Supporting Interactive Graph Exploration with Edge Plucking

Nelson Wong and Sheelagh Carpendale

Department of Computer Science, University of Calgary,
2500 University Drive N.W. Calgary, Alberta T2N 1N4
{yw, sheelagh}@cs.ucalgary.ca

Abstract. Excessive edge density in graphs can cause serious readability issues, which in turn can make the graphs difficult to understand or even misleading. We introduce Edge Plucking, an interactive tool that enables clarification of node-edge relationships by bending edges while preserving node positions. Edge Plucking extends the types of interactive graph exploration tools that have been designed to address edge congestion.

1 Introduction

Many types of information can be represented as a graph. Usually the information's entities are mapped to nodes and the relationships between these entities are mapped to edges. Examples include telecommunication networks, the World Wide Web, airline routes, software entity relationship diagrams, and road maps. Creating layouts of node-link diagrams for these types of information can aid one's comprehension of the inter-relationships within the information. For instance, patterns become apparent such as the tendency to hub and stroke structure in the Web and the location of heavily serviced regions for communication networks. However, creating comprehensible diagrams is hard even with small information sets and gets increasingly difficult as the size of these information sets increases. When the data set includes a lot of relationships between entities, drawing these relationships often results in dense regions with multiple edge-crossings and overlapping edges, which in turn can obscure nodes and any extra visuals under these regions. We call this *edge congestion* [19]. Edge congestion can make graphs difficult for people to understand, or even misleading. For example, Fig. 1a shows a real world graph of the airline routes flown by NorthWest Airlines (November 2001 in-flight magazine) and Fig. 1b shows a set of Parallel Coordinates [7] representing the attributes of several generations of plants.

Since creating a comprehensible graph layout that successfully handles edge congestion is known to be difficult [4], we have been exploring the possibilities of interactive solutions. For our first interactive approach we focused on minimizing the disturbance to the graph layout and created a localized edge-bending approach BubbleLens [2] (Fig. 2b), which successfully disambiguated some edge congestion such as when an edge crossed over a node and made it possible to read underlying

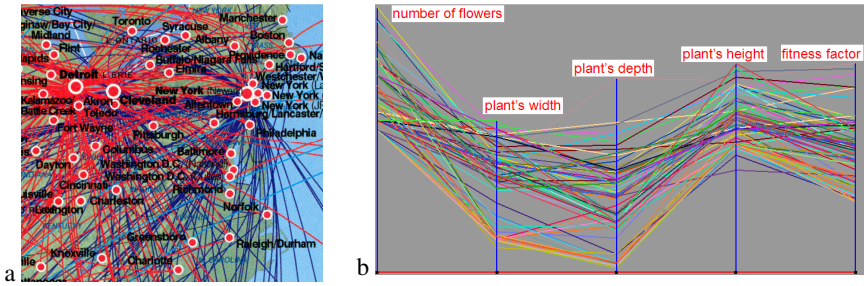

Fig. 1. Examples of edge congestions: a) Airline routes from NorthWest Airlines, "World Traveler", November, 2001; b) Parallel Coordinates of plant generations

graphics and labels. However, it also introduced additional bends in the edges (Fig. 2b). Our next approach used splines (Fig. 2c) to eliminate the extra edge bends but affected the entire length of an edge, creating interactive disturbances over large parts of the graph. A study [20] showed that the spline-based approach was more usable and generally preferred. We refined this approach to create EdgeLens [20]; however, through observations and discussions with users, the possibility of a different interaction metaphor arose. The metaphor behind EdgeLens is a virtual force field that pushes edges aside. In our interactions in the real world, we are more likely to pick up and move those things that are in the way. This coupled with the evidence of empirical support for the usefulness of motion of selected edges in querying graphs [16] has led to Edge Plucking. With Edge Plucking one can catch edges with the mouse cursor and simply pull them aside. In this paper we describe the development of Edge Plucking.

The paper is organized as follows. In the next section we present the related work. This is followed by a discussion of the ambiguities that can arise from edge congestion. In Section 4 we describe the concept of Edge Plucking and outline its development. Then in Section 5 we apply Edge Plucking to address the edge congestion problem, and compare and contrast Edge Plucking with EdgeLens. This followed by a conclusion in Section 6.

2 Related Work

There are several directions from which the problem of edge congestion has been approached. These include graph layout, magnification techniques, filtering methods,

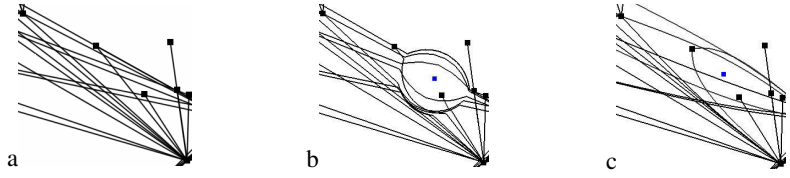

Fig. 2. Our previous interactive graph exploration approaches: a) the original graph; b) BubbleLens is applied; c) SplineLens is applied

and some interactive approaches.

Graph Layout: Through changing the layout by moving node positions it is possible to lessen edge congestion. However, in practice optimal solutions are difficult to find [4, 6, 18] and not all graphs can be drawn without edge crossings. In addition, graphs that visualize particular data may have constraints on layout positions that reflect data semantics. For example in Fig. 1a the nodes are cities and have geographic locations. If their positions were moved much information would be lost. Similarly in Fig. 1b the nodes in this visualization represent data points and as such their locations are specified. Even if these layouts could be reorganized to lessen edge congestion much meaning would be lost. Another related layout strategy is the idea of curving edges [3, 9]. This has been applied globally. Instead, we let users interactively curve selected edges.

Magnification: Different methods for enlarging objects of interest to reveal more detail have been explored. In a *full zoom* the whole representation is enlarged. As a smaller area of the representation fills the available display space, contextual information is lost. To maintain context, zooming has been extended to *insets*, *overview&detail* presentations, and *fisheyes*. Insets and overview&detail presentations show two views; one at original scale and one enlarged. The organization of the two views may vary and connections may be drawn between them [17], however context is adjacent rather then integrated. With fisheye views the area of interest is enlarged in place and context adjusted or compressed in order to provide space [10]. All images in Fig. 3 show the same graph. Fig. 3a is the unadjusted view while Fig. 3b to 3d show the application of different zoom methods; Fig. 3b a full zoom, Fig. 3c an inset, and Fig. 3d a fisheye lens. For both full zoom and the inset no additional information is gained. The confusing part is bigger but not clarified. The fisheye view (Fig. 3d) creates increased ambiguity because nodes that are near the focus are moved and enlarged. The magnified node of interest occludes a larger area of the graph and covers up some of the edges that are not attached to the node.

Filtering: Edge congestion can be relieved by removing less important edges, thus revealing only the important relationships in the graph [11]. To be effective this technique requires a good method to distinguish "important" from "unimportant" edges. Filtering also interferes with context: while the intended edges are visible, we lose information about how they relate to other, now invisible, edges. Partial filtering

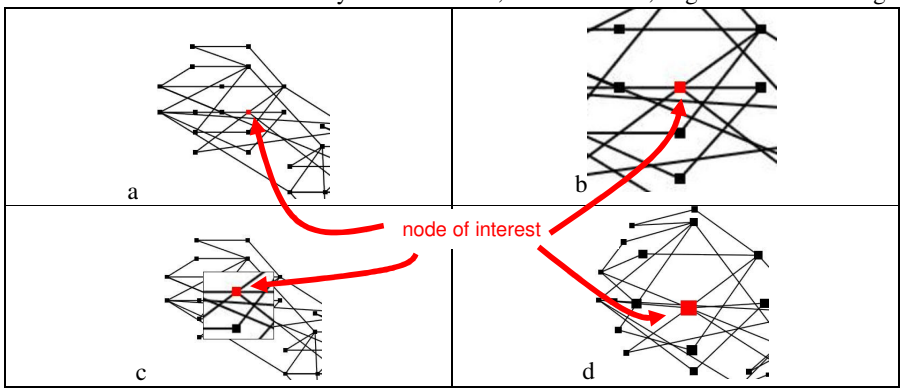

Fig. 3. Magnification techniques on a graph: a) original graph; b) full zoom; c) inset; d) fisheye

has also been used to remove the central portions of edges interactively. This is the technique used in SeeNet [1], SeeNet3D [3], and the visualization of MBone [12]. This filtering method leaves an indication that there was an edge and shows edge's direction, but precise relationships are harder to determine because the originally connected nodes are now visually disconnected.

Interaction: Interactive techniques have the advantage of being temporary, enabling the return to the original graph layout with ease. Several of the techniques mentioned above can be used interactively. However, the interactive versions—layout adjustment [15], various zooming techniques [8], and partial filtering [1, 3, 12]—have the same issues as discussed above, though readability issues are somewhat lessen by the ability to change the view and revert to original views. The two most closely related techniques are BubbleLens [2] and EdgeLens [20], both of which are interactive techniques that preserve node positions and temporarily bend edges to better reveal node-edge relationships. Edge Plucking extends these ideas with a more natural interaction metaphor and provides new facilities for exploring graph structure.

3 Readability Issues in Edge Congested Graphs

One method used to judge the quality of a layout has been to apply graph drawing aesthetics. These include such things as minimizing edge crossings, minimizing edge bends, maximizing minimum angles between edges, and aiming for continuity, symmetry and orthogonality. They have been developed gradually by the research community and have been studied empirically [13]. They provide means for assessing a graph layout, while we are interested in developing interactions that can improve the readability of a given layout through exploration. To do this we re-examined these aesthetics in terms of readability and possible ambiguities:

Occlusion

– **Edges obscuring edges:** It is possible to have multiple edges that are close to or actually overlapping each other. This can cause confusion when identifying node-edge relationships. Fig. 4 shows six different possible node and edge configurations in a three-node graph. Graph A is ambiguous and can be interpreted as any of the graphs B to F. In graph B, the two end nodes are connected to the centre node by two separate edges. Graph C has only one edge connecting both end nodes. In graphs D and E, the end nodes are connected but only one of these connects to the center node. Nodes of graph F are fully connected. Even a simple three-node graph can be ambiguous, creating difficulties when identifying

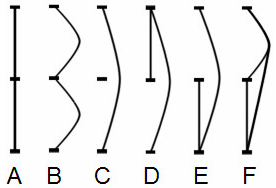

Fig. 4. Different possible node-edge relationships in a three-node graph

relationships between nodes.

- **Edges obscuring nodes:** If an edge crosses a node it can be difficult to assess whether this edge is incident to this node or is merely crossing it.
- **Edges obscuring other information:** When a large number of edges overcrowd a region they may obscure information in the background as well as other edges or nodes. The airline routes and Parallel Coordinates in Fig. 1 illustrate this problem.

Structure

- **Low angular resolution:** With low angular resolution, it can be difficult to distinguish between nearby edges (Fig. 5a). Node A has an edge connected to node B and another edge connected to node C, but it is difficult to see these connections.
- **Long edges:** It can be hard to follow long edges in graphs; this problem is exacerbated in edge congested graphs. For example, in Fig. 5b it is difficult to see which nodes are actually connected to node A. When following a long edge that is close to several other long edges, it is easy to lose track of the edge of interest.
- **Edge bends:** Bends or sharp corners can affect the readability of graphs [14] because sharp bends may be confused with nodes.
- **Inflection points:** Observations suggest that inflection points make edges hard to follow, although this hypothesis has yet to be formally tested. Graph readability is further worsened when edges cross at inflection points. For instance, the graph in Fig. 5c is difficult to read because a crossover occurs at the inflection point. It is difficult to tell whether node A is connected to node B or D, and whether node C is connected to node B or D.

While each of these issues can give rise to ambiguity in themselves, they tend to exacerbate each other. The higher the edge density the more likely it is that several of these issues will be present.

4 Edge Plucking

Edge Plucking is intended to provide one with the possibility of moving edges, one at a time or in a group, to achieve some clarification of graph structure, node and edge relationships or associated information. This explanation will begin with a description of the metaphor or concept of Edge Plucking, after which we will outline our design criteria and then explain the development and interaction.

The Edge Plucking metaphor comes from the action of plucking a guitar string.

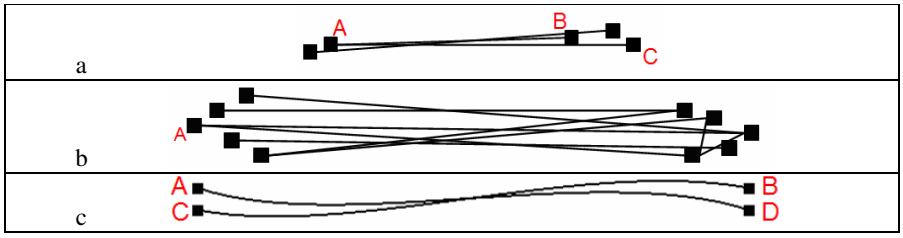

Fig. 5. Structural confusion: a) low angular resolution; b) long edges; c) inflection points

Fig. 6. Running a finger down a set of closed Venetian blinds

One plucks a guitar or violin string to make a sound but the action involves controlling the string at one point and causing it to move. The same action can be used on Venetian blinds, allowing one to peek through the blinds (Fig. 6). When one releases the blinds, they will return to their original shape. Edge Plucking mimics this plucking action for use in graph exploration, temporarily bending edges to mitigate edge congestion and clarify readability issues. The interaction concept behind Edge Plucking is to use a natural plucking action to mitigate edge congestion.

4.1 The Design Criteria

We are interested in developing an interactive tool that supports graph exploration and is capable of resolving layout ambiguities. In particular our design goals are:

- Reduce edge congestion by allowing edges to be moved aside.
- Alleviate ambiguity by providing a method by which ambiguous components can be separated.
- Provide naturalistic visual response by following the plucking metaphor. For instance, the edge should stay with the mouse cursor and should bend in an intuitive manner.
- Minimize introduction of edge crossings, sharp edge bends and inflection points.
- Preserve node properties such as size, shape, and location.
- Aid graph exploration by making it possible to interactively wiggle one or more edges [16].

The intention is to achieve this interactivity while minimizing layout disturbance and keeping interactions temporary.

4.2 The Development of Edge Plucking

A simple approach would be to separate an edge into two lines when it is plucked. The two lines would connect at the mouse cursor, with one line attaching to each node (Fig. 7). This is simple and reasonably effective but we felt that the visual response is not natural and the connecting point of the two straight lines forms a sharp angle. Another approach would be to use a spline with a control point at the cursor. Once again the edges can be moved aside but interaction is difficult because the mouse and the edges that are being moved are not connecting (Fig. 8).

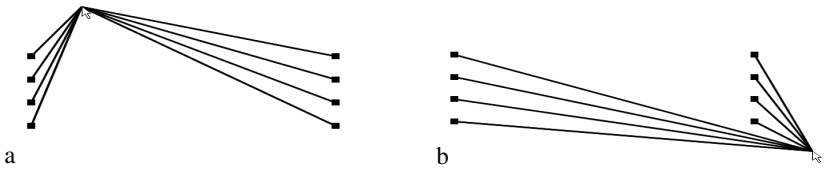

Fig. 7. Plucking action when the edge is separated into two straight lines

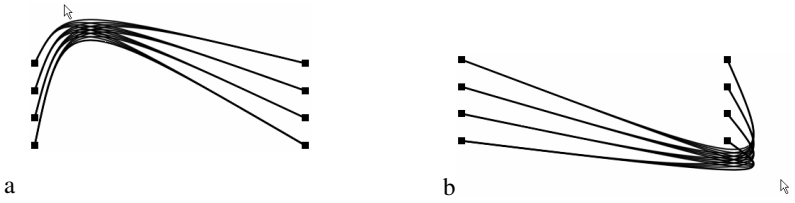

Fig. 8. Plucking action with a single spline, note distance from the cursor

To get the visual response we were looking for we tried several different splines including interpolating variations [5]. The following is the most satisfactory results we obtained. Visually it combines maintaining a direct connection with the mouse cursor and slightly curved edges. In addition it includes a variant response depending on the location of the point on the edge from which plucking is initiated. This variation in visual response appears quite natural in that the longer part of the edge appears more stretched and the shorter part of the edge seems to have somewhat less tension. Also, the shape of the curve gets tighter, in that the bend gets much sharper, as the pluck point approaches the node. While this does introduce the possibility of a sharp bend, it also keeps the shape of the edge more neatly within the boundaries set by the edge's nodes. This is illustrated in Fig. 9: Fig. 9a is the untouched edge; Fig. 9b through 9d shows different shapes of a plucked edge from smooth to a sharper curve.

A plucked edge is composed of two connecting cubic Bézier curves. They are controlled by seven control points (*cp*), where *cp1*, *cp2*, *cp3*, and *cp4* control one curve, and *cp4*, *cp5*, *cp6*, and *cp7* control the other curve. Control points *cp1* and *cp2* have the same location at node *n1*, and control points *cp6* and *cp7* are both located at node *n2*. The middle control point, *cp4*, the one that is shared by both curves is located at the mouse cursor *mc*.

When the mouse cursor touches the edge at *mc*, the edge is considered as two line segments joined at *mc*. The first step is to assess which line segment *n1* to *mc* or *mc* to *n2* is shorter. The control point on the shortest line segment will be calculated first,

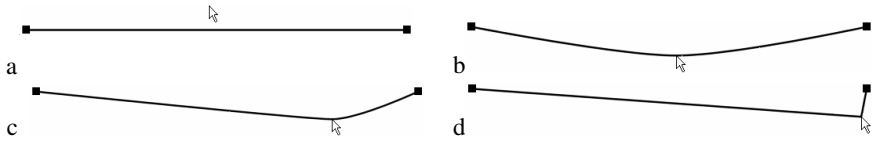

Fig. 9. An edges is being plucked in different ways: a) the original graph; b) the edge is plucked in the middle; c) the edge is plucked on the right side of the edge; d) the edge is plucked very close to the right node

and the remaining control point will be calculated afterwards. In Fig. 10 this is the segment *n1* to *mc* and *cp3* will be calculated using the formula:

$$dc = dn * r \quad (1)$$

where *dc* is the distance from *cp3* to *mc*, *dn* is the distance from *n1* to *mc*, and *r* is a number between 0 and 1. While *r* can be interactively adjusted, the current default for *r*, as used in the illustrations in this paper, was arrived at through observation and is *0.3*. The control point for the longer line segment, in Fig. 10 *cp5*, is placed on longer line segment using the same distance *dc* just calculated for the shorter line segment. In the example in Fig. 10 *cp5* is placed on the segment *mc* to *n2* at a distance *dc* from *mc*. This places the two calculated control points equidistant from *mc* and all three are

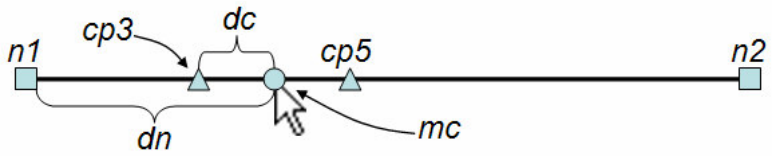

Fig. 10. Illustrate how *cp3* and *cp5* are calculated

on the original edge and thus are collinear.

At this point the mouse has touched the edge and we are ready for the edge to be plucked. The distance from *cp3* to *mc* equals the distance from *mc* to *cp5* and all three control points are collinear. When the mouse cursor moves, and consequently *mc*, this co-linear relationship will be maintained. The three points move in the same direction and distance as *mc* moves, maintaining a parallel relationship to the original edge (Fig. 11). That is, all three points will be moved in unison. Maintaining this co-linearity and using cubic Bezier splines, in spite of the fact that the end control points must be doubled at the nodes in order to get four control points per segment, does provide us with C^2 continuity at the location of the mouse cursor. This visual continuity seems important to provide the feedback that the edge is still a single edge and is responding as a unit.

Note that the locations of *cp3* and *cp5* depend on how close *mc* is to *n1* or *n2*. This dependency is essential to the characteristic of variable smoothness of plucked edges and allows the shape of the edge to be different on each side of the cursor. The closer the mouse cursor is to the centre of the edge, the smoother the plucked edge will be. As an opposite effect, the closer the cursor is to the end of the edge, the sharper the plucked edge will be. However, the previous discussion about visual continuity remains true even when the curve at *mc* is much sharper due to the choice of a pluck point close to a node.

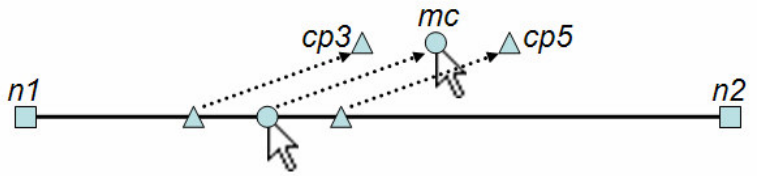

Fig. 11. *cp3* and *cp5* move in the same direction and distance as the cursor

4.3 The Interactive Control of Edge Plucking

To pluck edges, one must touch the edge(s) with the cursor, click, hold down the mouse button, and move the mouse as desired. Moving across edges while holding the mouse button down collects the edges touched under the cursor. Continued motion will pull the edges with the cursor. Edges can then be either released and they will return to their original positions or edges can be pinned at any chosen location.

One can pin edges that are being plucked by clicking the right mouse button while still holding the left mouse button down. Once edges are pinned, they shapes and locations are fixed, i.e., they will not be affected by other plucking actions. Pins can be removed at will by right clicking where the pins are located, at which point in time all edges fixed by the removed pins will return to their original position. However, one cannot remove any pins when the cursor is still plucking edges. This allows the possibility to pin multiple edges to the same location at different times. Note that in this situation where multiple pins are placed at the same location, they cannot be removed individually. This means that when right clicking on that location, all pins are removed and all edges are returned to their original shapes. Also note that pinned edges cannot be unpinned individually. This is one of many possible ways to control Edge Plucking, and we are not claiming that this is the best way. Whether this control method is efficient or not is yet to be determined.

5 Applying Edge Plucking

In this section we show how Edge Plucking can be used to address the graph readability issues introduced in Section 3. Then we contrast the use of Edge Plucking with the use of EdgeLens.

1. **Edges obscuring edges.** Fig. 12 shows how Edge Plucking can be used to disambiguate overlapping edges. The node-edge relationship in the graph in Fig. 12a is unclear. With Edge Plucking, we can see the actual configuration of the graph (Fig. 12b). This is one of the graphs used in Fig. 4.
2. **Edges obscuring nodes:** Plucking an edge near a node will reveal whether the edge is connected to or passing across the node (Fig. 12). Or in the case where many edges obscure the structure of the graph, Edge Plucking can be used to move the edges away from a region of interest to reveal hidden structure (Fig. 13). However, it may be difficult to identify which edge sections on one side of the cursor are connecting to which sections on the other side as in Fig. 13b.
3. **Edges obscuring other information:** One can pluck an edge or collection of edges and move them aside to read background information (Fig. 14c).
4. **Low angular resolution.** In Fig. 15a, the angular resolution of the node on the left is low, but can be interactively increased when Edge Plucking is applied (Fig. 15b).
5. **Long edges:** Ware and Bobrow's [16] study confirmed that motion can help graph exploration. When a long edge (or group of edges) is plucked, it draws attention to its nodes when wiggled.

Fig. 12. Using Edge Plucking to disambiguate a three-node graph

Fig. 13. Using Edge Plucking to reveal hidden nodes

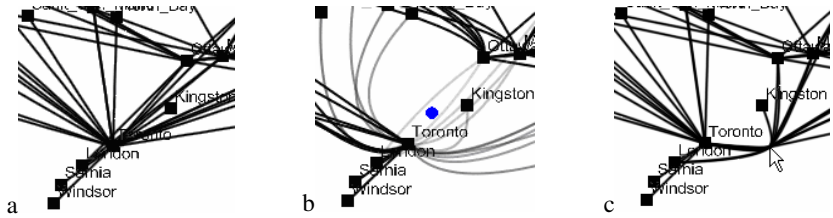

Fig. 14. A graph with the label "Toronto" occluded: a) the original graph; b) EdgeLens is applied; c) Edge Plucking is applied

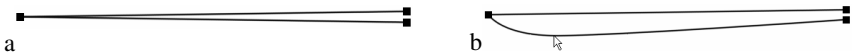

Fig. 15. Using Edge Plucking to increase angular resolution

6. **Edge bends and inflection points:** Since plucking an edge changes its shape it can be used to clarify ambiguities arising from edge bends and inflection points.

While both Edge Plucking and EdgeLens are able to address these readability issues in edge congested graphs, Edge Plucking does have advantages over EdgeLens in certain situations. For instance, in the situation of revealing background information, EdgeLens reveals the background partly by pushing edges aside and partly through use of translucency (Fig. 14b); in contrast, Edge Plucking can much more directly just move the occluding edges aside (Fig. 14c).

Since Edge Plucking can be applied to one or more edges, while EdgeLens is applied to a given region, Edge Plucking seems more amenable to the piece by piece exploration of graph structure. For example, consider exploring the small portion of a circular graph structure shown in Fig. 16a. Here there are seven nodes connected by several edges that are all lying along the slight circular arc. Applying an EdgeLens to the region of interest does reveal the structure (Fig. 16b); however, it can be difficult to choose just the right position to achieve a reasonable spread to the edges and the structure is still somewhat difficult to decipher. Instead, one can use Edge Plucking (Fig. 16c to 16h) more deliberately. Following through from Fig 16c each edge is plucked and pinned to the side, gradually revealing the structure. Pinning all six edges clarifies the relationships between nodes. While Edge Plucking is effective with larger

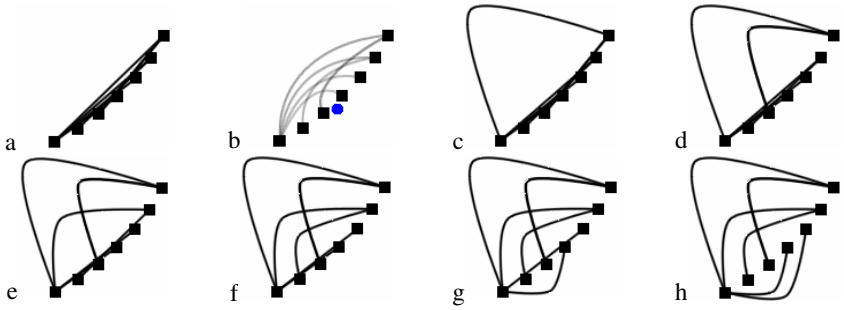

Fig. 16. Exploring a cluster of nodes in a graph: a) the original graph; b) EdgeLens is applied; c) through h) Edge Plucking is used separate the edges within the cluster of nodes

graphs, we have illustrated this discussion with small graphs so that the static images will be clear. Much of the effectiveness is a result of the motion plucking causes.

In our experience, Edge Plucking and EdgeLens facilitate different graph exploration techniques. EdgeLens provides a quick overview of basic structure, while Edge Plucking lends itself to more careful and precise exploration of the structure. We intend to continue with this comparison and use empirical means to determine which methods are best suited to which tasks.

6 Conclusion

In this paper we introduce Edge Plucking, an interactive technique for mitigating edge congestion in graphs. Edge Plucking has the following merits:

- interactively and temporarily moves edges;
- preserves node positions;
- separates overlapped edges;
- increases angular resolution;
- reveals hidden information that is occluded by congested edges; and
- makes long edges easy to follow.

In addition, Edge Plucking offers some advantages over the earlier interactive technique EdgeLens. The interaction metaphor behind EdgeLens is based on repelling edges from the cursor's location, the metaphor of plucking and moving edges aside may be more natural. Edge Plucking is more effective at clearing edges off background information and seems better suited to the exploration of graph structure. As we continue this work we will study these differences empirically.

References

1. Becker, R., Eick, S., and Wilks, A.: Visualizing Network Data. IEEE Transactions on Visualization and Computer Graphics, Vol. 1(1), (1995) 16-28.

2. Carpendale, S. and Rong, X.: Examining Edge Congestion. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2001), Interactive posters: visualizing, Vol. 2, (2001) 115-116.
3. Cox, K., Eick, S., and He, T.: 3D Geographic Network Displays. ACM Special Interest Group on Management of Data Record, Vol. 25(4), (1996) 50-54.
4. Di Battista, G., Eades, P., Tamassia, R., and Tollis, I.: Algorithms for Drawing Graphs: An Annotated Bibliography. Computational Geometry: Theory and Applications, Vol. 4(5), (1994) 235-282.
5. Dyn, N., Levin, D., and Gregory, J.: A Four-Point Subdivision Scheme for Curve Design. Computer Aided Geometric Design, (1987) 257-268.
6. Herman, I., Melancon, G., and Marshall, M.: Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Transactions on Visualization and Computer Graphics, Vol. 6(1), (2000) 24-43.
7. Inselberg, A.: Multidimensional Detective. Proceedings of the IEEE Symposium on Information Visualization (InfoVis '97), (1997) 100-107.
8. Jankun-Kelly, T. and Ma, K.: MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes. Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2003), (2003) 59-66.
9. Lamping, L., Rao, R., and Pirolli, P.: A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. Proceeding of the ACM Conference on Human Factors in Computing Systems (CHI '95), (1995) 401-408.
10. Leung, Y. and Apperley, M.,: A review and Taxonomy of Distortion-Orientation Presentation Techniques, ACM Transactions of Computer-Human Interaction, Vol. 1(2), (1994) 126-160.
11. Mukherjea, S. and Foley, J.: Visualizing the World-Wide Web with the Navigational View Builder. Computer Networks and ISDN Systems, Vol. 27(6), (1995) 1075-1087.
12. Munzner, T., Hoffman, E., Clarify, K., and Fenner, B.: Visualizing the Global Topology of the MBone. Proceedings of the IEEE Symposium on Information Visualization (InfoVis '96), (1996) 85-92.
13. Purchase, H.: Evaluating Graph Drawing Aesthetics: Defining and Exploring a New Empirical Research Area. Computer Graphics and Multimedia: Applications, Problems and Solutions, DiMarco, J. (ed.), Idea Group Publishing, (2004) 145-178.
14. Purchase, H., Cohen, R., and James, M.: Validating Graph Drawing Aesthetics. Proceedings of the Symposium on Graph Drawing (GD '95), (1995) 435-446.
15. Storey, M.-A.D., and Müller, H.: Graph Layout Adjustment Strategies. Proceedings of the Symposium on Graph Drawing (GD '95), (1995) 487-499.
16. Ware, C. and Bobrow, R.: Motion to Support Rapid Interactive Queries on Node-Link Diagrams. ACM Transactions on Applied Perception, Vol. 1(1), (2004) 3-18.
17. Ware, C. and Lewis, M.: The DragMag Image Magnifier. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '95), Videos, Vol. 2, (1995) 407-408.
18. Wills, G.: NicheWorks—Interactive Visualization of Very Large Graphs. Journal of Computational and Graphical Statistics, Vol. 8(2), (1999) 190-212.
19. Wong, N.: EdgeLens: An Interactive Technique for Mitigating Edge Congestion in Graphs. MSc Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4, April (2005).
20. Wong, N., Carpendale, S., and Greenberg, S.: EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs. Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2003), (2003) 51-58.