Topology-Driven Force-Directed Algorithms*

Walter Didimo, Giuseppe Liotta, and Salvatore A. Romeo

Università di Perugia, Italy
{didimo,liotta,romeo}@diei.unipg.it

Abstract. This paper studies the problem of designing graph drawing algorithms that guarantee good trade-offs in terms of number of edge crossings, crossing angle resolution, and geodesic edge tendency. It describes two heuristics designed within the *topology-driven force-directed* framework that combines two classical graph drawing approaches: the force-directed approach and a planarization-based approach (e.g., the topology-shape-metrics approach). An extensive experimental analysis on two different test suites of graphs shows the effectiveness of the proposed solutions for the optimization of some readability metrics.

1 Introduction and Overview

Several empirical studies compare different aesthetic criteria for drawing graphs and show that the number of edges crossings often has the strongest impact on the readability of a diagram (see, e.g., [21,22]). As a consequence, it has been widely accepted that one of the primary optimization tasks of a good graph drawing algorithm is the minimization of crossings and a large body of literature has been devoted to this topic (see, e.g., [4,20]).

Cognitive experiments by Huang, Huang *et al.*, and Ware *et al.* provide new insights into the classical correlation between edge crossings and human understanding of graph drawings [18,19,24]. As these experiments show, the readability of a diagram not only depends on the edge crossings (which are unavoidable for dense non-planar graphs) but also on the "quality" of these crossings and on the "quality" of the curves that cross with each other. The experiments suggest that the query "Are two vertices adjacent in the drawing?" is easier to answer when the minimum angle formed by the crossing edges is as large as possible and when the curves that represent the edges are as straight as possible. The task of easily following the drawing of an edge from its source to its destination is also listed in the *NetViz Nirvana* of a recent study by Dunne and Shneiderman about HCI design principles for visual analytics [7].

This paper studies the problem of designing graph drawing algorithms that guarantee good trade-offs in terms of number of edge crossings, *crossing angle resolution* and *geodesic edge tendency*. The crossing angle resolution measures the value of the minimum angle formed by two crossing edges. The geodesic edge tendency measures how close a bent edge is to the straight-line segment connecting the end-points of the edge. To this aim, we adopt a graph drawing methodology, which we call *topology-driven force-directed* approach, that combines two classical graph drawing algorithmic frameworks: the *force-directed* approach and *planarization-based* approach.

* Work supported in part by the MIUR project AlgoDEEP prot. 2008TFBWL4.

U. Brandes and S. Cornelsen (Eds.): GD 2010, LNCS 6502, pp. 165–176, 2011.

The force-directed approach associates the input graph with a physical system of forces and it tries to place the vertices so that the total energy of the physical system is minimal. Only a few examples of force directed algorithms use edges with bends (see, e.g., [5,10,12]); the vast majority of the force-directed algorithms represent the edges as straight-line segments and thus the computed drawings are optimal in terms of geodesic edge tendency. Unfortunately, force directed algorithms may introduce unnecessarily many crossings. For example, the drawing of Fig. 1(a) computed with a force-directed algorithm of the OGDF Library[1] contains edge crossings even though the input graph is planar.

A planarization-based algorithm first computes a topology (i.e., an embedding) of the graph with a small number of edge crossings and then it applies a drawing algorithm that preserves this topology. Since the crossing minimization problem is NP-hard, a planarization-based algorithm typically relies on heuristics such as first computing a maximal planar subgraph and then inserting an edge per time (see, e.g., [4,15]). Each unavoidable edge crossing is replaced by a dummy vertex and a planar embedding of an augmented planar graph is obtained. The drawing algorithm typically preserves this computed topology by introducing bends along the edges; for example, the well-known *topology-shape-metrics* algorithm by Tamassia [23] is a planarization-based algorithm that computes orthogonal drawings. Planarization-based algorithms are likely to compute drawings with smaller number of crossings than force-directed algorithms and often with a better crossing angle resolution. However, the bends along the edges may strongly affect the geodesic edge tendency. See for example the orthogonal drawing of Fig. 1(b). This drawing, computed with the GDToolkit Library[2], depicts the same graph of Fig. 1(a). Note that, understanding whether the two black vertices are adjacent is more complicated in the orthogonal drawing because their connecting edge is far from being geodesic.

The main results in this paper are as follows:

- We describe ORTHFD and POLYFD, two graph drawing algorithms that were designed within the topology-driven force-directed framework (see Section 3). For example, Fig. 1(c) shows a drawing computed by algorithm ORTHFD. Observe that compared to Fig. 1(a), the drawing does not have crossings; compared to Fig. 1(b) the edge between the black vertices is easier to follow.
- We perform an experimental study that compares our algorithms with respect to some of the most effective force-directed and planarization-based algorithms described in the literature. The results show that the drawings computed with the topology-driven force-directed approach achieve better trade-offs in terms of number of edge crossings, crossing angle resolution, and geodesic edge tendency. They often provide good results with respect to some other important aesthetic criteria as well, such as number of bends and vertex angle resolution (see Section 4).

Throughout this paper we assume that the reader is familiar with the basic concepts of force-directed and planarization-based techniques (see e.g. [4,20]).

[1] http://www.ogdf.net/
[2] http://www.dia.uniroma3.it/∼gdt

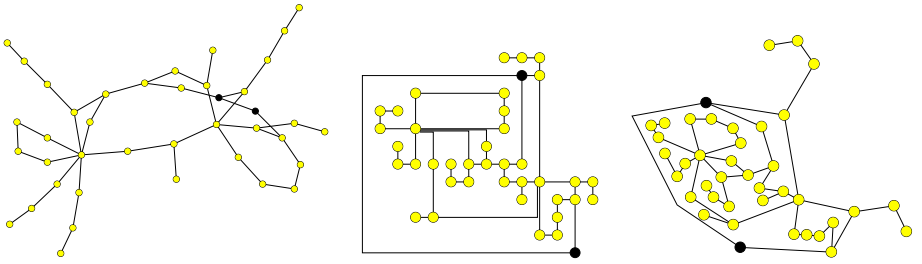


Fig. 1. Drawings of the same graph with different approaches: (a) A force-directed approach; (b) A planarization-based approach; (c) A topology-driven force-directed approach

2 Related Work

There are many examples in the literature describing force-directed algorithms that preserve a given topology (i.e. a given embedding). While we refer the interested reader to the accurate surveys in [8,9,10] for exhaustive lists of references, we mention here only those results that can be more closely related to the approach described in this paper.

Bertault describes a force-directed algorithm, called PRED, that preserves edge crossing properties [2]. Our work and the one of Bertault have some similarities but also substantial differences. Similar to PRED, our algorithmic framework uses force-directed techniques that do not increase the number of crossings with respect to an initial drawing of the graph. Differently from PRED, our work: (i) Combines crossing minimization heuristics and force-directed methods; (ii) Guarantees good crossing angle resolution by allowing bent edges; (iii) Enhances the force-directed model with a set of constraints that guarantees a good geodesic edge tendency in spite of the fact that the edges can bend.

Also, Dwyer *et al.* [9,10] have extensively studied sophisticated stress-optimization techniques that maintain a set of constraints, including a given topology. Similar to the algorithms in those papers, we allow bends along the edges. Differently from those papers, our primary goal is to find good trade-offs between number of edge crossings, crossing angle resolution, and geodesic edge tendency, while we do not focus on the stability problem in dynamic graph layout. This difference impacts on the developed techniques and on the computed drawings. For example, we do not consider the continuous network layout problem and we do not insist on preserving an initial mental map; in fact, our techniques may even change the initial topology if this change reduces the total number of edge crossings.

3 The Topology-Driven Force-Directed Approach

Let G be a graph. The *topology-driven force-directed* approach computes a drawing Γ of G into two main phases. Phase 1: A drawing Γ_0 of G is computed by applying a planarization-based algorithm. Phase 2: The final drawing Γ is obtained by applying a force-directed algorithm to Γ_0, with the constraint that Γ does not have more edge crossings than Γ_0.

Within this general approach, one can design several algorithms by combining different strategies for the two phases. In this work we describe two specific algorithms that we call ORTHFD and POLYFD, respectively. Both of these algorithms adopt a similar planarization step, that computes a planar embedding of G where crossings are replaced with dummy vertices, called *cross vertices*. In order to have a small number of edge crossings the planarization step uses a classical heuristic based on finding a shortest path in the dual graph for each edge insertion (see, e.g., [4]). Algorithms ORTHFD and POLYFD then construct a drawing Γ_0 of G that preserves the computed planar embedding. Algorithm ORTHFD uses the *topology-shape-metrics* approach for orthogonal drawings in the Kandinsky drawing convention [13]. According to this convention, the vertices are represented as squares of the same dimension and there may be parallel edges incident to vertices whose degree is larger than four. The orthogonal drawing is computed by the algorithm described in [3], which minimizes the total number of bends along the edges in $O(n^{\frac{7}{4}} \log n)$ time if the network flow algorithm of Garg and Tamassia is used [14]. We use the $O(n^2 \log n)$ algorithm described in [3].

Algorithm POLYFD computes a polyline drawing Γ_0 by applying the algorithm described in [6]. Γ_0 is constructed by finding a suitable orientation of the edges of the graph and by using this orientation to define a visibility representation of the graph. The polyline drawing is obtained collapsing the vertices of the visibility representation into points (see, e.g., [4]).

Phase 2 of ORTHFD and of POLYFD uses a common force-directed strategy, that works in different steps:

- **Step 1:** Every bend of Γ_0 is replaced with a dummy vertex, called a *bend vertex*. After this step all the edges are straight-line segments. Call Γ_1 the new drawing.
- **Step 2:** A force directed algorithm is applied, starting with Γ_1 as the initial drawing where the general physical model is that described in [11]: Each vertex is modeled as an electrically charged particle and each edge as a spring. The phisical model is augmented by additional constraints as described below:

  - Let $e = (u, v)$ be an edge of G that has k bends in Γ_0 and let $e_1, e_2, \ldots, e_{k+1}$ be the edges of Γ_1 forming e. Note that, if $k = 0$ then e_1 coincides with e; in this case we call e_1 a *straight edge*. With respect to Γ_1, denote by $\ell(e_i)$ the length of e_i, by $\ell(e)$ the sum of all $\ell(e_i)$, and by $d(u, v)$ the Euclidean distance between u and v. The spring corresponding to e_i is given a zero-energy length equal to $d(u, v)\frac{\ell(e_i)}{\ell(e)}$. Also, the stiffness of the springs that model the straight edges is smaller than the stiffness of the springs associated with the other edges. This is done to ensure that each edge approximates the straight line between its end-vertices, in order to obtain a good geodesic edge tendency.
  - Let $\overrightarrow{f(v)}$ be the resultant of all forces acting on v, and let d be the maximum distance by which v can be moved along the direction of $\overrightarrow{f(v)}$ without creating any new edge crossings. Vertex v is moved along the direction of $\overrightarrow{f(v)}$ by a quantity δ such that $\delta \leq \min\{d, |\overrightarrow{f(v)}|\}$. More precisely, if moving v by a quantity $|\overrightarrow{f(v)}|$ does not create crossings, then we set $\delta = |\overrightarrow{f(v)}|$, otherwise we compute δ by applying a binary search along the direction of $\overrightarrow{f(v)}$; we stop

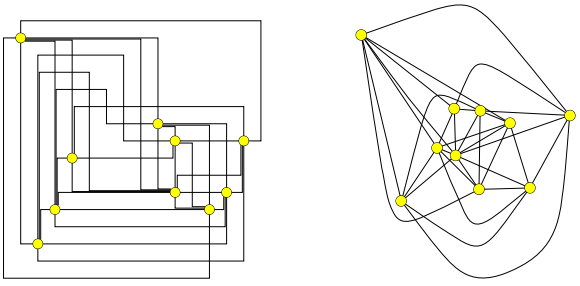


Fig. 2. Two drawings of the same graph computed with different algorithms. The drawing computed by ORTHFD, in Fig. 2(b), has six fewer crossings than the one computed by a planarization-based orthogonal drawer of `GDToolkit`, in Fig. 2(a). The edges in ORTHFD are represented by smoothed curves.

this search as soon as a placement for v is found that does not increase the number of edge crossings. Note that this procedure may lead to a reduction of the number of crossings computed by the planarization step. For example, the drawing computed by ORTHFD in Fig. 2(b) has six fewer crossings than the one in Fig. 2(a), computed by using a planarization-based orthogonal drawer of `GDToolkit`.

- **Step 3:** This is a post-processing step whose goal is to improve the crossing angle resolution. Let Γ_2 be the drawing at the end of Step 2 and let α be a target value for the minimum angle formed by two crossing edges. Let e_1, e_2 be two crossing edges of Γ_2 that form an angle smaller than α. In order to enlarge the crossing angle formed by $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ we adopt a technique similar to that described in [5]. Denote by c the crossing point of e_1, e_2. Define a disk δ centered at c such that δ does not intersect in Γ_2 any edge other than e_1 and e_2. Let p_i, q_i be the intersection points between δ and e_i ($i \in \{1, 2\}$). Edge e_i is split into the path $(u_i, p_i), (p_i, q_i), (q_i, v_i)$. Also, the straight-line edges $(p_1, p_2), (p_2, q_1), (q_1, q_2), (q_2, p_1)$ are added to the drawing. The four-cycle formed by these dummy edges is called the *cage* of crossing c. The zero-energy length of all edges forming the cages is set to be a constant smaller than the zero-energy length of the shortest edge of Γ_2; the stiffness of the edges in the cages is larger than the one of the other edges. Let Γ_2' be the drawing obtained from Γ_2' by inserting all cages. By applying a few iterations of the force-directed method to Γ_2', we obtain a new drawing where the cages are drawn as close as possible to squares, which enforces their diagonals to cross at large angles.
- **Step 4:** Let Γ_3 be the drawing at the end of Step 3. The algorithm iteratively removes unnecessary bends along the edges. A bend is unnecessary if its removal: (i) does not introduce new crossings; (ii) does not make the crossing angle resolution lower than the given threshold α.

4 Experimental Study

We tested algorithms OrthFD and PolyFD on two different test suites of graphs. The first test suite, called `Rome`, consists of 300 graphs randomly selected in the popular collection known as the "Rome graphs" [1]; we selected 30 graphs for each fixed number of vertices $n \in \{10, 20, \ldots, 100\}$. The graphs in this test suite are typically sparse (the average density of the Rome graphs is about 1.4) and they reflect the structure of graphs coming from real life applications in the field of databases.

The second test suite, called `Rand` consists of 300 randomly generated graphs, that are denser than the "Rome graphs". They have number of vertices $n \in \{10, 20, \ldots, 50\}$ and density $d \in [1.5 - 4.0]$. For each pair $\langle n, d \rangle$, a graph with n vertices and $m = d \cdot n$ edges was generated with a uniform probability distribution; we generated 60 graphs for each distinct value of n.

We initially compared the drawings computed by OrthFD and PolyFD with three effective force-directed and planarization-based algorithms, namely FM^3, Orth, and Poly. Algorithm FM^3 is the fast force-directed algorithm described by Hachul and Jünger [16]. Among the wide set of force-directed algorithms, we chose FM^3 because an experimental study showed that it typically computes drawings with smaller number of crossings and edge overlaps than other force-directed methods [17]. We used the implementation of FM^3 available in the open source library `OGDF`. Orth and Poly are the algorithms that compute an orthogonal drawing and a polyline drawing in Phase 1 of OrthFD and PolyFD, respectively. We used the implementation of these algorithms available in the `GDToolkit` Library; the algorithms Orth and Poly perform well in terms of number of bends and number of edge crossings [4]. However, after a few experimental measures we realized that Orth outperformed Poly in all aesthetics. Also Poly was often giving rise to drawings with overlapping edge bends, which was making the final drawings rather confusing. For example, Fig. 3 shows five drawings of the same graph, each computed by one of the above algorithms. It is immediate to see that the drawing of Fig. 3(d) is less readable than the others when trying to answer basic queries, such as identifying the neighbors of the black vertex (white vertices). Therefore, we decided to restrict the experimental comparison to algorithms Orth, FM^3, OrthFD, and PolyFD. We remark that the problem of overalpping edge bends does not occur with algorithm PolyFD, where the repulsive forces acting between pairs of vertices do not allow two bends to be drawn at the same point.

In the experimental setting of algorithms OrthFD and PolyFD we set the minimum threshold α for the crossing angle resolution to be $\alpha = 30°$. This is motivated by the work of Ware et *al.* [24], who observe that crossing angles smaller than 30° are more likely to cause visual confusion when reading a drawing of a graph.

In the following we present the charts summarizing our experimental results. In all charts we use bars with the following colors for the different algorithms: (a) FM^3 - White color; (b) Orth - Light gray color; (c) OrthFD - Dark gray color; PolyFD - Black color.

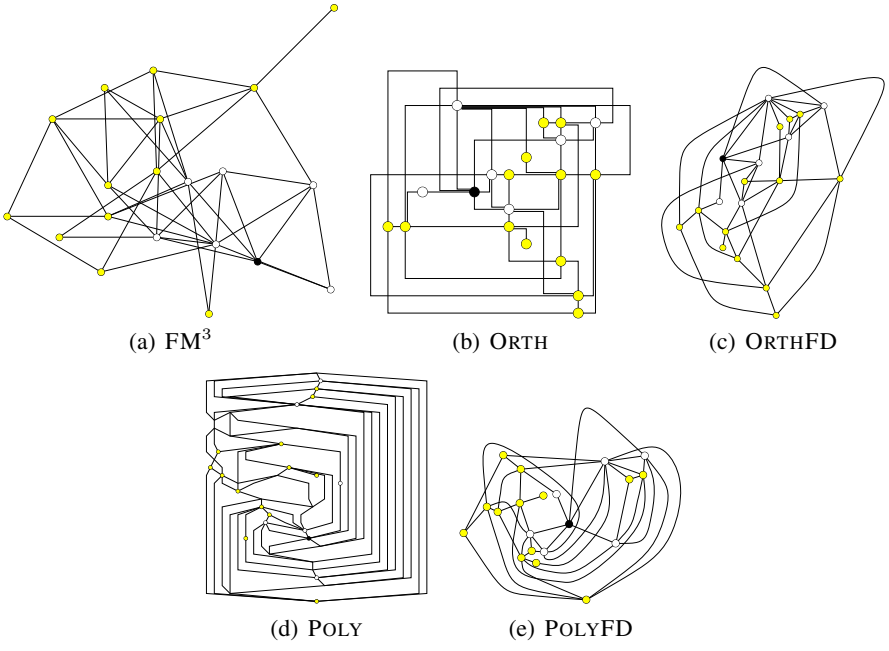


Fig. 3. Five drawings of the same graph computed with different algorithms

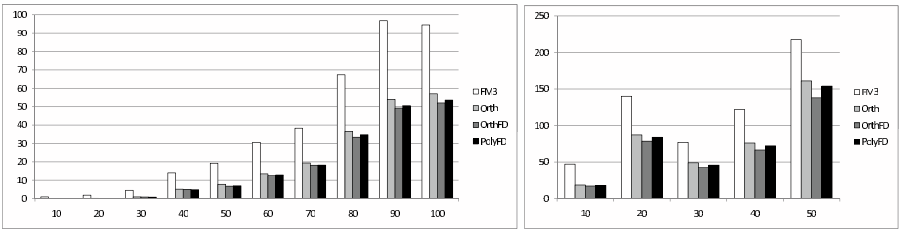
Number of Edge Crossings

Fig. 4(a) and Fig. 4(b) show that the number of edge crossings of ORTH, ORTHFD, and POLYFD is, on average, half that computed by FM^3. This behavior is a consequence of the impact of the planarization heuristic used in ORTH, ORTHFD, and POLYFD. The experiments also confirmed the observation made in the previous section that Phase 2 of the topology-driven force-directed framework can further reduce the number of edge crossings of a planarized drawing. In particular, the drawings computed by ORTHFD contain on average 10% less edge crossings than ORTH. We finally remark that some of the drawings computed by FM^3 presented edge overlaps (37 in the `Rome` test suite and 62 in the `Rand` test suite). The other algorithms never created edge overlaps.

For example, Fig. 3(a) has 59 edge crossings, Fig. 3(b) and Fig. 3(e) have 17 edge crossings, while Fig. 3(c) has 16 edge crossings.

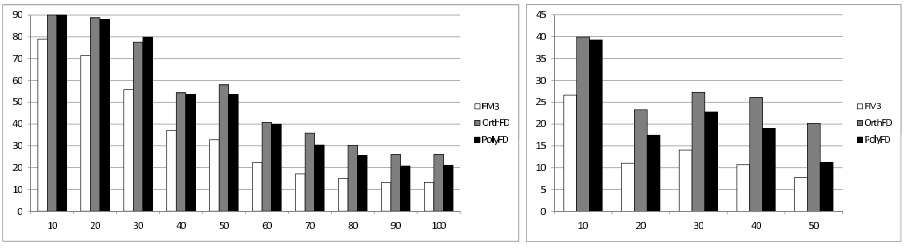
Crossing Angle Resolution

Fig. 5(a) and Fig. 5(b) show the crossing angle resolution in the drawings computed by FM^3, ORTHFD, and POLYFD, that is, the value of the minimum angle formed by any two crossing edges. Clearly, the crossing angle resolution of the drawings computed by ORTH is always optimal (i.e., $90°$) and hence we do not report the data for this algorithm. If a drawing is planar, it was assigned an optimal crossing angle resolution. From the charts it can be seen that both ORTHFD and POLYFD outperform FM^3.



(a) Number of edge crossings - `Rome` (b) Number of edge crossings - `Rand`

Fig. 4. Number of edge crossings for the two test suites of graphs (average values over the number of vertices)



(a) Crossing angle resolution - `Rome` (b) Crossing angle resolution - `Rand`

Fig. 5. Crossing angle resolution for the two test suites (average values over the number of vertices)

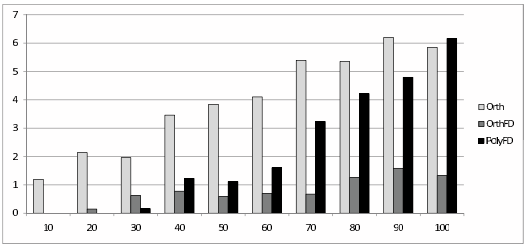
In particular, the average crossing angle resolution of the drawings computed by ORTHFD is always above $30°$ for number of vertices up to 80 in the `Rome` test suite. Also, the crossing angle resolution of the drawings computed by FM^3 on the `Rand` graphs is often very poor (around $10°$), while it is maintained always above $20°$ by ORTHFD.

Also, we observed that the percentage of edge crossings with an angle smaller than $30°$ is about 6% on average for the drawings computed by FM^3, while it is about 2% for the drawings computed by algorithms ORTHFD and POLYFD. This data is particularly relevant if considered together with the observation that the number of edge crossings created by FM^3 is typically much higher than the one created by ORTHFD and POLYFD.

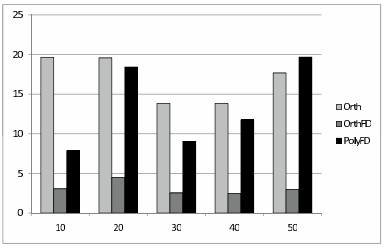
For example, Fig. 3(a) has two crossings that form an angle less than $30°$, all edge crossings in Fig. 3(c) form angles larger than $30°$, and Fig. 3(e) has one edge crossing that forms an angle less than $30°$.

Geodesic Edge Tendency

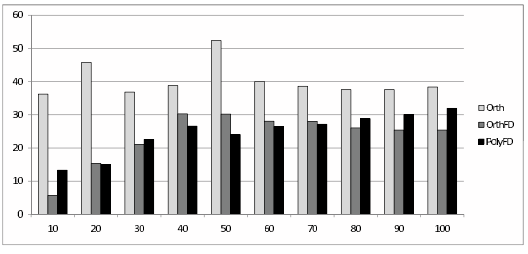
Clearly, the drawings computed by FM^3 are optimal in terms of geodesic edge tendency, because they have straight-line edges only. To measure the geodesic edge tendency of the drawings computed by algorithms ORTH, ORTHFD, and POLYFD we considered two different parameters: (i) The percentage of edges (u, v) that are not monotone in



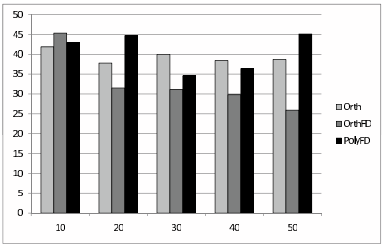
(a) Percentage of non-monotone edges - Rome



(b) Percentage of non-monotone edges - Rand



(c) Ratio (as percentage) between max. dist. from (u, v) to $\overline{uv}$ and $|\overline{uv}|$ - Rome



(d) Ratio (as percentage) between max. dist. from (u, v) to $\overline{uv}$ and $|\overline{uv}|$ - Rand

Fig. 6. Geodesic edge tendency (average values over the number of vertices)

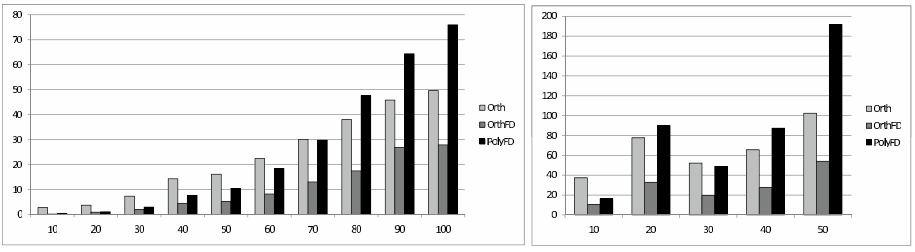
the direction of the straight segment $\overline{uv}$; small percentages of such edges indicate good geodesic edge tendency. (ii) The ratio between the maximum distance from an edge (u, v) to the segment $\overline{uv}$ and the length of $\overline{uv}$. Small values indicate good geodesic edge tendency.

Fig. 6(a) and Fig. 6(b) show that the number of non-monotone edges in the drawings computed by ORTHFD is between than $2 - 4\%$, and dramatically improves the values for the drawings computed by ORTH, namely by 82% on average. Also POLYFD gives some improvement with respect to ORTH for almost all instances, but this improvement is not so relevant as for ORTHFD. For example, Fig. 3(b) has six non-monotone edges, all edge in Fig. 3(c) are monotone, and Fig. 3(e) has four non-monotone edges.

Concerning the second parameter used to measure the geodesic edge tendency, Fig. 6(c) and Fig. 6(d) show the ratio (expressed as percentage) between the maximum distance from an edge (u, v) to the segment $\overline{uv}$ and the length of $\overline{uv}$. It can be observed that for the Rome graphs both the two topology-driven force-directed algorithms significantly improve the results of algorithm ORTH. More precisely, this percentage is on average 40% for ORTH, 23% for ORTHFD, and 24% for POLYFD. For the Rand graphs Orth and PolyFD perform similarly (about 40% on average), while OrthFD perform better for almost all instances (about 32% on average).

Bends and Vertex Angles

From the experiments we also observed that ORTHFD works well also in terms of number of bends and vertex angle resolution. Namely, Fig. 7 shows that the ORTHFD



(a) Total number of bends - Rome (b) Total number of bends - Rand

Fig. 7. Total number of bends for the two test suites (average values over the number of vertices)

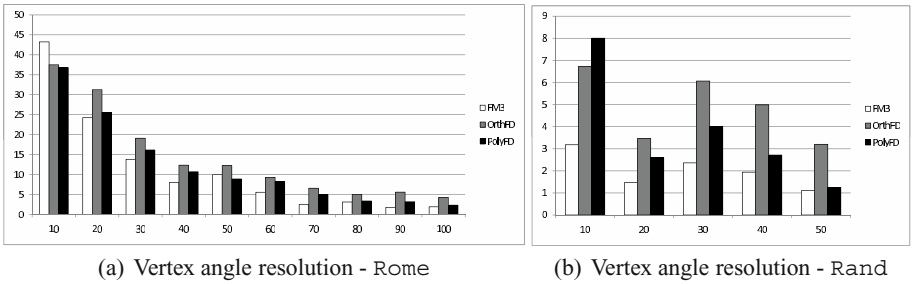
reduces the total number of bends of ORTH by 60% on average, both for the Rome and for the Rand graphs. On the contrary, POLYFD usually performs worse than the other two algorithms, because the drawings computed in Phase 1 by POLYFD contain many more bends than those computed by ORTH.

About the vertex angle resolution, we excluded from the comparison the drawings computed by Orth, where two edges incident to the same vertex may form an angle of zero degree (if they are parallel). Fig. 8 shows that ORTHFD and POLYFD perform better than FM^3 except for the sample of 10 vertices in the Rome graphs. In particular, the vertex angle resolution of the drawings computed by ORTHFD is on average 2.5 times higher than that of the drawings computed by ORTH on the Rand graphs.

For example, Fig. 3(b) has 38 edge bends, Fig Fig. 3(c) has 10 edge bends, and Fig. 3(e) has 34 edge bends. Also, Fig. 3(a) has vertex angle resolution equal to $0.64°$, Fig. 3(c) has vertex angle resolution equal to $11.2°$, and Fig. 3(e) has vertex angle resolution equal to $5.68°$.

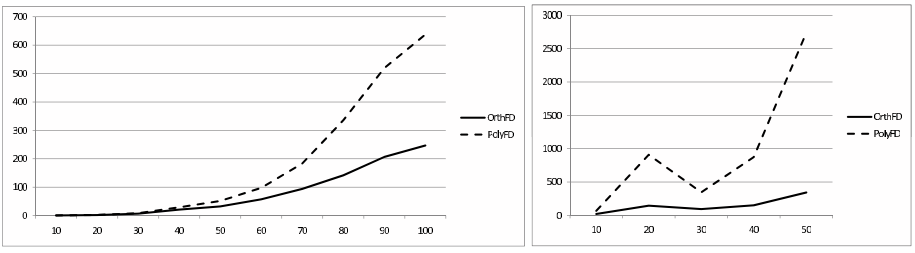
Running Time

We designed our algorithms to take into account several readability metrics, which typically conflict with each other. The good performances in terms of aesthetic criteria are



(a) Vertex angle resolution - Rome (b) Vertex angle resolution - Rand

Fig. 8. Vertex angle resolution (average values over the number of vertices)



(a) Running time - Rome (b) Running time - Rand

Fig. 9. Running time (in seconds) of algorithms ORTHFD and POLYFD (average values over the number of vertices)

however paid in terms of computational efficiency. A theoretical analysis shows that the algorithms require $O((n + b)(m \log m))$ time, where n, m, and b are the number of vertices, edges, and bends in the drawing. The time performances of ORTHFD and POLYFD are reported in Fig. 9. The algorithms were executed on a PC Intel Core Duo 2.66 GHz and 2GB RAM. From the charts, it is possible to see that the Rand graphs are computationally much harder than the Rome graphs, due to their higher density. However, the time required by ORTHFD is much smaller than that required by POLYFD.

5 Conclusions and Open Problems

In this paper we concentrated on the design of graph drawing algorithms that guarantee a good trade-off between number of edge crossings, crossing angle resolution, and geodesic edge tendency. We adopted the topology-driven force-directed framework and experimentally studied the performances of two algorithms, ORTHFD and POLYFD designed within this framework. The experimental analysis shows that algorithm ORTHFD has a better trade-off between number of edge crossings, crossing angle resolution, and geodesic edge tendency than existing force-directed and planarization-based algorithms. Also, it behaves rather well in terms of number of bends and in the vertex angle resolution. Furthermore ORTHFD generally outperformed POLYFD in all the experiments we have executed.

As our experiments show, algorithm ORTHFD can be reasonably applied to graphs having up to 100 vertices. It would be interesting to design graph drawing algorithms that perform equally well with respect to the aesthetic criteria taken into account in this paper and that can be applied efficiently to larger graphs. More experimental results that compare our technique with other force-directed algorithms are also of interest.

References

1. Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F.: An experimental comparison of four graph drawing algorithms. Comput. Geom. 7, 303–325 (1997)
2. Bertault, F.: A force-directed algorithm that preserves edge crossing properties. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 351–358. Springer, Heidelberg (1999)

3. Bertolazzi, P., Di Battista, G., Didimo, W.: Computing orthogonal drawings with the minimum number of bends. IEEE Trans. on Computers 49(8), 826–840 (2000)
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing. Prentice-Hall, Upper Saddle River (1999)
5. Didimo, W., Liotta, G., Romeo, S.A.: Graph visualization techniques for conceptual web site traffic analysis. In: PacificVis, pp. 193–200. IEEE, Los Alamitos (2010)
6. Didimo, W., Pizzonia, M.: Upward embeddings and orientations of undirected planar graphs. J. Graph Algorithms Appl. 7(2), 221–241 (2003)
7. Dunne, C., Shneiderman, B.: Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts. Technical report (2009)
8. Dwyer, T.: Scalable, versatile and simple constrained graph layout. Comput. Graph. Forum 28(3), 991–998 (2009)
9. Dwyer, T., Marriott, K., Schreiber, F., Stuckey, P.J., Woodward, M., Wybrow, M.: Exploration of networks using overview+detail with constraint-based cooperative layout. IEEE Trans. Vis. Comput. Graph. 14(6), 1293–1300 (2008)
10. Dwyer, T., Marriott, K., Wybrow, M.: Topology preserving constrained graph layout. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 230–241. Springer, Heidelberg (2009)
11. Eades, P.: A heuristic for graph drawing. Congressus Numerantium 42, 149–160 (1984)
12. 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)
13. Fößmeier, U., Kaufmann, M.: Drawing high degree graphs with low bend numbers. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996)
14. Garg, A., Tamassia, R.: A new minimum cost flow algorithm with applications to graph drawing. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 201–216. Springer, Heidelberg (1997)
15. Gutwenger, C., Mutzel, P., Weiskircher, R.: Inserting an edge into a planar graph. Algorithmica 41(4), 289–308 (2005)
16. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005)
17. Hachul, S., Jünger, M.: Large-graph layout algorithms at work: An experimental study. J. Graph Algorithms Appl. 11(2), 345–369 (2007)
18. Huang, W.: Using eye tracking to investigate graph layout effects. In: APVIS, pp. 97–100 (2007)
19. Huang, W., Hong, S.-H., Eades, P.: Effects of crossing angles. In: PacificVis, pp. 41–46. IEEE, Los Alamitos (2008)
20. Kaufmann, M., Wagner, D. (eds.): Drawing Graphs. Springer, Heidelberg (2001)
21. Purchase, H.C.: Which aesthetic has the greatest effect on human understanding? In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 248–261. Springer, Heidelberg (1997)
22. Purchase, H.C., Carrington, D.A., Allder, J.-A.: Empirical evaluation of aesthetics-based graph layout. Empirical Software Engineering 7(3), 233–255 (2002)
23. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM Journal on Computing 16(3), 421–444 (1987)
24. Ware, C., Purchase, H.C., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information Visualization 1(2), 103–110 (2002)