EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs

Nelson Wong[*]
Department of Computer Science
University of Calgary

Sheelagh Carpendale[†]
Department of Computer Science
University of Calgary

Saul Greenberg[‡]
Department of Computer Science
University of Calgary

Abstract

An increasing number of tasks require people to explore, navigate and search extremely complex data sets visualized as graphs. Examples include electrical and telecommunication networks, web structures, and airline routes. The problem is that graphs of these real world data sets have many interconnected nodes, ultimately leading to *edge congestion*: the density of edges is so great that they obscure nodes, individual edges, and even the visual information beneath the graph. To address this problem we developed an interactive technique called EdgeLens. An EdgeLens interactively curves graph edges away from a person's focus of attention without changing the node positions. This opens up sufficient space to disambiguate node and edge relationships and to see underlying information while still preserving node layout. Initially two methods of creating this interaction were developed and compared in a user study. The results of this study were used in the selection of a basic approach and the subsequent development of the EdgeLens. We then improved the EdgeLens through use of transparency and colour and by allowing multiple lenses to appear on the graph.

CR Categories: I.3.6[Computer Graphics]:Interaction Techniques

Keywords: Navigation, graph layout, distortion lens, information visualization, edge congestion, interactive visualization

1 Introduction

In everyday life, we come across many types of information that we wish to understand better. Often it is the complex relationships within the information that are of particular interest, such as the trade relationships among cities, or the physical connections inherent in telephone systems, power grids, airline routes and road maps. Graphs are a popular method for representing this kind of complex information, allowing us to visualize trade routes by representing the cities as nodes and trade relationships as edges. Yet real world data sets tend to be huge, and as their size increases, so does the complexity of the graph layout. In practice, this leads to *edge congestion,* where excessive edge density in a region leads to edge-crossings and overlapping edges, which in turn obscures nodes and any extra visuals under these regions. Because of edge congestion, people often have difficulties in understanding the information represented by the graph. For example, Figure 1 shows a real world graph of NorthWest Airlines routes taken from their in-flight magazine. In it, we see that edge congestion is severe enough to create regions entirely covered by edges. It is difficult to tell if an air route passes over a city or stops at it. It is hard to read the information on the underlying graphic, e.g., lacks and labels. It is hard to trace a particular route because of how edges overlap each other, yet this is not a particularly dense graph.

[*]e-mail: yw@cs.ucalgary.ca
[†]e-mail: sheelagh@cs.ucalgary.ca
[‡]e-mail: saul@cs.ucalgary.ca

IEEE Symposium on Information Visualization 2003,
October 19-21, 2003, Seattle, Washington, USA
0-7803-8154-8/03/$17.00 ©2003 IEEE

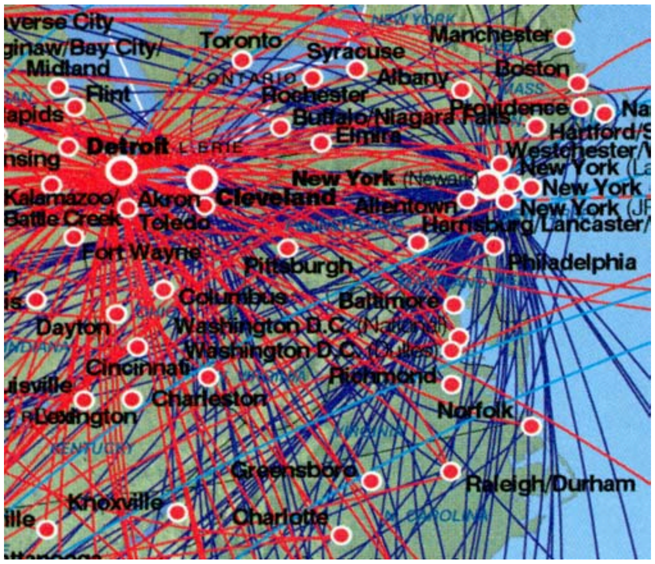

Figure 1. Airline routes from NorthWest Airlines, "World Traveler", November, 2001.

In this paper we introduce the concept of EdgeLens, an interactive method for managing edge congestion. After defining the edge congestion problem (Section 2) and describing how others have approached this problem (Section 3), we outline the EdgeLens approach and two particular ways it can be realized (Section 4). Through our user study (Sections 5 and 6), we show how the EdgeLens, based on cubic Bézier, proved superior. Finally, we describe the EdgeLens algorithm and offer several refinements (Section 7).

2 The Edge Congestion Problem

Graph layout is a challenging problem [Di Battista et al. 1994]. It increases in difficulty as the size and complexity of the data increases, and can be particularly onerous if one wants to incorporate readability considerations into the layout [Purchase 2000]. To add to this problem, some data visualizations further constrain the layout to reflect data semantics, e.g., relative node

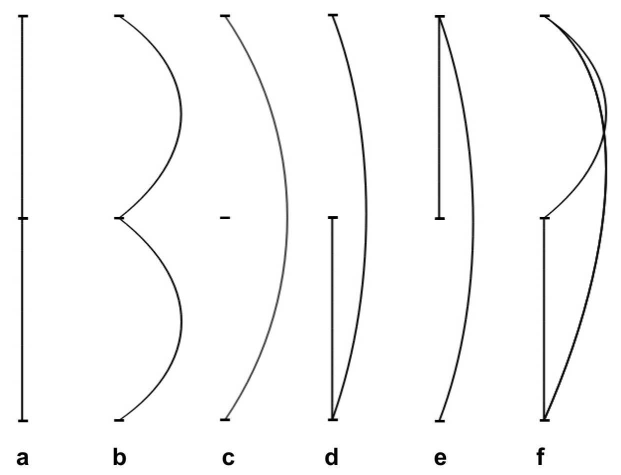

Figure 2. Ambiguity: the three node graph on the left could be as it appears or any of the configurations to the right.

positioning in relation to the rest of the data. Our concern in this paper is how edges appear within a graph, for previous studies have shown that edge placement can cause considerable difficulty in how people read graphs [Purchase 1997].

One method for managing edge layout is to alter the graph layout itself, e.g., by moving nodes to positions that minimize edge densities, crossovers and/or occlusions. This approach has been shown to be difficult [Di Battista et al. 1994] and is not amenable to all graph layouts. For example, Figure 1's graph of airline routes represents cities as nodes, where nodes are located at certain relative locations to reflect the city's geographic semantics and should not be moved. Even if the layout could theoretically be reorganized to eliminate edge crossings, much geographic meaning would be lost.

While the semantics of node positioning can add value to the visualization, they also introduce the possibility of ambiguities due to the way edges overlap. Figure 2 illustrates the inherent ambiguities in a simple three-node graph. If we represent connected edges as straight lines, we get Figure 2a: the viewer has the impression that the central node is connected by an edge to the node on the top and by another edge to the node on the bottom. Yet this may not be the case. If we constrain edges to their connected nodes but otherwise 'curve' the edges to reveal what is underneath, we can see other possibilities: perhaps the end nodes only connect to the middle node (2b), or only the end nodes are connected (2c), or while the end nodes are connected only one of these connect to the central node (2d+e), or it is in fact fully connected (2f).

Aside from edge occlusion, other ambiguities exist. When an edge passes under a node, the reader cannot tell if it is connected or unconnected. If many edges are drawn over or near a node, they can obscure that node and its labels. If information exists under the node, as in Figure 1's outline of North America, that information becomes hard to see. It is this problem of edge congestion that we address in this paper. To foreshadow what is to come, the curving of edges in Figure 2 is exactly what the EdgeLens does: by distorting the edge shape, the reader can disambiguate the configuration to see how nodes are truly connected and open up the graph to reveal underlying information.

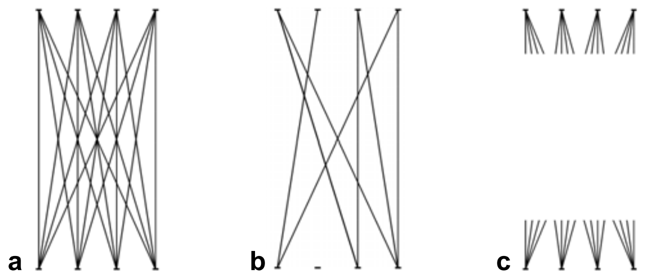

Figure 3. Various filtering approaches: from the graph on the left the centre graph removes several edges and the right most graph removes the central portion of the edges.

3 Previous Approaches to Edge Congestion

Many different attempts have been made to address or at least minimize the problem of edge congestion.

Layout: Manual or computational graph layout can potentially minimize edge crossing. In practice, optimal solutions are difficult to find [Di Battista et al. 1994; Herman 2000; Wills 1999,]. Interactively moving nodes is another way of addressing this problem. However, it fails if a node's position is important due to its semantic meaning (such as in Figure 1). We want to avoid the semantic confusion that can be created when the nodes are moved. The idea of curving edges [Cox et al. 1996; Lamping et al. 1995] has been applied globally. Instead, we let users interactively curve selected edges.

Filtering: Filtering relieves congestion by removing 'unimportant' edges, thus revealing only the important relationships in the graph [Consens et al. 1992; Furnas 1986; Mukherjea 1995]. For example, Figure 3b shows a filtered view of the graph in Figure 3a, where the edges that remain were judged more relevant than the filtered edges. This only works if we have a way to distinguish 'important' from 'unimportant' edges. Another problem is that filtering interferes with context: while we see particular edges, we lose how they relate with other now invisible edges (Figure 3b). Filtering has also been used partially to remove the central portions of edges interactively (Figure 3c). This leaves indications that there was an edge and shows its direction [Becker et al. 1995], but precise relationships are harder to determine because connections are now left to the 'minds eye'.

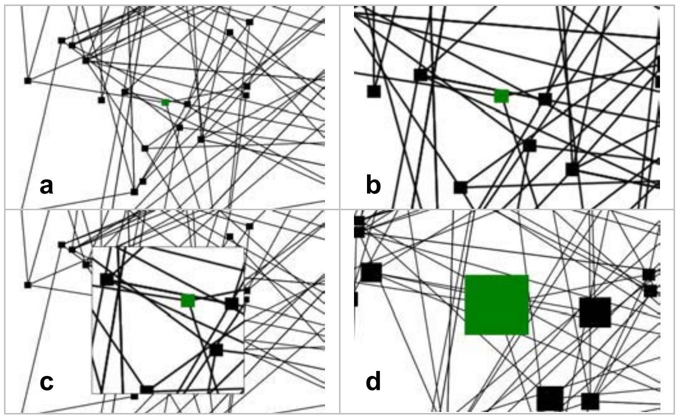

Figure 4. Magnification alone does not help: top left show an ambiguous node; top right, full zoom; bottom left, an inset; bottom right, a fisheye.

Magnification: Magnification enlarges areas of a graph – either linearly or non-linearly – so they can be seen in greater detail. Many approaches now exist: insets [Ware and Lewis 1995], fisheye views and other distortion based approaches [Carpendale et al. 1995; Furnas 1986; Keahey and Robertson 1996; Keahey and Robertson 1997; Lamping et al. 1995; Leung and Apperley 1994; Sarkar and Brown 1992], Magic Lenses [Bier et al. 1993], and zoomable user interfaces [Bedersen and Hollan 1994]. The problem is that enlarging does not necessarily reduce edge congestion.

For example, Figure 4a shows a portion of a graph layout with an ambiguous node highlighted in green. The full zoom in 4b and the magnified inset in 4c offer no further clarification. While Figure 4d uses a fisheye distortion to magnify nodes, this actually makes it harder to tell which edges are incident to the now magnified green node because it occludes a larger area.

4 The EdgeLens: an Interactive Approach

Our goal is to relieve problems caused by edge congestion. Our method is to develop the EdgeLens: an interactive technique that respects the semantics of node layout, disambiguates edge and node overlapping, and clarifies details about the graph structure.

To explain, we first assume that the locations of the nodes have a semantic meaning, and consequently nodes should not be moved. We also assume that the meaning of an edge is in its actual attachment to nodes: as long as the edge remains visible and attached to the appropriate nodes its semantics will remain intact. Given this, the basic idea behind the EdgeLens is to interactively move edges without detaching them from the nodes, while keeping all the nodes stationary. Figure 2b-f show a simple example: while the nodes and edge connections are intact, the lines defining the edges are moved by distorting their shape. In this manner we hope to disambiguate graph layouts without changing their meaning.

The EdgeLens borrows from ideas in detail-in-context distortion-based viewing. It too uses a linear lens with a point focus, and a radius that limits the range of its effects. It differs from a detail-in-context lens in that:

- We separate the effects of displacement and magnification, and use displacement only. This distinction has previously been mentioned [Leung and Applerley 1994; Keahey and Robertson 1996], and has been applied in 3D access distortion [Carpendale et al. 1997].
- The distortion is applied only to the edges, and not to any other part of the graph or underlying image. That is, the data is divided into two discrete parts, the edges and everything else. The effects of the distortion field are applied to the edges only.

We initially developed two types of EdgeLenses using a *Bubble* and a *Spline* approach to distort lines. Both are implemented using Elastic Presentation Framework (EPF) [Carpendale and Montagnese 2001].

The Bubble approach: This approach affects the local area only, as defined by the lens radius. As Figure 5a illustrates, all edges are provided with bend points and drawn as line segments from bend point to bend point. The bend points that are within the lens radius are displaced using a linear lens from EPF and the edge is redrawn [Carpendale and Rong 2001].

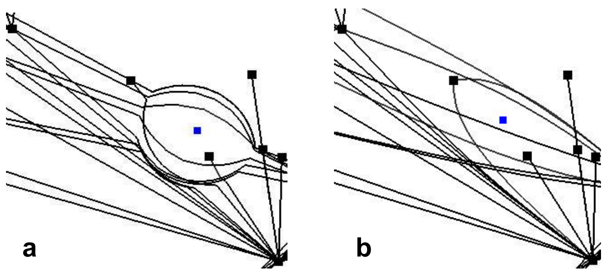

Figure 5. Two EdgeLens approaches. a) Bubble; b) Spline.

The Spline approach: With this approach, control points are calculated for all edges that fall within the influence of the lens. Then the distortion field is applied to these control points and the control points are used to apply a cubic Bézier to the edge. A cubic Bézier curve has four control points and interpolates the first and last control points. This creates a smooth curve that extends from node to node (Figure 5b).

As seen in Figure 5, the algorithmic, visual and interactive effects of these two approaches are very different. The Bubble approach only affects a small localized area, while the Spline approach shifts the edges it touches along their entire length. The complexity of the Bubble approach is dependent on the number of edge-bend-points and these in turn control the smoothness of the curve. The complexity of the Spline approach is dependent on the spline-control-points. At this point, rather than chose the preferred interaction based on algorithmic elegance, we ran a user study to compare these two approaches and to help us select the method that we would develop into the EdgeLens.

5 Comparing the Bubble and Spline Approaches

The Bubble and Spline approaches to the EdgeLens both mitigate edge congestion problems by revealing nodes and edges that would otherwise be occluded. We wanted to refine and extend one of these approaches, but we did not yet know which was better at managing edge congestion in practice. Consequently, we ran a controlled user study comparing people's performance and preferences when using the Bubble *vs* Spline approaches to path-finding within a geographic graph representing airline routes in Canada (Figure 6). The study and our results are discussed here.

Hypothesis: Because we had no *a priori* notion as to which lens would be better overall, we began with a null hypothesis:

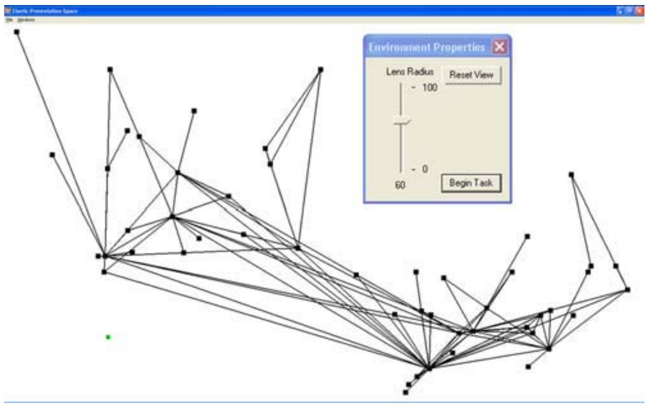

Figure 6. The software used in the study.

There is no difference in people's performance when using the Bubble *vs.* Spline approach to perform path-finding tasks of varying difficulty in a graph. Performance is measured by the *path quality* (incorrect, sub-optimal, and optimal) of the path found, the participant's *certainty of correctness* that they chose a correct path with a given lens (using a five point Likert measure anchored at very uncertain to very certain), and the *time* (seconds) to find the path.

From this hypothesis, independent variables are the Lens Type (2) X Task Difficulty (4). Dependent variables are path quality, certainty of correctness, and time. In actuality, we did expect that harder tasks would take longer to do, but wanted to check that there was no interaction between Lens Type and Task Difficulty.

Participants: We recruited 16 participants (8 males and 8 females) with formal education in computer science. All were experienced with both graphs and computers, but had no prior experience with EdgeLenses.

Materials: We created a practice graph, and a main graph that illustrated airline routes (edges) between major Canadian cities (nodes). City nodes were located at their approximate geographic position. As visible in Figure 6, the main graph contains several areas of congestion: the overlapping edges means some edges occlude one or more others, and some nodes are partially obscured.

We then created eight route-finding tasks of varying difficulty with this graph: easy, medium-easy, medium and hard. There are two tasks per category. Easy tasks are paths with only two intermediate nodes. In contrast, hard tasks have five or more intermediate nodes. They all go through overlapping edges and congested areas. Each task used different starting and destination nodes. Two otherwise identical versions of the software implemented either the Bubble or Spline approach as just described. The software would load a graph, and the people could then explore that graph by moving the particular lens around it with their cursor. People could also adjust the lens radius through a graphical slider. The software timed how long it took to do each task. A post-session questionnaire collected each participant's preferences between the two lenses as well as their comments.

Design and Method: The experimental design was within-subjects for both the lens type and task difficulty. Lens type was counter-balanced to minimize learning effects. That is, we randomly assigned half the subjects to start with the Bubble approach, while the other half started with the Spline approach. The sequence of events is as follows.

1. A participant is seated in a quiet room in front of a computer that displays the EdgeLens system running a particular lens.
2. The experimenter then explained graph terminology and concepts (nodes and edges), showed the participant how to control the lenses with the software, and explained what he or she had to do in each trial. In particular, participants were asked to imagine that they were a travel agent, where they were using this software to look for an optimal airline route from one city to another, perhaps including stopovers. Optimal routes are those that pass through the least number of cities possible, and do not go through a city more than once.
3. The participant then began with a practice session with no assigned tasks, where he or she interacted with a practice graph until comfortable with the use of that lens.
4. The participant was then given each task in turn, ordered by difficulty from easy, to medium-easy, to medium, to hard. For each task:
   - the experimenter coloured the start, end and (optional) stopover city that comprised the task question, explained what had to be done, and asked the participant to think aloud while doing the task;
   - the participant clicked a button to start the timing;
   - while exploring the graph with the lens, the participant selected the desired route by clicking the intermediary nodes (cities);
   - the experimenter videotaped user actions, and wrote down participant comments, how they used the EdgeLens, and the route picked.
   - the participant clicked the button to stop timing.
5. Steps 3-4 would be repeated with the other lens. Tasks differed from set to set, but were equivalent in difficulty.
6. Participants answered the post-session questionnaire.

6 Results

Path quality: All participants produced a path for each trial, for each lens at a particular task difficulty. A path was graded as *optimal* if it passed through the least number of cities (nodes) possible; *sub-optimal* if it was correct but passed through more nodes than necessary; and *incorrect* if the path did not meet the task specifications. All results are plotted in Figure 7, where each bar shows the ratio of optimal, sub-optimal, and incorrect paths for a particular lens and task difficulty.

The chart clearly shows that, in all cases, participants using the Spline approach had far fewer incorrect paths per task type when compared to the Bubble approach. Similarly, people produced more optimal paths with the Spline *vs* Bubble approach (pairwise t-test $p<.001$). We reject the null hypothesis for path quality, as people produce more optimal paths and fewer incorrect paths with the Spline *vs* the Bubble at comparable tasks.

Certainty of correctness: As part of the post-test questionnaire, participants were asked: "How certain are you with the correctness of the airline routes you picked?" They responded on a 5 point scale ranging from very uncertain to very certain for both lens types. Results are plotted in Figure 8. A pairwise t-test shows there is a statistically significant difference ($p<.001$). As indicated in the chart, in almost all cases participants had a higher certainty of correctness with the Spline approach. All but two

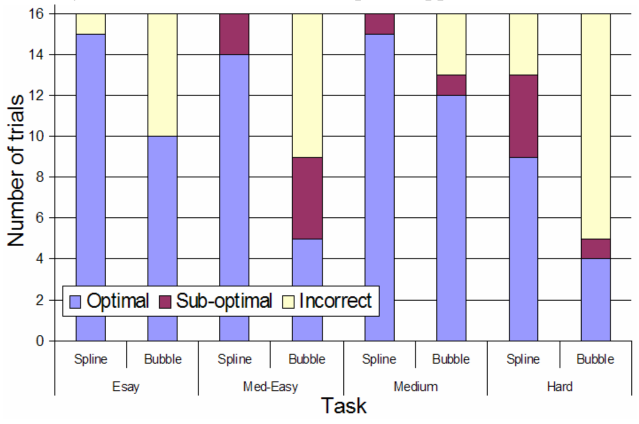

Figure 7. Spline approach led to fewer incorrect paths and more optimal paths per task type.

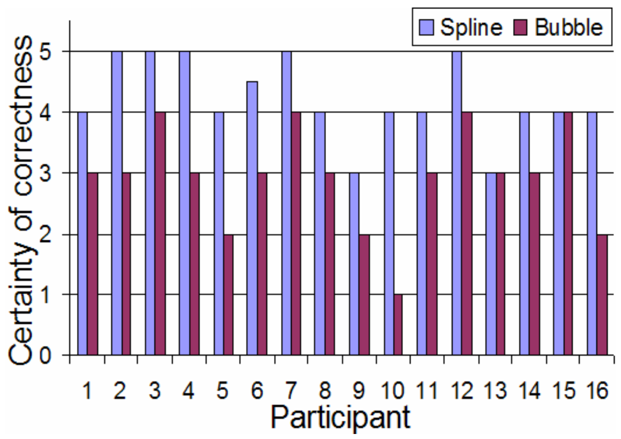

Figure 8. Participants were more certain that their paths were correct when using the Spline approach.

participants gave it a high score between 4 and 5.

Thus we reject the null hypothesis for certainty of correctness, for we saw that people have greater certainty that they found a correct path with the Spline *vs* the Bubble approach over all tasks.

Time: We logged the time required for each task to be completed. Thus we had 16 data points (one per participant) for each lens at a particular task difficulty, for a total of 128 data points. Figure 9 plots the means as a chart, where each bar shows the mean time for each level of task difficulty to be completed using the Bubble approach and the Spline approach.

We analyzed this data with a two-way repeated measures analysis of variance: Lens Type (2) X Task Difficulty (4). Results show a main effect for Task Difficulty (*F(3,45)=46.699, p<.001*), but no interaction (*F(3,45)=1.130, p=.347*) and no significant Lens Type effect (*F(1,15)=3.360, p=.087*). Though we cannot reject the null hypothesis for time, we saw that people on average are faster at completing path-finding tasks with the Spline *vs* the Bubble at comparable tasks. We also saw that overall time increases with both the Bubble and Spline methods when performing path-finding tasks of varying difficulty in a graph.

Participants' preferences and comments: The post-test questionnaire asked subjects which EdgeLens they preferred, and to comment on their choice. Every single participant preferred the Spline over the Bubble approach. Comments were generally negative about the Bubble approach, and positive about the Spline approach. Some examples are included here.

- Spline EdgeLens
  - "Very good looking – appealing."
  - "Works great for identifying if a path exists between two points."
  - "It actually identifies routes (edges) very well."
- Bubble EdgeLens
  - "Edges bend in a weird way."
  - "Awkward & not useful."
  - "Harder to use and the effect is not as clear."
  - "I don't like this lens at all."

Summary: In the quantitative and qualitative measures, the Spline approach outperforms the Bubble approach when people use them for path-finding tasks of varying difficulty.

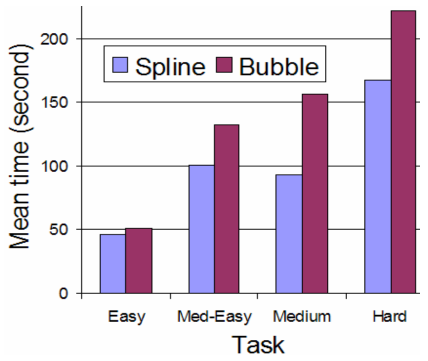

Figure 9. Participants complete tasks faster when using the Spline approach.

7 The EdgeLens: implementation details and refinements

Because the study results clearly suggest that the Spline approach is superior, we based our refined implementation of the EdgeLens on the Spline approach. We also describe how we enhanced its effect with the selective use of transparency, with colour, by providing multiple EdgeLenses in a graph.

An EdgeLens has a centre, a magnitude, and a radius of influence. A user can interact with an EdgeLens by placing and moving the EdgeLens centre, which is drawn as a small coloured square or circle. Moving and adjusting the EdgeLens produces the visible effect of moving edges. The user can also adjust the magnitude and shape of the edge displacement through a control such as a slider (e.g., as in Figure 6).

The Algorithm: there are four basic steps to create an EdgeLens, annotated in Figure 10.

Step 1: Decide which edges will be affected by the EdgeLens. If a perpendicular line can be drawn from a point on an edge in the graph to the EdgeLens centre and this point is within the EdgeLens' radius of influence, then that edge will be affected by the EdgeLens. The point on the edge from which the perpendicular line can be drawn is the seed control point (s_c). In Figure 10, the edge is affected by *e-lens*$_1$, but not by *e-lens*$_2$.

Step 2: Use the seed control point (s_c) to calculate the displacement. The seed control point is displaced using a

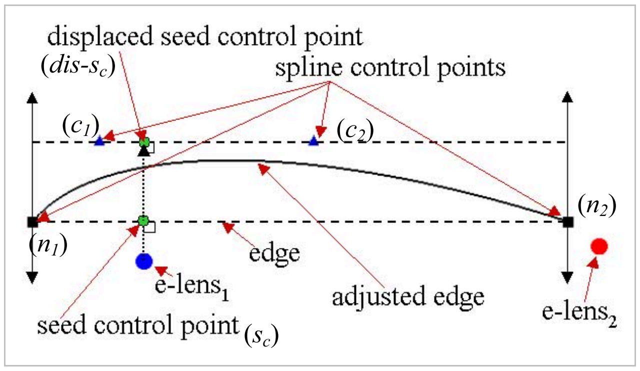

Figure 10. The workings of an EdgeLens.

radial distortion function, such that the distortion drops off smoothly from the centre until it reduces to no effect. In particular, we use the 2D result of a point-focus linear lens from EPF library [Carpendale and Montagnese 2001] to calculate the amount of displacement and the location of the displaced seed control point (*dis-*s_c). The magnitude of the displacement factor can be set by the user, and this in turn affects the resulting location of *dis-*s_c.

Step 3: Calculate the control points for the spline. We use *dis-*s_c to calculate the two control points c_1 and c_2 with one on each side of *dis-*s_c. These three points are lined up on a straight line, which is parallel to the unadjusted edge (Figure 10). The actual position of c_1 can be calculated based on the formula: $dc = de * r$ where de is the distance from s_c to n_1, dc is the distance from *dis-*s_c to c_1, and r is a number between 0 and 1. Similarly, c_2 uses the same formula with the distance from s_c to n_2 be de, the distance from *dis-*s_c to c_2 be dc. Notice that c_1 and c_2 will always stay in between n_1 and n_2. We can change the shape of curved edges by adjusting the ratio (r). Doing so, we move the locations of control points c_1 and c_2 and thus change the way edges curve.

Step 4: Draw a curved edge with a cubic Bézier curve. The control points, n_1, c_1, c_2 and n_2 are used to draw the curve.

In Edgelens the affected edge is pushed to the right when it is approached from the left, it returns through neutral as the EdgeLens passes across it and then is pushed to the left when the EdgeLens is on the right. The EPF library provides us with the freedom to choose the type of drop-off integration, the method by which more than one lens affect each other and to easily adjust the displacement and the lens radius. While EPF calculations are done in 3D, for the EdgeLens to operate in 2D, we simply use the 2D result or back-projection to find the location of the displaced seed control point, *dis-*s_c. For further explanations of EPF see [Carpendale et al. 1995; Carpendale and Montagnese 2001]. It is also possible to create a single EdgeLens from a simple distortion function such as *displace = (2 * dis)/(1 + (dis/l-rad))*, where *displace* is the displacement, *dis* is the distance between the lens centre and the seed control point, and *l-rad* in the lens' radius of influence. The new location of the displaced seed control point, *dis-*s_c, will be the lens centre plus the displacement, *dis*. Creating multiple EdgeLens with a formula like this would require investigating how several lenses interact with each other.

Transparency: As a person moves the EdgeLens focal point, the motion of the affected edges makes the emerging graph structure quite noticeable. However, when these edges are stationary, additional visual cues can be helpful in discriminating affected edges from unaffected ones. One solution is to adjust the transparency level of the curved edges. For example, Figure 11a shows a graph with considerable edge density, and Figure 11b applies an EdgeLens to this graph to reveal two previously hidden nodes. Figure 11c adds transparency to the curved edges, which further clarifies the structure. An additional advantage of transparency is that users can now see through clusters of edges, revealing information that would otherwise have been obscured by solid edges. Figure 12 shows an example, where the labels Kingston and Toronto are now clearly visible even though they overlap some of the now-transparent edges.

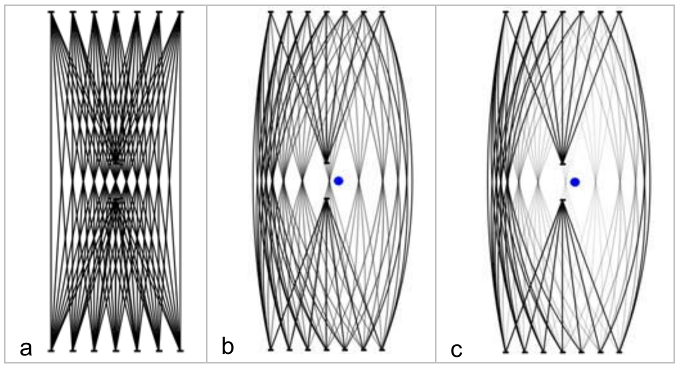

Figure 11. Applying transparency. a) a graph with considerable edge crowding; b) an EdgeLens reveals hidden structure; c) transparency makes this more clear.

Excluding Edges: People sometimes want to understand edge relationships between specific nodes, and consequently they may want to apply the EdgeLens selectively to move all but the desired edges to the side. For example, Figure 13a shows a graph where a person has a special interest in the edges connected to the 2nd node down from the top-right; because the EdgeLens moves these edges aside as well, their connections are hard to see. To solve this problem, nodes are selectable. When nodes are selected, their attached edges are excluded from the effect of the EdgeLens. These nodes and unaffected edges are coloured to further assist reading the graph structure. As seen in Figure 13b, the user has selected the (now red) node, and when the lens is moved near this excluded node, all connected edges (coloured red as well) are kept straight.

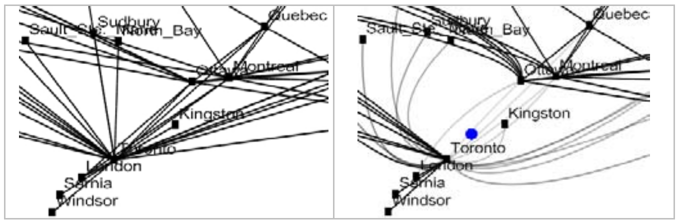

Figure 12. EdgeLens plus transparency reveals labels.

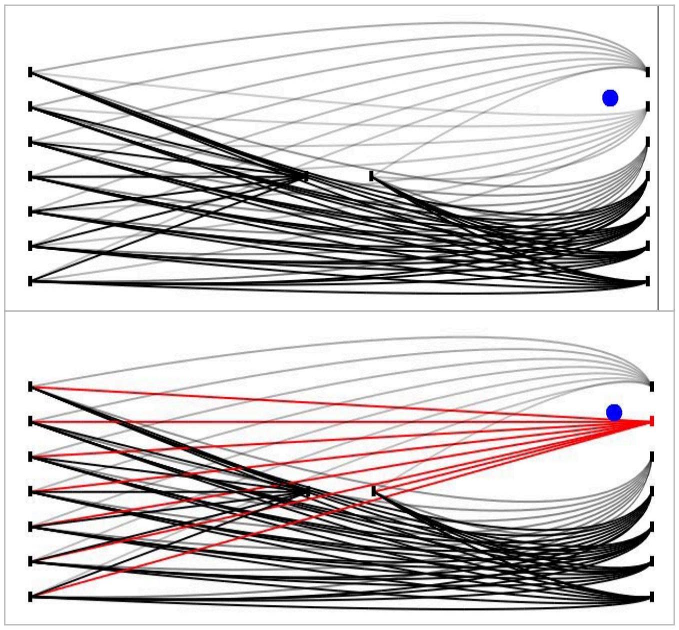
Figure 13. The edges of a selected node are excluded from the EdgeLens effect and coloured red.

Multiple EdgeLenses: When people are exploring and comparing different parts of a graph, it would be convenient if there were more than one EdgeLens available. With multiple EdgeLenses one can position a lens on a section of a graph and use another lens to examine other areas of the same graph.

Our algorithm smoothly integrates multiple EdgeLenses. Every curved edge has two control points, other than those that are located at the nodes, which are used to determine the way it curves. All three images in Figure 14 show an edge under the influence of two EdgeLenses, where the centre point for one of the lenses is shown as a red circle and the other as a blue circle. The control points for this edge, as created by each EdgeLens, are displayed as triangles, each coloured red and blue to match the centre point. The green triangles are the control points that are actually used to displace the edge. They are located by averaging the *x* and *y* locations of control points generated by the two EdgeLenses. As seen in Figure 14, the visual effect is that the edge balances its shape to reflect the interaction between the two lenses.

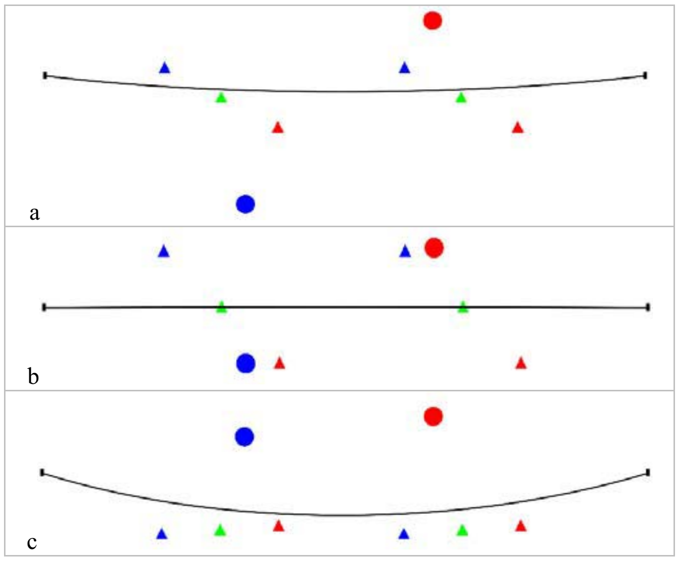

Figure 14. The effect of two EdgeLenses on a single edge: a) red EdgeLens is to the top of the edge and the blue EdgeLens is to the bottom – the curve effect is lessened; b) the location of the red and blue EdgeLenses are balanced and counteract the effect of each other; c) as the blue EdgeLens moves to the top joining the red EdgeLens on the same side the combined effect curves the edge.

In general, the algorithm can be applied to more than one EdgeLens. As described in Step 3 of EdgeLens algorithm, each lens determines the positions of two control points with one on each side of the lens. When there are *n* EdgeLenses affecting the edge, the average of *x*-coordinates of all control points on the same side of the lens is the resulting *x*-coordinate of the final control point on that side. Similarly the *y*-coordinate of the final control point is the average of all *y*-coordinates.

To show how this works on a complex graph, Figure 15 illustrates with a graph of a portion of the Department of Computing Science web site at the University of Calgary. Image A is a simple radial layout; image B shows a single EdgeLens revealing some of the graph structure; image C shows three EdgeLenses revealing detail in different areas.

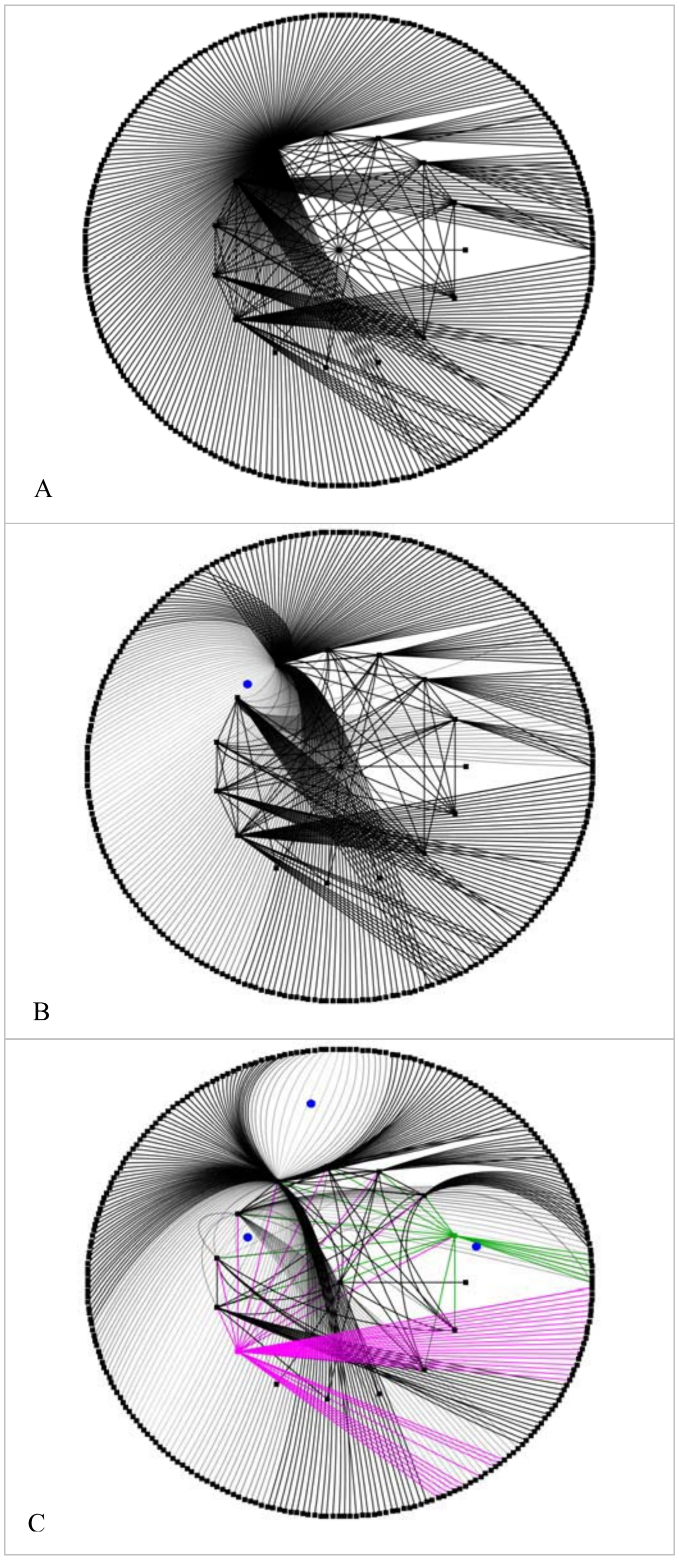

Figure 15. This graph represents a subset of the web pages of the Department of Computing Science at the University of Calgary. Image A, is a simple radial layout; image B, shows single EdgeLens revealing some of the graph structure; image C shows three EdgeLenses with selected edges excluded from the effect.

8 Conclusions

The primary contribution in this paper is to describe the development of the EdgeLens, an interactive solution that lets people explore graphs containing considerable edge congestion.

The EdgeLens works because it:

- maintains the nodes in original layout,
- interactively moves edges,
- helps to removes ambiguities,
- clarifies graph structure, and
- reveals hidden information underneath the graph structure.

We developed and offered two possible interaction candidates, Bubble and Spline, and we saw through a user study that the Spline-based approach was much preferred and significantly helped participants with their tasks.

Subsequently, we described the EdgeLens algorithm in detail. We also provided enhancements to its use: the ability to exclude and colour selected edges from the EdgeLens effect, the ability to change the transparency, shape and displacement of edges, and the ability for people to create multiple EdgeLenses in a single graph. The EdgeLens effect makes a powerful new tool for exploring information and relationships in information-dense graphs.

Acknowledgements

This research was supported in part by Intel Inc. and the National Science and Engineering Research Council of Canada. The authors would like to thank the members of Interactions Lab, University of Calgary for ongoing help and support.

References

BECKER, R., EICK, S., AND WILKS, A. 1995. Visualizing Network Data. In *IEEE Transactions on Visualization and Computer Graphics 1*, 1, 16-28.

BEDERSEN, B., AND HOLLAN, J. 1994. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *Proceedings of ACM UIST'94*, ACM Press, 17-26.

BIER, E., STONE, M., PIER, K., BUXTON, W., DEROSE, T. 1993. Toolglass and Magic Lenses: The See-Through Interface. In *Proceedings of ACM SIGGRAPH 93*. Computer Graphics Proceedings, Annual Conference Series, ACM, 73-80.

CARPENDALE, M.S.T., COWPERTHWAITE, D. J. AND FRACCHIA, F. D. 1995. Three-Dimensional Pliable Surfaces: For Effective Presentation of Visual Information. In *Proceedings of ACM UIST'95,* ACM Press. 217–226.

CARPENDALE, M.S.T., COWPERTHWAITE, D.J. AND FRACCHIA, F. D. 1997. Extending distortion viewing from 2D to 3D. In *IEEE Computer Graphics and Applications 17*, 4, 42–51.

CARPENDALE, M.S.T. AND MONTAGNESE, C. 2001. A framework for unifying presentation space. In *Proceedings of ACM UIST'01,* ACM Press. 61-70.

CARPENDALE, M.S.T., AND RONG, X. 2001. Examining Edge Congestion. In *Proceedings of ACM CHI'01.* 115-116.

CONSENS, M., CRUZ, I., MENDELZON, A. 1992. Visualizing queries and querying visualizations. *ACM SIGMOD Record 21*, 1, 39-46.

COX, K., EICK, S., AND HE, T. 1996. 3D Geographic Network Displays. *ACM SIGMOD Record 25*, 4, 50-54.

DI BATTISTA, G., EADES, P., TAMASSIA, R., AND TOLLIS, I.G. 1994. Algorithm for Drawing Graphs: an Annotated Bibliography. *Computational Geometry 4*, 235-282.

FURNAS, G. 1986. Generalized Fisheye Views. In *Proceedings of ACM CHI'86*, ACM Press, Human Factors in Computing Systems Proceedings, 16-23.

HERMAN, I., MELANCON, G., AND MARSHALL, M. 2000. Graph visualization and navigation in information visualization: A survey. In *IEEE Transactions on Visualization and Computer Graphics 6*, 1, 24-43.

KEAHEY, T. AND ROBERTSON, E. 1996. Techniques for nonlinear magnification transformations. In *Proceedings of the IEEE Conference on Information Visualization*, IEEE Computer Society Press, 38–45.

KEAHEY, T. AND ROBERTSON, E. 1997. Nonlinear magnification fields. In *Proceedings of IEEE Conference on Information Visualization,* IEEE Computer Society Press, 51-58.

LAMPING, J., RAO, R., PIROLLI, P. 1995. A focus and context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of ACM CHI'95*, ACM Press, 401–408.

LEUNG, Y. K. AND APPERLEY, M. D. 1994. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction 1*, 2, 126-160.

MUKHERJEA, S., FOLEY, J. 1995. Visualizing the world-wide web with the navigational view builder. *Computer Networks and ISDN Systems, Special Issue on Third International World Wide Web Conference 27*, 6, 1075-1087.

PURCHASE, H. 1997. Which aesthetic has the greatest effect on human understanding? In *Symposium on Graph Drawing 97, vol. 1353 of Lecture Notes in Computer Science.* Springer-Verlag, 248-261.

PURCHASE, H. 2000. Effective information visualization: a study of graph drawing aesthetics and algorithms. *Interacting with Computers 13*, 2, 477-506.

SARKAR, M., AND BROWN, M. H. 1992. Graphical Fisheye Views of Graphs. In *Proceedings of ACM CHI'92*, ACM Press, 83-91.

WARE, C. AND LEWIS, M. 1995. The DragMag Image Magnifier. *ACM CHI '95 Video Program.*

WILLS, G. 1999. NicheWorks – Interactive Visualization of Very Large Graphs, *Journal of Computational and Graphical Statistics 8*, 2, 190-212.