Force-Directed Lombardi-Style Graph Drawing

Roman Chernobelskiy*, Kathryn Cunningham*, Michael T. Goodrich†,
Stephen G. Kobourov*, and Lowell Trott†

*Dept. of Computer Science, Univ. of Arizona, Tucson, AZ, USA
†Dept. of Computer Science, Univ. of California, Irvine, CA, USA

Abstract. A *Lombardi drawing* of a graph is defined as one in which vertices are represented as points, edges are represented as circular arcs between their endpoints, and every vertex has perfect angular resolution (angles between consecutive edges, as measured by the tangents to the circular arcs at the vertex, all have the same degree). 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 both produce near-Lombardi drawings, with one being slightly better at achieving near-perfect angular resolution and the other being slightly better at balancing vertex 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).

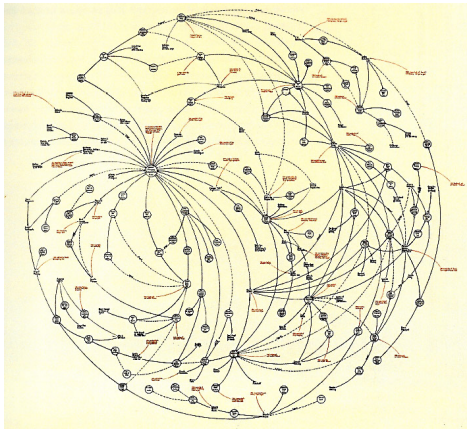

Fig. 1. Mark Lombardi's WFC 1970-84 [26].

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, not force-directed, but, as they show, there are several types of graphs that cannot be drawn perfectly as Lombardi drawings. Thus, these negative results motivate a relaxation of their requirement that drawings achieve perfect angular resolution at every vertex.

Nevertheless, we know from experimental studies that angular resolution has a significant impact on the readability of a graph [29, 30]. 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 known for the balanced types of drawings they produce, in terms of vertex and edge placement, using straight-line edges (e.g., see [1, 2, 7, 17–19]). 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, and, indeed, the work of Mark Lombardi suggests that they don't. (See also Fig. 2.) Therefore, the approach we are interested in studying in this paper is that of designing 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 even more natural than can be achieved using straight-line edges.

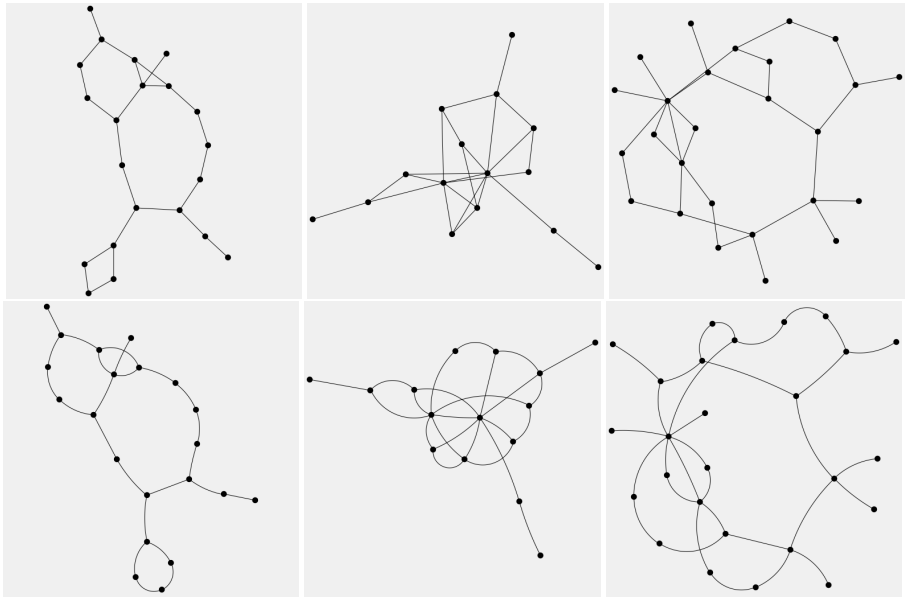

Fig. 2. Examples of standard straight-line and Lombardi-style drawings.

1.1 Related Work

There are several graph drawing methods that use circular-arc edges or curvilinear poly-edges. For example, Goodrich and Wagner [21] 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. Several groups of researchers have also studied *confluent drawings* [9, 14, 15, 23], which bundle edges together in smooth curves so as to reduce crossings.

There is, of course, a wealth of existing work on force-directed graph drawing. So, rather than review all of this previous related work, let us refer the reader to some of the excellent books and articles that survey the subject (e.g., see [1, 2, 7, 17–19]), and focus here on the most related papers on this topic. Holten and van Wijk [24] 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 [16], 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 [28] 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.

There is also considerable amount of additional work studying angular resolution for drawings that addresses the problem in the straight-line setting [8, 20, 27]. Polyline edges have also been studied in the context of drawing planar graphs with good angular resolution [21, 22, 25]. In addition, 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 producing *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 is based on a technique of applying forces at the tangents where edges meet vertices, so as to spread those tangents out as much as possible. Our second approach is instead based on a technique of using dummy vertices on each edge with repellent forces to "push out" the circular arcs representing edges, so as to provide an aesthetic "balance."

We have implemented the two algorithms and tested them on several graphs. 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 vertices.

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, vertices are treated as steel rings and the edges are springs that obey Hooke's Law [13]. 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 [17]. 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 spring length.

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 arc. 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 in such a way 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.

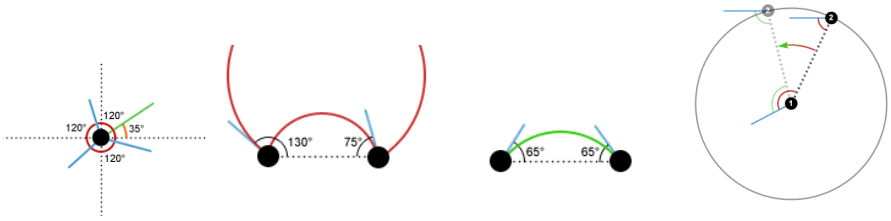

Fig. 3. Lombardi forces move and rotate vertices so that all corresponding tangents have matching angles, allowing for feasible circular arcs. (a) An illustration of pre-assigned tangents for a given degree-3 vertex. The angles between the tangents are equal and remain fixed, while the vertex itself 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 common circular arc between them; (c) If the angles are equal, there is a unique circular arc between them; (d) The tangential force for vertex 2 with respect to vertex 1 attempts to achieve matching outgoing angles from the two vertices.

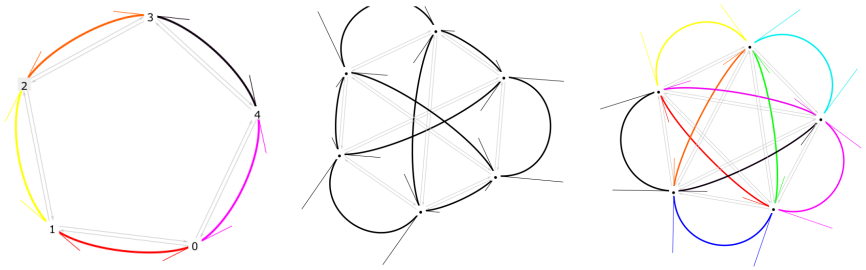

Fig. 4. Perfect Lombardi drawings of C_5, $K_{3,3}$ and K_5, with shown tangents.

The magnitude of the force is proportional to the distance between this optimal position and the current position, and the direction is 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 vertices, 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. 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 $K = 0.3$, $M = 600$, $A = 0.9$, $B = 0.5$; see Fig. 4 for examples on some simple graphs.

Note that as described the algorithm assumes a fixed order of the neighbors around each vertex. Fixing such an order can be very limiting in computing a Lombardi drawing. With this in mind we allow for modifying the order of adjacencies by shuffling the tangents at the beginning of each iteration to find a lower energy state. If the number of tangents (i.e., degree of a vertex) is small, we try every permutation to find the one with the minimal energy. If the number of tangents is large we move one tangent at a time. The energy function that we try to minimize with this shuffling is similar to the rotational force. Recall that when we compute the rotational force, we were interested in the net force for each vertex. With this shuffling force, however, we are interested in the total force (the sum of absolute values of each contributing rotation).

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 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 con-

nected vertices with minimal loss of angular resolution. As the change in tangents may be too drastic, we improve the angular resolution by another round of force-directed simulation. Rather than change the position and rotation of vertices as we did before, here we only modify tangent angles (which were fixed before). This new force is based on the observation that a tangent should be in the middle of two tangents clockwise and counter-clockwise from it. We find this midpoint and compute the force which is proportional to the required rotation to get the tangent there: $F_{NL} = C \times \Delta angle$, where $\Delta angle$ is the difference between current angle and optimal angle, and C is a constant. In order to maintain the "equal tangent angles" rule, we compute the force using both of the tangents along an edge, and apply it equally to both of them.

For near-Lombardi drawings we need a measure of quality. As all edges are still circular arcs, the only violations 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, defined by the average deviation from perfect angular resolution. To compute this value we add the deviations over all angles and all vertices $\sum_{v \in V} \Delta angle/2|E|$ and scale to the 0-100 range dividing by 1.8 (as the maximum value of $\Delta angle$ is 180).

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 90's.

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

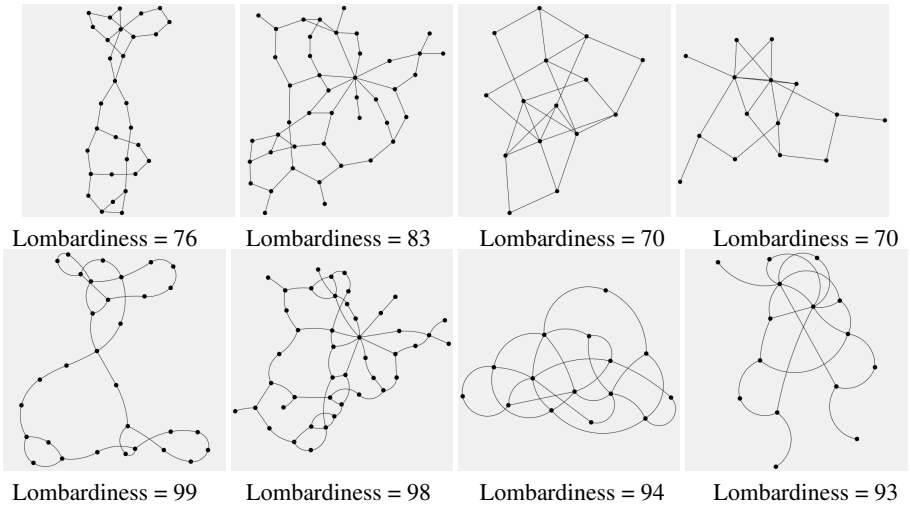

Fig. 5. Standard force-directed drawings (above) and near-Lombardi drawings (below).

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 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.

As mentioned above, one of the simplest forms of the force-directed approach is to use linear springs for edges and charged particle repulsion for vertices, to achieve near-uniform edge length and node distribution respectively. In this approach, each edge in the input graph is represented by a spring, and each node a charged particle. If adjacent nodes are spread too far, the difference between the spring length and its resting state is large, so the nodes will be strongly pulled together. As nodes grow close, the repulsion forces, whose strength is calculated relative to the inverse-square of the distance, pushes vertices apart. Thus, the forces act to negotiate the nodes into equilibria.

Practically speaking, this gradient-descent relaxation is done by calculating the sum of forces acting on each node, in turn, calculating spring forces from neighbors and repulsive forces from all other nodes. This summation results in an update vector, which is then applied so that the node is moved a small amount in the direction of the update vector. This update vector is calculated for all nodes, and then they are adjusted. This small vectored movement is then iterated until the graph achieves a minimal (or approximately near-minimal) energy state.

Our force-directed algorithm based on the use of dummy vertices functions in two phases. Each phase consists of iterated force-directed updates, as in the standard gradient-descent approach. The first phase places the nodes of the graph. This phase proceeds as has been described above, and is standard for straight-line force-directed algorithms.

Once we have placed the nodes of the graphs we begin our second phase. First, we assign to each edge an additional "dummy" vertex that is placed at the midpoint of that edge. A valuable observation is that once the endpoints of an edge have been placed, only one more point is required to uniquely determine a circular arc 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).

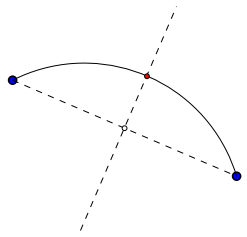

Fig. 6. Points along the perpendicular bisector will determine an arc.

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 the 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. 7). Using u' we will make a small movement of the edge node, maintaining a circular arc edge. The edge nodes are updated iteratively until they reach equilibrium. At this point, our algorithm completes.

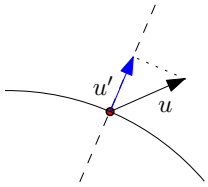

Fig. 7. The update vector u' used will be the projection of the sum force vector u.

In Figure 8, we provide a scan of a drawing of Lombardi and the result of our method applied to the same graph.

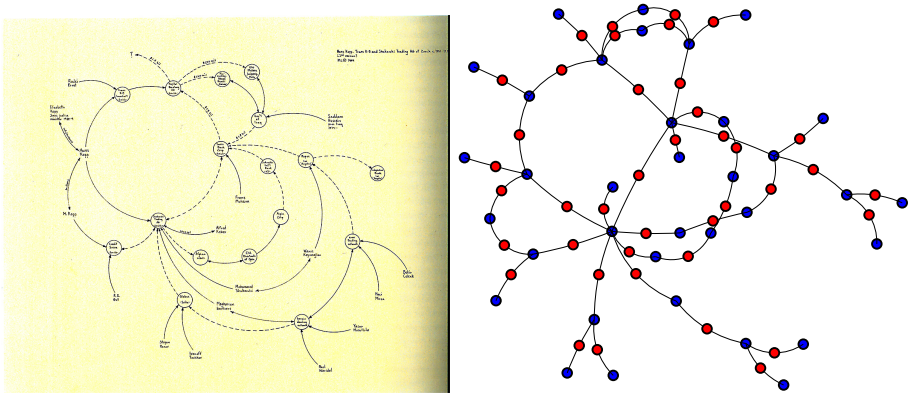

Fig. 8. The graph for Lombardi's Hans Kopp, Trans K-B and Shakarchi Trading [26], shown as rendered by Lombardi and as rendered by our force-directed method based on the use of dummy vertices.

In Figure 9, we show the evolution of our algorithm through various substeps of the two phases.

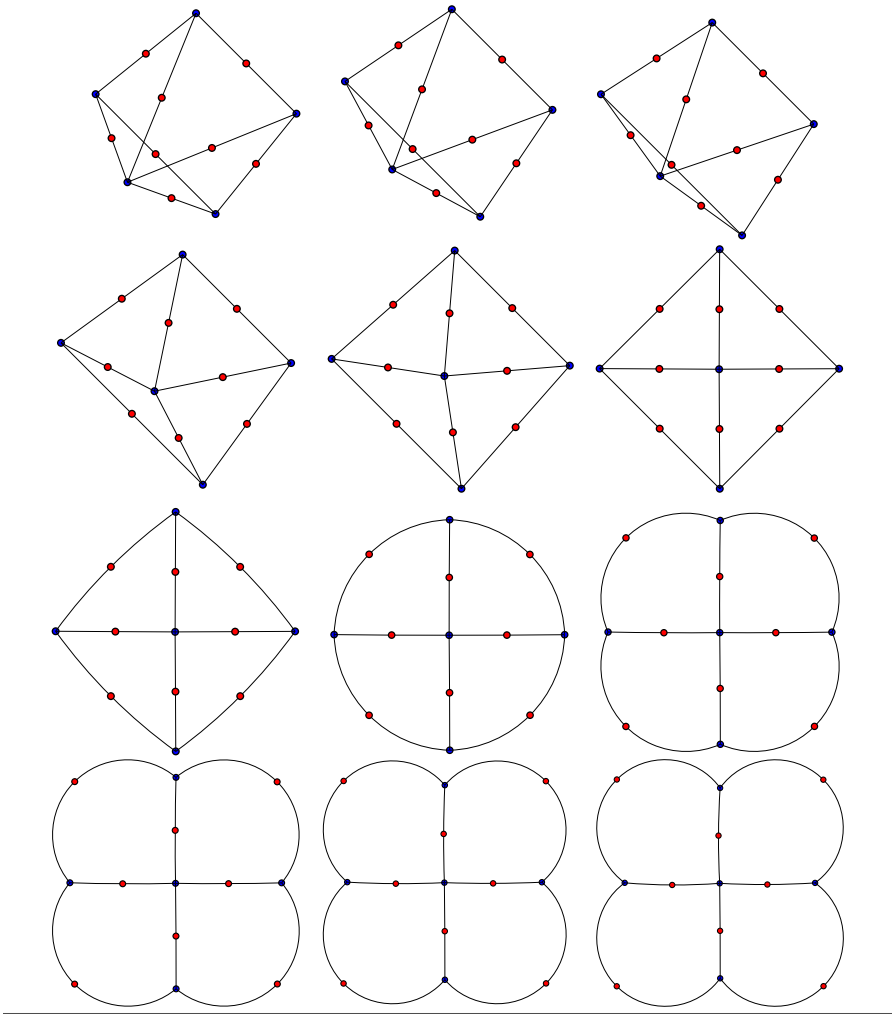

Fig. 9. A five-node graph with center node initially displaced. Selected stages of the force-directed placement are shown. The top two rows show phase 1 and the bottom two show phase 2.

4 A Comparative Analysis

In this section, we provide a comparative analysis of our two methods. Already from the visual examples of graphs generated by the two methods (e.g., see Fig. 13 in the appendix), we can see that the two approaches result in varied aesthetic qualities.

As mentioned above, both algorithms use forces to increase angular resolution in their final drawings, but their alternative approaches can have different results. 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 in a scatter plot, which is shown Fig. 10. The data show a near-perfect separation regarding the Lombardiness of the three approaches.

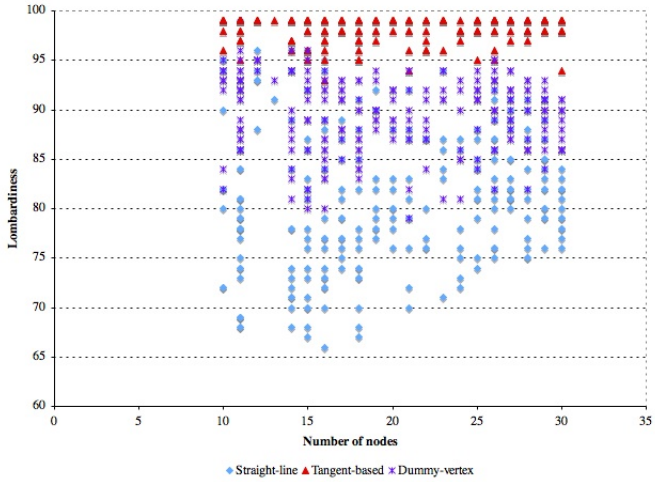

Fig. 10. A scatter plot of the Lombardiness of a collection of 250 graphs with straight-line, tangent-based, and dummy-vertex embeddings (with many data points overlapping).

Both of our approaches work very well at increasing the distance between graph vertices. This is done as a separate step in the dummy-vertex approach, and during this phase it is the primary focus of the algorithm. The tangent-based method sacrifices some of this distribution of nodes for the sake angular resolution, but still results in fairly uniform final locations, albeit not always as balanced as in the dummy-vertex approach (e.g., see Fig. 13 in the appendix).

An interesting advantage demonstrated by 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 algorithm's tendency to spread vertices tends to reduce crossings in simple cases. 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 vertices 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, especially when taking the dummy-vertex approach. 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, " "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. F. Brandenburg, M. Himsolt, and C. Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In F. Brandenburg, editor, *Graph Drawing*, volume 1027 of *LNCS*, pages 76–87. Springer, 1996.
2. U. Brandes. Drawing on physical analogies. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 71–86. Springer-Verlag, 2001.
3. U. Brandes and B. Schlieper. Angle and distance constraints on tree drawings. In *Proc. 14th Int. Symp. on Graph Drawing (GD 2006)*, pages 54–65, London, UK, 2007.
4. U. Brandes, G. Shubina, and R. Tamassia. Improving angular resolution in visualizations of geographic networks. In *2nd TCVG Symp. Visualization (VisSym '00)*, pages 23–32, 2000.
5. U. Brandes and D. Wagner. Using Graph Layout to Visualize Train Interconnection Data. *J. Graph Algorithms Appl.*, 4(3):135–155, 2000.

6. C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. *Discrete Comput. Geom.*, 25(3):405–418, 2001.
7. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.
8. G. Di Battista and L. Vismara. Angles of planar triangular graphs. *SIAM J. Discrete Math.*, 9(3):349–359, 1996.
9. M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Algorithms Appl.*, 9(1):31–52, 2005.
10. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Löffler. Planar and Poly-Arc Lombardi Drawings. In *19th Symp. on Graph Drawing*, 2011. Under submission.
11. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Drawing Trees with Perfect Angular Resolution and Polynomial Area. In *Proc. 18th Int. Symp. on Graph Drawing (GD 2010)*, 2010.
12. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Lombardi Drawings of Graphs. In *Proc. 18th Int. Symp. on Graph Drawing (GD 2010)*, 2010.
13. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
14. D. Eppstein, M. T. Goodrich, and J. Y. Meng. Delta-confluent drawings. In P. Healy and N. S. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2005.
15. D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent layered drawings. *Algorithmica*, 47(4):439–452, 2007.
16. B. Finkel and R. Tamassia. Curvilinear graph drawing using the force-directed method. In J. Pach, editor, *Graph Drawing*, volume 3383 of *LNCS*, pages 448–453. Springer, 2005.
17. T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. – Pract. Exp.*, 21(11):1129–1164, 1991.
18. P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry: Theory and Applications*, 29(1):3–18, 2004.
19. P. Gajer and S. Kobourov. GRIP: Graph drawing with intelligent placement. In J. Marks, editor, *Graph Drawing*, volume 1984 of *LNCS*, pages 104–109. Springer, 2001.
20. A. Garg and R. Tamassia. Planar drawings and angular resolution: algorithms and bounds. In *2nd European Symposium on Algorithms*, pages 12–23, London, UK, 1994.
21. M. T. Goodrich and C. G. Wagner. A framework for drawing planar graphs with curves and polylines. *J. Algorithms*, 37(2):399–421, 2000.
22. C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. In *6th International Symposium on Graph Drawing*, pages 167–182, 1998.
23. M. Hirsch, H. Meijer, and D. Rappaport. Biclique edge cover graphs and confluent drawings. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *LNCS*, pages 405–416. Springer, 2007.
24. D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28:983–990, 2009.
25. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.
26. M. Lombardi and R. Hobbs. *Mark Lombardi: Global Networks*. Independent Curators, 2003.
27. S. Malitz and A. Papakostas. On the angular resolution of planar graphs. *SIAM J. Discrete Math.*, 7(2):172–183, 1994.
28. N. Matsakis. Transforming a random graph drawing into a Lombardi drawing. *arXiv ePrints*, abs/1012.2202, 2010.
29. H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th Symposium on Graph Drawing (GD)*, pages 248–261, 1998.
30. H. C. Purchase, R. F. Cohen, and M. James. Validating graph drawing aesthetics. In *Proceedings of the 3rd Symposium on Graph Drawing (GD)*, pages 435–446, 1996.

Appendix

In this appendix, we include some examples of Mark Lombardi's art and some more drawings generated by our two Lombardi Spring Embedder algorithms. In Fig. 11 we can see that near-perfect angular resolution is a good approximation to what the artist used. In Fig 12 we can also see that groups of vertices follow well-defined trajectories.

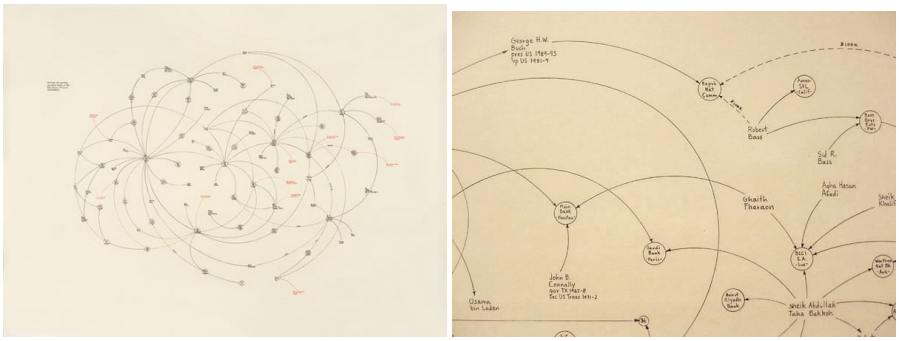

Fig. 11. Examples of Mark Lombardi's art; (a) The topmost vertex could probably be realized with perfect angular, but it is not, probably to allow for clear and well-defined trajectories of a number of vertices in the actual drawing. (b) Several vertices deviate from the normal perfect angular resolution, probably to accommodate more important nodes, which often have very well distributed vertices, unless there is a "direction" in the underlying story.

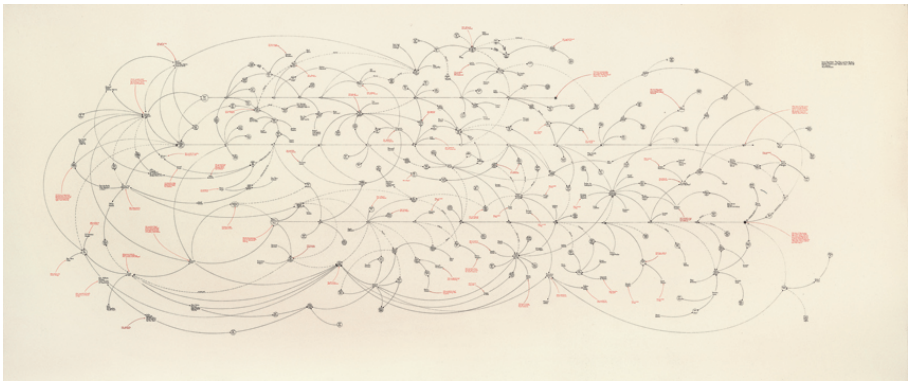

Fig. 12. A large example of Mark Lombardi's, showing several multi-vertex trajectories and hard-wired left-to-right time component.

We include a few more examples of graphs drawn by our Lombardi spring embedder algorithms, which illustrate some of the complementary strengths of the two approaches, in Fig. 13.

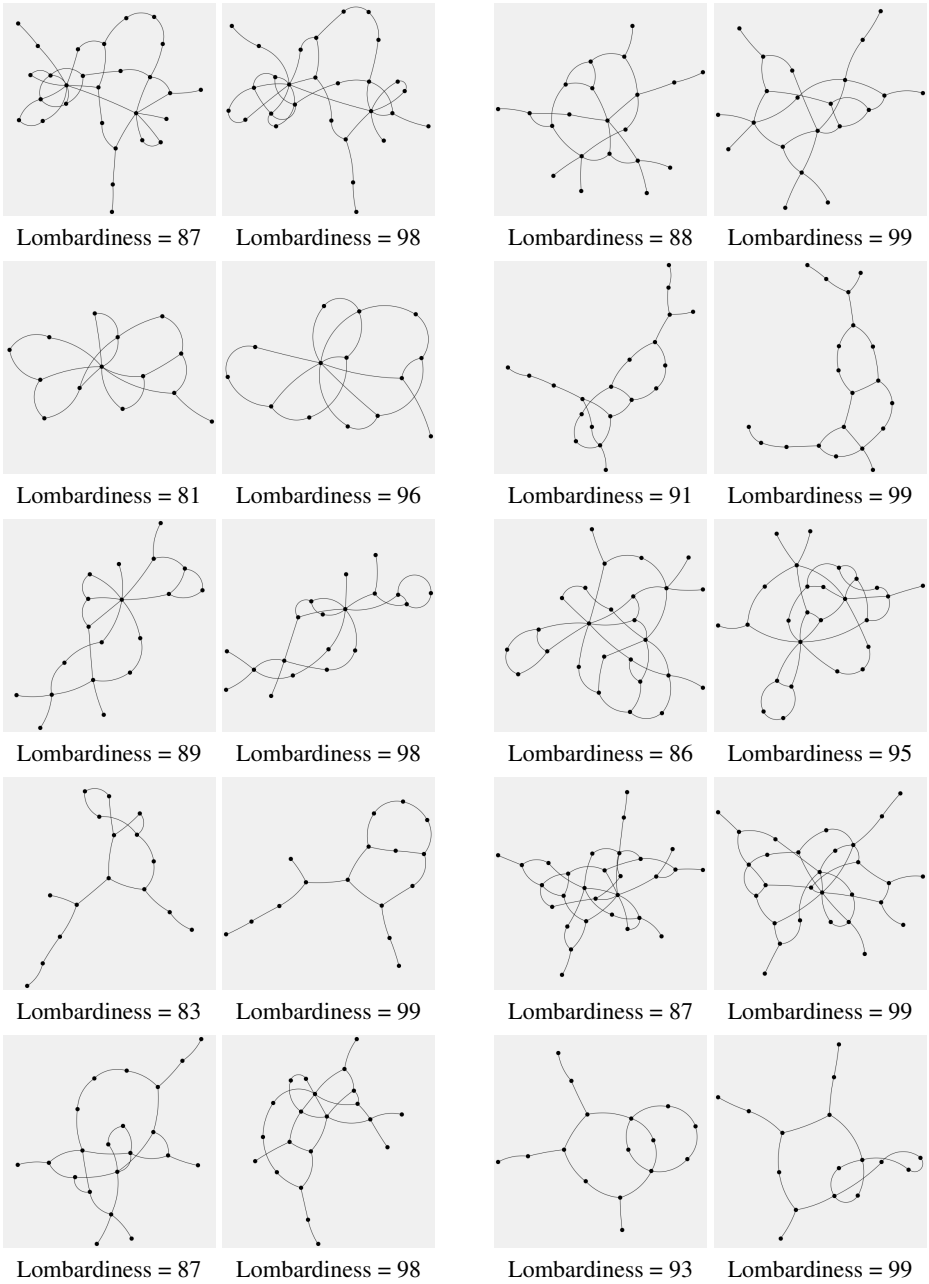

Fig. 13. Some example Lombardi-style drawings using the two force-directed approaches. For each pair, the drawing on the left was done using the dummy-vertex approach and the drawing on the right was done using the tangent-based approach. The Lombardiness score for each is given below.