Force-Directed Lombardi-Style Graph Drawing*

Roman Chernobelskiy[1], Kathryn I. Cunningham[1], Michael T. Goodrich[2], Stephen G. Kobourov[1], and Lowell Trott[2]

[1] Department of Computer Science, University of Arizona, Tucson, AZ, USA
[2] Department of Computer Science, University of California, Irvine, CA, USA

Abstract. A *Lombardi drawing* of a graph is one in which vertices are represented as points, edges are represented as circular arcs between their endpoints, and every vertex has perfect angular resolution (equal angles between consecutive edges, as measured by the tangents to the circular arcs at the vertex). We describe two algorithms that create "Lombardi-style" drawings (which we also call *near-Lombardi* drawings), in which all edges are still circular arcs, but some vertices may not have perfect angular resolution. Both of these algorithms take a force-directed, spring-embedding approach, with one using forces at edge tangents to produce curved edges and the other using dummy vertices on edges for this purpose. As we show, these approaches produce near-Lombardi drawings, with one being slightly better at achieving near-perfect angular resolution and the other being slightly better at balancing edge placements.

1 Introduction

The American artist, Mark Lombardi, was known for his drawings of social networks of conspiracy theories, which use circular arcs for edges and have a nice aesthetic placement for both vertices and edges (e.g., see Figure 1).

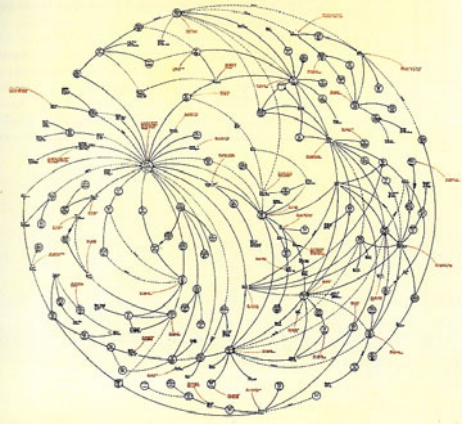


Fig. 1. Mark Lombardi's WFC 1970-84 [24]

* Research funded in part by NSF grants CCF-0545743 and CCF-1115971.

M. van Kreveld and B. Speckmann (Eds.): GD 2011, LNCS 7034, pp. 320–331, 2012.

Inspired by Lombardi's work, Duncan *et al.* [11, 12] introduce the concept of a *Lombardi drawing*, which is a drawing that uses circular arcs for edges and achieves the maximum (i.e., *perfect*) amount of angular resolution possible at each vertex. Their methods are deterministic and not force-directed, but, as they show, there exist graphs that do not have perfect Lombardi drawings. These negative results motivate a relaxation of the requirement that drawings achieve perfect angular resolution at every vertex.

At the same time, experimental studies have shown that angular resolution has a significant impact on the readability of a graph [27, 28]. Thus, our goal in this paper is to study the degree to which one can achieve good angular resolution at vertices by using the Lombardi-inspired approach of embedding edges as circular arcs.

Force-directed layout algorithms, also known as "spring embedders," are well-known for the "organic" type of drawings they produce, in terms of vertex and edge placement, using straight-line edges (e.g., see [7, 16–18]). Still, straight-line segments rarely occur in nature; hence, it is not clear that humans prefer straight-line segments for the sake of graph readability. With this in mind, we consider force-directed graph-drawing algorithms that allow for circular-arc edges and include forces that tend to spread those edges more evenly around vertices. We feel this approach can result in drawings that appear more "alive" than can be achieved using straight-line edges; see Fig. 2.

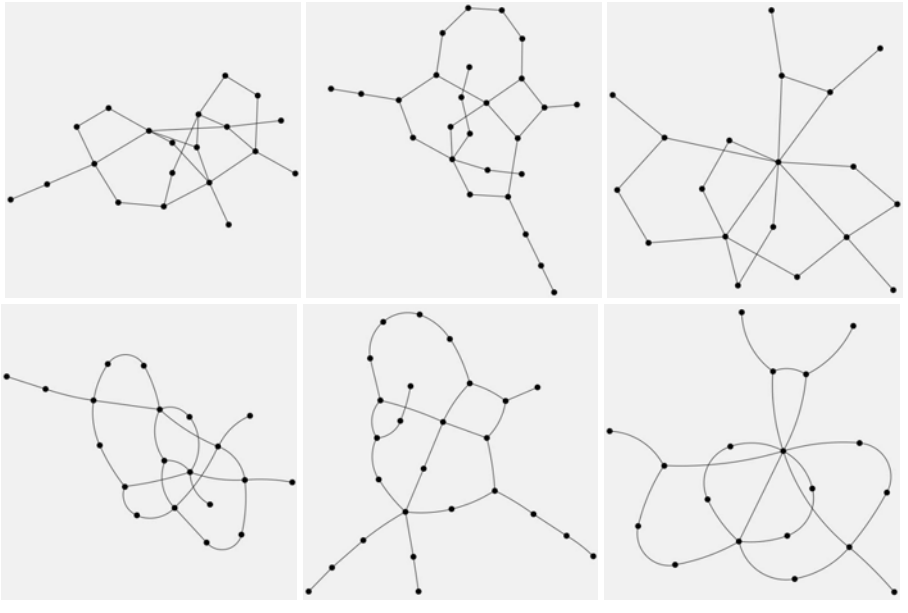


Fig. 2. Examples of standard straight-line and Lombardi-style drawings

1.1 Related Work

There are several graph drawing methods that use circular-arc edges or curvilinear poly-edges. For example, Goodrich and Wagner [20] give algorithms for drawing planar graphs using Bézier splines for edges, and Cheng *et al.* [6] describe a scheme for drawing graphs using circular arc poly-edges. *Confluent drawings* [9, 14, 22], bundle edges together in smooth curves so as to reduce crossings.

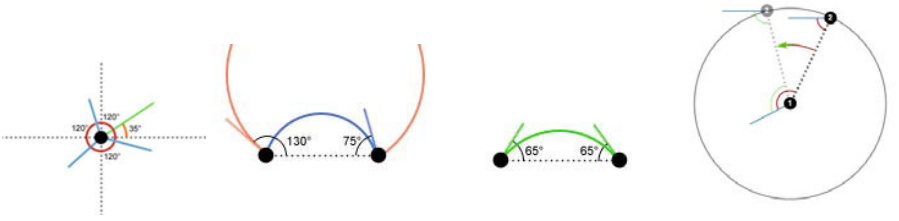
There is also a great deal of prior work on force-directed graph drawing and we refer the reader to some excellent surveys (e.g., see [1, 2, 7]). Holten and van Wijk [23] give a force-directed method for producing an edge-bundled drawing that is similar to a confluent drawing. Brandes and Wagner [5] describe a force-directed method for drawing train connections, where the vertex positions are fixed but transitive edges are drawn as Bézier curves (see also [3]). Finkel and Tamassia [15], on the other hand, describe a force-directed method for drawing graphs using curvilinear edges where vertex positions are free to move. Their method is based on adding dummy vertices, as one of our methods does, but their dummy vertices serve as control points for Bézier curves, rather than circular arcs, and their drawings do not achieve locally-optimal edge resolution at the vertices. Matsakis [26] describes a force-directed approach to producing a Lombardi-style drawing, which is based on iteratively visiting each vertex v and making adjustments locally with respect to v. Unfortunately, he does not evaluate his method experimentally and it is not clear that it always converges.

Angular resolution in the straight-line setting is also a well-studied problem [8, 19, 25]. Polyline edges have also been considered in the context of drawing planar graphs with good angular resolution [20, 21]. Finally, rotating optimal angular resolution templates for each vertex in the fixed position setting has been studied as well [4].

1.2 Our Results

In this paper, we describe two force-directed algorithms for *Lombardi-style* (or *near-Lombardi*) drawings of graphs, where edges are drawn using circular arcs with the goal of maximizing the angular resolution at each vertex. Our first approach calculates lateral and rotational forces based on the two tangents defining a circular arc between two vertices. In contrast, the second approach uses dummy vertices on each edge with repulsive forces to "push out" the circular arcs representing edges, so as to provide an aesthetic "balance." Another distinction between the two approaches is that the first one lays out the vertex positions along with the circular edges, while the second one works on graphs that are already laid out, only modifying the edges.

We have implemented both algorithms and tested them on a subset of the Rome graph library. We provide experimental evidence that our approaches yield drawings that have both a visual appeal and an increased angular resolution. We give explicit demonstrations for well-known symmetric graphs, random graphs, and even a graph drawn by Lombardi himself. We also provide a comparative analysis of the two approaches, which suggests that the tangent-based method is in general better at achieving the highest angular resolution possible, while the dummy-vertex approach is better in general at balancing the placement of edges.



(a) (b) (c) (d)

Fig. 3. Lombardi forces move and rotate vertices so that all corresponding tangents have matching angles, allowing for feasible circular arcs. (a) An illustration of pre-assigned tangents for a given degree-3 vertex. The angles between the tangents are equal and remain fixed, while the vertex itself can be rotated by changing its orientation with respect to the origin (currently 35° as indicated by the green line); (b) If the angles differ, then there cannot be a circular arc tangential to both tangents; (c) If the angles are equal, there is a unique circular arc; (d) The tangential force for vertex 2 with respect to vertex 1 moves vertex 2 to make the tangent angles equal.

2 A Tangent-Based Lombardi Spring Embedder Formulation

Force-directed algorithms treat a given graph as a physical system, where the vertices represent points in an N-body mechanics problem. In the system set up by Eades [13], vertices are treated as steel rings and the edges are springs that obey Hooke's Law. Fruchterman and Reingold describe a model in which a strong nuclear force attracts two protons within the atomic nucleus at close range, while an electrical force repels them at farther range [16]. Although inspired by physics, most force-directed algorithms do not attempt to mimic physical laws precisely.

Similar to most force-directed layout algorithms, our tangent-based Lombardi spring embedder assigns a force to each vertex and aims to minimize the overall energy of the system. There are three forces which affect vertex position, and one force which affects the radius of the circular arcs between a pair of vertices connected by an edge.

The attractive force, F_a, pulls vertices connected by edges closer together. It is applied to every pair of vertices connected by an edge as follows:$F_a = (d - k)/d$, where d is the current distance between the two vertices and k is a constant representing the ideal distance between them. The repulsive force, F_r, pushes vertices apart. It is applied to every pair of vertices using the following formula: $F_r = k^2/d^3$.

The tangential and rotational forces make it possible for circular arcs to be drawn between vertices, while maintaining a perfect (or near-perfect) angular resolution. To help compute the two forces, we augment each vertex with an orientation and fixed tangents, which dictate how to draw the arcs. The angles between the tangents are equal and remain fixed, while the vertex itself can be rotated by changing its orientation with respect to the origin. Note that the angle of a tangent at one vertex must equal the angle of a tangent at the other vertex for an arc to be possible between them. Here, angles are measured with respect to the segment connecting two vertices; see Fig. 3.

The tangential force, F_t, attempts to move vertices so as to make a circular arc possible between any pair of vertices connected by an edge. To compute this force we need to find the optimal position of a vertex with respect to its neighbor. The magnitude of

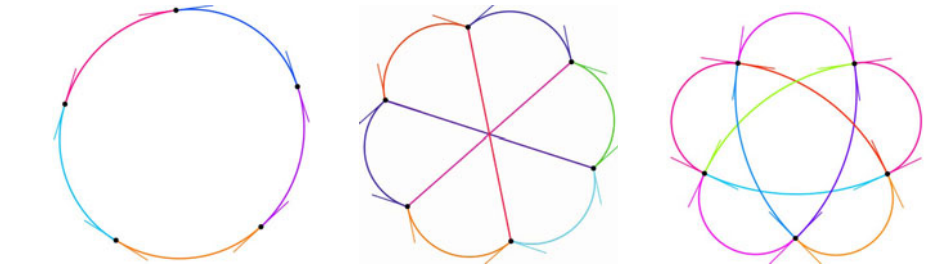


Fig. 4. Perfect Lombardi drawings of C_5, $K_{3,3}$ and K_5, with shown tangents

the force is proportional to the distance between this optimal position and the current position, and the direction is straight towards the optimal position. It is applied to every pair of vertices that share an edge as follows: $F_t = A \times d$, where d is the distance between current and optimal positions, and A is the tangential force constant.

The rotational force, F_ρ, does not attempt to move a vertex, but to rotate a vertex and its tangent template so as to make the tangent angles match, thereby making the arc between two vertices possible. To compute the rotational force we find the optimal angle of a tangent and subtract the current angle as follows:$F_\rho = B \times \Delta angle$, where $\Delta angle$ is the rotation required and B is a small constant. For each vertex v, the three appropriately scaled movement forces are added together to the rotational force in order to determine the overall force acting on the vertex: $F(v) = F_a + F_r + F_t + F_\rho$.

The following cooling function is used to determine the magnitude of the force in terms of the number of iterations: $T(i) = (T_0 * (M - i))/M$, where i is the iteration number and M is the maximum number of iterations (adapted from graphviz). T_0 is calculated as follows: $T_0 = K * \sqrt{n}/5$, where K is the ideal spring length and n is the number of vertices. Experimentally determined values for the constants we use are $K = 0.3$, $M = 600$, $A = 0.9$, $B = 0.5$. For many small graphs the algorithm succeeds in computing perfect Lombardi drawings; see Fig. 4.

Note that as described, the algorithm operates on a fixed ordering of the tangents around each vertex. This strict order hinders our algorithm as it attempts to find a Lombardi drawing. To give our algorithm more flexibility, we add a "shuffling" method that can modify the relative order of tangents around a vertex. This rearrangement occurs at the beginning of each iteration, and tangents are reordered only if a lower energy state is found within a reasonable number of calculations. In this case, the energy we seek to minimize is the total amount of rotation generated by all tangents to match the angle of their counterpart across their edges. This is the sum of the absolute value of the rotation generated by each tangent, rather than the net sum of the rotations, as we calculated when finding the rotational force. If the degree of a vertex is small, we calculate the total amount of rotation for every permutation of tangent orderings to find the one with the least energy. If the number of tangents is large, we test a subset of tangent orderings created by all possible pairwise swaps.

2.1 A Tangent-Based Near-Lombardi Spring Embedder

As not all graphs are Lombardi graphs [10], and our algorithm cannot guarantee that it will find a Lombardi drawing even if one does exist, when needed we relax the perfect

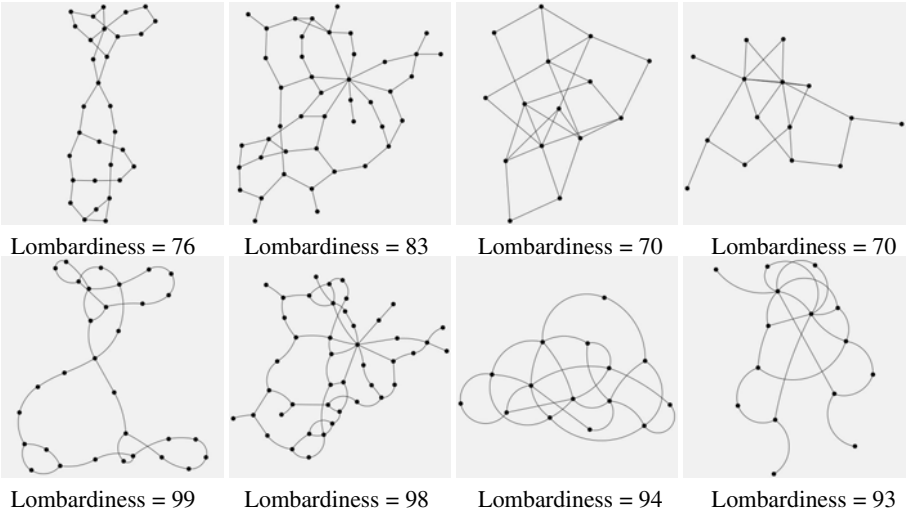


Fig. 5. Standard force-directed drawings (above) and near-Lombardi drawings (below)

angular resolution constraint. If the above tangent-based Lombardi Spring Embedder has failed to find optimal positions for every vertex, we modify the tangents of vertices which have infeasible edge constraints.

For tangents of adjacent vertices that have unequal angles, we move each tangent to the average of both tangents' positions. Now we can draw circular arcs between all connected vertices with minimal loss of angular resolution. As this change in tangent angles may drastically affect the angular resolution, we improve the resolution with another round of force-directed simulation. Rather than rotate or change the position of vertices as before, here we only modify angles between adjacent tangents (which were fixed before). This new force is based on the observation that, ideally, a tangent should bisect the angle formed by the two tangents immediately clockwise and counter-clockwise from itself. We find this midpoint and compute the force proportional to the required rotation to move the tangent to this location: $F_{NL} = C \times \Delta angle$, where $\Delta angle$ is the difference between the current angle of the tangent and the angle that bisects the neighboring tangents, and C is a constant. In order to maintain equal tangent angles for both tangents across an edge, which is necessary to draw a perfectly circular arc, we compute this force for both of the tangents incident to the edge, average it, and then apply it equally to both.

2.2 Lombardi Metric

For near-Lombardi drawings we need a measure of quality. As edges produced by the algorithm are always perfect circular arcs, the only violations of the Lombardi criteria are at vertices where perfect angular resolution could not be achieved. With this in mind, we define the *Lombardiness* of a drawing to be a number in the range 0 to 100, based on the average deviation from perfect angular resolution across all inter-tangent angles. This deviation is the difference between the actual angle measure and the measure of

the angle if its vertex had perfect angular resolution: $|a - \frac{2\pi}{d}|$, where a is the actual angle measure and d is the degree of a's vertex. To find the Lombardiness of a given graph, we compute this value for all inter-tangent angles, and then find the average. We scale this average to the 0-100 range by dividing by π (as the maximum value of the deviation is π). We then subtract this value from 100 to get the final score:

$$Lombardiness = 100 - \frac{1}{\pi \times 2|E|} \sum_{a \in A} |a - \frac{2\pi}{d}|.$$

Note that this measure of Lombardiness can be applied to all drawings we compute, as well as to straight-line drawings computed by a standard force-directed method (after all, straight-line segments are circular arcs with radius infinity). Fig. 5 shows several pairs of graphs drawn with a standard force directed embedder and with our tangent-based Lombardi spring embedder, along with their Lombardiness scores. For 80% of the 5451 graphs in the Rome library with 50 vertices or less, we obtain Lombardiness scores of 98 or higher, while very few have scores in the low 90s.

A web-enabled demo, as well as complete python source code, image libraries, and several movies illustrating this tangent-based algorithm at work can be found at `http://lombardi.cs.arizona.edu`.

3 A Dummy-Vertex Approach to Lombardi-Style Drawings

Brandenburg *et al.* have experimentally shown [1] that different force-directed methods can produce results with various trade-offs for aesthetic criteria. Thus, to allow for freedom with respect to these criteria, we built a second Lombardi-style force-directed method that allows for choice in the underlying force-directed algorithm. This method relies on a simple two-step process so as to allow an augmentation that can be applied to existing force-directed approaches. The first step involves using an existing straight-line force-directed method to place vertices and fix the order of edges around them, and the second step applies a force-directed approach based on the use of dummy vertices to maximize angular resolution at the vertices through the use of circular-arc edges.

Once the nodes have been placed with a user-selected force-directed method we begin our second phase. First, we assign to each edge an additional "dummy" vertex that is placed at the midpoint of that edge. Note that once the endpoints of an edge have been placed, only one more point is required to uniquely determine a circular arc

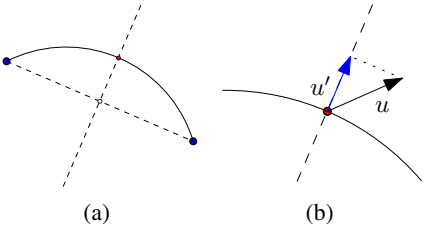


Fig. 6. (a) Points along the perpendicular bisector will determine an arc. (b) The update vector u' used will be the projection of the sum force vector u.

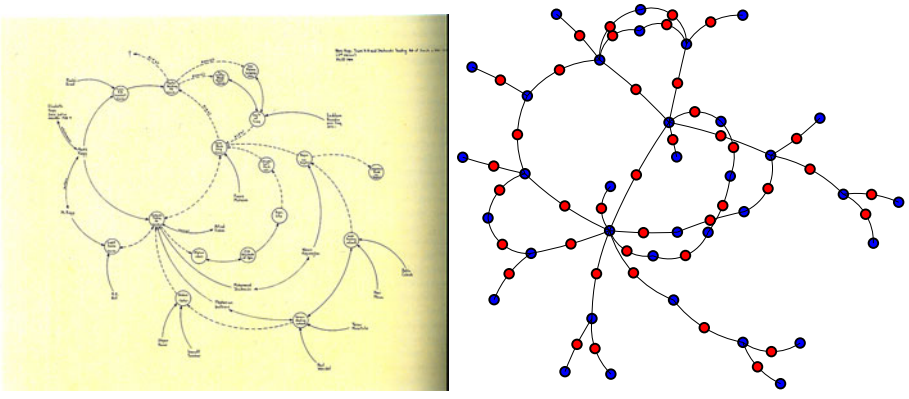


Fig. 7. Lombardi's Hans Kopp, Trans K-B and Shakarchi Trading [24], shown as rendered by Lombardi and as rendered by our dummy-vertex force-directed method

between these points. Thus, we can describe all possible arcs between nodes by the set of points along the perpendicular bisector of their straight-line connection; see Fig. 6(a).

The responsibility for moving an edge will be given to the additional node we have added to that edge. We then proceed to use the force-directed method to place these edge-nodes. Each (dummy) edge-node will consider the nodes that it connects as neighbors, and the partial edges as springs with a fractional resting length. Moreover, each edge node will repulse from all other nodes, both the original graph nodes and other edge-nodes. The sum force vector is calculated as before, but will be used to move the node in a modified way. If u is the sum force update vector we consider only its motion along the perpendicular bisector. This projection will determine a new update vector u' that we will use to move the edge node; see Fig. 6(b). Using u' we can determine the movement of the edge-node, while maintaining a circular arc edge. The edge-node positions are updated iteratively until an equilibrium is reached.

In Figure 7, we provide a scan of a drawing of Lombardi and the result of our method applied to the same graph. In Figure 8, we show the evolution of our algorithm through various substeps of the two phases.

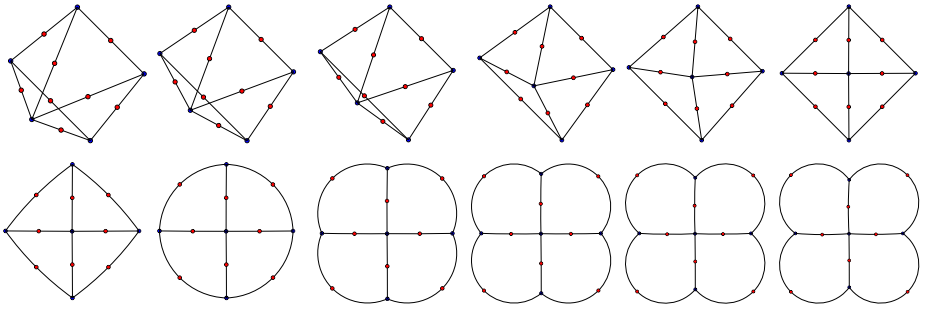


Fig. 8. A 5-node graph with center node initially displaced. Selected stages of the force-directed placement are shown. The top row shows phase 1 and the bottom phase 2.

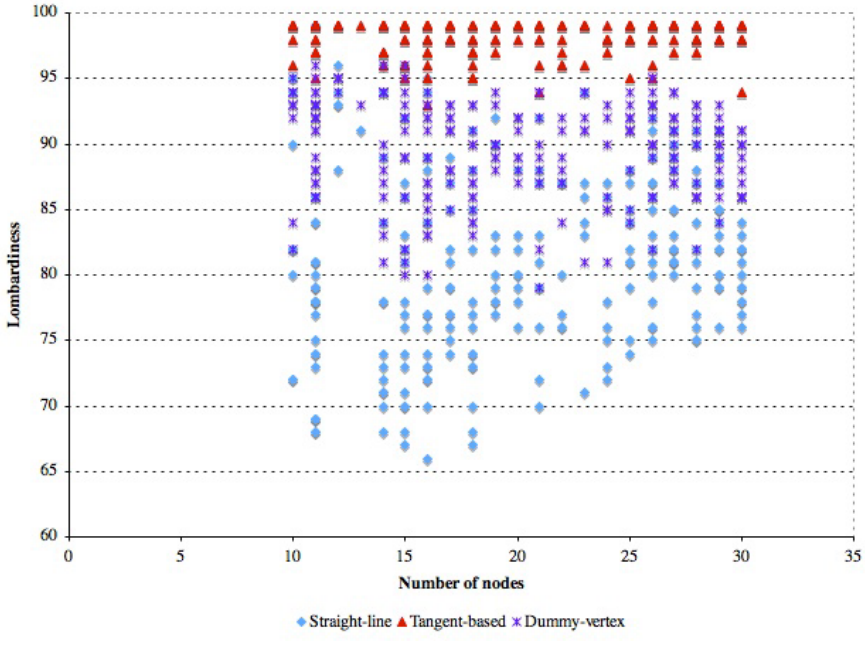


Fig. 9. A scatter plot of the Lombardiness of a collection of 250 graphs with straight-line, tangent-based, and dummy-vertex embeddings (with many data points overlapping)

4 A Comparative Analysis

In this section, we provide a small comparative analysis of our two methods. From the visual examples of drawings generated by the two methods it can already be seen that although both use circular arcs and aim to provide near-perfect angular resolution, the drawings seem to optimize different aesthetic qualities; see Fig. 10. For instance, in the tangent-based approach, the tangents of a node's edges have direct control over the angular resolution of that node, and this method does not take node positions as fixed. Thus, the tangent-based approach is able to achieve near-perfect angular resolution on all nodes. The dummy-vertex approach, on the other hand, starts from node positions determined by a straight-line force-directed method and moves edges into open space using dummy vertices. Since it does not directly consider the angle of other outgoing edges incident on the same vertex, it is not as successful in approaching perfect angular resolution. Nevertheless, it does improve angular resolution over straight-line drawings. To verify these observations, we performed an experimental analysis involving 250 graphs in the Rome library, and visualized their Lombardiness scores against their size; see Fig. 9. The data confirms this, showing a near-perfect separation between the three approaches (the third one given by the straight-line drawing).

The primary difficulty in drawing comparisons between these two methods is that they are inherently different, as evidenced by a visual comparison; see Fig. 10. While the tangent-based method arranges both vertices and edges, the dummy-vertex method focuses on edges alone. And since the underlying force-based layout algorithm used

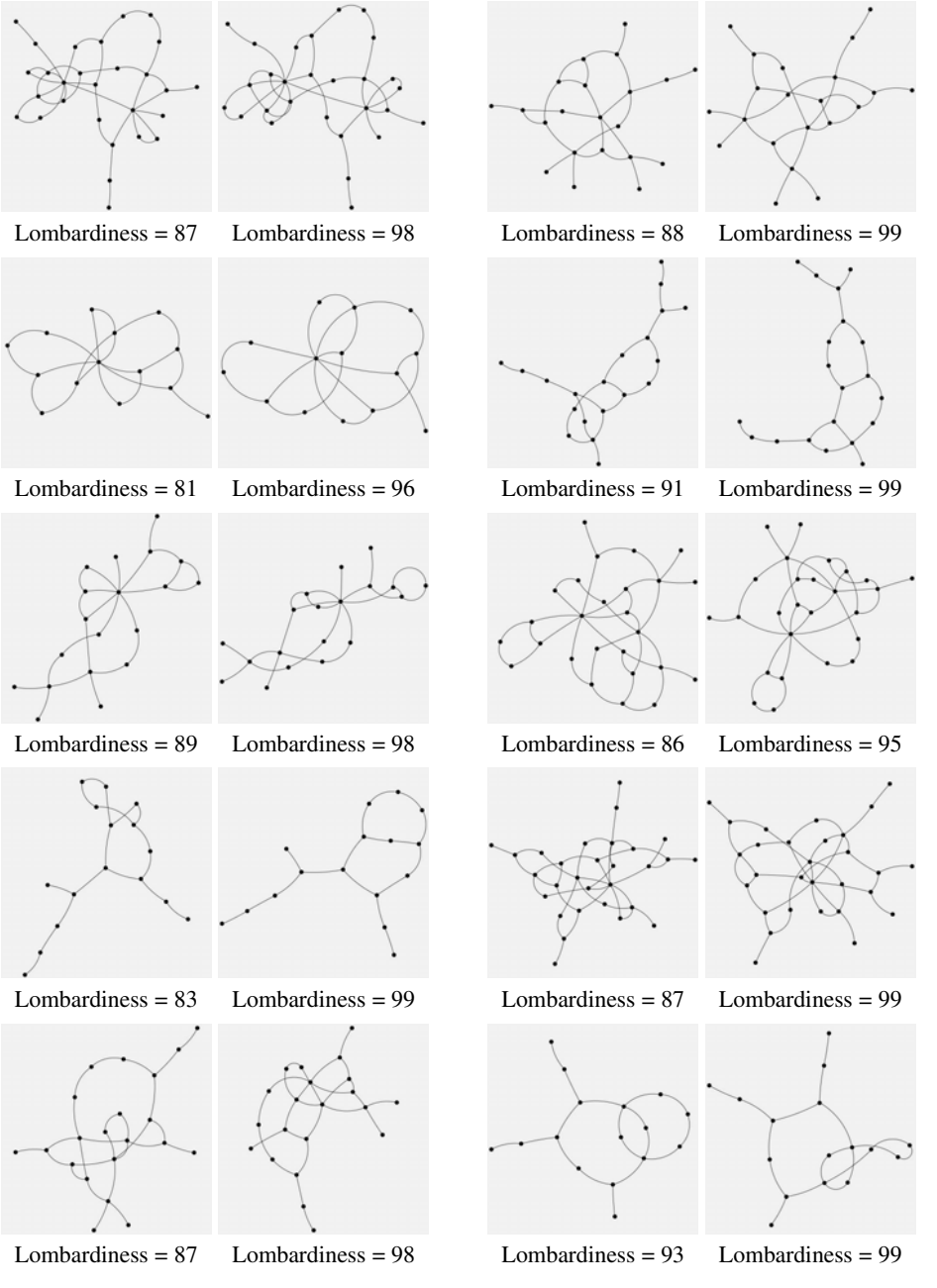


Fig. 10. Some example Lombardi-style drawings using the two force-directed approaches. For each pair, the drawing on the left was done using the dummy-vertex approach and the drawing on the right was done using the tangent-based approach. The Lombardiness score for each is given below.

for the dummy-vertex approach has such a significant influence on the end result, it is difficult to find metrics that compare the two approaches and ignore the underlying layout algorithm used.

An interesting advantage of the dummy-vertex approach is the increase in the distances between vertices and non-incident edges over both the straight-line drawings and results of the tangent-based approach. The reason for this is intuitive: each dummy vertex is repulsed by other edges and graph vertices, and so edges resist getting too close to other edges or non-incident vertices. This helps alleviate a distraction when edges pass too close to non-incident vertices, which a reader can mistake for an adjacency.

One additional observation we can make concerns edge crossings. While neither algorithm directly attempts to prevent crossings in the final drawings, both algorithms have a tendency to spread vertices, which might reduce crossings. In some cases, the tangent-based method will create crossings in its pursuit of better angular resolution, while in other cases, the dummy-vertex method allows edge crossings to occur because it takes the vertex positions as given from a straight-line force-directed method.

5 Conclusion and Future Work

We demonstrated two extensions of the spring-embedder paradigm for creating Lombardi and near-Lombardi drawings. A feature that can often be seen in Mark Lombardi's art is that many edges follow common trajectories. This feature is not included in the definition of a Lombardi drawing [12], but does occur frequently in drawings obtained by our spring embedders. While previous work on using cubic Bézier curves for good angular resolution is similar in spirit, the resulting drawings do not have vertices following common trajectories.

There are several natural directions to explore in future work, including alternative formulations of spring forces, a multi-level version that would scale to larger graphs, as well as possible use of this approach along with confluent drawing and edge bundling. A very informal user feedback indicates some aesthetic appeal of the drawings produced by the Lombardi spring embedder. Some keywords and phrases associated with these types of drawings were "more natural, " "like balloon animals," "blobby," "cute and cuddly," in contrast with the traditional straight-line realizations which were more "jagged" and "angular."

Acknowledgments. We would like to thank Ulrik Brandes and Alexander Wolff for useful discussions and suggestions.

References

1. Brandenburg, F., Himsolt, M., Rohrer, C.: An Experimental Comparison of Force-Directed And Randomized Graph Drawing Algorithms. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 76–87. Springer, Heidelberg (1997)
2. Brandes, U.: Drawing on Physical Analogies. In: Kaufmann, M., Wagner, D. (eds.) Drawing Graphs. LNCS, vol. 2025, pp. 71–86. Springer, Heidelberg (2001)
3. Brandes, U., Schlieper, B.: Angle and Distance Constraints On Tree Drawings. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 54–65. Springer, Heidelberg (2007)
4. Brandes, U., Shubina, G., Tamassia, R.: Improving angular resolution in visualizations of geographic networks. In: 2nd TCVG Symp. Visualization, pp. 23–32 (2000)

5. Brandes, U., Wagner, D.: Using Graph Layout to Visualize Train Interconnection Data. J. Graph Algorithms Appl. 4(3), 135–155 (2000)
6. Cheng, C.C., Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Drawing planar graphs with circular arcs. Discrete Comput. Geom. 25(3), 405–418 (2001)
7. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall PTR, Upper Saddle River (1998)
8. Di Battista, G., Vismara, L.: Angles of planar triangular graphs. SIAM J. Discrete Math. 9(3), 349–359 (1996)
9. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: Visualizing non-planar diagrams in a planar way. J. Graph Algorithms Appl. 9(1), 31–52 (2005)
10. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Löffler, M.: Planar and Poly-Arc Lombardi Drawings. In: van Kreveld, M., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 308–319. Springer, Heidelberg (2011)
11. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Drawing Trees with Perfect Angular Resolution and Polynomial Area. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 183–194. Springer, Heidelberg (2011)
12. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Lombardi Drawings of Graphs. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 195–207. Springer, Heidelberg (2011)
13. Eades, P.: A heuristic for graph drawing. Congressus Numerantium 42, 149–160 (1984)
14. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent layered drawings. Algorithmica 47(4), 439–452 (2007)
15. Finkel, B., Tamassia, R.: Curvilinear Graph Drawing Using the Force-Directed Method. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 448–453. Springer, Heidelberg (2005)
16. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. Softw. – Pract. Exp. 21(11), 1129–1164 (1991)
17. Gajer, P., Goodrich, M.T., Kobourov, S.G.: A multi-dimensional approach to force-directed layouts of large graphs. Comp. Geometry: Theory and Applications 29(1), 3–18 (2004)
18. Gajer, P., Kobourov, S.G.: GRIP: Graph dRawing with Intelligent Placement. Journal of Graph Algorithms and Applications 6(3), 203–224 (2002)
19. Garg, A., Tamassia, R.: Planar drawings and angular resolution: algorithms and bounds. In: 2nd European Symposium on Algorithms, London, UK, pp. 12–23 (1994)
20. Goodrich, M.T., Wagner, C.G.: A framework for drawing planar graphs with curves and polylines. J. Algorithms 37(2), 399–421 (2000)
21. Gutwenger, C., Mutzel, P.: Planar Polyline Drawings With Good Angular Resolution. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 167–182. Springer, Heidelberg (1999)
22. Hirsch, M., Meijer, H., Rappaport, D.: Biclique Edge Cover Graphs and Confluent Drawings. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 405–416. Springer, Heidelberg (2007)
23. Holten, D., van Wijk, J.J.: Force-directed edge bundling for graph visualization. Computer Graphics Forum 28, 983–990 (2009)
24. Lombardi, M., Hobbs, R.: Mark Lombardi: Global Networks. Independent Curators (2003)
25. Malitz, S., Papakostas, A.: On the angular resolution of planar graphs. SIAM J. Discrete Math. 7(2), 172–183 (1994)
26. Matsakis, N.: Transforming a random graph drawing into a Lombardi drawing. arXiv ePrints, abs/1012.2202 (2010)
27. Purchase, H.: Which Aesthetic Has The Greatest Effect On Human Understanding? In: Di-Battista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997)
28. Purchase, H.C., Cohen, R.F., James, M.: Validating Graph Drawing Aesthetics. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 435–446. Springer, Heidelberg (1997)