Revisiting Partial Edge Drawing in Graph Diagrams

Min Lu

Abstract

Graph drawing is a fundamental problem in visualization. In this work, we present spiky links as a new means to visually encode the links in a graph. Unlike conventional links which use lines to explicitly connect nodes, spiky links are designed as two visually echoed spikes attached to the pair of connected nodes. The pair of spikes that represent the link between the nodes are not necessarily connected, but yet, based on the Gestalt Continuity principle, maintain a good perception of the connection between the nodes. The shape of the spikes is modeled and controlled by a uniform parameter, which we call *Stickiness*, by which the shape is distributed according to the density of the graph. The key advantage of using spiky links is that they significantly alleviate visual clutter and ease overdrawing. A controlled user study was conducted to examine the efficiency of spikes in representing connectivity in graphs. The results indicate that spiky links maintain the perception of node connectivity, and were overall preferred by participants, stating that they were more aesthetic and show less visual clutter than their node-link counterpart.

Keywords：Graph Visualization,Edge Drawing, Visual Clutter, Gestalt Principle

1 Introduction

Graph drawing is at the very core of visualization, and a large variety of graph visualization methods have been proposed to better reveal and understand the patterns of connectivity in network data[1].

Node-link diagram[2] is the most widely-used graph visualization method, where the connection among entities are simply drawn as lines or curves (called *links*) (Figure 1(left)). However, this intuitive visual encoding of graph edges does not scale well. Visual clutter becomes a severe problem as the number of nodes and links increase, in particular when many links cross each other in small regions. There are many approaches to tackle this overdrawing problem, such as graph layout algorithms that modify the node positions to reduce the amount of edge crossing[3-4], bundling of the edges[5-6], or simplifying the graph by drawing the meta structure of the graph[7]. These works all perform some sort of manipulation on the graph's structure or layout, but do not change the visual encoding of a given graph.

In this work, we present a new graph visualization method, re-encoding the graph links in order to reduce visual clutter. As shown in the right of Figure 1, instead of continuous links in a node-link diagram, we propose to draw *spiky links* (or simply *spikes*), which constitute a pair of strokes between nodes connected in the graph. The conceptual idea of spikes is based on the *Gestalt principle of Continuity*[8], which leverages the human perceptual tendency to continue a flow of visual elements, and thus perceive linked nodes from separate spikes. The key idea of using spikes rather than continuous links is that the visual encoding of edges is relaxed from continuous lines, to effectively reduce the visual clutter, while preserving the perception of connectivity.

We evaluate our method by first showing its effectiveness in reducing visual clutter by measuring the data-ink ratio[9] of several graphs. Then, we perform a controlled user study that shows that spiky links is as good as, and in some cases even better in perceiving graph connectivity compared to a regular node-link diagram. Furthermore, participants preferred spiky links, stating it is more aesthetic, visually clear and provides less visual clutter.

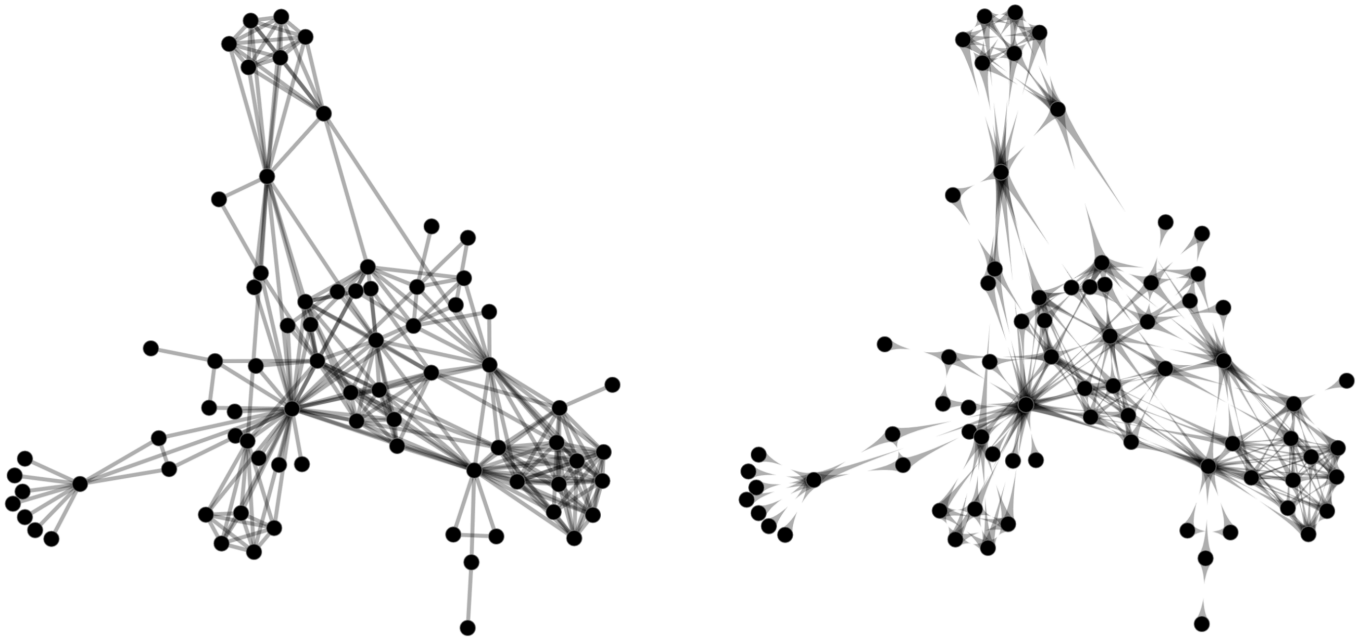

Figure 1: (Left) A conventional node-link diagram with continuous lines connecting the nodes. (Right) The same graph is drawn using spikes to visually encode the links, while significantly reducing clutter and easing overdrawing. Although the spikes are not necessarily connected, they maintain a good perception of connectivity.

2 Related works

In this section, we first discuss works related to node-link diagrams, and then review the topic of graph readability.

2.1 Node-link Diagram

Over the years, many graph visualization methods were developed[10], including node-link diagrams, such as NodeXL[11], or matrix representations for graphs, like in Matrix Zoom[12], MatrixExplorer[13] or Nodetrix[1] (which combines both types of representations). Due to its intuitive representation of edges, node-link diagram is clearly the most common way to visualize graphs. Node-link diagrams attract a large graph drawing community, which is dedicated to research that deals with the geometric representation and the layouts of graphs.

Various layout methods were proposed over the last decades[2]. Reingold and Tilford[14] were the first to present tree layout for graphs with intrinsic hierarchy. Along this line, numerous hierarchical layouts were proposed, such as the radial graph layout[15]. By modeling nodes and links as physical bodies tied with springs, Eades[16] first presented the Force-Directed layout algorithm in graph drawing. This work has then inspired various spring-based models, like in[17] or[18].

Unlike the works above, our work does not deal with the determination of the placement of graph nodes and links, but with the drawing of a given graph of a layout that is already determined. We propose a new visual

encoding of the edges in the graph. Our proposed method can be applied to visualize any node-link diagram with any kind of layout.

2.2 Partial Edge Drawing

The idea of partial edge drawing is to draw the edge partially instead of traditional complete link representations. One of early work in this direction is by Becker et al.[19], which separates the link into halves, and uses the width of halves to encode information in bi-direction. The link breaks when half width diminished. A closer idea to spiky link is by Rusu et al.[20], they propose asymmetric breaks in edges to avoid edge crossing. By distinguishing crossing edges as primary edge and secondary edge, the primary keeps fully connected while the secondary one takes a break at the crossing. Bunch of research work are performed along this research line. Burch et al.[21] propose acsymmetric partial drawing, only drawing the links starting at the start node and pointing to the end node. Later Burch et al.[22] develop interactive partial links that allows for applying only to regions of interest. Thereafter, some design variations of partial edge drawing are proposed. For example, Schmauder et al.[23] use partial edge in dynamic graphs, by splitting each link as segments as time steps to encode the temporal information. Misue and Akasaka[24] propose morphing partial edges (MED), in which links are morphed between partial and complete drawings.

Research efforts are also drawn into empirical study to learn the performance of partial edges compared with conventional links. Binucci et al.[25] performed a user study and show that the benefit of homogeneity in terms of readability overcomes the benefit of fewer crossings and more inks in partial edge drawing. Sathiyanarayanan and Priozzi[26] perform a user study to test Euler graph with complete edges and partial and show there is no significant difference in performance between them, but people prefer partial draw edge for visual-aesthetics. Burch[27] study how the link length and direction influence the judgement of target node. Some other research focus on mathematical modeling of partial links. Bruckdorfer and Kaufmann[28] mathematically formalize the PED and suggest several variants with Symmetry and Homogeneity in the breaks. Bruckdorfer et al.[29] study the symmetric and σ-homogeneous where lengths to be a given fraction σ of the edge lengths PED (SHPED), and explore the bounds of SHPED of graphs. Hummel et al.[30] propose efficient algorithms to solve the maximum-ink PEDs and SPEDs problem, to maximizing total length of PEDs while keeping the graph crossing-free.

Although the concept of partial links has been shown for decades, most of research focus on the symmetry, homogeneity and the proven of its property in different graphs mathematically. The visual style of partial edge style are mostly remained as direct broken lines. In this work, we revisit the concept of partial edge in graph drawing from the perspective of visual design and propose an advanced edge style, spiky link. Note that though sharing the similar idea of decreasing thickness, tapered links are proposed to encode the direction and draw asymmetric.

3 Method

3.1 Overview

In conventional node-link diagrams, nodes are linked by continuous lines. Figure 2(a) shows a classic node-link diagram. Linking a pair of nodes with a line provides a strong visual encoding of the connection. With an even distribution of ”ink” along the line, connectivity can be effortlessly traced when there is little clutter. However, continuous lines can easily induce visual clutter when coming across with others, especially for lines connecting long distance nodes. As can be seen in Figure 2(a), the middle-bottom region of the graph is heavily cluttered, as many lines are overdrawn. Nodes located inside this dense region are almost buried by overly crossed lines. It is difficult to tell the connections between those nodes, or even see the nodes themselves.

To provide better perception of the graph and its structure, we propose using a novel visual encoding, named *Spiky links*, which are designed as a pair of hand-shaking strokes with a shape of a spike, attached to the connected nodes. Figure 2(b) shows the same graph, as in (a), where the connected nodes are visually encoded with a pair of spiky links, which are not necessarily visually connected. The basic idea of spiky links is to indicate the connection by the two perceptually connected spike shapes, instead of explicitly connecting the lines. As shown in Figure 2(b), the node-link diagram drawn with spikes is less cluttered than the original. With the new link design, the original inner area, especially around the nodes, is now much lighter and is drawn with significantly less ink.

Figures 2(c) and (d) show a common variation of the graph in (a)(b), where the drawn size of the nodes reflects an attribute of the nodes. As nodes with small value are diminished into tiny dots, the clutter problem becomes more severe in the conventional node-link diagram. Small size nodes in dense regions are hardly perceived (Figure 2(c)). Even in less dense regions, it becomes quite hard to notice the nodes, which can sometimes only be estimated by the ends of the links. Using spiky links (Figure 2(d)), the nodes are clearer and better visually emphasized. With spiky links, it is easier to locate small nodes within dense regions, as well as perceive their connectivity information, such as the degree of the node or to which other nodes they links.

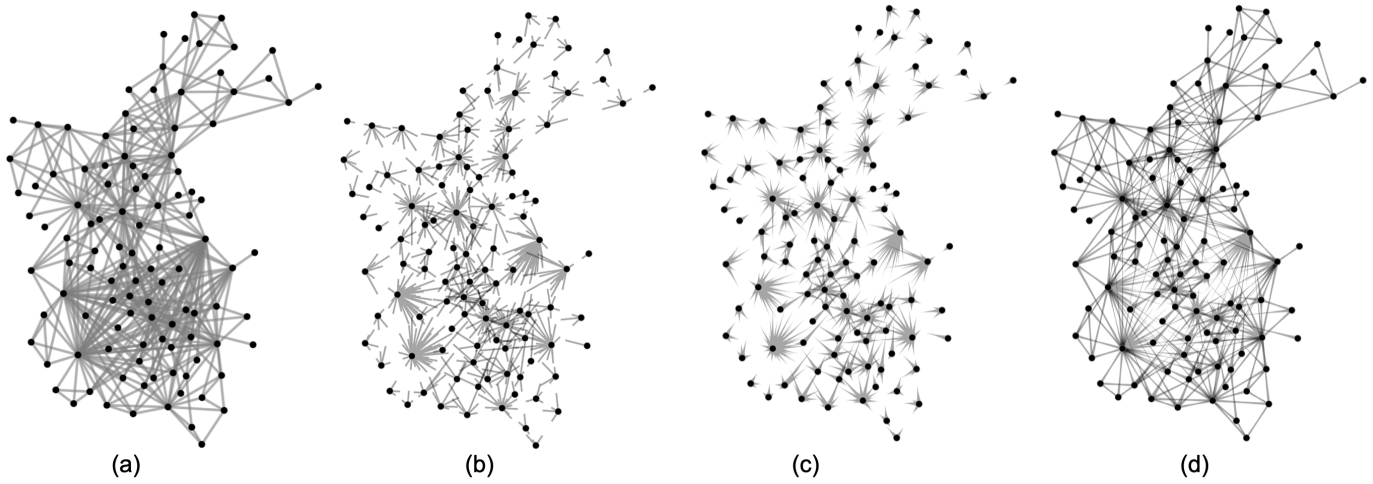

Figure 2: Spikes with different levels of *Stickiness*, spikes range from broken shapes to connected links. Stickiness is a trade-off design choice between connectivity indication and visual decluttering.

3.2 Spike Stickiness

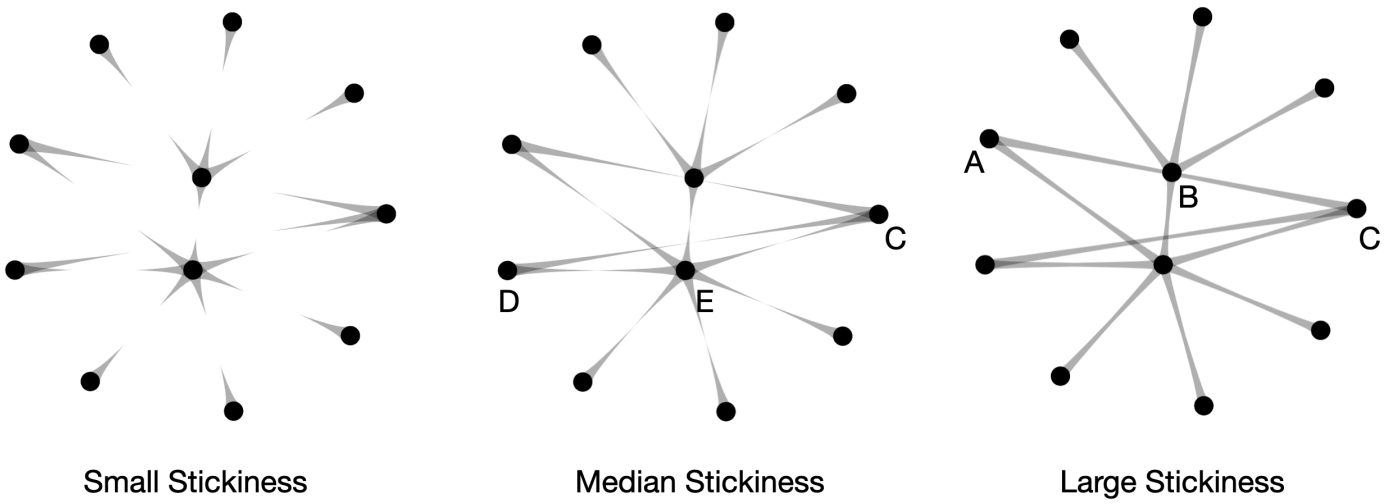

Figure 3: Spikes with different levels of *Stickiness*, spikes range from broken shapes to connected links. Stickiness is a trade-off design choice between connectivity indication and visual decluttering.

4 Implementation details

4.1 Comparing with released source codes

In my work, we use python to reproduce original work based on totally understanding viusalization design.Here we pay much attention to data processing. data processing can be divided into the following six part:

Procedure 1 readdata:changes the form of data from json to dictionary and creates global lists,which are all_node and all_links from converted dictionary of nodes and links respectively

Input: json files like nodes *names,group, pos* links *source, target, value, sourceGroupid, targetGroupid,* boundary *left, right, top, bottom*
Output: dictionary and list

Procedure 2 nodes_out:creates global lists,which are all_names and all_pos from the key of name and pos in all_nodesandall_links

Input: all_nodes *names, group and pos*
Output: names_pos *name X Y*
for *node* ***in*** *all _ nodes* **do**
| $allnames$ = append(*node*[′*name*′]) $allpos$ = append(*node*[′*pos*′])
end

Procedure 3 links_out:creates global lists,which are all_names and all_pos from the key of name and pos in all_nodesandall_links

Input: all_nodes *names, group and pos*
Output: names_pos *name X Y*
for *node* ***in*** *all _ nodes* **do**
| $allnames$ = append(*node*[′*name*′]) $allpos$ = append(*node*[′*pos*′])
end

Procedure 4 get_sourcetargetpos:Returns a dictionary ”position” whose values are list discribing the source and target positon by mutiplying raw positon and SURFACE_HEIGHT

Input: allnodes *straight, spike with different stickiness* alllinks *0.8* HEIGHT_SURFACE *500*
Output: newpos *X Y*
for *link* ***in*** *all_links* **do**
| $sourcepos$ = append(*node*[′*pos*′], H_l) $alllinks$ = Encoder(X_l)
end

Procedure 5 generate_stickyspike:returns thirdpos and handlepoint that are used when drawing curves

Input: drawstyle *straight, spike with different stickiness* encodeInfo *0.8*
Output: thirdpos *X Y* handlepoint *X Y*
for *i* ***in*** *range len(sourcepos)* **do**
| $angle\_some$ = atan2((*dx*, *dy*))
end
for *i* ***in*** *range (len(p1c))* **do**
| $p1c_{xi}$ = Vector1(*sourcepos*) $p1c_{yi}$ = Vector1(*targetpos*)
end
for *i* ***in*** *range (len(p2c))* **do**
| $p2c_{xi}$ = Vector2(*sourcepos*) $p2c_{yi}$ = Vector2(*targetpos*)
end

Procedure 6 getStickiness:returns stickiness which control the shape of spike

Input: encodeInfo *constant* prop *constant*
Output: stickiness *constant*

Procedure 7 getAnchors:return two pairs of pa and pb on source and target circles when we draw ”straight”and ”spike” edges

Input: sourcepos *X Y* targetpos *X Y* stickiness *0.4 0.5 0.8*
Output: p1a *X Y* p1b *X Y* p2a *X Y* p2b *X Y*

4.2 Experimental environment setup

The most hands-off way to draw different style of edges is using windows10.This can be done simply by the following user manual: First, install visual studio code and required python package.Pay attention to set the environment path when installing python. We highly recommend the version of python over 3.9 and2019 visual studio code. Second, install and import required python library such as json, copy. Third, install SVG plug-in component in vs code. Fourth, run drawing.py file. Fifth, right click the svg file to check the drawings

4.3 Interface design

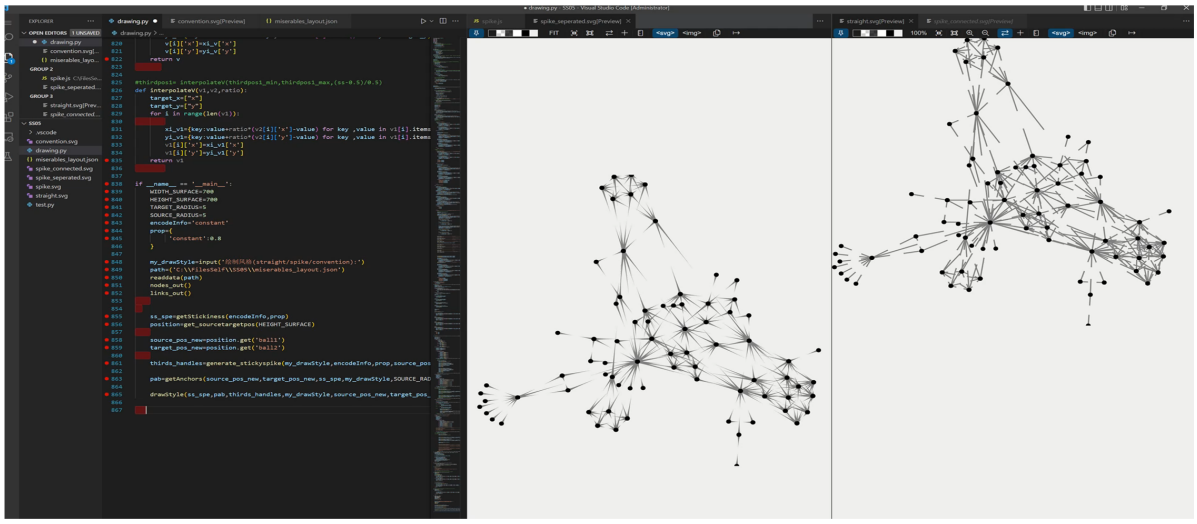

Figure 4: final Interface design

4.4 Main contributions

In the result display part, the interactive design of the static diagram is added to the original author’s work, such as the mouse hover effect, the ruler effect, the equal scale enlargement or reduction and download spiky pictures. More importantly, we can change the background of the same picture in my work, which is helpful to help readers judge the connection between two far points

5 Results and analysis

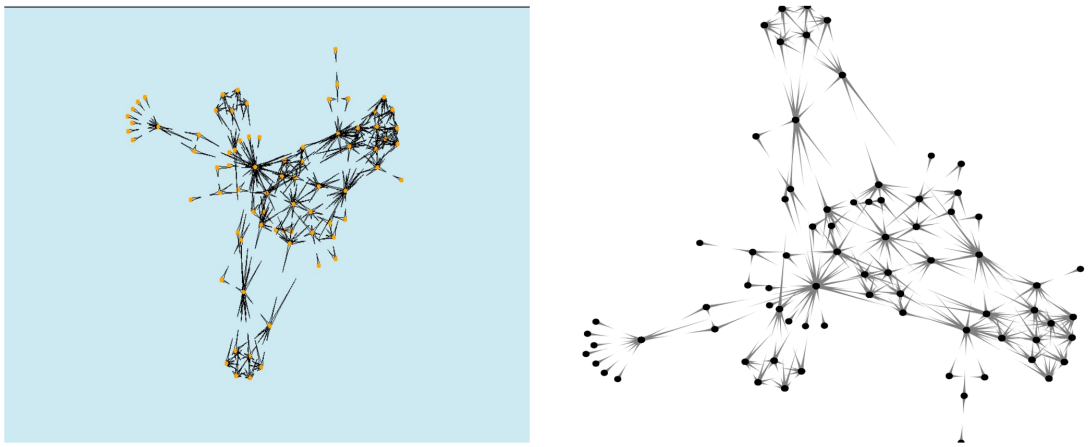

Figure 5: (Left) Aliasing node-link diagram with spiky links.(Right)The same graph is drawn using anti-aliasing spiky

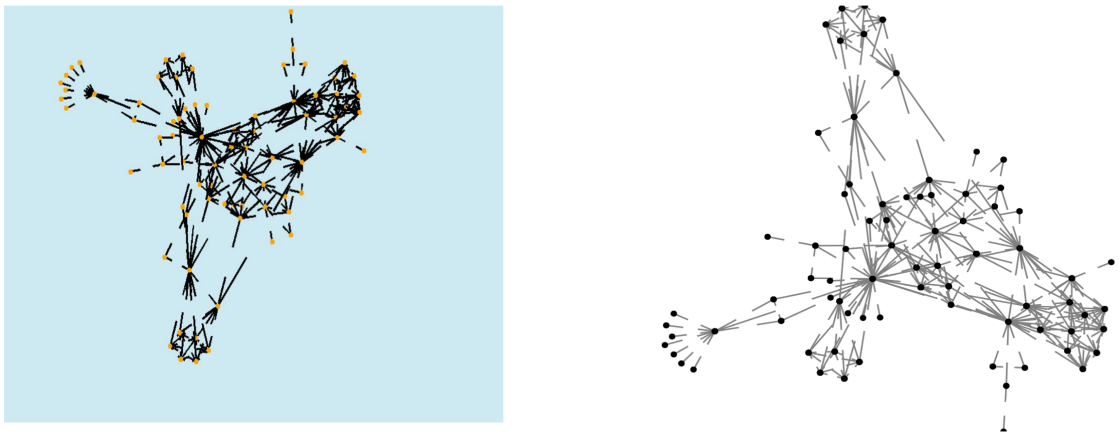

Figure 6: The same graph is drawn using aliasing(Left) and anti-aliasing (Right)with ¼ shape partial edge drawing links

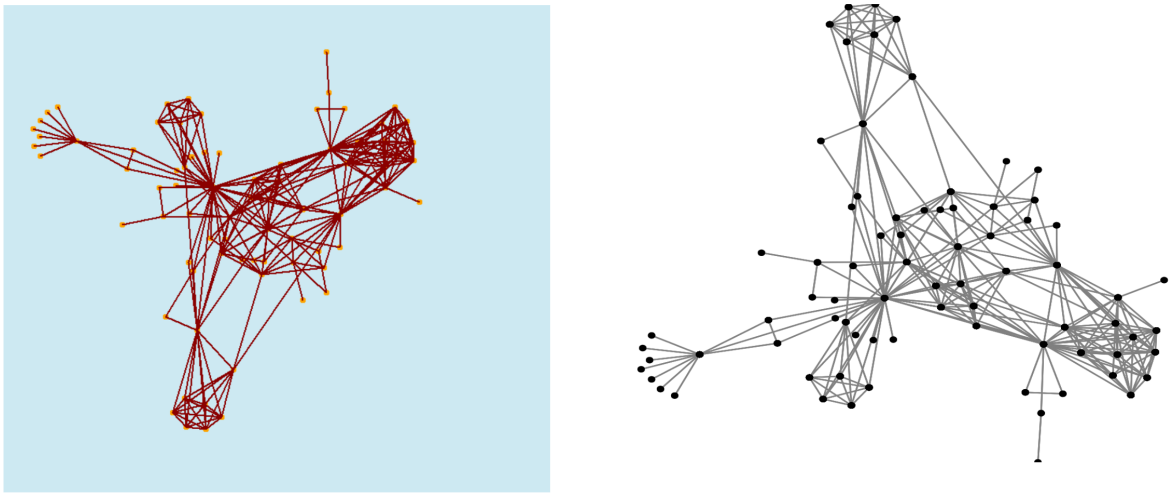

Figure 7: The same graph is drawn using aliasing conventional links(Left) and anti-aliasing links(Right)

6 Conclusion and future work

Further work can be done in the space of effect display, such as displaying spike graph in 3D space rather than 2D to explore the influence of display space on the readability of the same spike graph.In addition, add

vivid stories to the spike static graph to make the visualization dynamic and alive

References

[1] HENRY N, FEKETE J D, MCGUFFIN M J. NodeTrix: a hybrid visualization of social networks[J]. IEEE transactions on visualization and computer graphics, 2007, 13(6): 1302-1309.

[2] BATTISTA G D, EADES P, TAMASSIA R, et al. Graph drawing: algorithms for the visualization of graphs[M]. Prentice Hall PTR, 1998.

[3] BAUR M, BRANDES U. Crossing reduction in circular layouts[C]//International Workshop on Graph-Theoretic Concepts in Computer Science. 2004: 332-343.

[4] MATUSZEWSKI C, SCHÖNFELD R, MOLITOR P. Using sifting for k-layer straightline crossing minimization[C]//International Symposium on Graph Drawing. 1999: 217-224.

[5] HOLTEN D, VAN WIJK J J. Force-directed edge bundling for graph visualization[C]//Computer graphics forum: vol. 28: 3. 2009: 983-990.

[6] HURTER C, ERSOY O, TELEA A. Graph bundling by kernel density estimation[C]//Computer graphics forum: vol. 31: 3pt1. 2012: 865-874.

[7] DUNNE C, SHNEIDERMAN B. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2013: 3247-3256.

[8] WERTHEIMER M. Untersuchungen zur Lehre von der Gestalt. II[J]. Psychological Research, 1923, 4(1): 301-350.

[9] TUFTE E R. The visual display of quantitative information[J]. The Journal for Healthcare Quality (JHQ), 1985, 7(3): 15.

[10] HERMAN I, MELANÇON G, MARSHALL M S. Graph visualization and navigation in information visualization: A survey[J]. IEEE Transactions on visualization and computer graphics, 2000, 6(1): 24-43.

[11] SMITH M A, SHNEIDERMAN B, MILIC-FRAYLING N, et al. Analyzing (social media) networks with NodeXL[C]//Proceedings of the fourth international conference on Communities and technologies. 2009: 255-264.

[12] ABELLO J, VAN HAM F. Matrix zoom: A visual interface to semi-external graphs[C]//IEEE symposium on information visualization. 2004: 183-190.

[13] HENRY N, FEKETE J D. Matrixexplorer: a dual-representation system to explore social networks[J]. IEEE transactions on visualization and computer graphics, 2006, 12(5): 677-684.

[14] REINGOLD E M, TILFORD J S. Tidier drawings of trees[J]. IEEE Transactions on software Engineering, 1981(2): 223-228.

[15] JANKUN-KELLY T, MA K L. MoireGraphs: Radial focus+ context visualization and interaction for graphs with visual nodes[C]//IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714). 2003: 59-66.

[16] EADES P. A heuristic for graph drawing[J]. Congressus numerantium, 1984, 42: 149-160.

[17] CHEONG S H, SI Y W. Force-directed algorithms for schematic drawings and placement: A survey[J]. Information Visualization, 2020, 19(1): 65-91.

[18] FRUCHTERMAN T M, REINGOLD E M. Graph drawing by force-directed placement[J]. Software: Practice and experience, 1991, 21(11): 1129-1164.

[19] BECKER R A, EICK S G, WILKS A R. Visualizing network data[J]. IEEE Transactions on visualization and computer graphics, 1995, 1(1): 16-28.

[20] RUSU A, FABIAN A J, JIANU R, et al. Using the gestalt principle of closure to alleviate the edge crossing problem in graph drawings[C]//2011 15th International Conference on Information Visualisation. 2011: 488-493.

[21] BURCH M, VEHLOW C, KONEVTSOVA N, et al. Evaluating partially drawn links for directed graph edges[C]//International Symposium on Graph Drawing. 2011: 226-237.

[22] BURCH M, SCHMAUDER H, PANAGIOTIDIS A, et al. Partial link drawings for nodes, links, and regions of interest[C]//2014 18th International Conference on Information Visualisation. 2014: 53-58.

[23] SCHMAUDER H, BURCH M, WEISKOPF D. Visualizing Dynamic Weighted Digraphs with Partial Links.[J]. IVAPP, 2015, 2015: 123-130.

[24] MISUE K, AKASAKA K. Graph Drawing with Morphing Partial Edges[C]//International Symposium on Graph Drawing and Network Visualization. 2019: 337-349.

[25] BINUCCI C, LIOTTA G, MONTECCHIANI F, et al. Partial edge drawing: Homogeneity is more important than crossings and ink[C]//2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA). 2016: 1-6.

[26] SATHIYANARAYANAN M, PIROZZI D. Social network visualization: Does partial edges affect user comprehension?[C]//2017 9th international conference on communication systems and networks (COMSNETS). 2017: 570-575.

[27] BURCH M. A user study on judging the target node in partial link drawings[C]//2017 21st International Conference Information Visualisation (IV). 2017: 199-204.

[28] BRUCKDORFER T, KAUFMANN M. Mad at edge crossings? Break the edges![C]//International Conference on Fun with Algorithms. 2012: 40-50.

[29] BRUCKDORFER T, CORNELSEN S, GUTWENGER C, et al. Progress on partial edge drawings[C]//International Symposium on Graph Drawing. 2012: 67-78.

[30] HUMMEL M, KLUTE F, NICKEL S, et al. Maximizing ink in partial edge drawings of k-plane graphs [C]/ / International Symposium on Graph Drawing and Network Visualization. 2019: 323-336.