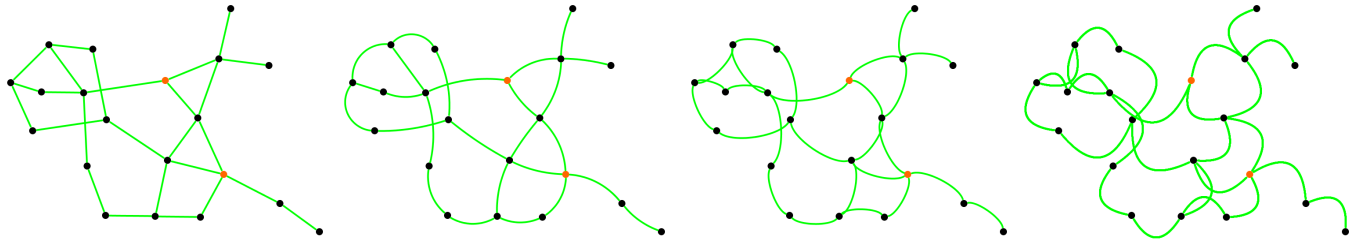
A User Study on Curved Edges in Graph Visualization

Kai Xu, Chris Rooney, Peter Passmore, Dong-Han Ham, and Phong H. Nguyen



(a) Straight edge (b) Lombardi layout (c) Slightly curved edge (d) Heavily curved edge

Fig. 1. Examples of edge types used in the cured edge study.

Abstract—Recently there has been increasing research interest in displaying graphs with curved edges to produce more readable visualizations. While there are several automatic techniques, little has been done to evaluate their effectiveness empirically. In this paper we present two experiments studying the impact of edge curvature on graph readability. The goal is to understand the advantages and disadvantages of using curved edges for common graph tasks compared to straight line segments, which are the conventional choice for showing edges in node-link diagrams. We included several edge variations: straight edges, edges with different curvature levels, and mixed straight and curved edges. During the experiments, participants were asked to complete network tasks including determination of connectivity, shortest path, node degree, and common neighbors. We also asked the participants to provide subjective ratings of the aesthetics of different edge types. The results show significant performance differences between the straight and curved edges and clear distinctions between variations of curved edges.

Index Terms—Graph, Visualization, Curved edges, Evaluation.

1 Introduction

It seems that straight lines seldom occur in natural objects and that humans actually prefer curved lines [3]. Thus it may not seem surprising that in aesthetics, curved lines are often to be preferred over straight ones, as found for example in Hogarth's serpentine Line of Beauty [16]. Other examples include the works by American artist Mark Lombardi [15], who is famous for portraying the social networks behind financial and political scandals with curved edges. Manually produced network diagrams, such as metabolic pathways and metro network maps, often contain curved edges as well. A increasing number of automatic techniques for producing network visualizations with curved edges are being developed. One example is the visualization of professional networks on LinkedIn[1], in which curved edges are used to depict the relationships among the users. Started by the work of Holten [17] and Dickerson et al. [7], there are an increasing number of techniques that utilize the flexibility of curved edges, i.e., the possibility to change curvature direction and level, to reduce visual complexity by re-routing edges as curved lines and "grouping" or "bundling" neighboring edges together. Such methods are commonly known as either *edge bundling* or *confluent drawing* approaches respectively. Additionally, there are methods that aim to automatically produce graph visualizations with a style similar to that of the works by Mark Lombardi, with the Lombardi Spring Embedder algorithm [6] being such an example.

Many examples are available to demonstrate the results of graph visualization with curved edges. However, there have been limited number of studies to empirically evaluate their effectiveness on common graph-related tasks. The study by Bar and Neta [3] mainly tested the psychological reaction to straight and curved lines, not the performance of graph-related tasks. Another study [30] compared hierarchical edge bundling against node-link diagrams with five software developers. However, the data and tasks were software engineering-specific, and can not be easily generalized.

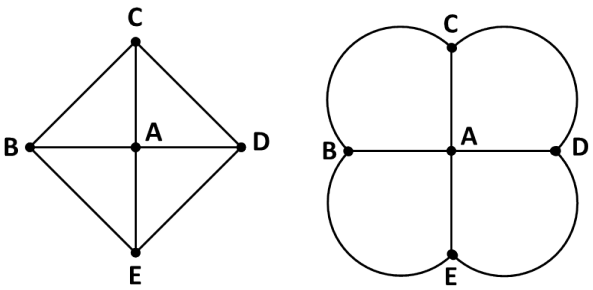
In this paper we present the results from two experiments studying the impact of edge curvature on general graph readability. Fig. 1 shows the different types of edges used in the experiments. The first experiment compared graphs with *constant* edge curvatures level, i.e., all the edges in a graph have the same level of curvature. The second experiment examined the case that edges have *varying* curvature within a graph, such as those generated by the Lombardi force-directed layout [6] (Fig. 1(b)). The experiment results showed significant differences between the task performance using straight and curved edges and also among different types of curved edges.

The remainder of the paper is organized as follows. Section 2 describes previous work on curved-edge visualization and graph visualization evaluation. Section 3 gives an overview of the experimental design. The details of the two experiments, including the setup and procedures, and results, are described in Section 4 and 5 respectively. Finally, Section 6 concludes the paper with the discussion of possible future work.

- *Kai Xu is with Middlesex University, E-mail: k.xu@mdx.ac.uk*
- *Chris Rooney is with Middlesex University, E-mail: c.rooney@mdx.ac.uk*
- *Peter Passmore is with Middlesex University, E-mail: p.passmore@mdx.ac.uk*
- *Dong-Han Ham is with Chonnam National University, donghan.ham@gmail.com*
- *Phong H. Nguyen is with Middlesex University, E-mail: p.nguyen@mdx.ac.uk*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

[1] http://inmaps.linkedinlabs.com



(a) Node *A* has perfect angular resolution, but not the other nodes. (b) A Lombardi drawing with all the nodes have perfect angular resolution

Fig. 2. Lombardi drawing example.

2 Related Work

2.1 Graph Visualization with Curved Edges

Curved edges have been long used to illustrate self loops and multiple edges between a pair of nodes [5], which are not possible with straight line segments. Examples include the "Arc Diagram" [33] in which all the nodes are placed on a horizontal line and half-circled lines are used to draw the connections among them. The PivotGraph [34] is another example in which curved edges are used to draw the aggregated relationship between nodes. The Gephi [4] network analysis software also uses curved edges to show multiple directed edges between nodes. Algorithms have been proposed to produce curved edges to avoid node-edge overlapping such as those used in the GraphViz package [13]. In this case, curved edges are supplementary to straight ones, which account for the majority of the edges.

Curved edges allow more display space compared to their straight counterparts and potentially reduce visual clutter. The same number of edges can occupy more space when drawn as curves than straight lines. This can alleviate edge cluttering problems, which is common in real-world graphs such as "small-world" networks [35] that have small graph diameter and densely clustered regions. One of the early works that exploited this property was the EdgeLens [36]. It is an interaction technique that dynamically re-draws the edges close to a point of interest from straight to curved to reduce edge clutter.

Curved edges also appear in many visualization techniques including drawing edges on Treemaps [11], linking semantic substrates in a visualization [29], and label edges [37]. In the last method, edges are replaced with long curved edge labels. Curved edges are the key feature of edge bundling and confluence drawing approaches, which "group" or "bundle" neighboring edges together to reduce visual complexity. Such approaches have gained considerable research attention recently, with a number of following techniques based on the same idea. Latest work, such as [10, 12, 27], provides a comprehensive list of such methods. A recent paper by Riche et al. [28] includes a detailed discussion on the design space of curved edges in node-link diagrams.

There are also efforts aiming to automatically generate graph visualizations with a style similar to the art work by Mark Lombardi. Duncan et al. [8] defined the *Lombardi drawing of graphs* that uses curved edges to generate graph visualization with *perfect angular resolution*. A node has *perfect angular resolution* if the angles between neighboring incident edges are all the same. For instance, Node *A* in Fig. 2(a) has perfect angular resolution, but other nodes do not. A graph visualization has *perfect angular resolution* if all the nodes have perfect angular resolution. Fig. 2(b) shows the Lombardi drawing of the network in Fig. 2(a) where curved edges are used to achieve perfect angular resolution. For Node *B*, *C*, *D*, and *E*, the incident edge angle is formed by the tangents to the edges at the node. Given that not all networks have a Lombardi drawing [8], Chernobelskiy et al. proposed a force-directed algorithm [6] that uses circular arcs to make the network layout as close to a Lombardi drawing as possible.

2.2 Graph Visualization Evaluations

There has been a rich literature on user studies of graph visualization techniques. These range from the early work by Purchase et al. [26] on the effectiveness of graph drawing aesthetics to the more recent work on 3D graphs [32], visual representation of directed edges [18, 19], and readability of dynamic graphs [2]. Please refer to a recent survey [14] for more details. In most of these studies, users were asked to perform graph-related task(s) on drawings produced by different visualization techniques. Their task completion time and accuracy are recorded and used to compare the relative effectiveness among methods. We adopted a similar approach in our study.

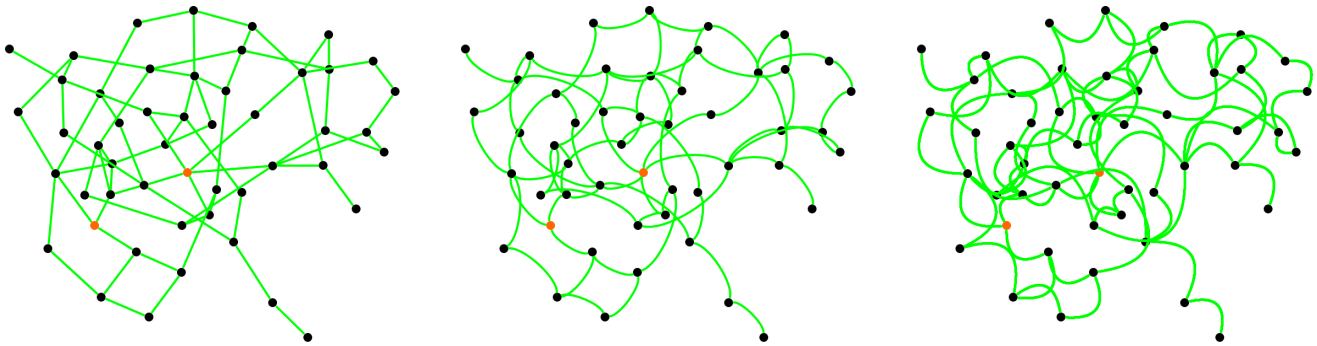
Despite its popularity, there are relatively few evaluations available on graph visualization techniques using curved edges. In the study by Bar and Neta [3] it was hypothesized that sharp transitions in contours might convey a sense of threat and thus lead to a negative bias. In their task subjects were briefly presented with real-life object and artificial pattern stimuli of various curvature and asked to make a forced choice judgment as to whether they liked the stimulus or not. Their results showed that curved lines (i.e., less angular) were the preferred choice. The work by Telea et al. [30] was a qualitative study comparing hierarchical edge bundling against node-link diagrams with software engineering-related data. Five software developers were asked to rate their preference among the two types of visual representations based on their experience of using them during the experiment. The results showed that all participants strongly preferred hierarchical edge bundling to node-link diagrams.

Studies that are not on curved edges but related to our experiments include the work by Purchase [25], where aesthetic criteria of graph visualization are ranked according their impact on graph readability. There were five aesthetics criteria included in the study: edge bends, edge crossings, angular resolution, edge orthogonality, and symmetry. The results showed that edge crossings have the most significant impact, edge bends and graph symmetry have some impact, whereas improving orthogonality or angular resolution had little effect on graph readability. Ware and Bobrow [31] demonstrated that a key component to the readability of shortest paths (one of the tasks used in our experiments) was to produce a continuous chain of links. In a similar study using an eye tracking device, Huang and Eades [20] found that for path searching tasks the edges incident to nodes concerned, edges going toward to the target node, and the edges alongside the paths affect drawing readability and trigger extra eye movements.

3 Experiment Design

The study consisted of two experiments to examine the impact of edge curvature on graph readability. The first experiment studied the difference among graph edges with different levels of curvature. Three curvature levels are compared: straight (i.e., zero curvature), slightly curved, and heavily curved. The second experiment included an additional curve edge type produced by the Lombardi layout algorithm. In such a case, curved edges were only used if they help improve angular resolution. Also, edge curvature is chosen to maximize the angular resolution of the resulting graph visualization, so edge curvature varies among edges. This results in a graph visualization with a mixture of straight and curved edges with changing curvature levels. Breaking the study into two experiments allowed us to include more tasks without making it too long, and the results from the first experiments were used to inform the design of the second experiment. This approach is similar to that of the previous studies [18, 19], which successfully examined a relatively large number of conditions.

Our study aimed to examine the general impact of edge curvature on graph readability. This requires a wide range of graph-related tasks in the experiment to ensure the comparison is comprehensive. There are a large number of possible tasks according to existing taxonomies [1, 22]. In practice, some tasks are more clearly defined and thus widely used than the others. For instance, topology-related tasks, such as finding the shortest path length between two nodes, are very well defined and included in many graph visualization-related evaluations. However, tasks such as identifying clusters and overall graph structure are relatively less clearly defined and used more rarely in



(a) Straight edge (STL) (b) Slightly curved edge (SCL) (c) Heavily curved edge (HCL)

Fig. 3. Examples of graph visualizations used in the first experiment.

studies. Finally, the practical concerns of not making an experiment overly long limit the number of tasks that can be included. For our study, we chose a simple topology task in the first experiment that allowed us to identify the curve edge type that makes performance significantly worse than other edge types. In the second experiment, we used four different tasks to make the comparison more comprehensive. We did not include any overview-type tasks, such as describing the general structure of a graph, because we felt further research is required before such tasks can be readily used in evaluation studies. As a result, we did not include any edge bundling or confluent drawing techniques, because they are known to excel in revealing overall network structure but not particularly suitable for topology-based tasks [17], which are the ones used in our study.

To understand the generic impact of edge curvature, we chose undirected abstract graphs instead of examples from a specific domain. The graphs used in the experiment were generated with the model proposed by Ware et al. [31]. This model results in a connected graph (no disconnected nodes) with reasonable edge density (Fig. 3). It is shown to produce more realistic graphs (closer to real-world graphs) than the Erdős-Renyi random graph model [23]. In such a model, each node has a chance p $(0 < p < 1)$ to connect to one other randomly selected node and $1 - p$ chance to connect to two other randomly selected nodes. In our study, p is set to 0.5 to allow for a medium edge density.

4 Experiment 1 - Constant Edge Curvature

4.1 Graph and Visualization Generation

The aim of the first experiment is to understand the impact of constant edge curvature (i.e., all the edges within a graph share the same curvature level) on graph readability. We chose three curvature levels: straight edge (zero curvature), slightly curved edge, and heavily curved edge (Fig. 3). These are referred as STL, SCL, and HCL respectively. A four-point Bézier curve was plotted for SCL, and three point Bézier curve was plotted for HCL. The SCL is plotted the same way as curved edges in Gephi [4]: two intermediate control points were generated to set the curvature on the line (Fig. 4). Firstly a displacement factor d_2 was calculated which was 20% of the Euclidean distance between the two end points (d_1). Then starting at either end the intermediate control point was positioned by traveling down the straight line between the endpoints by a distance of d_2 to create a point p, and then traveling out from p along the tangent to the straight line by a distance of d_2 to create the intermediate control point. For the heavy curve, a single intermediate control point was generated such that an equilateral triangle was formed between it and the two end points (Fig. 4). The Bézier curve formula was then applied to both to generate SCL and HCL.

The graphs were generated using the model discussed in the previous sections. There are three graph sizes: 20, 50, and 100 nodes. Ten graphs were generated for each size, which results in 30 different graphs. Three curvature variations (STL, SCL, and HCL) are generated for each graph, which resulted in 90 graph visualizations in total. We used the spring-embedder algorithm [9] provided by the Gephi software package [4], and the same layout is used for all three edge variations to avoid any confounding factor introduced by the graph layout. In applications such as Gephi, the edge curve is used to indicate edge direction, i.e., the edge curve always goes counter-clock wise from the starting point to the end point. However, we used undirected graphs so there is no direction associated with each edge.

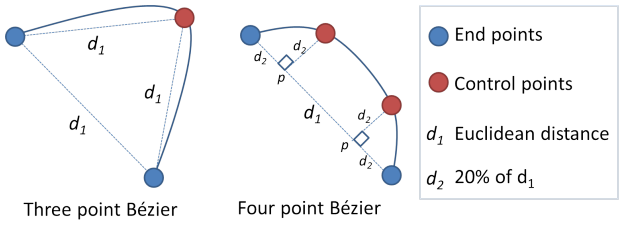
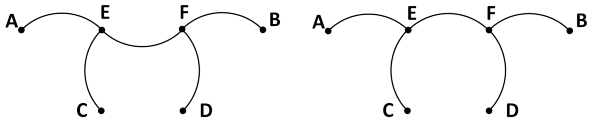


Fig. 4. Generation of the SCL and HCL edge.



(a) Path $AEFB$ is smooth but path $CEFD$ is not. (b) Path $CEFD$ is smooth but path $AEFB$ is not.

Fig. 5. "Smooth" path optimization conflicting example.

There are two options to draw a curved edge between a pair of nodes, either above or below the straight line that connects the two end nodes. Two possible optimizations were considered when choosing between these two options for each edge: improve path continuity and reduce edge crossings, which both have been shown to help improve graph readability. It is possible to arrange the curved edges to make the path between a pair of node A and B "smooth". For instance, path $AEFB$ in Fig. 5(a) is "smoother" than that in Fig. 5(b). However, it is not possible to make all the path as smooth as possible. For example, in Fig. 5 it is not possible to have both path $AEFB$ and $CEFD$ smooth. Besides, there are a very large number of possible paths ($O(|N|^2)$, where $|N|$ is the number of nodes), and it will be computationally expensive to optimize for all paths. The other possible optimization is to reduce edge crossing number. Changing the bend direction of a curved edge usually reduces the edge crossings in the part of graph the edge was in but increases the edge crossings in the other part of graph. To find a global optimal it requires to compare to all cases, which is exponential to the number of edges ($O(2^{|E|})$, where

$|E|$ is the number of edges). Such optimization is usually not practical without more efficient heuristics. Both optimizations are non-trivial research questions, and we are not aware of any existing work that can be applied to them. As a result, we decided not to use any optimization and assign the curve edge side randomly. While this is not an optimal solution, it does represent the state-of-the-art knowledge on curve edge crossing reduction or path smoothness optimization.

4.2 Design, Procedure, and Participants

We used a repeated-measures design with the edge curvature and graph size as the within-subjects independent variables. During the experiment, participants were asked to identify whether a path of length two existed between two nodes (i.e., path-finding task). The two nodes used for the path finding task were randomly selected. As previously discussed, there are 90 graph visualizations in total, 10 for each of the nine curvature-size combinations (one *block*). The nine blocks were randomly ordered and seeded differently for each participant to remove any effect of learning.

A pilot test was conducted to examine the appropriateness of the experimental system and the levels of the experimental factors, which included the user interface features of the application software used in the experiment, the three edge types, and the three graph sizes. The pilot test was conducted with 12 participants and a within-subject design approach. In the pilot test, the two main experimental factors were the three edge types and the three graph sizes. The results of the pilot test showed that the effects of the two main factors were statistically significant at the 0.01 significance level in the two measures, which indicated that the three levels of the two experimental factors could be appropriate for the purposes of the main experiment.

Before the experiment started, participants were asked for personal information, given a brief description of the task, and given three practice trials. During practice, feedback was provided as to whether the answer was correct. If not, the correct answer was shown to help the participant identify where they went wrong. No such feedback was provided during the experiment, which started after the practice. Participants were presented with the nine blocks, and given a short break after each block. After the experiment, participants were given a short questionnaire to collect subjective feedback. They were asked which edge type they preferred aesthetically and which they felt was most effective for the tasks. Results were recorded on a five-point Likert scale. Finally, participants were asked to provide any general comments.

A custom Java application was built to display the graphs and record user inputs. Majority of the participants used a 23 inch monitor with 1920x1080 resolution. Participants always sat straight in front of the display with normal viewing distance. When displayed the graphs were scaled so that each node occupied approximately the same amount of screen real estate regardless of the number of nodes in the graph. As a result, a graph with 50 nodes occupied about 50% display area of that of a graph with 100 nodes. For each graph, the two reference nodes presented to the participant were colored orange while the remainder were colored black, and edges were colored green to make them distinct from nodes (Fig. 3). The different coloring of nodes and edges was necessary to avoid the confusion of node-edge overlap with node-edge connection: edges were connected to the outer edge of a node (rather than the center) and were plotted on top of the nodes if the two overlapped (e.g., Fig. 3). It would be difficult to differentiate these two conditions if nodes and edges shared the same color. Each trial started by displaying only the two orange reference nodes so they could be clearly identified; two seconds later the remainder of the graph was displayed. Participants were instructed to press "y" on the keyboard if a path of length two could be identified and "n" otherwise. A progress bar was shown on the top indicating the percentage of the trials that had been completed. Fig. 6 shows the experiment user interface.

A total of twenty-eight subjects volunteered to participate in the study and all had normal or corrected-to-normal vision. Among the participants, there were 22 males and 6 females. Their ages ranged from 18 to 53 with an average of 29. The participants self-reported their previous experience with graph visualization: 8 rated 'none or very little', 17 rated 'medium', and the remaining 3 rated 'extensive'. They were from diverse social-economical background and included university students and staff (both academic and non-academic).

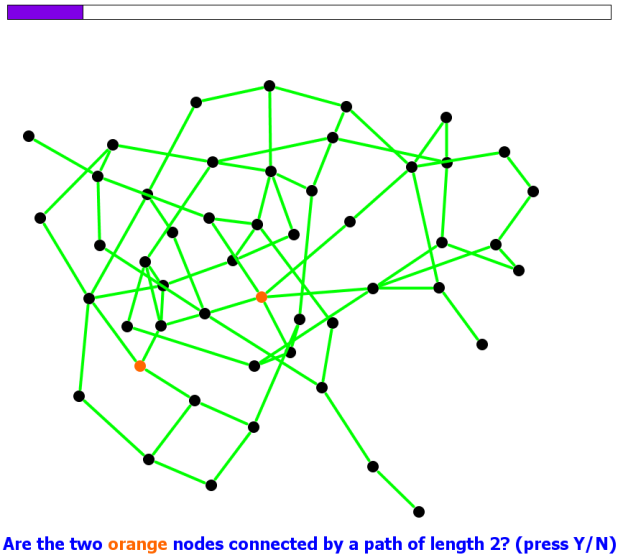


Fig. 6. The experiment user interface.

4.3 Hypotheses

H1: We predicted that steeper arcs would be detrimental to performance. As the curvature level increases, so does edge length (the node positions are fixed). This may also lead to additional edge crossings and/or less smooth paths.

H2: As the number of nodes increases, a graph will appear to be more cluttered. Therefore, we predicted performance would decrease with the graph size.

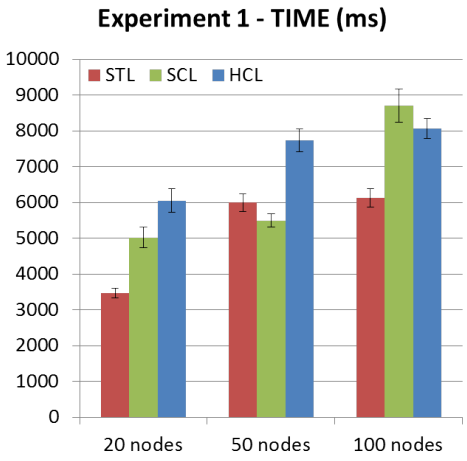
H3: We expected the participants would prefer straight edge for effectiveness, but slightly curved edges for aesthetics.

4.4 Results

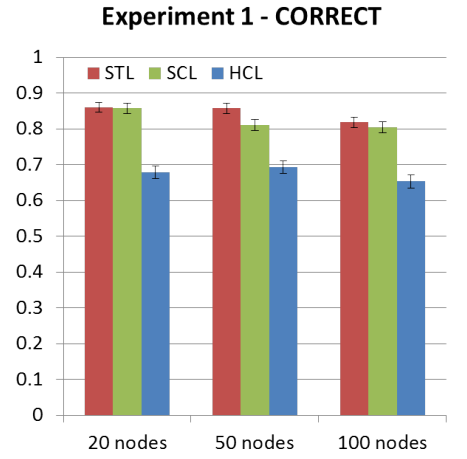
This study used two objective measures: time to answer (TIME, in milliseconds) and the number of correct answers (CORRECT) and two subjective measures: user preference on effectiveness (PREF-EFFECTIVE) and look (PREF-LOOK) of edge type. For the two subjective measures, this study used a 5-point Likert-type scale where 1 = strongly disagree and 5 = strongly agree.

Fig. 7 shows the mean with the 95% confidence interval of each experimental condition in all the measures. A generalized linear model (GLM) ANOVA was used for the analysis of TIME and CORRECT. Subjects were manipulated as a random factor, whereas the two within-subjects variables were considered as fixed factors. The Tukey's test was used to conduct a pairwise comparison between two levels of TIME and CORRECT. As the data of the two subjective measures were ranked order, the Friedman test, which is a nonparametric test for k correlated samples, was used. Before conducting a GLM ANOVA for TIME and CORRECT, the basic assumptions of the ANOVA were checked by the examination of residuals. It was found necessary to apply variance-stabilization techniques to the TIME data to rectify violations of two assumptions: normality and equality of variance, and a logarithmic transformation was applied. Residual plots indicated that the underlying assumptions of the ANOVA were not significantly violated in the case of CORRECT.

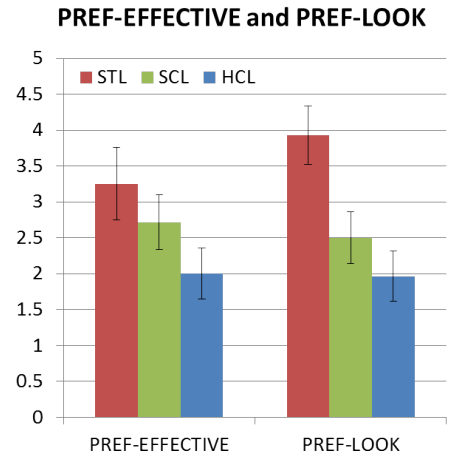
TIME: The ANOVA results showed that all the main effects were statistically significant at the 0.05 significance level: edge type ($F(2,54) = 16.31$, $p < .01$) and number of nodes ($F(2,54) = 26.64$, $p < .01$). The interaction effect between two within-subject factors did not show a statistical significance ($F(4,108) = 1.89$, $p = .117$).



(a) TIME (millisecond)

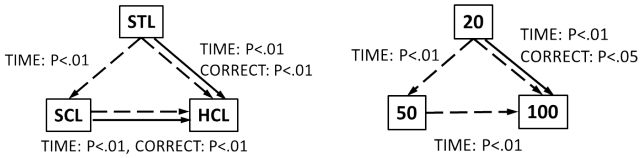


(b) CORRECT (percentage)



(c) PREF-EFFECTIVE and PREF-LOOK (Subjective Measurements)

Fig. 7. Results of the Experiment on Constant Edge Curvature.



(a) Edge type (b) Graph size

Fig. 8. Summary of the significant difference of TIME and CORRECT. There is one directed edge for each statistically significant difference, and the origin performs significantly better (faster/more accurate) than the destination. A dashed line means TIME is better and a solid line means CORRECT is better.

Tukey's test results showed that STL is significantly faster than SCL ($p < .01$) and HCL ($p < .01$), and SCL is also significantly faster than HCL ($p < .01$). The average TIME of STL, SCL, and HCL are 5198.1ms, 6405.1ms, and 7280.9ms respectively. In terms of graph size, 20-node graph is significantly faster than 50-node graph ($p < .01$) and 100-node graph ($p < .01$), and 50-node graph is also significantly faster than 100-node graph ($p < .01$). The average TIME of 20, 50, and 100 nodes are 4840.1ms, 6406.3ms, and 7629.7ms respectively. Fig. 8 summarizes the significant differences.

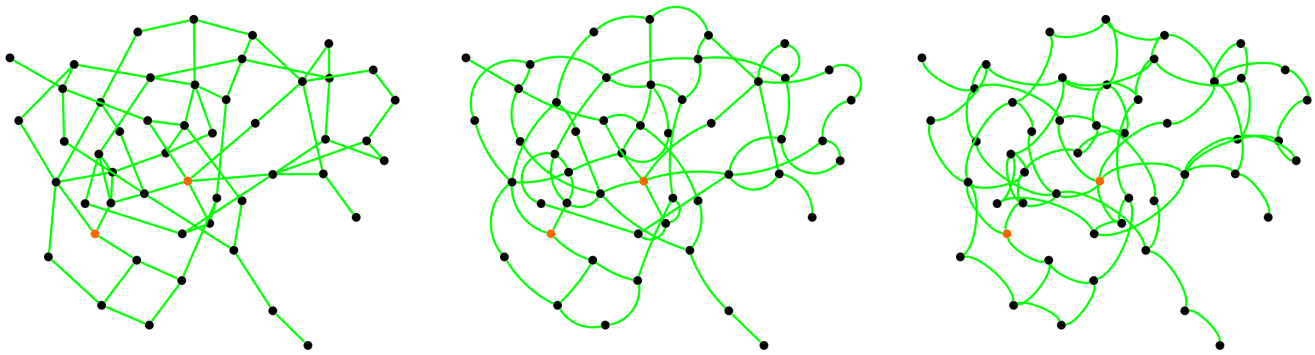
CORRECT: The main effect of edge type was statistically significant at the 0.01 significance level ($F(2,216) = 61.20$, $p < .01$). The main effect of number of nodes was also significant at the 0.05 level ($F(2,216) = 3.05$, $p < .05$). The interaction effect of the two factors was not significant ($F(4,216) = 0.60$, $p = .660$). Tukey's test revealed that STL had a significant difference from HCL at the 0.01 significance level but no difference from SCL. SCL showed a significant difference from HCL at the .01 level. The average CORRECT of STL, SCL, and HCL are 0.845, 0.824, and 0.675 respectively. In the case of number of nodes, there was only a significant difference between 20 and 100 at the .05 significance level. The average CORRECT of 20, 50, and 100 node are 0.799, 0.787, and 0.758 respectively. The results are summarized in Fig. 8.

PREF-EFFECTIVE and PREF-LOOK: For the two subjective measures, the Friedman test showed that there was a statistically significant difference among three edge types (PREF-EFFECTIVE ($\chi^2_F = 16.83$, $p < .01$; adjusted for ties) and PREF-LOOK ($\chi^2_F = 16.83$, $p < .01$; adjusted for ties)). Fig. 7(c) shows that STL is the most preferable edge type in both subjective measures.

4.5 Discussion

The results show that STL has the best performance, in terms of both TIME and CORRECT. This agrees with our hypothesis H1. The results also show that performance decreases with the curvature level: as the curvature increases from zero (STL) to medium (SCL) and then high (HCL), both TIME and CORRECT drop. Again this is in agreement with our hypothesis H1. While this is hardly surprising, we investigated further to see whether edge crossing number is the deciding factor as found in a previous study [25]. We computed the edge crossing number for all the graphs used in the experiment. Fig. 9 shows the average number of edge crossings (with 95% confidence level) for each edge type in the three different graph sizes used. Tukey's test shows that there is no significant difference between the edge crossing number of SCL and HCL while both are significantly larger than that of the STL. This indicates that edge crossing number is unlikely to be the deciding factor for task performance in the experiment, because SCL is significantly better than HCL for both TIME and CORRECT but there is no significant difference in the edge crossing number of the two. This implies that *path distance* (i.e., the Euclidean path length) and "smoothness" (i.e., the level of direction change at each intermediate node along a path) are more important factors in task performance.

The results also show that the time taken increases with the number of nodes for each curvature condition, which agrees with our hypothesis H2. However, this is to a much less extent than the impact of curvature level as there is only significant difference between graphs with



(a) Straight edge (STL) (b) Lombardi layout (LBD) (c) Slightly curved edge (SCL)

Fig. 10. Examples of graph visualizations used in the second experiment.

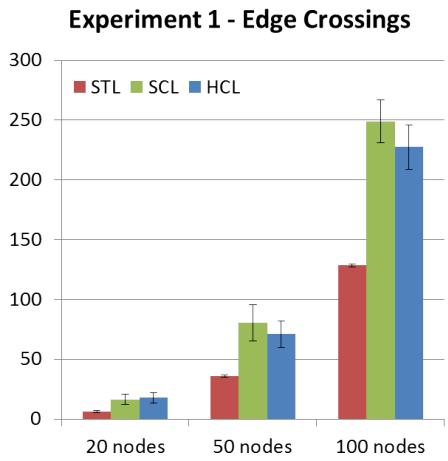


Fig. 9. The average number of edge crossings in the first experiment with 95% confidence level.

20 and 100 nodes with $p < .05$. One possible explanation is that path finding is a local task, i.e., it does not require checking the entire graph. In this experiment, "visual density" was kept relatively stable across different graph sizes and as a result the path finding performance may be less affected by the graph size increase.

Participant preference partly agrees with our hypothesis H3. Participants judged STL to be more effective for the path finding task, which agrees with the hypothesis. However, they ranked STL over SCL and HCL for aesthetics, which disagrees with our hypothesis H3. This is surprising because it is different from the results of previous studies [3, 30]. The latter could be the result of having the preference ranking after completing the tasks; the advantage of using STL for the task could have an impact on the aesthetics choice.

5 Experiment 2 - Varying Edge Curvature

5.1 Graph and Visualization Generation

Based on the results of the first experiment, it is clear that increasing edge curvature only is detrimental to performance. As a result, HCL was excluded from the second experiment. What was missing from the first experiment was a method that was specifically designed for graphs with curved edges. The force-directed layout is designed for graphs with straight edges and as discussed there is no readily available solution to reduce edge crossing or improve path smoothness when curved edges are used. As the study was being conducted, a new layout [6] that was designed for the curved edge graphs became available. As discussed earlier, the new layout is mainly designed to improve angular resolution, however, it also improves the 'smoothness' of a graph path. For instance, the path *ABCA* is "smoother" in Fig. 2(b) than that in Fig. 2(a) as a direct result of the improved angular resolution at node *B* and *C*. Besides, straight edges are used whenever angular resolution is already optimal. This reduces edge length (a straight edge is alway shorter than a curved one when end node positions are fixed) and potentially also reduce edge crossings. Therefore, there are three edge types in this experiment: STL, SCL, and Lombardi (LBD).

The graphs are generated using the same model as before. Given the relatively less significant effect of graph size in the first experiment, we replaced the graph size of 20 nodes with 200 nodes. This allowed us to examine the readability of relatively large graphs. Therefore, the graph sizes in this experiment are 50, 100, and 200. Eight graphs were generated for each size, which resulted in 24 different graphs in total.

A force-directed Lombardi algorithm is used to produce layout for each graph. More specifically, the variation that calculates lateral and rotational forces based on the two tangents defining a circular arc between two nodes (the first approach in the paper [6] was used). Different edge curvature variations were applied once the node positions were fixed by the layout: LBD uses the edges produced by the algorithm (all edges are circular arcs). STL and SCL replaced the edges with straight line segment and a Bézier curve (the same as the SCL condition in the first experiment) respectively. As a result, the three variations of one graph share the same node position but have different edge types. Fig. 10 shows the three variations of one graph with 50 nodes.

5.2 Design, Procedure, and Participants

To make the comparison more comprehensive, three new tasks were added. The four tasks used in this experiment are:

- Path of length two (PATH): to decide if there is a path of length two connecting the two randomly selected nodes (same as the first experiment).
- Shortest path length (LENG): to find the length of the shortest path between two randomly selected nodes.
- Connection number (CONN): to find the number of edges connected to a randomly selected node.
- Common neighbors (NEIG): to find the number of nodes connected to two selected nodes.

The experiment employed a mixed design with one between-subject factor (edge type) and two within-subjects factors (task type and graph nodes). Each participant experienced one edge type, but all tasks and graph sizes. Such a type of experimental design is also called a split-plot design [21]. This design allowed the inclusion of new tasks and avoided the experiment being overly long. The edge type a participant experiences was selected randomly at the beginning of the experiment. As a result, the number of participants for each type was not exactly

the same (STL 21, LBD 23, and SCL 21). Each participant performed four types of tasks on three different graph sizes, with eight graphs for each size. Therefore, each participant performed 96 trials in total.

The procedure is similar to that of the first experiment. Before the experiment started, participants were asked for personal information, followed by a choice of edge type for aesthetic preference. The aesthetics ranking was moved to the beginning of the experiment to remove any potential impact of the edge type utility for completing the tasks. After that, participants were given a brief description of the task, followed by two practice trials. Feedback was given during the practice but not during the experiment. The experiment was broken into 12 blocks, with each block containing 8 trials for one graph size/task combination. A short break was given after each block.

A Java application similar to the one used in the first experiment was built to display the graphs and record user inputs. The only difference is that participants were instructed to press a number between "0" and "9" for tasks requiring a number for answer. The viewing setup is also similar to that of the first experiment.

A total of 65 subjects voluntarily participated in the study, all had normal or corrected-to-normal vision. To avoid any learning effect, we excluded anyone who had participated in the first experiment. Among the participants, there were 45 male and 20 female. Their age ranged from 18 to 52 with an average of 30.3. The participants self-reported their previous experience with graph visualization: 17 rated 'none or very little', 42 rated 'medium', and the remaining 6 rated 'extensive'. They were from diverse social-economical background including university students and staff (both academic and non-academic).

5.3 Hypotheses

H1: We predicted that LBD would improve performance compared to SCL. As discussed earlier, LBD should improve path "smoothness", edge length, and potentially number of edge crossings as the result of angular resolution optimization.

H2: We predicted that performance using LBD will be similar to that of STL. As the results of the first experiment indicated, path distance and smoothness play an important role in deciding the performance. While LBD is likely to have longer path distance, its paths are smoother.

H3: Similar to the first experiment, we predicted performance would decrease as the graph size grows, with a potentially more significant effect on graphs with 200 nodes.

H4: We expected the participants to prefer LBD for aesthetics. As discussed earlier, we suspect having the preference rating at the end in the first experiment had negative impact on curved edge types. LBD may also be more appealing than SCL with its better angular resolution, path smoothness, and number of edge crossings.

5.4 Results

Two objective measures were used as in the first experiment: time to answer (TIME, in milliseconds) and the number of correct answers (CORRECT). TIME is defined as the time taken to solve a problem in an *experimental condition* (one level of graph type $\times$ one level of task type $\times$ one level of the number of nodes). CORRECT is defined as the number of correct answers in the eight repetitions of the same experimental condition. For example, a "7" is recorded if seven out eight questions of the same experimental condition were answered correctly. Fig. 11 shows the mean of TIME and CORRECT with the 95% confidence interval. Both TIME and CORRECT data are further broken down according to graph size and task type.

Before conducting a generalized linear model (GLM) ANOVA for TIME and CORRECT, the basic assumptions of the ANOVA were checked by the examination of residuals. Three assumptions, which are normality, independence, and equality of variance, are investigated by using normal probability plot of residuals, plot of residuals in time sequence, and plot of residuals versus fitted values, respectively.

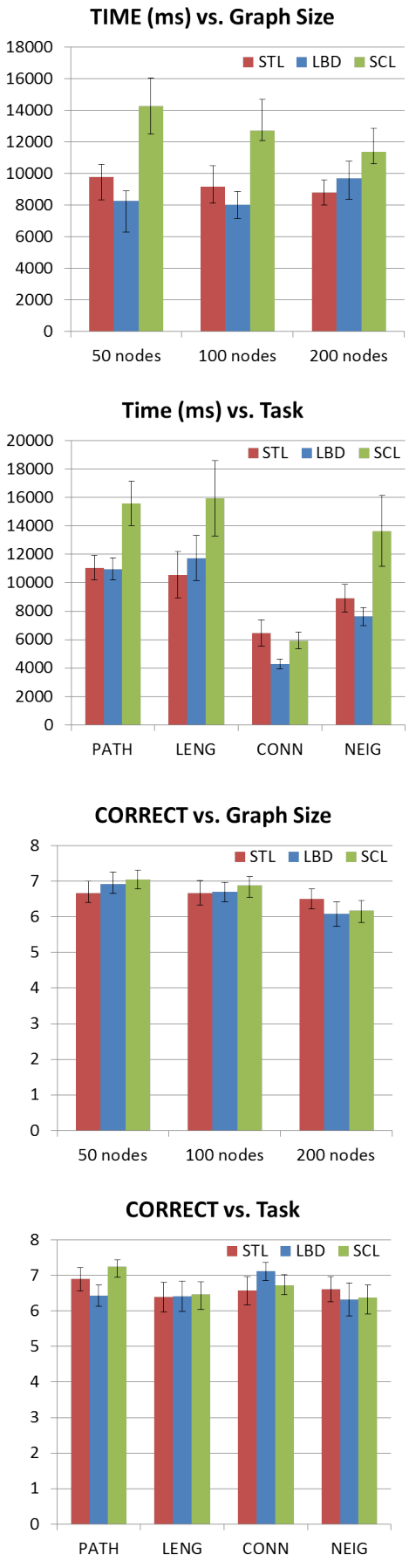
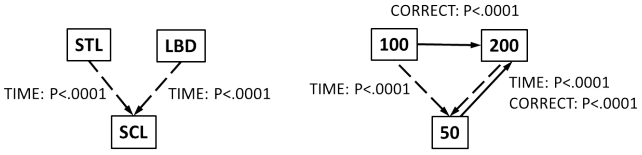


Fig. 11. Results of the Experiment on Varying Edge Curvature.



(a) Edge type (b) Graph size

Fig. 12. Summary of the significant difference of TIME and CORRECT. There is one directed edge for each statistically significant difference, and the origin performs significantly better (faster/more accurate) than the destination. A dashed line means TIME is better and a solid line means CORRECT is better.

Through the investigation, it was found necessary to apply variance-stabilization techniques to the time data to rectify violations of two assumptions: normality and equality of variance. According to the guidance and procedures described by Montgomery [24], a logarithmic transformation was applied to the time data. Residual plots indicated that the underlying assumptions of the ANOVA were not significantly violated in the case of CORRECT. The Tukey's test was used to conduct a pairwise comparison between two graph types in terms of TIME and CORRECT.

TIME: As stated above, logarithmic transformation of TIME data was used for the ANOVA test. The ANOVA results showed that all the main effects were statistically significant: edge type: $F(2,62) = 3.23, p < .05$; task type: $F(3,186) = 78.34, p < .01$, and graph size: $F(2,124) = 10.72, p < .01$. Tukey's test results showed that both STL and LBD are significantly faster than SCL ($p < .0001$ in both cases), but there is no significant difference between them. The average TIME of STL, LBD, and SCL are 9248.1ms, 8655.3ms, and 12778.9ms respectively. For graph size, Tukey's test results revealed that both 100 nodes and 200 nodes are significantly faster than 50 nodes ($p < .0001$ in both cases), but there is no significant difference between them. The average TIME of 50, 100, and 200 node are 10768.1ms, 9960.1ms, and 9954.0ms respectively. Fig. 12 summarizes the pairwise comparison results from the Tukey's test.

CORRECT: The main effect of graph type was not statistically significant ($F(2,62) = 0.14, p = .866$). However, the two other main effects were statistically significant at the .05 significance level: task type ($F(3,186) = 3.82, p < .01$) and the number of nodes ($F(2,124) = 18.67, p < .01$). Tukey's test results revealed that both 50 nodes and 100 nodes are significantly more accurate than 200 nodes ($p < .0001$ in both cases), but there is no significant difference between them. These results are also shown in Fig. 12. The average CORRECT of 50, 100, and 200 node are 6.88, 6.75, and 6.26 (correct answers out of eight questions) respectively.

ANOVA tests were conducted to examine whether there is any significant difference in TIME and CORRECT for each task. Results showed that only the CORRECT of the PATH task ($p < .05$) and the TIME of the NEIG task ($p < .05$) varied significantly among the edge types. Tukey's test results showed that for the PATH task, SCL is significantly more accurate than STL ($p < .05$) and LBD ($p < .0001$), and STL is significantly more accurate than LBD ($p < .01$). For this task the average CORRECT of SCL, STL, and LBD are 7.24, 6.89, and 6.43 respectively. For the NEIG task, there is no significant difference in TIME between LBD and STL, but both are significantly faster than SCL ($p < .0001$ in both cases). For this task the average TIME of LBD, STL, and SCL are 7629.8ms, 8912.4ms, and 13648.7ms respectively.

Figure 13 shows the aesthetics preference data and the y-axis is the number of participants that selected that edge type as the preferred one. STL is the clear leader selected by 50 participants (out of 65).

5.5 Discussion

The results regarding SCL and STL are consistent with the conclusions from the first experiment. Tasks performed on STL took less time than those on SCL, but there is no significant difference in accuracy level. The addition of three new tasks and a larger graph size did not lead to any different conclusion. Similarly, LBD is shown to be faster than SCL but with no significant difference in accuracy level. This agrees with our hypothesis H1. There is no significant difference in TIME or CORRECT between LBD and STL, which is consistent with H2. Similar to the first experiment, we computed the edge crossing number for all the graphs used in the experiment. Fig. 14 shows the average number of edge crossings (with 95% confidence interval) for each edge type in the three different graph sizes used. Tukey's test showed that SCL has significantly more edge crossings than LBD ($p < .001$), which in turn has significantly more edge crossings than STL ($p < .001$). This confirms the result from the first experiment that edge crossing number is unlikely to be the deciding factor for task performance, as there is no significant different between STL and LBD for TIME or CORRECT but their edge crossing numbers are significantly different. As a result, path distance and smoothness are more important factors, which agrees with H2.

As expected, the 200 node graph size had the worst result for CORRECT, which is consistent with H3. As the graph size increases, some tasks become harder. However, it is surprising to see that the 50 node graph size had the worst TIME. Learning effect is a possible contributing factor. 50 node graphs are always shown first, followed by 100 node and then 200 node graphs. Participants might became more familiar with the tasks when they encountered them again in the set of graphs with 100 and 200 nodes. According to the first chart in Fig. 11, the difference among the mean completion time for different graph sizes is not very substantial.

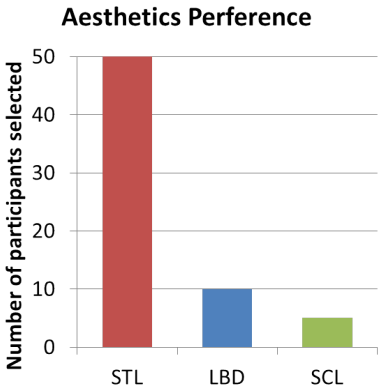


Fig. 13. Aesthetics preference in the second experiment.

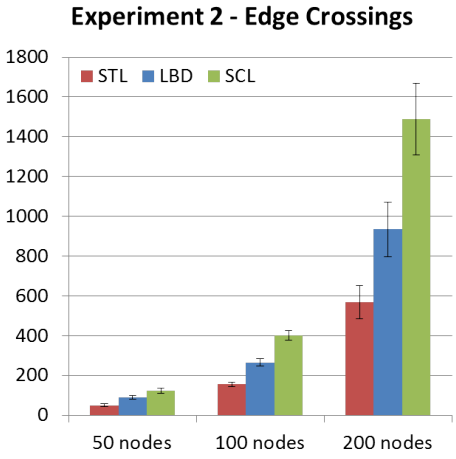


Fig. 14. The average number of edge crossings in the second experiment with 95% confidence level.

Participants ranked STL over SCL and LBD for aesthetics, which disagrees with our hypothesis H4. This is consistent with the results from the first experiment and showed that having the choice before or after the tasks did not make any difference. The preference is very strong with about 77% of participants selected STL as the most aesthetically pleasing edge type.

6 Conclusions and Future Work

We studied the impact of edge curvature on graph readability through two experiments. The results show that there are clear distinctions among different types of curved edge techniques. Introducing uniform edge curvature had a detrimental impact on graph readability and this negative effect increased with curvature level. On the other hand, Lombardi layout, which only uses curved edges when they improve angular resolution, had no significant difference in accuracy or task completion time when compared to straight edges. This makes Lombardi layout a good alternative to straight edges, especially in cases where curved edges are preferred. During the study we also found that edge crossing is unlikely to be the most important factor for readability of graphs with curved edges, because it has a weak correlation with the task performance. This is different from the results of the readability study of straight-edge graphs [25]. Instead, path distance and smoothness play a more important role.

The aesthetic preference for straight edges is very strong. The majority of the participants selected the straight edges as the most aesthetically pleasing when the choice was made both before and after the tasks. This is different from the conclusion from a previous study[3], which used real objects and artificial patterns in the experiment instead of graphs. This could be the result of lack of exposure to graphs with curved edges for many participants, but currently straight edge is the clear choice for edge curvature aesthetics.

We would like to acknowledge that the results presented in this paper are limited by the experimental parameters, and some of the conclusions may not be applicable to the general comparison of straight and curved edges for graph visualization. There are a group of methods, commonly known as edge bundling or confluent drawing techniques, are not included in this study. More importantly, while research on straight-edge graph visualization has been ongoing for decades, curved edges only start to gain research traction recently and there are still many challenging problems waiting to be solved. One such problem is the development of a simple and effective heuristics for reducing the number of edge crossings. For future work, we plan to expand the types of tasks used in the experiment. Most tasks used in this study focused on local topology and did not cover aspects such as overall graph structure. Expanding task types would also allow us to include popular edge bundling and confluent drawing methods, which are not covered in this study.

Acknowledgments

The authors would like to thank Roman Chernobelskiy for the help on the implementation of the force-directed Lombardi-style algorithm. A short version of the first experiment was presented as a poster at Diagrams 2012.

References

[1] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the Proceedings of the IEEE Symposium on Information Visualization*, pages 111–117, Washington, DC, USA, 2005. IEEE Computer Society.

[2] D. Archambault, H. C. Purchase, and B. Pinaud. Difference map readability for dynamic graphs. In *Proceedings of the 18th international conference on Graph drawing*, pages 50–61. Springer-Verlag, 2011.

[3] M. Bar and M. Neta. Humans prefer curved visual objects. *Psychological Science*, 17(8):645–648, 2006.

[4] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2009.

[5] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[6] R. Chernobelskiy, K. I. Cunningham, M. T. Goodrich, S. G. Kobourov, and L. Trott. Force-directed lombardi-style graph drawing. In *Proceedings of the 19th International Symposium on Graph Drawing*, pages 320–331. Springer-Verlag, 2011.

[7] M. Dickerson, M. T. Goodrich, and J. Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. In *Proceedings of the 11th International Symposium on Graph Drawing*, pages 1–12. Springer-Verlag, 2003.

[8] C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nollenburg. Lombardi drawings of graphs. In *Proceedings of the 18th International Symposium on Graph Drawing*, pages 195–207. Springer-Verlag, 2010.

[9] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[10] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2364–2373, 2011.

[11] J.-D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying graph links on treemaps. In *IEEE Symposium on Information Visualization Compendium*, 2003.

[12] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 187–194. IEEE Computer Society, 2011.

[13] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 30(11):1203–1233, 2000.

[14] C. Gorg, M. Pohl, E. Qeli, and K. Xu. Visual representations. In *Human-Centered Visualization Environments*, pages 163–230. Springer, 2006.

[15] R. Hobbs. *Mark Lombardi: Global Networks*. Independent Curators Inc.,U.S., 2003.

[16] W. Hogarth. *The Analysis of Beauty*. Yale University Press, 1753.

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

[18] D. Holten, P. Isenberg, J. J. van Wijk, and J.-D. Fekete. An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *IEEE Pacific Visualization Symposium*, pages 195–202, 2011.

[19] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 2299–2308, 2009.

[20] W. Huang and P. Eades. How people read graphs. In *Asia-Pacific Symposium on Information Visualisation*, pages 51–58, 2005.

[21] R. E. Kirk. *Experimental Design: Procedures for Behavioral Sciences*. Wadsworth Publishing, 1994.

[22] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the AVI workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pages 1–5. ACM Press, 2006.

[23] G. Melancon. Just how dense are dense graphs in the real world?: a methodological note. In *Proceedings of the AVI workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pages 1–7. ACM Press, 2006.

[24] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, 2008.

[25] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing*, pages 248–261. Springer-Verlag, 1997.

[26] H. C. Purchase, R. F. Cohen, and M. James. Validating graph drawing aesthetics. In *Proceedings of the Symposium on Graph Drawing*, pages 435–446. Springer-Verlag, 1996.

[27] G. Quercini and M. Ancona. Confluent drawing algorithms using rectangular dualization. In *Proceedings of the 18th International Symposium on Graph Drawing*, pages 341–352. Springer-Verlag, 2010.

[28] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 506–513. ACM, 2012.

[29] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12:733–740, 2006.

[30] A. Telea, O. Ersoy, H. Hoogendorp, and D. Reniers. Comparison of node-link and hierarchical edge bundling layouts: A user study. In D. A. Keim,

A. Pras, J. Schönwälder, and P. C. Wong, editors, *Visualization and Monitoring of Network Traffic*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[31] C. Ware and R. Bobrow. Supporting visual queries on medium-sized node-link diagrams. *Information Visualization*, 4(1):49–58, 2005.

[32] C. Ware and P. Mitchell. Visualizing graphs in three dimensions. *ACM Transactions on Applied Perception*, 5(2):1–15, 2008.

[33] M. Wattenberg. Arc diagrams: Visualizing structure in strings. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 110–116, 2002.

[34] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of International Conference for Human-Computer Interaction*, pages 811–819, 2006.

[35] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[36] N. Wong, M. S. T. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, 2003.

[37] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote, and J. Thomas. Dynamic visualization of graphs with extended labels. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 73–80, 2005.