through replacing the set of edges with the same set of source and target nodes by a special node called edge concentration node. The number of crossings is reduced sub-

*e-mails: zhouhong@cse.ust.hk, xiaoru.yuan@gmail.com, weiwei@cse.ust.hk, huamin@cse.ust.hk, baoquan@cs.umn.edu

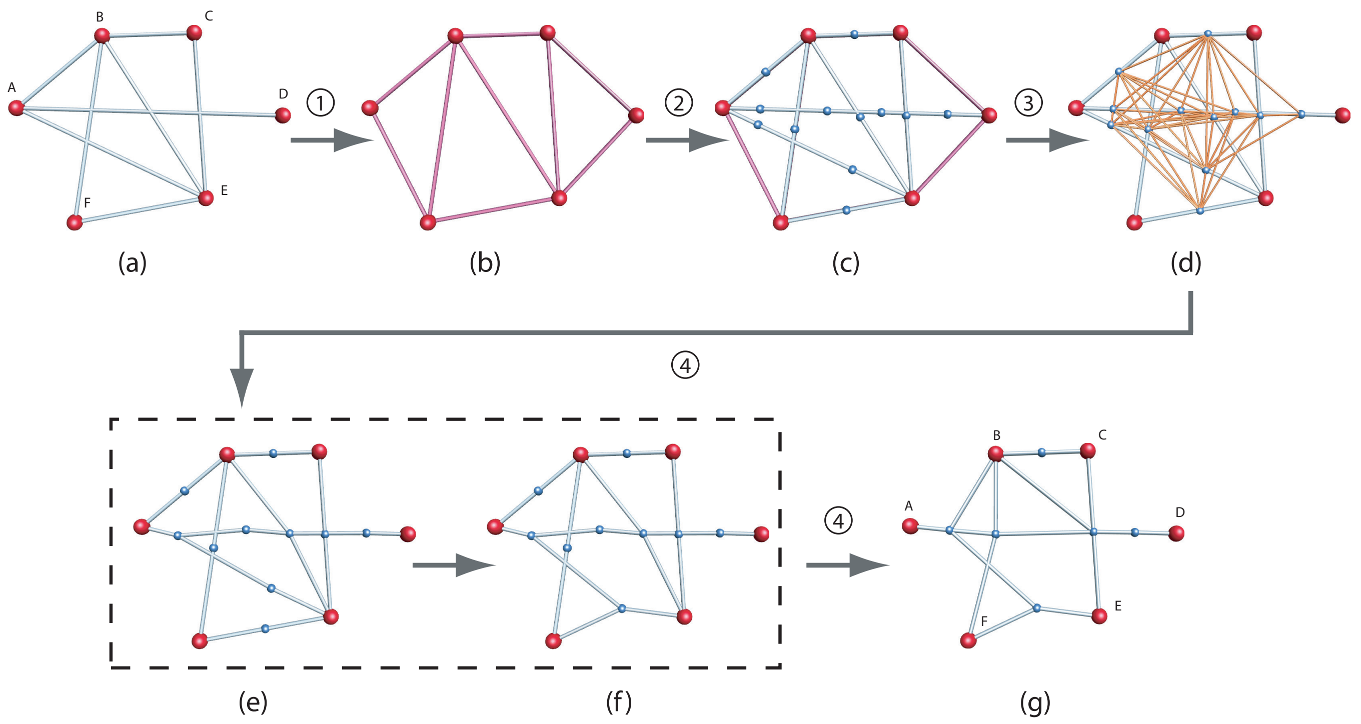


Figure 1: Hierarchical edge clustering algorithm overview. Step 1: performing Delaunay triangulation; Step 2: sampling the graph edges; Step 3: building the control map of the graph; Step 4: generating edge hierarchy (hierarchical clustering). The figures (e) and (f) enclosed in the dotted rectangle are the intermediate graphs during the hierarchical clustering.

sequently when the number of edges is reduced. Lin [11] further studied the computational complexity of Edge Concentration and showed that the problem is NP-hard.

Schemes of planar graphs [4] have been investigated to totally avoid edge crossing with certain constraints. Carpendale and Rong [3] described an edge-displacement algorithm which interactively adjusts the graph layout. In their algorithm, only edges are shifted to provide sufficient space to clarify relationships, while nodes remain at their original positions. EdgeLens [19] interactively curves edges of the node-link diagrams away from the center of focus (center of the lens) without changing the node positions. Sufficient space can be provided by the EdgeLens operations to discern ambiguous node and edge relationships and to reveal the underlying information while still preserving the node layout. However, the visual clutter is only reduced locally, and large-scale patterns cannot be effectively revealed. Above methods rely on the interaction with the user and are not applicable to static media such as magazine printouts. Edge Plucking [18] introduced by Wong and Carpendale allows users to interactively pluck edges apart to clarify node-edge relationships.

Hierarchical methods have been applied to graphs with known node hierarchy structures to avoid visual clutter and provide better abstractions of graphs [9]. Phan et al. [15] proposed a method for generating layouts of flow maps using hierarchical clustering. The algorithm minimizes edge crossings and distorts node positions while maintaining their relative positions. Their method mainly targets at single-source node-link diagrams. It is not clear how to extend it to general multiple source graphs. Recently, Gansner and Koren [7] presented an idea of minimizing an energy function for rerouting edges to achieve hierarchical edge clustering. The edge density is reduced by coupling groups of edges as bundled splines that share part of their routes. However, their method is specifically designed for circular layout graphs, while our method is for general node-link graphs. TopoLayout, a feature-based algorithm organizing multilevel undirected graph layouts by topological features, has been proposed by Archambault et al. [1]. The graph hierarchy is formed by collapsing subgraphs into single nodes according to recursively detected topological features. Instead of clustering the nodes, our method clusters the edges to build the hierarchy, which can keep the geographic meanings of the nodes in the graph. Qu et al. [16] proposed a controllable edge clustering method based on Delaunay triangulation to reduce visual clutter for large networks. It groups edges together which are represented by curved lines according to their geometry information. That method only allows merging on the Delaunay edges, while our new method can provide more flexibility and scalability to the edge clustering process.

3 Hierarchical Edge Clustering Algorithm

In this section, we first give the definition of node-link graphs. We then briefly overview our proposed energy-based hierarchical edge clustering algorithm which consists of four steps, i.e., performing Delaunay triangulation, sampling the graph edges, building the control map, and generating the edge hierarchy. We then discuss the details of each step followed by a brief description of the implemented interface enabling user exploration on the clustering results.

3.1 Basic Definition

A graph $G = (V,E)$ comprises a set V of n nodes, and a set E of m edges, where $V = \{v_i | i = 1,...,n\}$, $E = \{(v_i,v_j)|v_i,v_j \in V, i \neq j\}$. Each edge (v_i,v_j) connects nodes v_i and v_j. In the scope of this paper, we only consider undirected graphs, in which edge (v_i,v_j) is equivalent to (v_j,v_i).

We do not change the node positions during the clustering with the assumption that a relatively good initial layout of nodes has been computed based on other methods [13, 14] and the location of each node is unchanged during the clustering. For certain applications such as visualizing airline flight routes, it is important to preserve the node location.

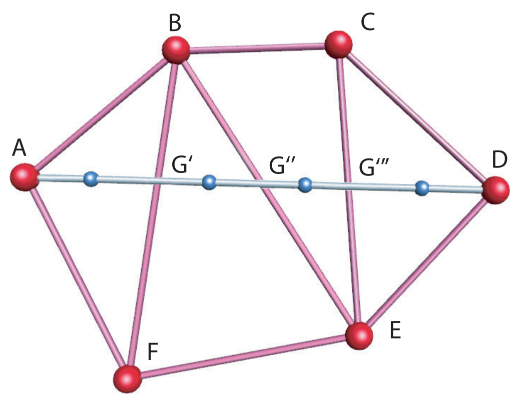


Figure 2: Sampling the graph edges based on Delaunay triangulation. Edge AD is intersected with Delaunay triangulation edges BF, BE, and CE. Four segments AG', $G'G''$, $G''G'''$, and $G'''D$ are sampled for edge AD.

Before clustering, in the representation of the graph G, each edge (v_i, v_j) is a straight line connecting two nodes v_i and v_j, as shown in Figure 1(a). Our edge clustering algorithm merges edges and displays a new edge layout in the form of polylines or curves according to the graph topology and edge distribution.

3.2 Algorithm Overview

As illustrated in Figure 1, our energy-based hierarchical edge clustering algorithm can be divided into the following four steps:

- Step 1: *Performing Delaunay Triangulation.* Delaunay triangulation is computed based on the node set V;

- Step 2: *Sampling the Graph Edges.* Each edge in the set E is subdivided into segments according to its intersections with the Delaunay edges computed in Step 1;

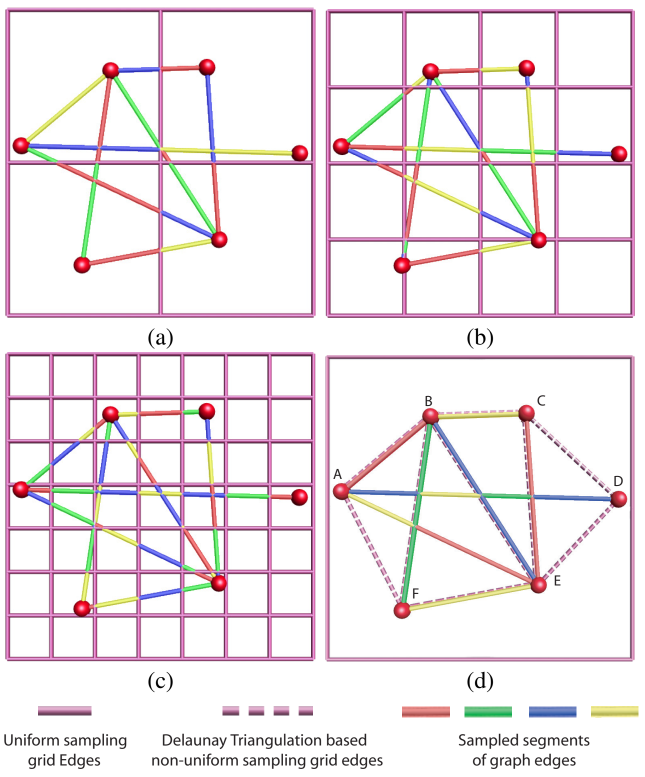
- Step 3: *Building the Control Map.* Sampled segments, represented by the midpoint of each, are filtered to form a new graph $G'(V', E')$, based on the criteria of Euclidian distance and the relative positions of each node pair in the set V';

- Step 4: *Generating the Edge Hierarchy.* Nodes in the new graph G' are hierarchically clustered based on our energy optimization. The edges in the original graph G are clustered accordingly.

In the following subsections, we describe each step in detail.

3.3 Step 1: Performing Delaunay Triangulation

Similar to the controllable and progressive edge clustering method [16], we first compute a Delaunay triangulation of the nodes in the original graph G. Delaunay triangulation is a classic method to generate triangles from points. Delaunay triangulation maximizes the minimum angle of all the angles of the triangles in the triangulation and tends to avoid sharp angles, thus the shapes of the generated triangles are usually very good. An example triangulation of nodes in the graph shown in Figure 1(a) is illustrated in Figure 1(b). The Delaunay edges are marked with red color. Recently, some GPU-accelerated Voronoi diagram and Delaunay triangulation methods have been proposed [8], thus thousands of points can now be triangulated in real time. In our algorithm, we take the advantages of Delaunay triangulation for the graph edge sampling in the next step.



(a)
(b)
(c)
(d)
A
B
C
D
E
F
Uniform sampling grid Edges
Delaunay Triangulation based non-uniform sampling grid edges
Sampled segments of graph edges

Figure 3: Comparison between uniform sampling and our proposed non-uniform sampling scheme based on Delaunay triangulation: (a) - (c) uniform sampling schemes with regular grids at difference sampling rate; (d) Delaunay-triangulation-based non-uniform sampling scheme.

3.4 Step 2: Sampling the Graph Edges

In the previous step, we obtain a triangulation of the node set V. As shown in Figure 1(a) and (b), each edge in the original graph G is either overlapped or intersected with one or multiple Delaunay edges. An edge $(u, v) \in E$, intersecting with k Delaunay edges can be subdivided into $k+1$ sampling segments according to the intersection positions. As illustrated in Figure 2, edge AD intersects Delaunay edges BF, BE, and CE at points G', G'', and G''' respectively. Line segments AG', $G'G''$, $G''G'''$, and $G'''D$ are sampled.

Based on Delaunay triangulation, we non-uniformly sample edges according to the geometry of the graph. In Figure 3, our non-uniform sampling scheme is compared with the grid-based uniform sampling scheme on the same graph illustrated in Figure 2. In Figure 3(a), the graph edges are sampled with a 2×2 grid. The sampling grid edges are drawn in the purple color. Each graph edge is segmented according to its intersections with the grid edges. Edges are segmented into 17 fragments, each painted with the color of red, green, blue, or yellow. In Figure 3(b) and (c), the graph edges are sampled under the increased rates, with the grid size of 4×4 and 7×7 respectively.

We observe that with regular sampling, segments with very short length are usually generated in two scenarios. In the first scenario, a graph edge intersects the corner of one grid cell, such as the segments colored with red in Figure 3(a) and (b). In the second scenario, a graph node is very close to the grid cell boundary and the node has edge(s) corssing the nearest grid cell boundary. One example is the lowest node (node F in Figure 1) in Figure 3(c). Based on the above observation, the uniform sampling that is solely based on the edge length instead of the graph geometry may result in too many or too few subdivided edge segments with undesirable length, either causing low performance or generating unpleasant results.

Figure 3(d) shows all the segments generated with our Delaunay-triangulation-based scheme. Delaunay triangulation edges are drawn in dashed lines to be distinguishable from the original graph edges. In contrast to uniform sampling, our sampling scheme considers not only the edge information itself, but also the information of the neighboring nodes, e.g., the overall topology of the graph.

3.5 Step 3: Building the Control Map

Based on the sampling result of the second step, we build a control map for further processing. We represent each sampled segment with its midpoint (blue points shown in figure 1) and form a new complete graph $G'(V',E')$, in which V' is the set of the midpoints of sampled segments in G and E' is the set of edges connecting each pair of nodes in V'. We call graph G' the control map of graph G. By merging and rearranging the nodes in the control map, we can cluster and reroute the edges in the original graph G. Before further processing, the edges, in graph G', that satisfy the following criteria are first filtered out.

1. The edge linking two nodes (representing two sampled segments in graph G) which are originally on the same edge in graph G;

2. The edge linking two nodes with Euclidian distance larger than a threshold d specified by the user.

After filtering, only a subset of edges in E' remains, as shown in Figure 1(d). This filtered control map G' forms the basis of the further hierarchical edge clustering in the next step.

3.6 Step 4: Generating the Edge Hierarchy

In this step, we build the hierarchical clustering of nodes in the control map G' and thus generate the hierarchical edge clustering in G accordingly. During this step, the nodes of G' are merged and moved. The edges in graph G can be drawn as polylines or curves with control points from graph G'. The edges in E' are first sorted according to their energy U computed with Equation 1:

$$\begin{aligned} U(u',v') &= -D_{u'v'} - \Delta\theta(u',v') \cdot \ln(D_{u'v'}+1) \\ &\quad -\Delta\alpha(u',v') \cdot \ln(D_{u'v'}+1) \end{aligned} \tag{1}$$

where $D_{u'v'} = ||p_{u'} - p_{v'}||$ is the Euclidian distance of node u' and v', and $\Delta\theta(u',v')$ is the angle between the two sampled edges associated with u' and v'. When both u' and v' are original nodes in the initial control map, $\Delta\alpha(u',v')$ is the same as $\Delta\theta(u',v')$.

The edge with the highest energy is collapsed to form one single node. Intuitively, nodes with close or parallel corresponding sampled segments are likely to be merged together. After merging a node pair, the control map graph $G'(V',E')$ is updated and the energies of updated edges are recomputed. The new pairs are inserted into the priority queue according to the energy. For the newly formed node w' which is the merging of node w'_1 and w'_2, its corresponding sampled segment direction is computed as the average direction of the segments associated with w'_1 and w'_2. In Equation 1, if u' is the merging of nodes $u'_1, u'_2, ..., u'_m$, and v' is the merging of nodes $v'_1, v'_2, ..., v'_n$ of the initial control map, then the sampled segment directions of $u'_1, u'_2, ..., u'_m$ are averaged for the direction of u'. The segment direction of v' is computed similarly. Value $\Delta\theta(u',v')$ is computed using the average segment directions and $\Delta\alpha(u',v')$ is computed as the sum of pairwise angle between set $(u'_1, u'_2, ..., u'_m)$ and $(v'_1, v'_2, ..., v'_n)$.

During the merging of node pair (u',v') in V', the following operations are performed:

1. *Update the node list V'*: Remove u' and v' from the node list V' and insert new node $u'v'$ into V'. The position of $u'v'$ is initialized as the midpoint of u' and v';

2. *Update the edge list E'*: Remove the edges associated with u' and v' and make new edges with the end node $u'v'$. The edges are being filtered based on the criteria described in the step of building the control map (Step 3) before being added into graph G'.

3. *Optimize the location of $u'v'$*: The location of $u'v'$ is optimized to minimize the overall energy of graph G'.

Our energy definition is partially inspired by the edge-repulsion LinLog energy model [14]. The first term in Equation 1 can be interpreted as the attraction between close segments. The closer the two segments are, the higher possibility the merging has. Thus, in Equation 1, the smaller the value of $D_{u'v'}$, the larger the energy is. The second term models the repulsion between segments with small Euclidian distance but large direction difference. In this term, the angular distance is multiplied by the logarithmic value of the Euclidian distance. Large angular distance makes small energy, thus two segments with very different directions have a low possibility to be merged even though they have small Euclidian distance. The term $\Delta\theta(u',v')$ is recorded in radians, therefore, the value ranges from 0 to $\pi/2$. The value of term $D_{u'v'}$ is positive. If the input to the logarithmic operation is less than 1, the curve becomes very sharp as the input value increasing. The last term in Equation 1 is used to prevent the merging of the nodes whose corresponding edges in the original graph G are almost perpendicular to each other. Similar to the second term, this term also considers both angular and Euclidian distances. The difference is that the angular distance of $\Delta\theta(u',v')$ is recorded for the segments in the original graph G instead of the control map G'.

The energy model for location optimization is following:

$$U(G') = -\sum_{(u',v')\in E'} D_{u'v'} - \sum_{(u',v')\in E'} (\Delta\theta(u',v') \cdot \ln(D_{u'v'}+1)) \tag{2}$$

where $D_{u'v'}$ and $\Delta\theta(u',v')$ are defined the same as in Equation 1. Equation 2 is for the overall energy of graph G'. Compared with Equation 1, the first two terms are similar, and the third term of Equation 1 is removed. The removed term is to count the pairwise angles between the nodes in the initial control map. For the overall energy of a graph, this term is a constant value and can be ignored for optimization.

The above merging operation is repeated until there is no new candidate node pair left in the queue.

3.7 User Interaction

We provide an interface allowing users to explore the node-link graphs with different level of details. The implemented user interface consists of two regions, as shown in Figure 4. The left is the display region. The graphs are displayed with shaded tubes (edges) and spheres (nodes). The right is the parameter region. Two types of parameter controls are provided to the user. The control panel for the computation parameter is used for adjusting the edge clustering computation, i.e., the distance threshold discussed in Section 3.5 and 3.6. The display parameter panel provides users with the access to explore the clustering results. In our algorithm, each edge is segmented and represented by a series of control points. The resulting hierarchically clustered edges can be drawn in either Catmull-Rom splines or Bezier curves upon the user's selection. In this paper, all the curves shown are drawn with Catmull-Rom splines in which specified curves pass all of the control points. Our user interface supplies an easy-to-use scrolling bar for user exploration. The user can slide the scrolling bar to examine the edge clustering results at different levels interactively.

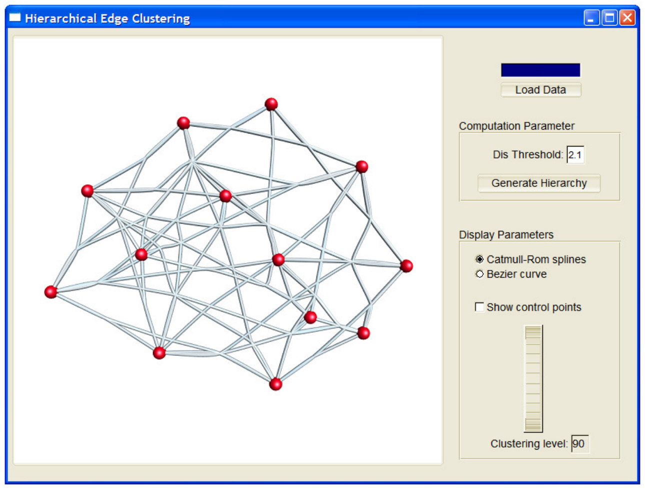


Figure 4: The user interface of our system using energy-based hierarchical edge clustering. Left: display region; Right: parameter region accepting user input.

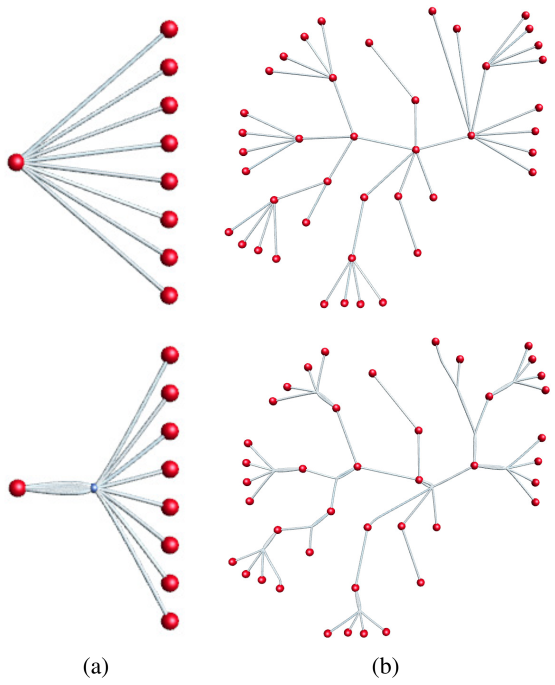


Figure 5: Results of hierarchical edge clustering. The images in the top row are the initial graphs. The images in the bottom row are the corresponding clustered graphs drawn with curved edges.

4 Experimental Results and Discussion

We applied our energy-based hierarchical edge clustering algorithm to several node-link diagrams. Figure 5 shows the clustering results on two graphs with simple topologies. The top images are the original graphs, while the bottom ones are the corresponding clustered results. In Figure 5(a), one array of nodes are all connected to a single center node to form a single bundle. As illustrated in Figure 5(b), a tree graph is clustered by our algorithm. All edges are grouped as expected. Figure 6 shows the hierarchical edge clustering structure of an artificial dataset with 13 nodes and 57 edges. Figure 6(a) is the original node-link graph. The highest level of edge clustering generated by our hierarchical edge clustering algorithm is illustrated in Figure 6(b). Figure 6(c)-(e) show the intermediate graphs during the hierarchical clustering from low to high levels. Graph edges are clustered progressively as the level of hierarchy increasing.

We further applied our algorithm on a flight route map. The original graph is shown in Figure 7(a), with 83 cities and 187 flight routes displayed. Each graph node represents a major city in China, while each graph edge represents a flight connecting two cities. The hierarchical clustering result is illustrated in Figure 7(b). Airline routes are clustered together according to their geometry and topology structures. Similar routes are combined together to reduce visual clutter. We observe that undesirable results can be generated for a few routes. Such as the route of the direct flight between Beijing and Urumqi (the most northwestern city in the map) is distorted in the clustered result and becomes difficult to trace. We are currently investigating possible solutions for such cases. One possibility is to let the user directly restrict the deformation degree for certain graph edges.

We implemented our algorithm on a Dell OptiPlex GX280 desktop with a single Intel Pentium 4 3.2GHz CPU and 1GB Memory. The results in Figure 5 can be computed interactively. Computing the hierarchy for graphs in Figure 6 and Figure 7 takes 0.985 seconds and 9.125 seconds respectively. Our implementation is not highly optimized for speed. Therefore, for large size graphs, the pre-computation may cost several minutes. The GPU-accelerated Delaunay triangulation methods [8] and local optimization constraints can be applied to speed up the performance.

One advantage of our method is that once pre-computed, the whole hierarchical clustering structure can be stored for further uses. After that users can interactively explore the clustering results at continuous level of details clustering results with an easy-to-use scrolling bar.

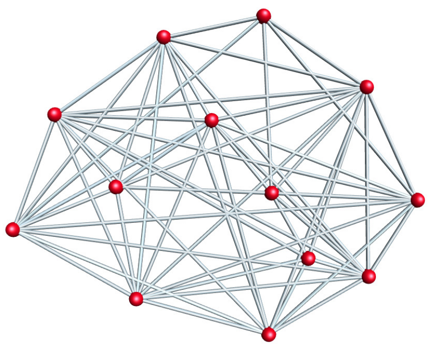
5 Conclusions and Future Work

In this paper, we have presented a novel energy-based hierarchical edge clustering method for graphs. Our method hierarchically clusters control points of edges to minimize a tailored energy function. In the sampling stage, we adapt Delaunay triangulation for the non-uniform sampling of graph edges. Comparing with uniform sampling schemes using regular grids, our Delaunay-triangulation-based scheme considers the topological structure of the graph and generates more desirable results. Through hierarchical edge clustering, our method can reduce visual clutter and provide a better understanding of the graphs. The results shown in the paper demonstrate the effectiveness of our method in clustering edges. Our algorithm can be applied to data with complex interconnections and can provide good high level abstractions of complex graphs.

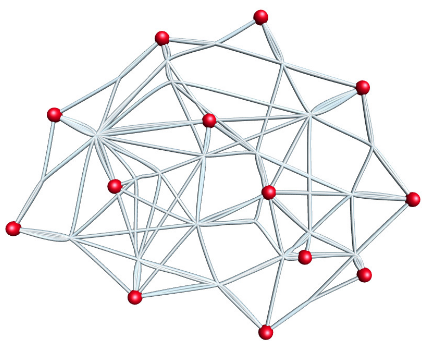
We plan to further investigate the effectiveness of our method on large graphs. We will test whether the diagrams with our hierarchical edge clustering are more readable than the raw diagrams under some defined visual clutter measures. We are also interested in conducting user studies to validate our technique. In addition, we are looking into an intuitive interface for users to control the abstraction level of the clustering. In some cases, it is desirable to allow users to modify the clustering results interactively, e.g., to modify the locations of certain control points.

Acknowledgments

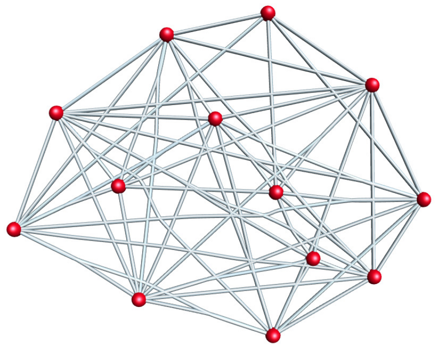
We would like to thank the anonymous reviewers for their valuable comments. This work is supported by Hong Kong RGC grants CERG 618705 and 618706.



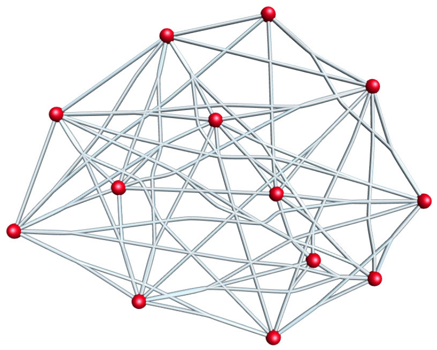
(a)



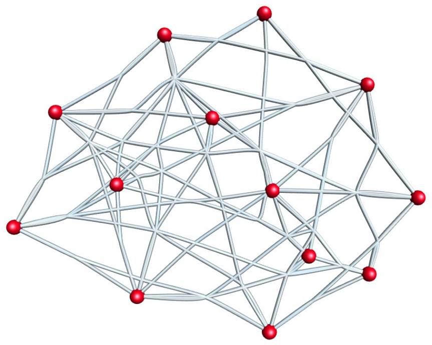
(b)



(c)



(d)



(e)

Figure 6: Results of hierarchical edge clustering on an artificial dataset with 13 nodes and 57 edges: (a) the original node-link graph; (b) the highest level of edge clustering based on our hierarchical edge clustering algorithm; (c), (d), and (e) the intermediate graphs during the hierarchical clustering from low to high levels.

References

[1] D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[2] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of the SIGGRAPH '93*, pages 73–80, 1993.

[3] M. Carpendale and X. Rong. Examining edge congestion. *CHI '01 extended abstracts on Human Factors in Computing Systems*, pages 115–116, 2001.

[4] M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Algorithms Appl.*, 9(1):31–52, 2005.

[5] A. Formella and J. Keller. Generalized fisheye views of graphs. In *Proceedings of the 3rd International Symposium on Graph Drawing*, pages 242–253, 1995.

[6] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, 1986.

[7] E. R. Gansner and Y. Koren. Improved circular layouts. In *Proceedings of the 14th International Symposium on Graph Drawing*, pages 386–398, 2006.

[8] K. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. *ACM Transactions on Computer Graphics*, 33:277–286, 1999.

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

[10] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2):126–160, 1994.

[11] X. Lin. On the computational complexity of edge concentration. *Discrete Appl. Math.*, 101(1-3):197–205, 2000.

[12] F. J. Newbery. Edge concentration: A method for clustering directed graphs. In *Proceedings of the 2nd International Workshop on Software Configuration Management*, pages 76–85, 1989.

[13] A. Noack. An energy model for visual graph clustering. In *Proceedings of the 11th International Symposium on Graph Drawingg*, pages 425–436, 2003.

[14] A. Noack. Energy-based clustering of graphs with nonuniform degrees. In *Proceedings of the 13th International Symposium on Graph Drawing*, pages 309–320, 2005.

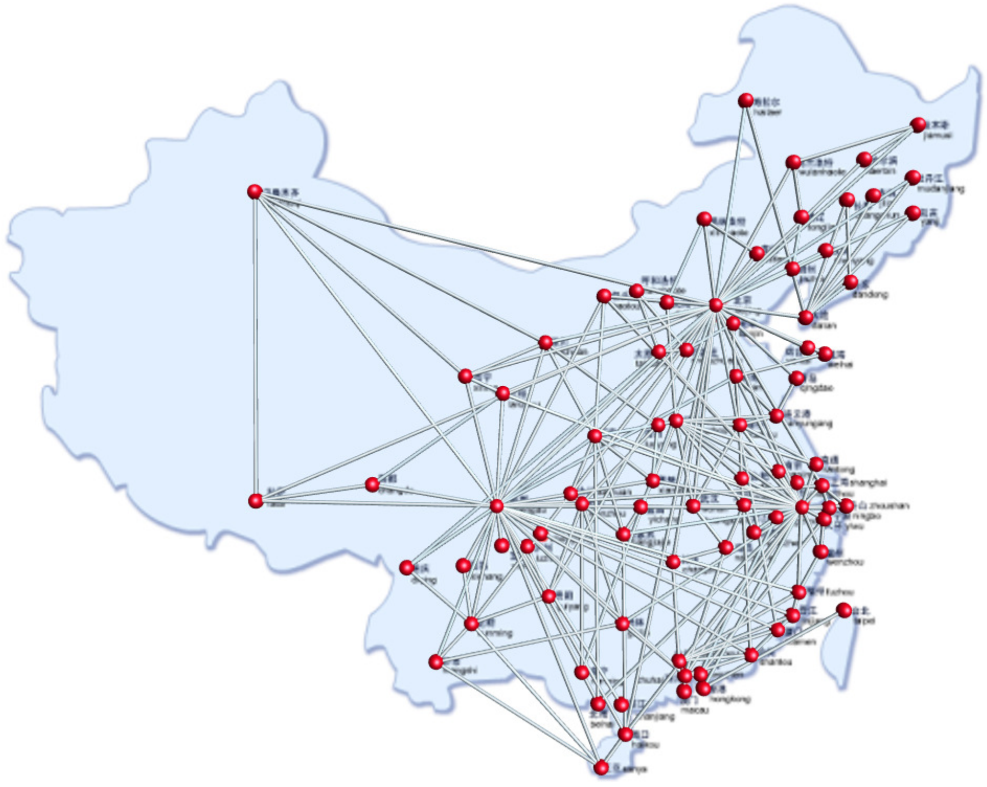
[15] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. *IEEE Symposium on Information Visualization 2005*, pages 219–224, 2005.

[16] H. Qu, H. Zhou, and Y. Wu. Controllable and progressive edge clustering for large networks. In *14th International Symposium on Graph Drawing*, pages 399–404, 2006.

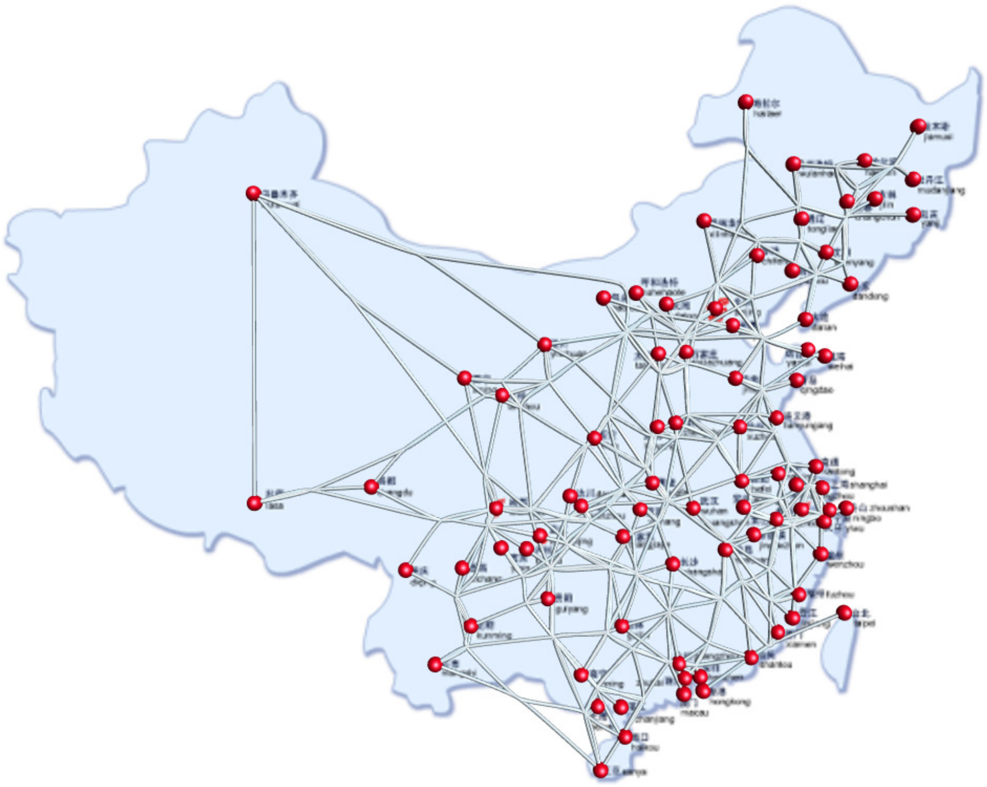
[17] C. Ware, H. C. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[18] N. Wong and M. Carpendale. Interactive poster: Using edge plucking for itneractive graph exploration. In *Poster in the IEEE Symposium on Information Visualization (2005)*, 2005.

[19] N. Wong, M. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. *IEEE Symposium on Information Visualization 2003*, pages 51–58, 2003.



(a)



(b)

Figure 7: Results of hierarchical edge clustering on an airline flight route map with 83 nodes and 187 edges: (a) the original node-link graph; (b) the edge clustering result. (Data source of the background map: *http://www.airchina.com.cn/*)