First publ. in: Data visualization 2000 : proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization in Amsterdam, The Netherlands, May 29-31, 2000 / W. de Leeuw ... (eds.). Wien: Springer, 2000, pp. 23-32

Improving Angular Resolution in Visualizations of Geographic Networks*

Ulrik Brandes[1], Galina Shubina[2], and Roberto Tamassia[2]

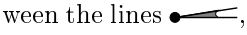
[1] University of Konstanz, Department of Computer and Information Science, Box D 188, 78457 Konstanz, Germany, Ulrik.Brandes@uni-konstanz.de
[2] Brown University, Department of Computer Science, Providence, Rhode Island 02912-1910, USA, {gs,rt}@cs.brown.edu

Abstract. In visualizations of large-scale transportation and communications networks, node coordinates are usually fixed to preserve the underlying geography, while links are represented as geodesics for simplicity. This often leads to severe readability problems due to poor angular resolution, i.e. small angles formed by lines converging in a node. We present a new method using automatically routed cubic curves that both preserves node coordinates and eliminates the resolution problem. The approach is applied to representations in the plane and on the sphere, showing European train connections and Internet traffic, respectively.

1 Introduction

Since nodes in large-scale transportation networks, such as airline flight plans, train connection maps, or extracts of the Internet, have a given geographic location, we call these networks *geographic networks*. Typical visualizations use given node coordinates either directly, or only apply an appropriate projection (e.g., from the surface of a sphere to a plane) to retain the viewers familiarity with the underlying geometry [3]. In general, the exact routing of connections is not important, so that links are often represented as geodesics (straight-lines or great circles). While computationally and visually simple, this approach does not take into account the perceptual organization of the resulting visualization.

Prior work on improving the visual quality has focused on moderate repositioning of close or overlapping nodes [17, 10], and re-routing of edges cutting through nodes or other features [1, 8]. Since we address a different criterion of layout quality, our work can be potentially used in conjunction with these approaches.

Severe readability problems in visualizations of geographic networks stem from small angles formed by lines converging in a common node. Small angles cause viewers to perceive filled-in areas between the lines  , causing "blob"-effects and making it difficult to tell lines apart.

* Research partially supported by the U.S. National Science Foundation under grants CCR-9732327 and CDA-9703080, by the U.S. Army Research Office under grant DAAH04-96-1-0013, and by the German Academic Exchange Service (DAAD, Hochschulsonderprogramm III).

We present a new method that modifies a given visualization so that all angles formed by incident lines are of sufficient size. This is achieved by replacing straight-line and great-circle connections with cubic curves. While visually still simple, cubic curves allow to prescribe angles between incident curves at will. A computationally fast and simple method is introduced and demonstrated working on two real-world data sets.

The remainder of this paper is organized as follows. In Sect. 2, we provide some terminology and background on properties of angles in network visualizations. Our approach is described in Sect. 3 and some extensions are presented in Sect. 4. Finally, it is applied to visualizations in the plane and on the sphere in Sect. 5, using data from European train and ferry schedules, and the multicast backbone of the Internet, respectively.

2 Background on Angular Resolution

Networks are conveniently described as graphs $G = (V, E)$, where v is a set of *vertices* (nodes), and E is a set of *edges* (links). Without loss of generality we consider only undirected graphs without loops and multiple edges, so that every edge is an unordered pair of vertices.

A network visualization is a drawing of the graph, i.e. a mapping of vertices to points in the plane or in space together with a mapping of edges to curves connecting the points of their respective vertices. We confine ourselves to drawings that map vertices into the plane or the surface of a sphere.

Assume, we are given a drawing of a graph in the plane, such that every edge is represented as a straight line. The *local angular resolution* at vertex $v \in V$ is the minimum angle between a pair of edges incident to v. The minimum angle between any pair of edges incident to any $v \in V$ is called the *angular resolution* of the drawing and introduced in [9].

A trivial upper bound on the local angular resolution at vertex $v \in V$ is $\frac{2\pi}{d_G(v)}$, where $d_G(v)$ is the *degree* of v, defined as the number of edges incident to v. It is shown in [9] that every simple graph has a straight-line drawing with angular resolution $\Omega(\frac{1}{\Delta(G)})$, where $\Delta(G)$ is the maximum degree of any vertex in G. For planar graphs, i.e. graphs that can be drawn in the plane without crossing edges, it is shown how to construct drawings with asymptotically optimal angular resolution. However, these drawings are in general not planar.

Every planar graph has a planar straight-line drawing with angular resolution $\Omega(\frac{1}{\alpha^{\Delta(G)}})$ for some constant $\alpha > 1$ [18], but there are planar graphs for which the angular resolution in any planar straight-line drawing is bounded by $\mathcal{O}(\sqrt{\frac{\log \Delta(G)}{\Delta(G)^3}})$ [12]. Note that maximizing the angular resolution over all planar straight-line drawings of a planar graph is $\mathcal{NP}$-hard [11].

It is shown in [16] how to obtain planar drawings with asymptotically optimal angular resolution when edges may be represented as sequences of straight lines. This result is improved in [13] and [14], where it is shown that only two bends per edge are needed, so that edges can be drawn as cubic curves.

Our setting differs from the above in that the mapping of vertices to points is already fixed. We do not know of other work on this particular problem.

To alter the angular resolution, we clearly must use polyline or curved edges as well. We currently use cubic Bézier curves [4], partially for the pragmatic reason that they are built into PostScript and LEDA's [19] graph editor. They are defined by a sequence of four *control points*, $b_0, \ldots, b_3$ (and thus three *control segments* $\overline{b_0b_1}$, $\overline{b_1b_2}$, and $\overline{b_2b_3}$), and have several advantages over other types of curves [5]. In particular, the tangents at b_0 and b_3 are collinear with the first and third control segment. The definitions of local and global angular resolution are hence easily generalized to drawings with cubic Bézier curves.

To replace a straight-line edge with a cubic Bézier curve, we only have to place the inner control points b_2 and b_3, since b_0 and b_3 must be assigned to the positions given for the incident vertices. The placement is then divided into

1. determining directions for the first and third control segment, and
2. determining the lengths of the first and third control segment.

Since the resulting angular resolution is fully determined by the first step, we do not consider the second step in the remainder of this paper. As a simple heuristic the length of the first and third control segment is chosen proportional to the length of the straight-line edge. Future work should guarantee further desirable properties [15] of, potentially higher-order, curves (e.g., to retain planarity of a drawing). Figure 1 shows the principal combinations of directions of the first and third control segment and resulting cubic Bézier curves. Note that all of them are easy to trace for the human eye. Moreover, we will see later that our approach avoids the four high-curvature cases on the right hand side whenever possible.

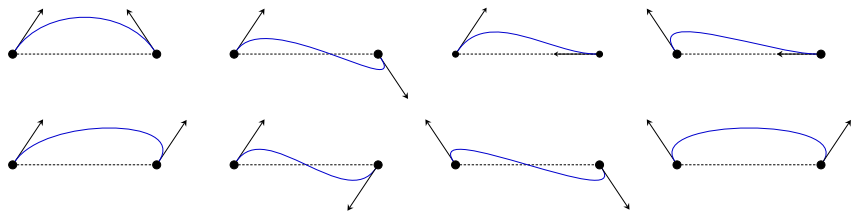


Fig. 1. Principal non-degenerate combinations of control segment orientations

3 Fixed Angular Resolution and Interesting Rotations

In this section, we show how to improve the angular resolution of a given straight-line drawing in the plane by choosing directions for the first and third control segment of Bézier curves replacing the straight lines.

On the one hand, these directions should yield good angular resolution. On the other hand, we also want them to resemble the straight-line directions as close as possible to avoid the high-curvature cases of Fig. 1, which also serves to keep the drawing simple. Note that we can treat control segments incident to one vertex independently from control segments incident to other vertices. Therefore, let $e_0, \ldots, e_{d_G(v)-1}$, be a counterclockwise ordering of the edges around some $v \in V$ in the given drawing (with ties broken arbitrarily), and denote by α_i, $i = 0, \ldots, d_G(v) - 1$, the angle between e_i and its counterclockwise neighbor.

Accordingly, we define $c_0, \ldots, c_{d_G(v)-1}$ to be the corresponding ordering of control segments incident to v. The angles between neighboring control segments are fixed to be $\frac{2\pi}{d_G(v)}$, thus ensuring optimal local, and hence global, angular resolution. Because of the simplicity requirement, we want to penalize the deviation of control segments from straight-line edges. Denote by x_i the angle between e_i and c_i, $i = 0, \ldots, d_G(v) - 1$, where $x_i > 0$, if e_i comes before, and $x_i < 0$, if e_i comes after c_i in the counterclockwise order around v. We call these quantities the *angular differences*. See Fig. 2 for an illustration and note that $x_3 < 0$.

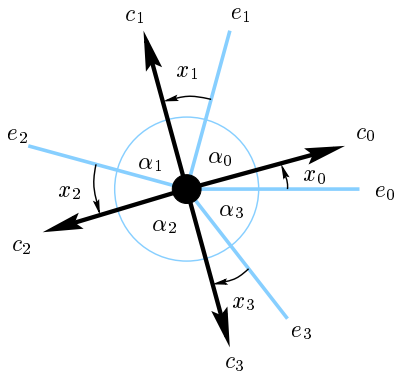


Fig. 2. Angles α_i between straight-line edges, and angular differences x_i

Any set of control segments satisfying the angle constraints is called a *rotation* at v. There are several ways to define desirable rotations. Since the angles between control segments are fixed. all angular differences of a vertex are dependent. It is easily seen that $x_1 = x_0 + \frac{2\pi}{d_G(v)} - \alpha_0$, and more general $x_i = x_0 + i \cdot \frac{2\pi}{d_G(v)} - \sum_{j=0}^{i-1} \alpha_j$ for $i = 0, \ldots, d_G(v) - 1$. We frequently use $y_i = i \cdot \frac{2\pi}{d_G(v)} - \sum_{j=0}^{i-1} \alpha_j$ to denote the *angular offset* caused by choosing x_0 as the independent variable. A straightforward objective is to minimize the largest absolute angular difference,

$$\max_{i=0,\ldots,d_G(v)-1} |x_i|. \tag{1}$$

We call this a *minimum* rotation.

Theorem 1. *The minimum rotation is unique, and linear-time computable.*

Proof. Since $x_i = x_0 + y_i$, $i = 0, \ldots, d_G(v) - 1$, the maximum absolute angular difference is the maximum of $|x_0 + \min_i y_i|$ and $|x_0 + \max_i y_i|$. This maximum becomes minimum for

$$x_0 = -\frac{\min_i y_i + \max_i y_i}{2},$$

since both absolute values become equal and every other x_0 increases either one.

We next observe that, if the distribution of angles between straight-line edges in a given drawing is well-behaved, the maximum absolute angular difference will be small.

Corollary 1. *In a minimum rotation of a vertex v with degree at least two, the maximum angular difference is* $\max_i |x_i| = \max_{r<s} \frac{1}{2} \cdot \left|(s-r)\frac{2\pi}{d_G(v)} - \sum_{j=r}^{s-1} \alpha_j\right| < \pi$.

Proof. The proof of Theorem 1 showed that $\max_i |x_i| = |x_0 + \min_i y_i| = \frac{1}{2} \cdot |min_i y_i - \max_i y_i| = \frac{1}{2} \cdot |max_i y_i - \min_i y_i|$. It follows that there are $r, s \in \{0, \ldots, d_G(v) - 1\}$ such that $\max_i |x_i| = \frac{1}{2}|y_s - y_r|$ and $r < s$, and there is no pair with a larger absolute difference. Since $y_s - y_r = (s-r)\frac{2\pi}{d_G(v)} - \sum_{j=r}^{s-1} \alpha_j$, the equality holds. Clearly, $0 < (s-r)\frac{2\pi}{d} < 2\pi$ and $0 \leq \sum_{j=r}^{s-1} \alpha_j \leq 2\pi$ for all pairs $r < s \in \{0, \ldots, d_G(v) - 1\}$.

Essentially, the corollary states that the angular differences are half as bad as the given distortion in the straight-line edges. Recall that angular differences less than or equal to $\frac{\pi}{2}$ exclude the high-curvature cases of Fig. 1.

The corollary also indicates that the minimum rotation may be dominated by one pair of straight-line edges. An alternative objective is therefore to minimize the sum of squared angular differences,

$$\sum_{i=0}^{d_G(v)-1} (x_i)^2, \tag{2}$$

weighing contributions more evenly. Such rotations are called *balanced*.

Theorem 2. *The balanced rotation is unique, and linear-time computable.*

Proof. Substituting for x_i reduces objective function (2) to a positive function in x_0. Thus, any x_0 satisfying $\frac{\partial}{\partial x_0} \sum_{i=0}^{d_G(v)-1} (x_0 + y_i)^2 = \sum_{i=0}^{d_G(v)-1} 2 \cdot (x_0 + y_i) = 0$ yields a minimum. Clearly, there is exactly one such x_0, and it is obtained by averaging over all offsets, i.e.

$$x_0 = \frac{-\sum_{i=0}^{d_G(v)-1} y_i}{d_G(v)} = \frac{1}{d_G(v)} \sum_{i=0}^{d_G(v)-1} \left(\sum_{j=0}^{i-1} \alpha_j - i \cdot \frac{2\pi}{d_G(v)} \right).$$

An immediate consequence of the proof explains the name of this rotation.

Corollary 2. *In a balanced rotation,* $\sum_{i=0}^{d_G(v)-1} x_i = 0$.

See Fig. 3 for a comparison of minimum and balanced rotation. For completeness we note that the seemingly less interesting *absolute* rotation, suggested by this corollary, minimizing $\sum_{i=0}^{d_G(v)-1} |x_i|$ is also unique, and that all angular differences for vertices with already optimal local angular resolution equal zero in any of these rotations.

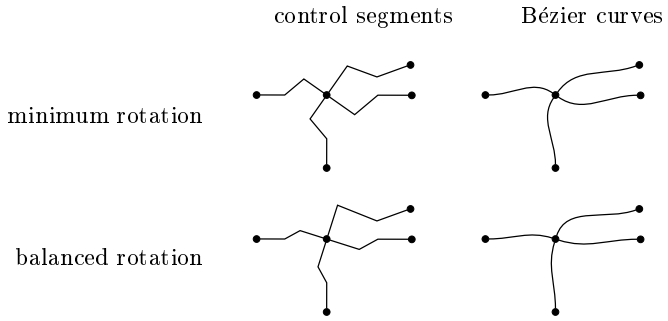


Fig. 3. Comparison of minimum and balanced rotation

4 Extensions

The following are some of several possible extensions that may be useful when a particular application yields additional requirements on the quality of a drawing.

Arbitrary angle constraints. From Corollary 1 we see that angular differences can grow quite big, when the given straight-line drawing has large angles, which typically results in edges with high curvature. In such situations it may be advantageous not to optimize local angular resolution exactly, but only up to a constant. For example, if there is an angle of at least π, all angles between control segments can be set to $\frac{\pi}{d_G(v)-1}$, except for one which is fixed to π. Note that we can impose any constraint on the angles between control segments, provided they add up to 2π, without affecting Theorems 1 and 2. Figure 4 shows some potentially useful templates for angles between control segments. The middle one is applied to vertices on the convex hull in Fig. 5.

Weighted angular differences. In an application with different types of edges, angular differences of some edges may be more important than those of others. If this importance can be quantified, the objective functions for rotations can be modified to accommodate weights, and analog theorems hold. Weights may be particularly useful when the straight-line edges incident to a vertex are of significantly different length.

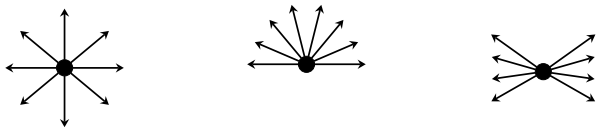


Fig. 4. Example templates for constraints on control segment angles

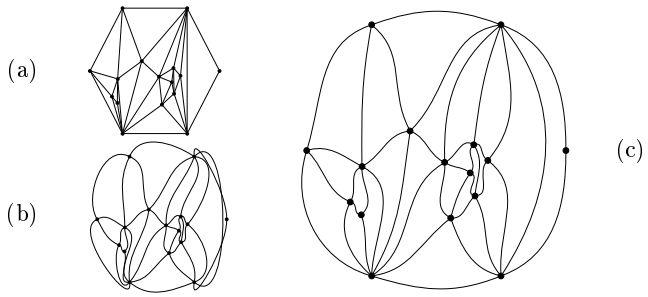


Fig. 5. A straight-line triangulation (*a*) and curved edges in a balanced rotation (*b*). Using the half-sided templates of Fig. 4 for vertices on the convex hull significantly improves the drawing (*c*)

Cyclic ordering. The order of control segments of a vertex need not the same as for the straight-line edges. Figure 6 depicts a situation in which a different ordering would be better. Allowing negative angles α_i, control segments can be put in any order.

Crossings. While there is no control over whether curved edges cross each other when a rotation is determined, crossings already existing in the straight-line drawing can be maintained by replacing them with a dummy vertex. Since angle constraints are arbitrary, we can thus constrain curves to cross, e.g., at a right angle or at the same angle as in the given drawing.

5 Application Examples

5.1 Train Connections

To analyze time table data of a number of European public transport systems (mostly trains), a graph is constructed in the following way [6]: Each station is represented by a vertex, and there is an edge between two vertices if there is at least one non-stop service between them at any time. This graph is analyzed, e.g., with respect to completeness, consistency, or changes between schedules, serving, e.g., quality control, international coordination, and pricing.

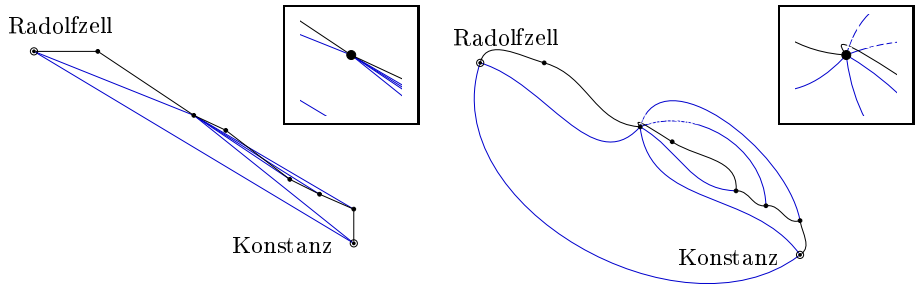
Each vertex has a geographic location, thus providing geographical context for visual data exploration. Part of the analysis is an automatic classification

into *minimal* and *transitive* edges, corresponding to tracks and high-speed connections passing by minor stations, respectively. By their very nature, many edges run almost in parallel when drawn as straight-lines. In particular, transitive edges overlap each other, and minimal edges (see Fig. 6(a)). Figure 6(b) shows a balanced rotation with the special half-sided template for vertices with an angle of at least π. The unnecessary detour of one minimal edge is due to the unfortunate ordering of straight-line edges. The system we are building will order edges based on their length and classification.

An elaborate algorithm for application-specific curved layout of transitive edges is presented in [6]. Users are very satisfied with the output, but running times are painful (7–45 minutes for the networks compared in Tab. 1). Using the approach described in this paper, we can easily afford to compute a new layout every time a network is displayed and thus support interactive querying. In fact, optimal rotations are computed faster than the editor (LEDA's [19] GraphWin class) renders the network. See also the larger example in Fig. 7.

instance	nodes	edges	minimum	balanced
`switzerland`	2218	3203	0.36 sec	0.36 sec
`italy`	2386	4370	0.51 sec	0.51 sec
`france`	4551	7793	0.81 sec	0.80 sec
`germany`	7083	9713	1.19 sec	1.18 sec

Table 1. Computation times for rotations (Sun UltraSparc, 440 Mhz, 256 MBytes)



(a) straight lines (b) balanced rotation

Fig. 6. Small network in southern Germany with one undesirable edge ordering

5.2 Internet Multicast Backbone

Internet connections represent another example of an organically grown, and growing, geographic network. To support their analysis, which is crucial for

maintenance and development, advanced systems such as [7, 20] provide effective interactive visualizations of network topologies. The environment of [20] is publicly available,[1] and generates a geometric scene description that can be explored in an interactive browser such as Geomview [2]. We replaced the generation of geometric output to demonstrate how to improve the angular resolution, and thus the perceptual quality, of the resulting visualizations.

The original system visualizes the topology of the MBone, the Internet's multicast backbone, by representing it as a geographic network on the globe, where connections (MBone tunnels) are shown as great circle segments elevated into space (Fig. 8(a)). The angles between great circles correspond to the angles between their projections into the plane tangent to their intersection. Our approach of orienting control segments is therefore easily generalized to deal with great circles on a sphere rather than lines in the plane. Once the inner control points of a curved edge are placed on the sphere, the actual curve is determined as follows. Assume the globe is represented as a sphere with unit radius, and $B(t) : [0, 1] \to \mathbb{R}^3$ is a cubic Bézier curve in space connecting points b_0 and b_3. The curve is projected onto the sphere and elevated into space according to

$$\frac{B(t)}{\|B(t)\|} \cdot (1 + c \cdot \arccos(b_0 \cdot b_3) \cdot \sin(t\pi)).$$

where c is an elevation constant. The resulting curves have essentially the same height as those in [7]. See Fig. 8(b).

6 Conclusions

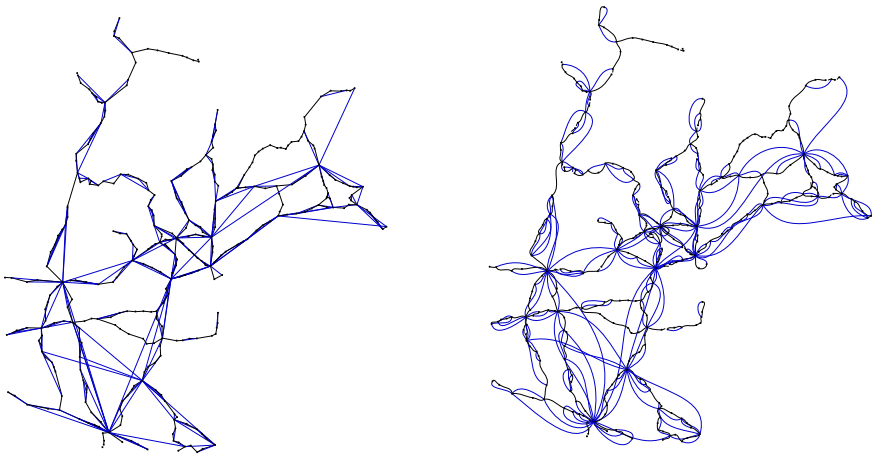
We presented a simple and extremely fast approach to improve the perceptual quality of visualizations of networks with underlying geography. The approach is novel in that it focuses on angular resolution rather than vertex positioning, and we expect many potential applications other than the two described.

There are, however, several interesting topics that need further research. First of all, our current formulation does not allow to control the introduction of new edge crossings. It would be nice to guarantee that an improved drawing has the same number of crossings as the input drawing, especially if the latter has no crossings at all. The small time table data example already showed that our approach is sensitive to the ordering of edges around a vertex. Is there a generally applicable rule to determine an ordering other than the one in the straight-line drawing? Since rotations are determined independently for each vertex, an optimal rotation may introduce unnecessarily many S-shaped curves, whereas a different rotation may be more pleasing. Can we introduce some form of dependency to make angular differences at both ends of an edge similar? Finally, we would like to improve the heuristic used to determine the length of a control segment. It may be advantageous to have control segments of similar length at a vertex, or at both ends of an edge.

[1] `http://oceana.nlanr.net/PlanetMulticast/`

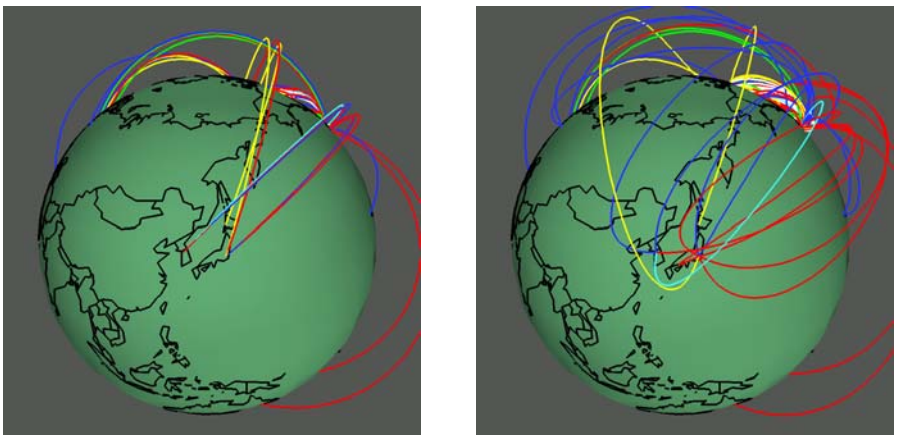
References

1. J. Abello and E. R. Gansner. Short and smooth polygonal paths. *Proc. 3rd Latin American Symp. Theoretical Informatics (LATIN '98)*, LNCS vol. 1380, pp. 151–162. Springer, 1998.
2. N. Amenta, S. Levy, T. Munzner, and M. Phillips. Geomview: A system for geometric visualization. In *Proc. 11th ACM Ann. Symp. Computational Geometry (SoCG '95)*, pp. C12–13. ACM, 1995.
3. R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Graphics*, 1(1):16–28, 1995.
4. P. Bézier. *Numerical Control*. John Wiley & Sons, 1972.
5. W. Boehm. On cubics: A survey. *Computer Graphics and Image Processing*, 19:201–226, 1982.
6. U. Brandes and D. Wagner. Using graph layout to visualize train interconnection data. *Proc. 6th Intl. Symp. Graph Drawing (GD '98)*, LNCS vol. 1547, pp. 44–56. Springer, 1998.
7. K. C. Cox, S. G. Eick, and T. He. 3D geographic network displays. *ACM SIGMOD Record*, 24(4):50–54, 1996.
8. D. P. Dobkin, E. R. Gansner, E. Koutsofios, and S. C. North. Implementing a general-purpose edge router. *Proc. 5th Intl. Symp. Graph Drawing (GD '97)*, LNCS vol. 1353, pp. 262–271. Springer, 1997.
9. M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F. T. Leighton, A. Symvonis, E. Welzl, and G. Woeginger. Drawing graphs in the plane with high resolution. *SIAM Journal on Computing*, 22(5):1035–1052, 1993.
10. E. R. Gansner and S. C. North. Improved force-directed layouts. *Proc. 6th Intl. Symp. Graph Drawing (GD '98)*, LNCS vol. 1547, pp. 364–373. Springer, 1998.
11. A. Garg. On drawing angle graphs. *Proc. 2nd Intl. Symp. Graph Drawing (GD '94)*, LNCS vol. 894, pp. 84–95. Springer, 1995.
12. A. Garg and R. Tamassia. Planar drawings and angular resolution: Algorithms and bounds. *Proc. 2nd European Symp. Algorithms (ESA '94)*, LNCS vol. 855, pp. 12–23. Springer, 1994.
13. M. T. Goodrich and C. G. Wagner. A framework for drawing planar graphs with curves and polylines. *Proc. 6th Intl. Symp. Graph Drawing (GD '98)*, LNCS vol. 1547, pp. 153–166. Springer, 1998.
14. C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. *Proc. 6th Intl. Symp. Graph Drawing (GD '98)*, LNCS vol. 1547, pp. 167–182. Springer, 1998.
15. H. Hagen. Bézier-curves with curvature and torsion continuity. *Rocky Mountain Journal of Mathematics*, 16(3):629–638, 1986.
16. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.
17. K. A. Lyons, H. Meijer, and D. Rappaport. Algorithms for cluster busting in anchored graph drawing. *Journal on Graph Algorithms and Applications*, 2(1):1–24, 1998.
18. S. Malitz and A. Papakostas. On the angular resolution of planar graphs. *SIAM Journal on Discrete Mathematics*, 7(2):172–183, 1994.
19. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. `http://www.mpi-sb.mpg.de/LEDA/`.
20. T. Munzner, E. Hoffman, K. Claffy, and B. Fenner. Visualizing the global topology of the MBone. *Proc. 1996 IEEE Symp. Information Visualization*, pp. 85–92, 1996.



(a) straight lines (b) balanced rotation

Fig. 7. Medium size time table graph: surroundings of Venice



(a) elevated great circles (b) balanced rotation

Fig. 8. Part of the Internet's multicast backbone: Korea/Japan