Triangulations with Circular Arcs*

Oswin Aichholzer[1], Wolfgang Aigner[2], Franz Aurenhammer[2], Kateřina Čech Dobiášová[3], Bert Jüttler[3], and Günter Rote[4]

[1] Institute for Software Technology, Graz University of Technology, Austria
[2] Institute for Theoretical Computer Science, Graz University of Technology, Austria
[3] Institute of Applied Geometry, Johannes Kepler University Linz, Austria
[4] Institut für Informatik, Freie Universität Berlin, Germany

Abstract. An important objective in the choice of a triangulation is that the smallest angle becomes as large as possible. In the straight-line case, it is known that the Delaunay triangulation is optimal in this respect. We propose and study the concept of a circular arc triangulation—a simple and effective alternative that offers flexibility for additionally enlarging small angles—and discuss its applications in graph drawing.

1 Introduction

Geometric graphs and especially triangular meshes (often called triangulations) are an ubiquitous tool in geometric data processing [4,17,26]. The quality of a given triangular mesh naturally depends on the size and shape of its composing triangles. In particular, the angles arising in the mesh are among the critical issues in main application areas like modeling, drawing, and finite element methods [26].

For practical purposes, quite often the Delaunay triangulation (see, e.g., [17]) is the mesh of choice, because it maximizes the smallest angle over all possible triangulations of a given finite set of points in the plane. Still, the occurrence of badly shaped triangles cannot be avoided sometimes, especially near the boundary of the input domain, or due to the presence of mesh vertices of high degree.

The situation becomes different (and interesting again) if the requirement that triangulation edges be straight is dropped. Indeed, certain applications are not confined to straight-line triangular meshes, or even are not really suited for it. In applications from graph drawing, for example, staying with straight edges might mean a hindrance to the readability of the drawing. Moreover, in finite element methods, the respective bivariate functions may be defined, in a natural way and with certain advantages, over 'triangles' with nonlinear boundaries. In these and other applications, the calculational and aesthetical benefits of a graph that potentially grants nice angles can be exploited fully only if curved edges are permitted.

In this paper, we want to encourage the use of so-called *arc triangulations*, which simply are triangulations whose edges are circular arcs. Maximizing the smallest angle in a combinatorially fixed arc triangulation of a point set can be

* Supported by FWF NRN 'Industrial Geometry' S92. A preliminary version of this work appeared as [1].

M. van Kreveld and B. Speckmann (Eds.): GD 2011, LNCS 7034, pp. 296–307, 2012.

formulated as a linear program (Section 2), which for most settings can even be transformed to a simple graph-theoretic problem (Section 3). This guarantees a fast solution of this (and of related) optimization problems for arc triangulations in practice and in theory. Moreover, the linear program will tell us whether a given domain admits an arc triangulation of a pre-specified combinatorial type, by checking whether its feasible region is nonempty. In particular, flips for arcs can be defined (Section 4), by optimizing the triangulation that is obtained after applying the flip combinatorially. Preliminary inspection shows that small angles tend to enlarge significantly under such heuristics.

We believe that arc triangulations constitute a useful tool especially in two important application areas—graph drawing and finite element methods. In particular, so-called π-triangulations (Section 5) can be used with advantage, based on the fact that arc triangles whose angles sum to π are images of straight triangles under a Möbius transformation. In view of graph drawing applications [10,15,23], it is desirable to extend our approach to optimizing angles in general plane graphs (Section 6). This cannot be done dirctly, but by completing the graph to a suitable triangulation (for example, its constrained Delaunay triangulation [9]), and treating the sums of triangulation angles between the graph arcs as single entities to be maximized. A simple and efficient method for optimally redrawing a straight-line graph with circular arcs is obtained. Applications to finite element methods will be discussed in the full version of this paper.

2 Angle Optimization

Consider a straight-line triangulation, $\mathcal{T}$, in a given domain D of the plane. No restrictions on D are required but, for the ease of presentation, let D be simply connected and have a piecewise circular (or linear) boundary. In general, $\mathcal{T}$ will use vertices. in the interior of D. Throughout the paper, we assume general position of the vertex set. We are interested in the following optimization problem: Replace each interior (i.e., non-boundary) edge of $\mathcal{T}$ by some circular arc, in a way such that the smallest angle in the resulting arc triangulation is maximized.

To see that this problem is well defined, notice that the optimal solution, call it $\mathcal{T}^*$, cannot contain negative angles: The smallest angle between arcs has to be at least as large as the smallest angle that arises in $\mathcal{T}$. As a consequence, for each vertex in S, the order of its incident arcs in $\mathcal{T}^*$ coincides with the order of its incident edges in the input triangulation $\mathcal{T}$. In other words, each arc triangle in $\mathcal{T}^*$ is *well-oriented*, i.e., it has the same orientation as its straight-line equivalent. Therefore, no overlap of arcs or arc triangles in $\mathcal{T}^*$ can occur. Interestingly, this is a specialty of triangulations; the last conclusion remains no longer true if faces with more than three arcs are present. An arc quadrangle, for instance, may have self-overlaps in spite of being well-oriented, whereas this is not possible for an arc triangle; see Figures 1 and 2. We postulate for the rest of this paper that arc triangles be well-oriented.

We now formulate the angle optimization problem as a linear program. For each straight-line edge $e = pq$ in the triangulation $\mathcal{T}$, we introduce two variables

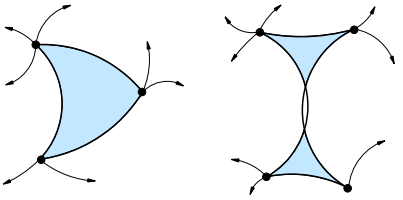

Fig. 1. Well-oriented arc triangle and quadrangle

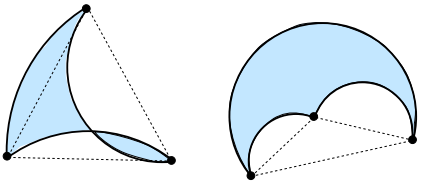

Fig. 2. These arc triangles are not well-oriented

ϕ_{pq} and ϕ_{qp}. The variable ϕ_{pq} describes the (signed) angle at which the circular arc $\widehat{pq}$ deviates to the left from the straight connection, when seen from p, and ϕ_{qp} describes this deviation angle, when seen from q. We have

$$\phi_{pq} = -\phi_{qp} \tag{1}$$

for all edges pq. For each edge e' of $\mathcal{T}$ on the input boundary ∂D, we fix the two deviation variables to the values $d_{e'}$ and $-d_{e'}$ given by ∂D. Thus, for a boundary edge $e' = pq$, we have

$$\phi_{pq} = -\phi_{qp} = d_{e'}. \tag{2}$$

We have $d_{e'} = 0$ if e' is supposed to stay a line segment. Alternatively, and preferably in certain applications, we could keep $\phi_{pq} = -\phi_{qp}$ variable and bound it by some threshold. The inequalities for the linear program now stem from the angles α_{qpr} arising in $\mathcal{T}$. The two edges pq and pr that define α_{qpr} are adjacent around p in the drawing, such that pr is the next edge counterclockwise from pq. We are interested in the angle between the corresponding two circular arcs, which is $\beta_{qpr} = -\phi_{pq} + \alpha_{qpr} + \phi_{pr}$, and we put

$$\delta \leq \beta_{qpr} \ . \tag{3}$$

The linear objective function L, which is to be maximized, is just $L = \delta$.

Clearly, maximizing δ will maximize the smallest angle $\beta_{\min}$ in the arc triangulation. Note that we may have $\beta_{\min} > \frac{\pi}{3}$ in $\mathcal{T}^*$ because, due to its piecewise circular shape, the sum of inner angles for ∂D may be larger than $\pi(h-2)$, for h being the number of vertices on ∂D. There are $O(n)$ (in)equalities and $O(n)$ variables, if n is the total number of vertices.

Sometimes the objective is to optimize not only the smallest angle, but rather to maximize lexicographically the sorted list of all arising angles, as is guaranteed by the Delaunay triangulation in the straight-line case. This can be achieved by repeatedly solving the linear program above, keeping angles that have been optimized already as constants. Care has to be taken however, because, depending on the solver, minimum angles do typically occur at several places, and the optimal ones among them have to be singled out. This type of problems has been called *lexicographic bottleneck optimization* in [6], in the context of combinatorial optimization problems. In [22] a general solution procedure in the context of

linear optimization is given, which amounts to repeatedly solving some slightly modified linear programs.

Angles larger than π may arise in the optimal triangulation. If this is undesirable in a particular application, constraints like

$$-\phi_{pq} + \alpha_{qpr} + \phi_{pr} \leq \gamma$$

for $\gamma < \pi$ may be added. In particular, choosing $\gamma = \pi - \delta$ will simultaneously decrease large angles, and thus will lead to arc triangles 'as equilateral as possible'. However, the demand of maximizing the smallest angle over the space of all possible arc triangulations (with the same combinatorics as $\mathcal{T}$) is then lost. Various other linear restrictions on angles can be added to the linear program, like fixing the angle sum in each arc triangle to π, or keeping each arc triangle inside the circumcircle of its three vertices. The relevance of these and other conditions will be substantiated in Sections 5 and 6. We consider the flexibility of our simple approach as an important feature in practice.

3 Graph-Theoretic Approach

The special setting of our linear program allows us to apply a purely graph-theoretic approach for its resolution.

Theorem 1. *The linear-programming problem of maximizing δ under restrictions* (1–3) *can be solved by a combinatorial (graph-theoretic) algorithm in $O(n^2)$ time.*

The remainder of this section gives a proof of Theorem 1. We have two variables ϕ_{pq} and ϕ_{qp} for each edge pq in the given straight-line triangulation, and the variable δ. Since a triangulation is a planar graph, there are $O(n)$ variables, $O(n)$ inequalities of type (3) induced by the angles between adjacent edges, and $O(n)$ equations of types (1) and (2).

First we consider a fixed value of δ and ask whether the system (1–3) is feasible. By using a method in [27] (see also [13,25]), we can transform the system into an equivalent system, in which every constraint has one of the following forms

$$X \leq Y + c, \tag{4}$$

$$X \leq 0 + c, \tag{5}$$

$$0 \leq Y + c, \tag{6}$$

where X and Y are two variables and c is a constant.

By substituting β_{qpr} we can easily rewrite (3) in this form, namely

$$\phi_{pq} \leq \phi_{pr} + (\alpha_{qpr} - \delta). \tag{7}$$

If we have bounds on the variables, $a \leq X \leq b$, we can also bring them into the desired form, and hence each equation (2) can be also handled, by first converting it into two inequalities.

We still have to deal with the equations (1) between 'opposite' variables. To this end, let us consider a system of inequalities of the form (4–6) in $2m$ variables $\mathcal{V} = \{x_1, \ldots, x_m, x'_1, \ldots, x'_m\}$ that come in 'opposite pairs'

$$x_i = -x'_i, \text{ for } i = 1, \ldots, m. \tag{8}$$

For a variable X, we will denote by $\bar{X}$ its opposite partner, $\bar{x}_i = x'_i$, $\bar{x}'_i = x_i$, $\bar{\bar{X}} = X$. The system we have at hands is of this form, with $\bar{\phi}_{pq} = \phi_{qp}$. Now, for each inequality of the form (4–6), we can form an equivalent *opposite inequality*, in which each variable is replaced by the opposite variable on the other side. For example,

$$X \leq Y + c \tag{4}$$

is turned into $\bar{Y} \leq \bar{X} + c$. In view of (8), the opposite inequality is equivalent to the original one. Thus, when we add all opposite inequalities, we will create some redundancy but we will not change the solution. It is easy to prove the following:

Lemma 1. *Consider a system of the equations* (8) *together with inequalities of the form* (4–6)*, that also contains with each inequality its opposite inequality. Then this system has a solution if and only if the system* without *the equations* (8) *has a solution.*

This means that we can ignore the equations (1), at the expense of doubling the number of inequalities. All inequalities have the form (4–6). By introducing a new variable Z_0 representing zero, the inequalities (5–6) that contain only one variable can also be brought into the standard form (4). This new system is equivalent to the original one: Since all inequalities now have the form (4), one can add an arbitrary constant to all variables without invalidating the inequalities, and thus one can assume, without loss of generality, that $Z_0 = 0$.

It is well known that a system of inequalities of the form (4) can be tested by checking whether an associated graph G has a negative cycle [7,27], and a solution can be found by a shortest path calculation. The graph G has a node for each variable, and for each inequality of the form (4) it contains an arc of weight c from X to Y. Moreover, consider an augmented graph G^+, that has an additional start node S and an edge of weight 0 from S to every node of G.

Lemma 2. *A system of inequalities of the form* (4) *has a solution iff the associated graph G (or equivalently, G^+) has no negative cycle. If a solution exists, it can be found by computing shortest distances from S to all nodes in G^+.*

The running time of this test, with the Bellman–Ford algorithm, is given by the number of nodes or variables ($2m = O(n)$ in our case), times the number of arcs or inequalities ($O(n)$ as well). Thus, finding a solution of the angle drawing problem for a given value of δ takes $O(n^2)$ time.

Now we will consider δ as a variable and come back to the problem of maximizing δ. This amounts to checking for a negative cycle in a graph whose weights are of the form $c - \delta$, for constants c and a parameter δ. This problem is known

as the *minimum cycle mean problem*: For a cycle with k edges the weight has the form $w - k\delta$, where w is the sum of all positive edge constants c along the cycle. The weight is negative for $\delta > w/k$. So w/k, the *mean weight* of the cycle, is the largest value for δ which does not result in a negative cycle. For the entire graph, this means that the largest possible value of δ for which the graph is free of negative cycles is determined by the minimum cycle mean. The minimum cycle mean problem has been solved in [19], and the algorithm takes the same running time as the Bellman–Ford algorithm, that is, $O(n^2)$ time, but it takes $O(n^2)$ space.

4 Flipping in Arc Triangles

The fact that every simple polygon can be triangulated with straight line segments is folklore. However, a domain D with piecewise circular boundary need not admit *any* triangulation, even if circular arcs may be used. It is known that a linear number of Steiner points is required in the worst case to ensure an arc triangulation [3].

One of the arising questions is: Given the domain D and a (combinatorial) triangulation $\mathcal{T}_c$ in D, possibly with (fixed) interior points, can $\mathcal{T}_c$ be realized by circular arcs? Clearly, if only straight-line edges are to be used, then this is merely a segment intersection problem. For deciding the general case, we can now utilize the linear program formulated in Section 2. A realizing arc triangulation exists if and only if the feasible region of the linear program is nonempty.[1] As a particularly nice feature, this enables us to define flip operations in arc triangulations, as is described below.

Consider some arc triangulation $\mathcal{A}$ in the domain D. Each interior arc $\widehat{pq}$ of $\mathcal{A}$ lies on the boundary of two arc triangles. Let r and s be the two vertices of these arc triangles different from p and q. Flipping $\widehat{pq}$ by definition means removing $\widehat{pq}$ from $\mathcal{A}$, establishing an arc between r and s combinatorially, and optimizing over the resulting triangulation. Note that 'well-oriented' in this case has to refer to the *combinatorial* order of the edges around a vertex of a triangulation.

For the linear program that describes this optimization problem, we have to know the angles α of the corresponding straight-line embedding; see Section 2. Note that after a flip, the straight-line realization of the graph is not necessarily a valid geometric triangulation. In such a case, the combinatorial order around a vertex is different from the geometric one. As a consequence, some angles α have to take negative values to obtain a valid setting for the linear program that optimizes δ. See Figure 4 for an example with the combinatorial order being 1 to 5, while the geometrical order is $1, 4, 2, 3, 5$.

Unlike for the original setting in Section 2, here a positive solution for δ is not guaranteed. In fact, the sign of the optimized value δ indicates whether or not the combinatorial triangulation (after a flip) is realizable as an arc triangulation.

[1] Note that the following related problem is NP-complete [20]: Given a point set S and *some* set E of straight-line edges on S, decide whether E contains a triangulation of S.

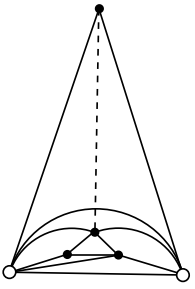

Fig. 3. A double edge connecting bottom vertices

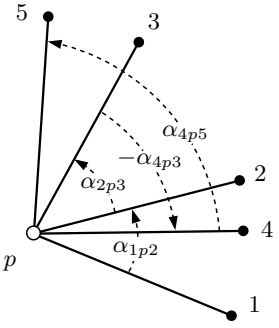

Fig. 4. Combinatorial order at p

If $\delta > 0$ after the optimization, then the new arc triangulation exists and contains a circular arc between r and s that satisfies the criterion of being geometrically well-oriented. In case of nonexistence (if $\delta \leq 0$), the combinatorial triangulation is not realizable as an arc triangulation, and we declare the arc $\widehat{pq}$ as non-flippable. Observe that an arc flip may change various circular arcs geometrically, as we optimize over their curvature afterwards.

Sometimes we may not want to perform an arc flip even if it exists. For example, flipping an arc a can lead to an inner vertex of degree 2, a property of arc triangulations which is possibly unwanted in the application. Arc a can easily be declared as not flippable, by putting the restriction that angles in triangles be less than π. Note that this does not necessarily prevent the occurrence of double-edges between two vertices of an arc triangulation. For example, see Figure 3, where all angles are smaller than π. However, a check if an edge already exists can be done before the optimization step, and thus does not have to be incorporated into the linear program.

Optimizing angles with arc flips is a powerful (though maybe costly) tool. We demonstrate the positive effect of sequences of such flips with Figures 5 and 6. A significant improvement over the Delaunay triangulation becomes possible (in fact, the smallest angle is doubled in this example) by reducing the degree of a particular vertex, v. Note that this configuration is quite 'robust' in the sense that v retains its high degree in the Delaunay triangulation even if the placement of the other vertices is changed moderately. Repeated appearance of patterns as in Figure 5 may lead to an overall poor quality of a given triangular mesh.

In general, we observe that small angles in a straight-line triangulation stem from one of two reasons: (1) The geometry of the underlying domain D (plus its vertex set) forces slim triangles in the vicinity of ∂D. These 'boundary effects' can usually be mildened by mere geometric optimization of the corresponding arc triangulation. (2) Vertices of degree k naturally impose an upper bound of $\frac{2\pi}{k}$ on the smallest arising angle. This situation can be remedied only with combinatorial changes, and in contrast to the straight edge case, this is indeed possible for arc triangulations. For straight edges, the combinatorics of the Delaunay triangulation is already optimal.

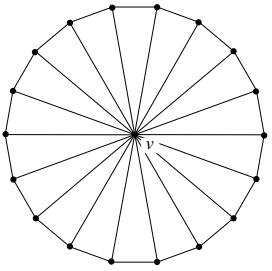

Fig. 5. Delaunay triangulation

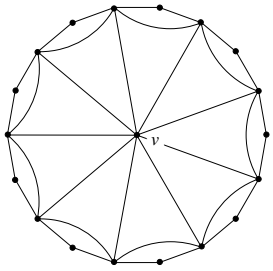

Fig. 6. Optimized arc triangulation

A challenging open question is whether repeated application of angle-improving arc flips always leads to the global optimum, that is, to the combinatorial type of arc triangulation which admits the largest possible minimum angle for the given domain. A more basic question is whether the set of combinatorial triangulations that are realizable as arc triangulations is connected by flips. We leave these problems as a subject for future research.

5 Special Arc Triangles

Before discussing the relevance of arc triangulations to the area of graph drawing, we have a look at special types of arc triangles. Recall from Section 2 the convention that arc triangles are geometrically well-oriented.

An arc triangle ∇ is termed a *π-triangle* if the sum of its interior angles is π. These triangles are interesting because they are images of a straight-line triangle under a unique Möbius transformation [24]. Moreover, any π-triangle is contained in the circumcircle of its vertices, a possibly useful regularity condition. We study arc triangulations that are composed of π-triangles. Such *π-triangulations* will not always exist, but they do, of course, if the domain D is a simple polygon, because every straight-line triangulation is a π-triangulation. If ∂D is composed of circular arcs, a necessary (though not sufficient) existence condition is that the sum of interior angles at the h boundary vertices of D is $\pi(h-2)$.

For the remainder of this section, let D be a simple polygon, and $\mathcal{T}$ be some straight-line triangulation in D. The geometry of any arc triangulation $\mathcal{A}$ in D that is combinatorially equivalent to $\mathcal{T}$ is determined by the vector $\Phi(\mathcal{A})$ of deviation angles ϕ_{pq}, for the interior arcs $\widehat{pq}$ of $\mathcal{A}$. (The opposite value, ϕ_{qp}, is fixed by ϕ_{pq}; see Section 2). Interpreting $\Phi(\mathcal{A})$ as a point in high dimensions, we can talk of the space of arc triangulations for $\mathcal{T}$. The next lemma is important in view of optimizing a given π-triangulation. Let us assume that there exists an arc triangulation for D where all interior angles are positive.

Lemma 3. *Let $\mathcal{T}$ have n vertices, h of which lie on the boundary of D. The dimension of the space of π-triangulations for $\mathcal{T}$ is $n-h$.*

The proof is omitted due to space constraints. Lemma 3 remains true if $\mathcal{T}$ is replaced by any π-triangulation of D. For applications, the input is most likely a

Table 1. Angle improvement in arc triangulations

angle sum	smallest angle	improvement over Delaunay
Delaunay (180°)	18.03°	0
180°	22.52°	25 %
179°–181°	22.92°	26 %
175°–185°	24.88°	38 %
170°–190°	27.53°	50 %
160°–200°	31.77°	72 %

straight-line triangulation, which is to be optimized into a π-triangulation with maximum smallest angle. The boundary of D might be given as a spline curve, approximated smoothly by circular arcs. The inner angle sum for D is $\pi \cdot h$ in this case (rather than $\pi(h-2)$), such that a π-triangulation does not exist. Still, the approximating circular arcs will be close to line segments for most practical data, such that an 'almost straight' π-triangulation is likely to exist. Also, one could start with some combinatorial triangulation suitable for D, to be able to treat a larger class of domains.

Table 1 shows experimental data for Delaunay meshes optimized into (almost) π-triangulations, for 500 random points, postprocessed to keep a certain inter-point distance as in realistic meshes. The gain is quite significant, especially if the condition on the angle sum is relaxed from π to a small interval around that value. For several applications, there is sometimes a certain threshold (typically around 25°) beyond which a mesh is considered as poor-quality [5].

Note that, by Lemma 3, optimization is only possible in subdomains of D where interior points are present. Thus, the diagonals of D defined by $\mathcal{T}$ (if any) separate optimizable subdomains from each other. Again, such diagonals are unlikely to appear in the dense meshes used in practical applications. In any case, extraneous points can be inserted into the π-triangulation while keeping all angle sums in arc triangles to π. In particular, we can put such points on arcs, in order to split obstructive diagonals of D.

6 Graph Drawing

Literature on drawing graphs nicely in the plane is large; see e.g. [10,23,28]. Most algorithms take as input an abstract graph G and produce a layout of the vertices of G such that the resulting straight-line (or orthogonal) drawing is aesthetically pleasing, and preferably is even optimal with respect to certain application criteria. On the theoretical side, bounds on the achievable angular resolution are known for various classes of graphs [16,21]. A characterization of all planar drawings of a triangular graph through a system of equations and inequalities relating its angles is given in [11].

Results for curvilinear drawings of graphs are comparatively sparse. See, for example, [8,18] and references therein, who give lower bounds and algorithms for drawing graphs on a grid with curved edges (including circular multiarcs), and [15] where a method based on physical simulation is proposed. In [14],

crossing-free drawings of graphs with circular arcs as edges are considered from an algorithmic viewpoint. The vertices are fixed and each edge has to be chosen from a given number of arcs. Recently, circular arc graphs with equiangular edges around each vertex have been studied in [12].

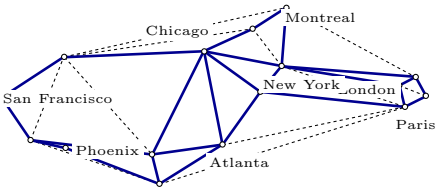

Fig. 7. IP backbone graph

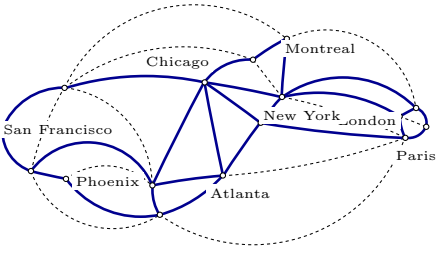

Fig. 8. Backbone optimally redrawn

Here we actually consider a simpler setting, namely, for a given planar straight-line embedding of a graph G, the problem of *redrawing* G with curved edges in an optimal way. In a redrawing, the positions of the vertices are kept fixed. This may be a natural demand, for instance, in certain geographical applications. Recently it has been shown [2] that redrawings of G with tangent-continuous *biarcs* or quadratic Bézier curves (parabolic arcs) always exist such that every vertex is pointed, i.e., has an incident angle of at least π. Potential applications concern labeling the graph vertices with high readability. Redrawing a plane graph G with circular arcs in a pointed way is not always possible.

Let us describe how maximizing the smallest angle in a circular arc redrawing of G can be achieved. It is tempting to apply the linear optimization method from Section 2 to G directly. This, however, bears the risk of arc overlaps getting out of control. (Recall that overlap-free optimization is guaranteed only for full triangulations. This is possibly the reason why this simple approach has not been used in practice yet.) One way out is to embed G in some triangulation $\mathcal{T}$ first, and treat respective sums of angles as single entities to be optimized. That is, for each angle ϱ in G, given by the concatenation of angles $\alpha_1, \ldots, \alpha_k$ in $\mathcal{T}$, we use the constraint $\delta \leq \beta_1, \ldots, \beta_k$, with each β_i expressed by the corresponding straight-line triangulation angle α_i and its two assigned deviation variables $\beta_i = -\phi_1 + \alpha_i + \phi_2$ as in Section 2.

The quality of optimization depends on the chosen triangulation, which will be subject of future research; cf. Section 4. Note that, however, even if we try out all possible triangulations, this may not lead to the optimal solution, as there are arc polygons that cannot be triangulated without additional vertices. If the optimal drawing contains such a face, then no triangulation will yield the optimum drawing.

If we wish to optimize the entire angle vector $\varrho_1, \ldots, \varrho_m$ for G, this can be achieved too, in an iterative way as before. Additional restrictions may be posed, like $\varrho_j < \pi$ or $\varrho_j < \frac{\pi}{2}$, in order to preserve obtuse or sharp angles in G.

The adjacency graph in Figures 7 and 8, and the layer graph in Figures 9 and 10 exemplify the effect of our circular arc redrawing method. The results

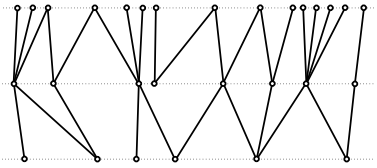

Fig. 9. A 3-layer graph

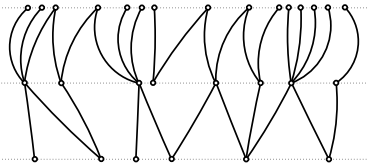

Fig. 10. Arc redrawing

seem satisfactory, in spite of the fact that vertices are required not to move. For geographic structures as in Figure 7, or certain graph structures arising in physics, this is quite often a desired property. Our results compare well to, e.g. [15], who use for optimization the additional freedom of placing vertices, though at a price of high computation cost. For our method, the number of vertices of the input graph is no limitation, as far as applications from graph drawing are concerned.

7 Open Questions

For non-triangulated regions in the input graph (compare the quadrangle in Figure 1), the requirement that arcs do not intersect induces a nonlinear constraint between the corresponding angles. It would be interesting to know if this constraint has some structure (for example, convexity), which would allow it to be accommodated in the optimization process. Further open questions raised here are the convergence of the angle-increasing arc flipping process in Section 4, and an extension of the presented results to three dimensions.

References

1. Aichholzer, O., Aigner, W., Aurenhammer, F., Čech Dobiášová, K., Jüttler, B.: Arc triangulations. In: Proc. 26th European Workshop Comput. Geometry, pp. 17–20 (2010)
2. Aichholzer, O., Rote, G., Schulz, A., Vogtenhuber, B.: Pointed drawings of planar graphs. In: Proc. 19th Ann. Canadian Conf. Comput. Geometry, pp. 237–240 (2007)
3. Aichholzer, O., Aurenhammer, F., Hackl, T., Juettler, B., Oberneder, M., Sir, Z.: Computational and structural advantages of circular boundary representation. Int'l J. Computational Geometry & Applications 21, 47–69 (2011)
4. Bern, M., Eppstein, D.: Mesh generation and optimal triangulation. Computing in Euclidean Geometry. LN Series on Computing, vol. 4, pp. 47–123. World Scientific (1995)
5. Boivin, C., Ollivier-Gooch, C.: Guaranteed-quality triangular mesh generation for domains with curved boundaries. International Journal for Numerical Methods in Engineering 55, 1185–1213 (2002)
6. Burkard, R.E., Rendl, F.: Lexicographic bottleneck problems. Operations Research Letters 10, 303–308 (1991)
7. Carré, B.: Graphs and networks. Oxford University Press (1979)

8. Cheng, C.C., Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Drawing Planar Graphs With Circular Arcs. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 117–126. Springer, Heidelberg (1999)
9. Chew, L.P.: Constrained Delaunay triangulations. Algorithmica 4, 97–108 (1989)
10. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing—Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
11. Di Battista, G.D., Vismara, L.: Angles of planar triangular graphs. SIAM J. Discrete Mathematics 9, 349–359 (1996)
12. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Lombardi Drawings of Graphs. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 195–207. Springer, Heidelberg (2011)
13. Edelsbrunner, H., Rote, G., Welzl, E.: Testing the necklace condition for shortest tours and optimal factors in the plane. Theor. Comput. Sci. 66, 157–180 (1989)
14. Efrat, A., Erten, C., Kobourov, S.G.: Fixed-Location Circular-Arc Drawing of Planar Graphs. Journal of Graph Algorithms and Applications 11, 145–164 (2007)
15. Finkel, B., Tamassia, R.: Curvilinar Graph Drawing Using The Force-Directed Method. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 448–453. Springer, Heidelberg (2005)
16. Formann, M., Hagerup, T., Haralambides, J., Kaufmann, M., Leighton, F.T., Symvonis, A., Welzl, E., Wöginger, G.: Drawing graphs in the plane with high resolution. SIAM J. Computing 22, 1035–1052 (1993)
17. Fortune, S.: Voronoi diagrams and Delaunay triangulations. Computing in Euclidean Geometry. LN Series on Computing, vol. 4, pp. 225–265. World Scientific (1995)
18. Goodrich, M.I., Wagner, C.G.: A framework for drawing planar graphs with curves and polylines. J. Algorithms 37, 399–421 (2000)
19. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23, 309–311 (1978)
20. Lloyd, E.L.: On triangulations of a set of points in the plane. In: Proc. 18th IEEE Symp. on Foundations of Computer Science, pp. 228–240 (1977)
21. Malitz, S., Papakostas, A.: On the angular resolution of planar graphs. In: Proc. 24th Ann., pp. 527–538 (1992)
22. Marchi, E., Oviedo, J.A.: Lexicographic optimality in the multiple objective linear programming: The nucleolar solution. European Journal of Operational Research 57, 355–359 (1992)
23. Nishizeki, T., Rahman, M.S.: Planar graph drawing. World Scientific (2004)
24. Pedoe, D.: A course of geometry for colleges and universities. Cambridge University Press (1970)
25. Rote, G.: Two solvable cases of the traveling salesman problem. PhD Thesis, TU Graz, Institute for Mathematics (1988)
26. Shewchuk, J.: What is a good linear element? Interpolation, conditioning, and quality measures. In: Proc. 11th International Meshing Roundtable, pp. 115–126 (2002)
27. Shostak, R.: Deciding linear inequalities by computing loop residues. Journal of the ACM 28, 769–779 (1981)
28. Sugiyama, K.: Graph Drawing and Applications for Software and Knowledge Engineers. World Scientific (2002)