Triangulations with Circular Arcs

Oswin Aichholzer[1] Wolfgang Aigner[1,3] Franz Aurenhammer[2]
Kateřina Čech Dobiášová[3] Bert Jüttler[3] 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 of a given point set is that the smallest angle becomes as large as possible. When triangulation edges are straight line segments, it is known that the Delaunay triangulation is the optimal solution. We propose and study the concept of a circular arc triangulation, a simple and effective alternative that offers flexibility for additionally enlarging small angles. We show that angle optimization and related questions lead to linear programming problems that can be formulated as simple graph-theoretic problems, and we define flipping operations in arc triangles. Moreover, special classes of arc triangulations are considered, for applications in finite element methods and graph drawing.

1   Introduction

Geometric graphs and especially triangular meshes (often called triangulations) are an ubiquitous tool in geometric data processing. They serve as partitioning and connectivity structures, and play a central role in geometric object representation and manipulation [3, 18, 29]. 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, contouring, interpolation, and finite element methods [29]. For example, triangles with small angles tend to lead to unsatisfactory conditioning for finite element methods.

For practical purposes, quite often the Delaunay triangulation (see, e.g., [18]) 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 'poor' triangles cannot be avoided sometimes, especially near the boundary of the input domain, or due to the presence of mesh vertices of high edge degree. Such vertices cannot be resolved by edge flips or other local operations, because the combinatorics of the Delaunay triangulation is already optimal for angles.

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 finite element methods, for example, the respective bivariate functions may be defined, in a natural way and with certain advantages, over 'triangles' with nonlinear boundaries. Also, in applications from graph drawing, staying with straight edges of a graph might mean a hindrance to the readability of its drawing. 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. Modeling triangulations this way bears several advantages if angles are to be optimized. A typical reason for the occurrence of small angles in a straight line triangulation is that the geometry of the vertex set forces slim triangles in the vicinity of the domain boundary. Such angles can now be enlarged, by optimizing the arc curvatures for the given triangulation. The second problem are vertices of high degree that result when the smallest angle is optimized for straight line triangulations (the Delaunay case). This situation can be faced now by applying angle-improving flips in arc triangles that reduce the vertex degree.

Maximizing the smallest angle in a combinatorially fixed arc triangulation of a point set can be 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 void. In particular, flips for arcs can be defined (Section 4), by optimizing the triangulation that is obtained when applying the flip combinatorially. Thereby, the existence check above gives us a tool to characterize flippable arcs. Preliminary inspection shows that small angles tend to enlarge significantly under such heuristics. Note that if we want to optimize equiangularity in an arc triangulation (i.e., maximize the sorted angle vector lexicographically) then we can do so as well. After having maximized the first k smallest angles, we keep them fixed in the linear program for maximizing the $(k+1)$st angle, for $k \geq 0$.

We believe that arc triangulations constitute a useful tool especially in two important application areas—finite element methods and graph drawing (Sections 6 and 7, respectively). In the isoparametric approach to finite element methods [20], for example, 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, we obtain an approximation by conformal (angle-preserving) Bézier patches when the sum of angles is optimized towards π. Moreover, if the angle sum is even equal to π, simple inverse geometry mappings can be constructed.

In view of graph drawing applications [10, 16, 26], it is desirable to extend our approach to optimizing angles in general plane graphs. However, our optimization method works only for full triangulations because, remarkably, an overlap of circular arcs is ruled out automatically by the structure of the triangles. To maximize small angles in a curvilinear (re)drawing of an arbitrary plane graph, we first complete the graph to a suitable triangulation (for example, its constrained Delaunay triangulation [22, 8]), and treat the sums of triangulation angles between two consecutive arcs of the graph around a vertex as single entities to be maximized. A simple and efficient method for optimally redrawing a straight line graph with circular arcs is obtained.

In several applications, the boundary of the underlying domain will be given as a polynomial spline curve. Such domains can be approximated in a convenient way using circular biarc splines [1, 12], and thus are naturally suited to triangulation by circular arcs.

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 piecewise circular (or linear) boundary. In general, $\mathcal{T}$ will use vertices in the interior of D. We are interested in the following optimization problem: Replace each 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. (Of course, straight arcs are allowed as well, if optimality demands.)

To see that this problem is well defined, we introduce the notion of a *well-oriented* arc triangle. Let $\mathcal{T}^*$ denote the optimal solution. Consider an arbitrary
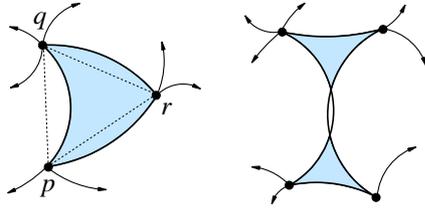
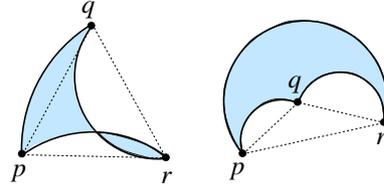Figure 1: Well-oriented arc triangle; overlapping arc quadrangle

Figure 2: These arc triangles are not well-oriented

arc triangle t^* in $\mathcal{T}^*$, and let t be its straight line equivalent in the input triangulation $\mathcal{T}$. Let p, q, and r be the vertices of the triangle t such that, when scanning the edges of t in this cyclic order, then t always lies to the right-hand side. We call the arc triangle t^* (geometrically) *well-oriented* if t^* behaves the same, i.e., it always stays to the right-hand side when its arcs $\widehat{pq}$, $\widehat{qr}$, and $\widehat{rp}$ are scanned in this order. Figures 1 and 2 give illustrations. We have the following property:

Observation 1 *All arc triangles in $\mathcal{T}^*$ are well-oriented.*

Proof: We observe that $\mathcal{T}^*$ 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 the underlying vertex set S, the order of its incident arcs in $\mathcal{T}^*$ coincides with the order of its incident edges in $\mathcal{T}$. This outrules the two possible cases of non-well-orientedness, namely, self-intersection or warping; see Figure 2 again. $\square$

Interestingly, this property is a specialty of triangulations; the last conclusion in the proof above remains no longer true if faces with more than three arcs are present. An arc quadrangle, for instance, may have self-overlaps even if no negative angles are present, whereas this is not possible for an arc triangle. 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 ϕ_{pq} and ϕ_{qp}. The variable ϕ_{pq} describes the (signed) angle at which the tangent at p to the circular arc $\widehat{pq}$ deviates to the left from the straight connection, and ϕ_{qp} describes this deviation angle, when considered at q. Figure 3 offers an illustration. 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 the (possibly piecewise circular) shape of ∂D. Thus, for a boundary edge $e' = pq$, we have

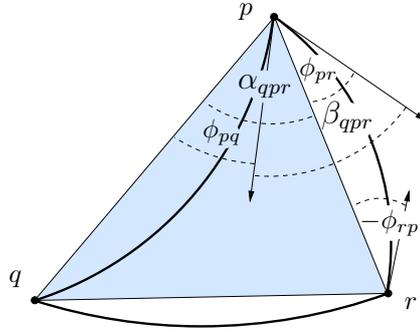$$\phi_{pq} = -\phi_{qp} = d_{e'}. \tag{2}$$

Figure 3: Angles of deviation

(We have $d_{e'} = 0$ if e' is a straight line segment, and $d_{e'} \neq 0$ if e' is a circular arc. Alternatively, and preferably in certain applications, we could keep $\phi_{pq} = -\phi_{qp}$ variable and bound it by some user-defined threshold that does not relate to ∂D.)

The inequalities for the linear program now stem from the angles α_{qpr} arising in $\mathcal{T}$. More precisely, α_{qpr} is the angle between two edges pq and pr which are adjacent around p in $\mathcal{T}$, such that pr is the next edge counterclockwise from pq. We are interested in the angle between the corresponding two circular arcs, which is $\beta_{qpr} = -\phi_{pq} + \alpha_{qpr} + \phi_{pr}$ (see Figure 3 again), and we put

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

such that δ bounds the smallest arising angle from below. 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 [5], in the context of combinatorial optimization problems. In [25] 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 may be meaningful in applications, 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, 6, and 7. 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 2 *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 2. 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 [30] (see also [14, 28]), 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.

Lemma 1 *Consider a system* (I), *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* (II) *without the equations* (8) *has a solution.*

Proof: It is clear that system (I) can only have a solution if the less restrictive system (II) has one. For a variable $x \in \mathcal{V}$ and its opposite $\bar{x}$ we now consider a solution pair $(x^*, \bar{x}^*)$ in system (II). Then it can be checked easily that $(x^\dagger, \bar{x}^\dagger)$, with

$$x^\dagger := \frac{x^* - \bar{x}^*}{2} \quad \text{and} \quad \bar{x}^\dagger := \frac{\bar{x}^* - x^*}{2},$$

is a solution for x and $\bar{x}$ in system (I). $\qquad\qquad\square$

For our setting, 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 [6, 30], and a

solution can be found by a shortest path calculation. The graph G has a node for each variable, and for each inequality

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

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 if and only if 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 Karp [21], 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 well known [9]. Again, 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 [1].

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.[1] For deciding the general case, we

---

[1]Note that the following related problem is NP-complete [23]: Given a point set S and *some* set E of straight line edges on S, decide whether E contains a triangulation of S.
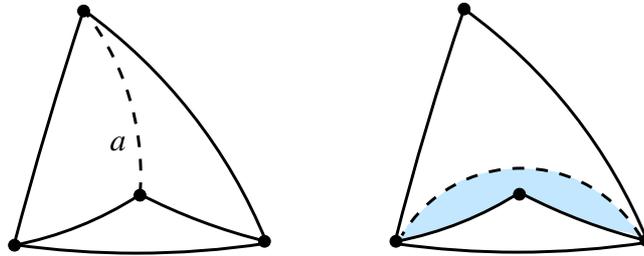
Figure 4: Flippable or not flippable?

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. 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, e.g., after a flip as depicted in Figure 4, the straight line realization of the graph is not 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 6 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. 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. Such arcs are not touched by the flipping process. On the other hand, flipping a flippable arc may lead to a double-arc between vertices; see Figure 5.

Observe that an arc flip may change various circular arcs geometrically (by optimizing over their curvature), whereas only a single arc is exchanged combinatorially. An arc flip thus is a geometrically global operation which is combi-
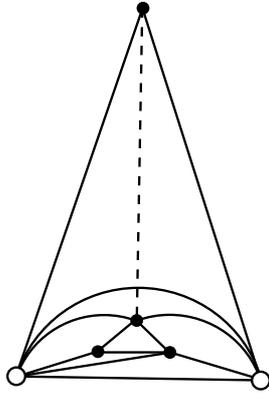
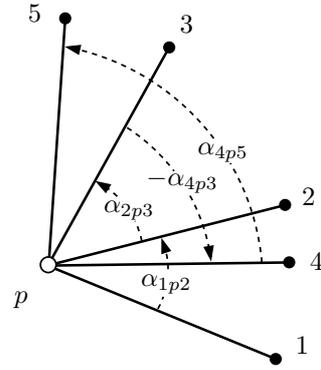Figure 5: Flipping the dashed arc creates a double-arc between the bottom vertices

Figure 6: Combinatorial order around the vertex p from 1 to 5

naturally local.

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; consult Figure 4 again. 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-arcs between two vertices of an arc triangulation. For example, see Figure 5, 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 7 and 8. A significant improvement over the Delaunay triangulation becomes possible, by reducing the degree of a particular vertex, v. (In fact, the smallest angle is doubled in the shown example.) 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 7 may lead to an overall poor quality of a given triangular straight line 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
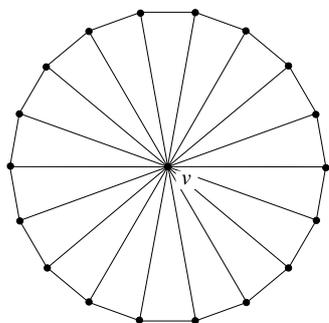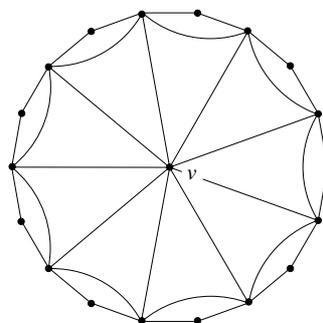
Figure 7: Delaunay triangulation

Figure 8: Flip-optimized arc triangulation

with combinatorial changes (for example, flips), 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.

A challenging open question is the following:

Does repeated application of angle-improving arc flips always lead to the global optimum, that is, to the combinatorial type of arc triangulation which admits the largest possible minimum angle for the given domain?

As a more basic question, we mention:

Is the set of combinatorial triangulations that are realizable as arc triangulations 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 areas of graph drawing and finite element methods, a look at various types of arc triangles is required. 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. Clearly, straight line triangles are π-triangles. We state a well-known geometric fact (cf. [27]).

Property 3 *Let ∇ be some arc triangle. The following three properties are equivalent.*
(a) ∇ is a π-triangle.
(b) The three supporting circles of ∇ intersect in a common point exterior to ∇.
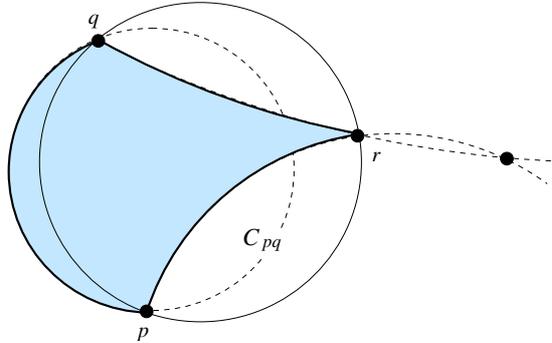
Figure 9: This arc triangle is not a π-triangle because it does not stay within its circumcircle

(c) ∇ is the image of a straight line triangle under a unique Möbius transformation.

Interestingly, π-triangles also fulfill a useful regularity condition.

Property 4 *Any π-triangle is contained in the circumcircle of its vertices.*

Proof: See Figure 9. Let p, q, and r be the vertices of a π-triangle ∇, and assume that there exists an arc of ∇, say $\widehat{pq}$, which lies outside of the circle spanned by p, q, and r. Then the circle C_{pq} supporting $\widehat{pq}$ does not enclose r. Otherwise, a change in the order of the arcs of ∇ is implied, and ∇ would not be well-oriented. But r lying outside of C_{pq} means that C_{pq} does not pass through any intersection point of the two circles supporting $\widehat{pr}$ and $\widehat{qr}$, respectively, because at least one of these arcs has to be bent inwards to allow an angle sum of π in ∇. This contradicts Property 3(b). $\qquad\square$

In view of the mentioned properties, it is worthwhile to study arc triangulations that are composed of π-triangles; we shall call them π-*triangulations*. Such 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 deviation 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 $n - h$.*

Proof: The space of *all* arc triangulations for $\mathcal{T}$ is full-dimensional, because each angle ϕ_{pq} (and with it ϕ_{qp}) can be modified individually within certain bounds so as to still guarantee the existence of an arc triangulation.[2] In order to keep the property that all arc triangles are π-triangles, we have to constrain the values of ϕ_{pq}. For each arc triangle ∇ with boundary vertices p, q and r, its three angles have to satisfy

$$\phi_{pq} + \phi_{qr} + \phi_{rp} = 0 \ . \tag{9}$$

Angles associated with boundary edges of $\mathcal{T}$ are zero. We obtain a system of t linear equations for e unknowns, where t and e are the numbers of triangles and inner edges of $\mathcal{T}$, respectively.

The structure of this linear system is determined by the dual graph, G, of $\mathcal{T}$. The nodes of G correspond to the equations (9), and the elementary cycles in G correspond to the degrees of freedom of the system. This can be seen by reducing G to a spanning tree, by adding constraints

$$\phi_{pq} = 0 \tag{10}$$

to the system for all interior arcs $\widehat{pq}$ but the ones dual to the tree. The system now possesses only the trivial solution, that is, the origin of the solution space, which corresponds to the straight line π-triangulation $\mathcal{T}$. Each time an edge e of G is put back (i.e., one constraint of the form (10) is removed), a unique elementary cycle is closed, and a single degree of freedom is gained, stemming from the corresponding linear equation: Namely, consider a traversal of this cycle in some direction, say clockwise. Then one can bend all arcs slightly in this direction, leaving the angle sum in each triangle unchanged. That is, indeed one more degree of freedom is gained when adding such an edge e.

The number of elementary cycles is, therefore, just $e - (t - 1)$, since a tree with t nodes has $t - 1$ edges. By Euler's formula, we have $e = (3n - h - 3) - h$ and $t = 2n - h - 2$. In conclusion, the system of equations (9) has $n - h$ degrees of freedom. $\qquad\square$

Lemma 3 remains true if $\mathcal{T}$ is replaced by any π-triangulation of D. For applications, the input is most likely a 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'

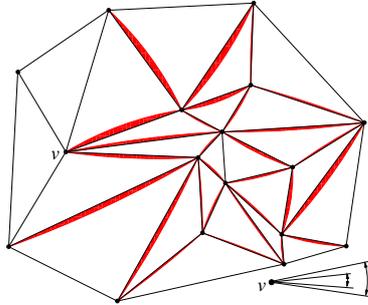[2]Here, like in all other parts of this paper, we assume general position of the underlying vertex set.

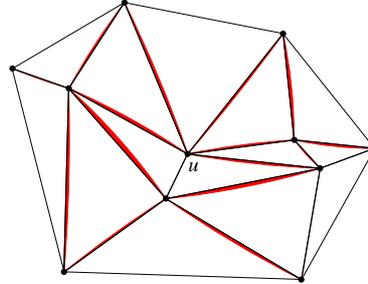Figure 10: Straight line triangulation and its angle-maximized π-triangulation superimposed

Figure 11: The Delaunay triangulation improves less but still more than 40%

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.

Figure 10 displays how a straight line triangulation is optimized into a π-triangulation. The change does not appear dramatic, but observe that the smallest angle (occurring at vertex v) almost doubles, from 9.7º to 19º. Figure 11 shows the same process for the Delaunay triangulation of a different set of points. The gain here is about 41%, from 13.9º to 18.6º, being attained at vertex u. In both cases, no arc flips have been applied.

Table 1 shows experimental data for a larger input (500 random points, postprocessed to keep a certain interpoint distance as in realistic meshes). We see that the gain reduces for larger Delaunay meshes but is still 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 25º) beyond which a mesh is considered as poor-quality [4].

angle sum	Delaunay min	min arc angle	gain
180º	18.03º	22.52º	25%
179º - 181º	18.03º	22.92º	26%
175º - 185º	18.03º	24.88º	38%
170º - 190º	18.03º	27.53º	50%
160º - 200º	18.03º	31.77º	72%

Table 1: Angle improvement in arc triangulations

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
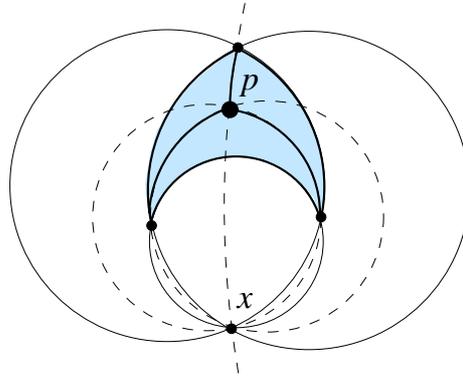
Figure 12: Inserting a vertex into a π-triangle

case, extraneous points can be inserted into the π-triangulation while keeping all angle sums in arc triangles to π. See Figure 12, where point p is inserted such that, for each of the three arc triangles created, the condition (b) in Property 3 is fulfilled. Note that p may be placed anywhere in the (shaded) arc triangle, because the dashed circles through x do always exist. In particular, we can choose p on some boundary arc, in order to split obstructive diagonals of D.

6   Finite Element Methods

In the isoparametric approach to finite element methods (FEM) [20], the computational domain is described by a collection of primitives P_i, each being the image of, e.g., a domain triangle Δ_i under a geometry mapping, G_i. More specifically, a set of *shape functions* ψ_{ij} is defined on Δ_i such that

$$G_i = \sum_j \psi_{ij} c_{ij} \ .$$

The coefficients c_{ij} typically stem from the vertices and the edge midpoints of the primitives. The *test functions* (finite elements) which span the discretization space are obtained by composing the shape functions with the inverse geometry mappings,

$$f_{ij} = \psi_{ij} \circ G_i^{-1} \ .$$

By suitably collecting functions defined on neighbouring triangles one obtains continuous test functions (conforming finite elements). Alternatively, one can use non-continuous test functions (non-conforming finite elements), which are better suited in some applications.

Clearly, any arc triangle can be represented by a geometry mapping, which is a rational quadratic Bézier triangle. The shape functions are the rational

$f(x, y) = (x^2 + y^2 - 1)\cos(y - 1)$ Straight triangles   Arc triangles
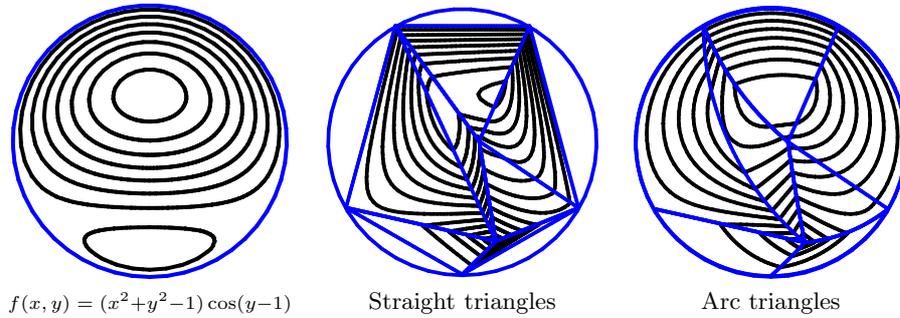
Figure 13: Approximation power of arc triangles versus straight triangles. The error for L^2 drops from 0.18 (straight) to 0.07 (arc), and for L^{max} from 0.30 to 0.15.

Bernstein basis functions. We may assume that this triangle is in standard form (i.e., the weights of the vertices are equal to one). Again, by collecting functions defined on neighbouring triangles one obtains continuous test functions. Thus, any arc triangulation can be used to define an isoparametric finite element method. The number of test functions is the same as in the case of quadratic elements defined on straight line triangles.

The use of arc triangles instead of straight line triangles offers two advantages. First, many interesting domains for numerical simulation possess piecewise circular boundaries (e.g., planar domains with circular holes, such as the famous 'plate with circular hole'). Such domains can be represented exactly by arc triangles, whereas triangles with straight edges can only approximate them. Thus, the use of arc triangles eliminates one source of discretization error in many cases, namely the approximation of the physical domain, which is required otherwise. Second, the additional degrees of freedom introduced by replacing straight lines with circular arcs provide a better distribution of the inner angles, and hence elements with better shapes. This allows to obtain smaller errors with the same number of triangles.

We demonstrate these two observations by a simple example. We consider a function on the unit disk satisfying homogeneous Dirichlet boundary conditions, and approximate it using finite elements derived from an arc triangulation and a straight triangulation, respectively, both consisting of 7 triangles. Conforming quadratic rational and quadratic polynomial finite element functions are used, respectively. In both cases, the number of degrees of freedom is equal to 10. The results are shown in Figure 13, where the function and its approximations are represented by level sets. The use of arc triangles leads to a significant reduction of the error, which is caused both by the better shape of the triangles and by the improved representation of the domain.

The case of π-triangles (Section 5) is of particular interest. Normally, for isoparametric FEM, the geometry mappings G_i are known explicitly only in

one direction. Sometimes it is desirable to use their inverses, for instance, if one wants to evaluate a test function, or even the solution, at a given point in the physical domain (which is represented as the collection of primitives). However, since G_i is known only in one direction, evaluating G_i^{-1} typically requires solutions of nonlinear systems of equations. If the primitives are π-triangles, the situation is much better. Recall from Property 4 that there exists a unique Möbius transformation which maps a straight line triangle (with the same interior angles) into the arc triangle. In this situation, the geometry mapping G_i (a rational quadratic Bézier triangle) is simply a composition of a projective mapping (a rational linear reparameterization, which is required for transforming the rational Bézier into standard form) with a Möbius transformation.

Consequently, the inverse geometry mapping G_i^{-1} is the composition of a Möbius transformation with a rational linear transformation. In this sense, π-triangles behave as good as straight triangles when used as geometric primitives for finite elements. In both cases, the geometry mappings and the inverse geometry mappings are given by simple closed form expressions. No numerical approximations are needed.

Möbius transformations are also conformal mappings. Therefore, the use of (possibly approximate) π-triangles paves the way for using geometry transformations which are approximations of conformal mappings, especially in the case of non-conforming finite elements, where no transformations of the rational Bézier triangles to standard form is required. If all domain triangles are to be chosen as equilateral ones, then one should optimize the arc triangles to get angles closer to $\frac{\pi}{3}$ than it would be possible in a straight line triangulation. In particular, arc flips (Section 4) can be applied at regions of the mesh where, typically, a few slim triangles diminish the overall quality. Note that a Möbius transformation maps circumcircles to circumcircles, and that π-triangles do not leave these circles, by Property 4.

Due to the simplicity of our optimization method (basic linear programming for which efficient large scale solvers exist), even bigger meshes can be angle-maximized within reasonable computation time.

7   Graph Drawing

Literature on drawing graphs nicely in the plane is large; see e.g. [10, 26, 31]. 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 [17, 24], including planar graphs. 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, [7, 19] and references therein, who give lower bounds and algorithms
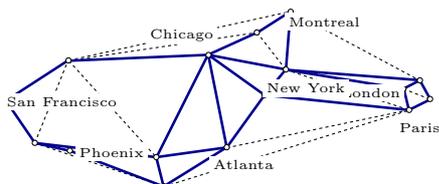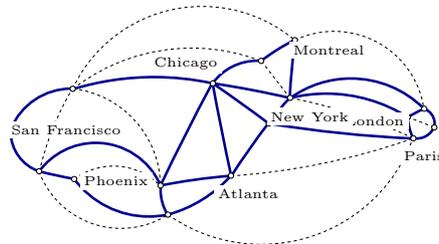
Figure 14: IP backbone graph

Figure 15: Backbone optimally redrawn

for drawing graphs on a grid with curved edges (including circular multiarcs), and [16] where a method based on physical simulation is proposed. In [15], 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 [13].

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, though.

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.[3]) 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; see Figure 16. 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

$$\delta \leq \sum_{i=1}^{k} \beta_i$$

with each β_i expressed by the corresponding straight line triangulation angle α_i

---

[3]This is possibly the reason why this simple approach has not been used in practice yet.
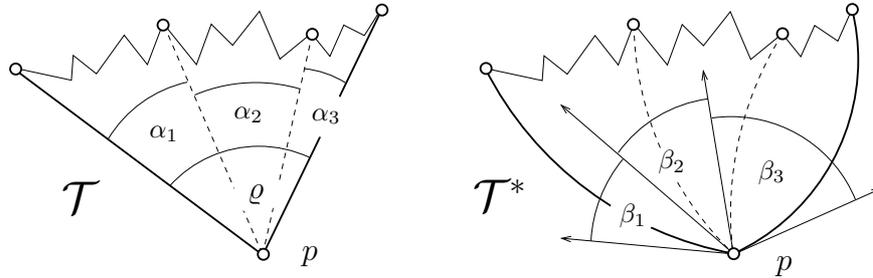
Figure 16: The angle ϱ in G as a concatenation of angles $\alpha_1, \alpha_2, \alpha_3$ in $\mathcal{T}$

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 this 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 14 and 15, and the layer graph in Figures 17 and 18 exemplify the effect of our circular arc redrawing method. The results seem satisfactory, in spite of the fact that vertices are required not to move. For geographic structures as in Figure 14, or certain graph structures arising in physics, this is quite often a desired property. Our results compare well to e.g. [16], 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.
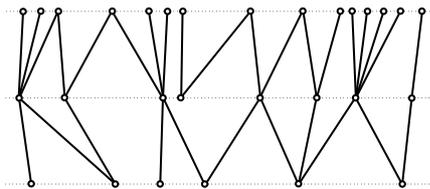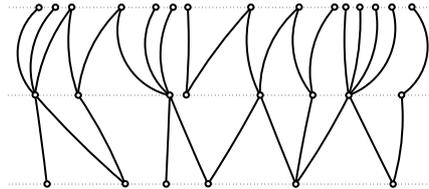

Figure 17: A 3-layer graph

Figure 18: Redrawing with arcs

8   Future Work

Circular arc triangulations are a flexible and computationally controllable structure with potential impact but, so far, with lack of interest from computational geometry. They lead to simple and fast graph redrawing procedures, and bear novel aspects for finite element methods (FEM). It is among the intentions of this paper to inspire research on theoretical foundation for the better approximation properties in FEM of arc triangles.

For the case of non-triangulated regions (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. In this context, we have the following conjecture. Note that it suffices to consider quadrangles, as the situation for regions with more than 4 arcs can be reduced to the quadrangle case.

For a convex quadrangle, let a and b be two sides that lie opposite to each other. Let the angle α be the deviation angle between the side a and an arc spanned by its two endpoints. In the same way, define the angle β for the side b.

Conjecture 1 *In the parameter space of α and β, the set of parameter pairs which correspond to non-intersecting arc pairs forms a convex region.*

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. In fact, arc triangles admit a simple generalization to three-space, by considering tetrahedral volumes with spherical faces. We call these volumes *spheroidal tetrahedra*. The edges of spheroidal tetrahedra are circular arcs, since any two spheres intersect in a circle. We will elaborate on the properties of such 3D primitives and their meshes in a forthcoming paper.

References

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Jüttler, M. Rabl, and Z. Sír. Computational and structural advantages of circular boundary representation. *Int. J. Comput. Geometry Appl.*, 21(1):47–69, 2011. `doi:10.1142/S0218195911003548`.

[2] O. Aichholzer, G. Rote, A. Schulz, and B. Vogtenhuber. Pointed drawings of planar graphs. In *Proceedings of the 19th Annual Canadian Conference on Computational Geometry, CCCG 2007*, pages 237–240, 2007. URL: `http://cccg.ca/proceedings/2007/09b5.pdf`.

[3] M. W. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F. K.-M. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, 1995.

[4] C. Boivin and C. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185–1213, 2002. `doi:10.1002/nme.542`.

[5] R. E. Burkard and F. Rendl. Lexicographic bottleneck problems. *Operations Research Letters*, 10(5):303–308, 1991. `doi:10.1016/0167-6377(91)90018-K`.

[6] B. Carré. *Graphs and Networks.* Oxford University Press, 1979.

[7] C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. In J. Kratochvíl, editor, *Graph Drawing*, volume 1731 of *Lecture Notes in Computer Science*, pages 117–126. Springer Berlin Heidelberg, 1999. `doi:10.1007/3-540-46648-7_12`.

[8] L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1–4):97–108, 1989. `doi:10.1007/BF01553881`.

[9] J. A. De Loera, J. Rambau, and F. Santos. *Triangulations – Structures for Algorithms and Applications.* Springer Verlag, 2010.

[10] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall, 1999.

[11] G. Di Battista and L. Vismara. Angles of planar triangular graphs. *SIAM J. Discrete Math.*, 9(3):349–359, 1996. `doi:10.1137/S0895480194264010`.

[12] R. S. Drysdale, G. Rote, and A. Sturm. Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs. *Computational Geometry*, 41(12):31–47, 2008. `doi:10.1016/j.comgeo.2007.10.009`.

[13] C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Lombardi drawings of graphs. *Journal of Graph Algorithms and Applications*, 16(1):85–108, 2012. `doi:10.7155/jgaa.00251`.

[14] H. Edelsbrunner, G. Rote, and E. Welzl. Testing the necklace condition for shortest tours and optimal factors in the plane. *Theoretical Computer Science*, 66(2):157–180, 1989. `doi:10.1016/0304-3975(89)90133-3`.

[15] A. Efrat, C. Erten, and S. Kobourov. Fixed-location circular arc drawing of planar graphs. *Journal of Graph Algorithms and Applications*, 11(1):145–164, 2007. `doi:10.7155/jgaa.00140`.

[16] B. Finkel and R. Tamassia. Curvilinear graph drawing using the force-directed method. In J. Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 448–453. Springer, 2005. `doi:10.1007/978-3-540-31843-9_46`.

[17] M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F. 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. `doi:10.1137/0222063`.

[18] S. Fortune. Voronoi Diagrams and Delaunay Triangulations. In D.-Z. Du and F. K.-M. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, 1995.

[19] M. T. Goodrich and C. G. Wagner. A framework for drawing planar graphs with curves and polylines. *Journal of Algorithms*, 37(2):399–421, 2000. `doi:10.1006/jagm.2000.1115`.

[20] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Dover Publications, New York, 2000.

[21] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978. `doi:10.1016/0012-365X(78)90011-0`.

[22] D. Lee and A. Lin. Generalized Delaunay triangulation for planar graphs. *Discrete & Computational Geometry*, 1(1):201–217, 1986. `doi:10.1007/BF02187695`.

[23] E. L. Lloyd. On triangulations of a set of points in the plane. In *18th Annual Symposium on Foundations of Computer Science*, pages 228–240, 1977. `doi:10.1109/SFCS.1977.21`.

[24] S. Malitz and A. Papakostas. On the angular resolution of planar graphs. *SIAM Journal on Discrete Mathematics*, 7(2):172–183, 1994. `doi:10.1137/S0895480193242931`.

[25] E. Marchi and J. A. Oviedo.   Lexicographic optimality in the multiple objective linear programming: The nucleolar solution. *European Journal of Operational Research*, 57(3):355–359, 1992. `doi:10.1016/0377-2217(92)90347-C`.

[26] T. Nishizeki and M. S. Rahman. *Planar Graph Drawing*. World Scientific, 2004.

[27] D. Pedoe. *A Course Geometry for Colleges and Universities*. Cambridge University Press, 1970.

[28] G. Rote. *Two solvable cases of the traveling salesman problem*. PhD thesis, TU-Graz, Institute for Mathematics, 1988.

[29] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *11th International Meshing Roundtable*, pages 115–126, 2002.

[30] R. Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, Oct. 1981. `doi:10.1145/322276.322288`.

[31] K. Sugiyama. *Graph Drawing and Applications for Software and Knowledge Engineers*. World Scientific, 2002.