Maximizing the Minimum Angle with the Insertion of Steiner Vertices

Shankar P. Sastry*

Abstract

We consider the problem of inserting a vertex inside a star-shaped input polygon at the location that maximizes the minimum angle in the resulting triangulation. An existing polynomial-time algorithm solves for the intersection of three polynomial surfaces (a prior paper indicates that these are eighth-degree polynomials) and computes the maxima of the curve of intersection of two such surfaces to solve the problem. We developed a similar technique through the geometric insight that at least two angles (typically, three) of the triangulation have to be identical at the optimal location. We combinatorially process the angles to compute the optimal location in each case. The worst-case complexity of the algorithm remains $O(n^3 \log n)$, but it is much easier to implement partly because our algorithm requires the solutions of an (at most) eighth-degree, *univariate* polynomial for each combination of the angles. We also modified the algorithm to lower the *expected* running time to $O(n^2)$ using a recursive, randomized algorithm for LP-type problems. We extend the algorithm by imposing constraints on the location of the Steiner vertex and solving the constrained optimization problem in a similar manner. We also extend the algorithm to simultaneously insert two vertices by considering all possible topologies and ensuring that the necessary conditions for local maxima are satisfied.

1 Introduction

We consider the problem of positioning a Steiner vertex that is connected to all the vertices of a star-shaped input polygon such that it maximizes the minimum angle in the resulting triangulation. The point has to be inside a convex feasible region so that no triangle lies outside the polygon [6, 7]. The algorithm that solves the problem may be used in Delaunay mesh refinment algorithms to insert addition vertices into a mesh at an optimal location or to carry out smoothing of the mesh vertices to improve the mesh quality.

In the context of mesh smoothing, Freitag and Plassman [6] developed a quadratic programming-based active-set approach to maximize the minimum angle by observing that it is a convex optimization problem with a nondifferentiable objective function. Recent research by Aronov et al. [2] has yielded polynomial-time algorithms to maximize the minimum angle. Their recent attempt [3] solves an LP-type problem by a computing the lower envelope of bivariate functions after solving eighth-degree polynomial equations. Their latest attempt [4] solves for the intersection of three bivariate polynomial surfaces or computes the maxima of the curve of intersection of two such surfaces. The surface intersection represents locations at which two or three angles of the triangulation are equal. These algorithms are described in Section 2. The active-set approach is easy to implement, but it is numerical in nature. The LP-type approach solves the problem exactly, but computing the lower envelope of bivariate functions is not easy. To obtain a maximum of the curve of intersection of two surfaces or a point of intersection of three surfaces is also hard in comparison with solving a univariate polynomial equation.

We use concepts from nondifferentiable optimization to develop our algorithm. We infer that the minimum angle is shared by two or more angles because it is always possible to improve the minimum angle (at the cost of the "better" angles) by appropriately moving the vertex. A brief background on this topic is presented in Section 3. Our approach is combinatorial and is similar to [4]. We take all combinations of possible locations of the minimum angles and return the "best" location of the Steiner vertex. Our algorithm is described in detail in Section 4. We observe that the number of possible combinations is $O(n^3)$, where n is the number of segments in the polygonal cavity, and we determine the optimal location by considering only those points where the gradients are suitably directed. We improve the expected running time to $O(n^2)$ using a recursive, randomized technique adapted from Clarkson's algorithm [5]. We also provide an algorithm for a generalized constrained optimization problem. We also extend the algorithm for the insertion of two vertices. These extensions are discussed in Section 5.

Our work has also provided some insight into local mesh quality improvement by vertex movement. Section 6 includes a discussion about improvements to our algorithm and future work.

*Scientific Computing and Imaging Institute, University of Utah, sastry@sci.utah.edu

2 Related Work

In this section, we discuss two of the most relevant algorithms in the evolution of our algorithm. The first algorithm is by Aronov and Yagnatinsky [4]. Although we developed our algorithm independently, their algorithm is similar to ours. The main difference is the geometric intuition; we believe ours is simpler to comprehend, and, therefore, easier to extend to tougher cases, as we shall see in Section 5. The second algorithm is the active set-based technique for mesh quality improvement [6]. We use concepts from this algorithm to accelerate our algorithm for realistic cases.

2.1 The Bivariate Surface Algorithm

Aronov and Yagnatinsky [4] begin by defining a feasible region in the domain where it is possible to construct a triangulation whose minimum angle is some z. The feasible region is bounded by circular arcs (that circumscribe an edge so that the angle subtended at the circumference is z) and line segments (that emanate at an angle z from the edges). The feasible region may be empty. For each value of z, we have arcs and lines on the domain. When they are lifted in the third dimension for all values of z, they form surfaces. In a prior paper [3], they have used eighth-degree, bivariate polynomials to define similar surfaces. They then claim that the optimal point is at a location where three surfaces intersect at a point or the maxima of curve formed by the intersection of two surfaces. The feasible maximum over the points returns the optimal triangulation. We use the gradients of the objective function to prove the claim in the language of nondifferentiable optimization and compute those points on the 2D plane. Our functions are eighth-degree, univariate polynomials.

2.2 The Active Set Method

Freitag and Plassman [6] solve the problem through vertex movement dictated by a numerical optimization algorithm. At a given location, they define the active set as the set of angle(s) with the minimum value. They use the gradient of the function defining the angle(s) to compute the direction in which the vertex should move in order to optimize it. The descent direction is computed by considering all convex combinations of active set gradients and choosing the one that minimizes the magnitude. If the active set changes during the vertex movement, the gradients and descent direction are recomputed. Similarly, we consider only those combinations of angles in which all the current active-set angle(s) are present.

3 Background

Since our objective function is nondifferentiable, we provide a background on necessary conditions[1] for the constrained optimization of such functions [8], which will also help us extend the algorithm for more complicated cases. In the subsequent sections, we will provide a geometric intuition behind the material presented here.

We assume that our objective function, $f(x)$, is defined as a minimum over a set of functions $f_i(x)$, $1 \leq i \leq n$. We define an active set at a location x as the maximum subset of functions whose value is equal to $f(x)$. A vector, $\vec{g}$ is defined as a subgradient of f at x if $\exists$ an ϵ neighborhood such that $f(x+\epsilon) - f(x) \geq \vec{g}^T\epsilon$. If only one function is present in the active set, the gradient of that function is the subgradient of f at x. If multiple functions are present, any *convex* combination of the gradients of the functions is also the subgradient of f at x. The set of all subgradients is called the subdifferential, $\partial f(x)$, of f at x.

For the unconstrained case, the necessary condition for x^* to be a local optimum is $0 \in \partial f(x^*)$. In other words, some convex combination of gradients of the functions in the active set should vanish, i.e., if $d < n+1$ functions are present in the active set in an $n-$dimensional space, the gradients should lie on some $d-$dimensional hyperplane and span both sides of all other $d-$dimensional hyperplanes at the origin. If $d \leq n+1$, the origin should lie inside the convex hull of the gradient vectors, which is easy to verify for any dimension.

For the constrained case, where $h_i(x) \leq 0$, $i \in [1,k]$, are the set of inequality constraints, a local optimum should satisfy the Karush-Kuhn-Tucker (KKT) conditions for nondifferentiable functions, i.e., $h_i(x^*) \leq 0$, $\lambda_i^* \geq 0$, $\lambda_i^* h_i(x^*) = 0$, and $0 \in \partial f(x^*) - \sum_{i=0}^{(k-1)} \lambda_i \partial h_i(x^*)$. These conditions imply that some convex combination of the gradients of the function(s) of the active set and the gradients of all the active constraint(s) should vanish.

It can also be shown that if some convex combination of m $n-$dimensional vectors vanishes (where $m \geq n+1$), there exists a set of $n+1$ vectors (from the original set of m vectors) for which some convex combination also vanishes. Thus, it is sufficient to consider all possible combinations of $n+1$ vectors rather than considering all combinations of $m > n+1$ vectors.

4 Algorithm

Based on the theory presented in the last section, we need to compute all possible locations where a set of angles are equal and some convex combination of their gradients vanishes. Since this is a 2D problem, we need to

[1] They hold true for a maximum, minimum, and saddle points.

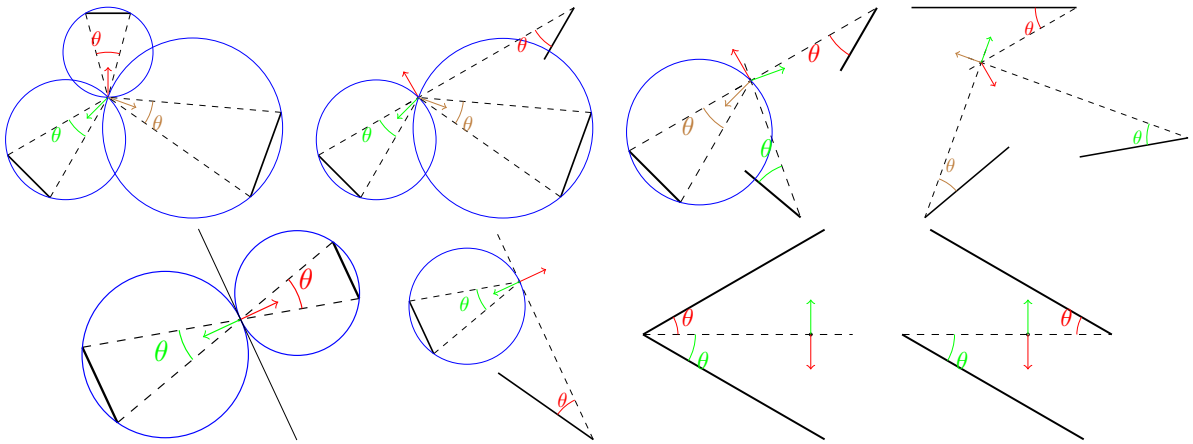

Figure 1: All possible ways in which a local maximum can occur. The thick edges are the input edges, and the dashed edges are the edges in the final triangulation. The circles (and some dashed edges) are potential locations of the Steiner vertex that result in an angle θ at some vertex. The arrows indicate the gradient for each angle. Notice how their convex combination can result in a zero vector.

only consider all possible sets of (no greater than) three angles in order to determine all possible local maxima. We also know that it is a convex optimization problem, so there exists only one local maximum, which is also the global maximum. Our algorithm considers all cases combinatorially and computes the global maximum.

In Fig. 1, we present all possible ways in which a set of three angles in the triangulation can be equal. If the input polygon has n sides, there are $2n$ angles on the edges and n angles at the Steiner vertex. The sets of two and three angles can be composed of either type of angles. All seven (eight, if the last two cases involving two angles on the edges are considered different) ways in which we can achieve equiangular configurations are shown in Fig. 1 along with the gradients for each of the angles.

In the top row of Fig. 1, we consider cases where three angles are equal. Consider the locus of points that results in an angle being θ. For angles at the Steiner vertex, the locus is a circle, and for angles at the edge, the locus is a straight line. The values of θ for which the three curves are concurrent need to be computed. In the bottom row, we consider cases where two angles are equal, where their gradients must be anti-parallel, i.e., the curves must be intersect tangentially.

We will now describe how a univariate polynomial function can be constructed with $\lambda = \cot\theta$ as the only variable. This is also called a rational univariate representation. The solution of the equation gives us the θ values at which a maximum can occur. The actual polynomial, however, is tedious to write down, but it is straightforward, so we leave that as an algebra exercise. First, we need to compute the circle that subtends an angle θ at the chord formed by an edge. The center of the circle should be at a distance λl from the mid point of the edge, where $2l$ is its length (see Fig. 2). The line that is at an angle θ can also easily be computed as shown in the figure. Second, as we now have the equations of the circles and/or lines (as functions of x, y, and λ), we compute their points of intersection in a pairwise manner. The point of intersection of two lines (as a function of λ) can be computed as a function of second-degree polynomials. The points of intersection of a line and a circle can be similarly computed. For two circles, however, we compute the equation of the line on which the two circles intersect.

We will consider each case in Fig. 1 separately, going from left to right (top row first). For the three angles at the Steiner vertex, we have the equations for three lines. We compute their points of intersection in a pairwise manner and equate one of their coordinates to get an eight-degree polynomial in λ. For two angles at the Steiner vertex and one at an edge, we have two lines whose point of intersection can be computed as a function of λ. We solve for the λ at which the angle subtended by one of the edges is θ. For an angle at the vertex and two at the edges, we are given a point. We again solve for λ at which the angle subtended by the edge is θ. When the angles are on three edges, we have three pairwise points of intersection. We equate the coordinates of the points of intersection to get an equation in λ. For the first two cases in the bottom row, the value of λ can be computed by ensuring that the roots of the resulting quadratic equation are equal because they intersect tangentially. The last two cases, however, are degenerate, and therefore, are not considered here; they are considered when three angles are considered.

To determine the maximum, we consider all sets of two and three angles and look for points at which their values are equal *and* some convex combination of their

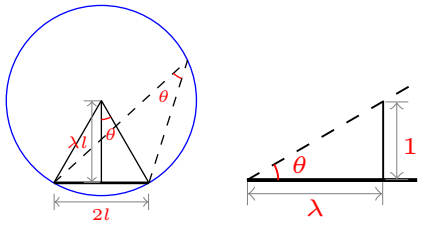

Figure 2: For a given $\lambda = \cot\theta$, the locus of points is either a circle or a line.

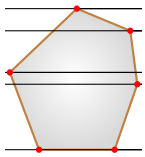

Figure 3: Preprocessing: determine the feasible region [7], and decompose the region into triangles/trapezoids in $O(n)$ time. For each point, determine the potential triangle/trapezoid using the binary search technique in $O(\log n)$ time, and determine if the point is inside or outside the triangle/trapezoid.

gradients vanish, which takes $O(n^3)$ time. Points that are outside the convex feasible region of the star-shaped polygon need to be discarded. It is possible to determine if a point is inside or outside the feasible region in $O(\log n)$ time after a line sweeping-based preprocessing algorithm (see Fig. 3) that is carried out at the beginning. Our global maximum has to occur at one of these locations that has not been discarded. The objective function is convex inside the feasible region [6]. At the points corresponding to smaller λ values (larger θ values), at least one of the angles associated with the largest λ value will be of less than the optimal value. Thus, the largest λ (smallest θ) corresponds to the global maximum. If not, it results in a contradiction. The worst-case complexity is, therefore, $O(n^3 \log n)$.

To improve the expected time, we adapt Clarkson's recursive algorithm [5] for our problem in which our $O(n^3)$ algorithm is a subroutine that terminates the recursion when n is small. In this algorithm, a subset of angles (the size of the subset is proportional to $O(\sqrt{n})$) is randomly chosen, and the optimal vertex location for those angles is recursively found. If the number of angles is small, our algorithm is used to carry out the optimization. For the optimal location found, if the minimum angle among the chosen subset also the minimum angle when all other angles are considered, we are done. If not, $O(\sqrt{n})$ more angles are randomly chosen from the set of angles that are smaller than the optimal angle for the chosen subset, and the algorithm is repeated. Although Clarkson's algorithm takes linear time for linear programming problems, as shown by Amenta et al. [1],

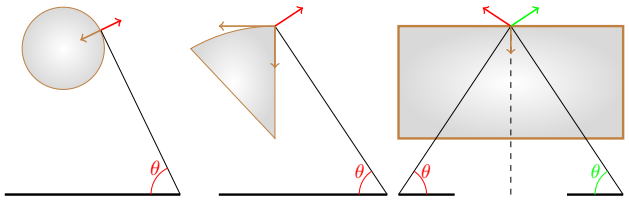

Figure 4: Three possible locations of a local maximum in a constrained problem. The vertex is constrained to be in the gray region. The gradients of the constraint(s) and the angle(s) are also shown.

the expected running time for our problem is quadratic because computing the minimum angle itself takes linear time.

5 Extensions

5.1 Constrained Optimization

The motivation for solving the constrained optimization problem is to gurantee termination of Delaunay refinement algorithm. Delaunay refinement terminates only if the newly inserted vertex is at least at a certain distance from other vertices in the domain. Thus, our inserted vertex must be sufficiently distant from the edges and vertices of the input polygon. Here, we consider a set of more general geometric constraints. Based on the theory of constrained optimization of nondifferentiable functions, we need to compute the locations where some convex combination of the gradients of the angle(s) and the constraint(s) vanishes. The three ways in which this can happen are depicted in Fig. 4. If the locus of points that results in a constant angle is tangential to a curve representing a constraint, the point of intersection is of interest. Similarly, when two constraint curves intersect and the gradient of an angle is suitably directed, a local maximum is present at the point of intersection. Also, the locus of points where two angles of the resulting triangulation are equal may meet a constraint. If the gradients are suitably directed, we have a local maximum. If there are $O(n)$ angles and $O(m)$ constraints, it takes $O(n^2m + nm^2)$ time to consider all cases. These cases are in addition to the cases for the unconstrained problem. For each case, it takes $O(m)$ time to determine if the location is within the constraints and $O(n)$ time to compute the minimum angle. This can be accelerated using the techniques described in the previous section. If the constraints are nonconvex or disjoint, we have to consider all combinations. We may also cache the sorted order of the angles in previous combinations to quickly discard a potential solution.

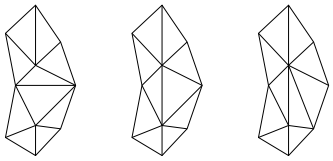

Figure 5: Three of the possible $O(n^2)$ topologies for insertion of two Steiner vertices.

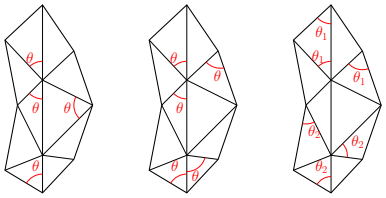

Figure 6: The possible choice of angles for a given topology; (left) underdetermined system of equations; (middle) zero-dimensional system of equations; (right) overdetermined system of equations.

5.2 Insertion of Two Vertices

In the case of inserting just one vertex, the topology of the resulting triangulation (connectivity of the vertices) is fixed, i.e., the Steiner vertex is connected to all the vertices of the input polygon. If two Steiner vertices are to be inserted, we have to consider multiple topologies as shown in Fig. 5. The two Steiner vertices may or may not be connected to each other, and each Steiner vertex is connected to a subset of polygon vertices. Exactly two vertices are present in both subsets (to avoid overlapping trianglesr). The common vertices divide the polygon into two sides, and each Steiner vertex is connected to all polygon vertices on its side. There are $O(n^2)$ such topologies to consider.

For the insertion of just one vertex, we actually solve a system of trivariate polynomials in x, y, and λ that we managed to reduce to a single variable. The system consists of three equations when three angles are being equated. When two angles are being equated, we still have three equations: two from the angles and an additional equation constraining the gradients to be anti-parallel. In the new case, we have a system of equations with five variables (x_1, y_1, x_2, y_2, and λ). Thus, five polynomial equations are needed to solve the problem. We also know that three equations need to be associated with the angles of triangles at each of the two Steiner vertices at the local maxima. Our choice of the angles should respect these conditions.

Consider the case where the Steiner vertices are not connected, which reduces to two instances of the single vertex insertion problem for each such topology. Thus, the optimal locations can be computed in $O(n^5 \log n)$ in the worst case.

Let us now consider the more interesting case where the Steiner vertices are connected. In this case, we need to choose two or three angles in the triangles associated with each of the Steiner vertices. Fig. 6 shows three possible ways this can be achieved. On the left, two of the angles chosen are common to both Steiner vertices, and only four angles have been chosen in total, which results in an underdetermined system of polynomial equations, which have an infinite number of solutions. As a result, the optimal value of λ will come at the cost of some other triangle. In the middle, only one angle is common to the Steiner vertices, and five angles have been chosen altogether. Thus, we have five equations with five variables, which is called a zero-dimensional system because it has a finite number of solutions. We have to consider all such cases in our algorithm. On the right, we have chosen six angles with each Steiner vertex being associated with three of them. This is an overdetermined system if we insist that the value of θ be the same for all angles. If they are different (as in the figure), we are solving a problem that is similar to the case where the two Steiner vertices are not connected. Note that the discussion above also holds when only two angle are chosen for a vertex because we have an additional equation in the form of their gradients being anti-parallel.

In order to determine the computational complexity, we have to analyze the number of ways in which these angles can be chosen. Let the degree of the vertices be d_1 and d_2, $d_1 + d_2 = n + 2$. We do not consider the case on the left in Fig. 6 because it results in an underdetermined system. For the case in the middle, there are only six ways in which the common angle can be chosen. There are $O(d_1^2)$ and $O(d_2^2)$ ways in which the other two angles can be chosen. The number of combinations is $O(n^4)$. The overall complexity of the number of possible cases for a given topology is $O(n^4)$ The number of possible topologies is $O(n^2)$. Thus, we consider $O(n^6)$ cases in total. Since feasible regions are hard to find in the fourth dimension, we carry out a linear search for each case to determine the smallest angle and, subsequently, the global maximum in $O(n^7)$ time. For the case on the right, the angles in the common triangles are not being considered. Therefore, they are two independent subproblems as in the case where the vertices are not connected. If the minimum angle is part of the common triangles, it will be handled in one of our earlier cases. If not, it can be handled in the case where the two vertices are not connected. Therefore, we do not consider this case in our analysis here.

In order to improve the expected running time, we may use a method similar to Clarkson's algorithm that we used for the single vertex insertion case. Note that it has to be used for every possible topology to obtain the most optimal location. We can extend this algorithm to get a "near-optimal" solution by considering changes in the topology (flip edges) only if it improves an existing

triangulation, but we cannot guarantee anything about the approximation of the solution obtained.

It is also possible to construct a univariate polynomial that is equivalent to the system of five pentavariate polynomials described earlier in this section. Consider the two angles of a vertex that do not belong to the common triangles. The locus of points that results in equal angles at the two corners can be described as a parametric equation in λ for both vertices, which fixes the location of the two vertices as a function of λ. We get a polynomial equation by enforcing the condition that the angle in the common triangle is also $\cot^{-1}\lambda$. In the case where only two angles are equated at a vertex, we need an additional parameter, say, t, to fix the location of one of the vertices. It is possible to compute the location of the other vertex as a function of λ and t and impose the angle condition as well as the gradient condition to obtain a system of two bivariate polynomials in λ and t.

6 Conclusion and Future Work

We used the theory of nondifferentiable optimization to develop our algorithm and extend it to tougher problems. Due to the firm theoretical ground, this research can be extended to investigate the complexity of the optimization of angles when an arbitrary number of vertices are inserted in 2D as well as 3D meshes. It can easily be seen that there is a combinatorial explosion of possible mesh topologies as the number of vertices increases, and they require solutions of larger systems of multivariate polynomials. A practical algorithm should account for this and prune the search space appropriately. We think that the prior research in mesh quality improvement [9] will help us in analyzing these problems. Mesh quality improvement techniques typically optimize one vertex at a time. Since we now know that two or three angles in 2D meshes (two, three, or four angles in 3D meshes) are identical in an optimal patch, reordering of vertices over which the optimization is carried out will bring about greater improvement in mesh quality. Prior research [9] has not taken advantage of this property. Our next research will look in this direction.

As mentioned in Section 4, the expected running time of our algorithm is improved by choosing a random subset of angles from the given angles in each subroutine. It is likely that the minimum angle occurs at the angle opposite to the shortest input edge or the smallest angle in the input polygon. Perhaps, assigning a higher probablity for such angles may yield faster results. Also, given an optimal Steiner vertex location for a subset of angles, we may choose additional angles by assigning a higher probablity to those angles (in the original set) that are smaller. A theoretical or emperical analysis of the improvement in the expected running time due to such modifications is also an interesting topic of research.

Acknowledgment

The work of the author was supported in part by the NIH/NIGMS Center for Integrative Biomedical Computing grant 2P41 RR0112553-12 and a grant from ExxonMobil. The author would also like to thank Christine Pickett, an editor at the University of Utah, for proofreading and finding numerous typos in the paper.

References

[1] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *J. Algorithms*, 30(2):302 – 322, 1999.

[2] B. Aronov, T. Asano, and S. Funke. Optimal triangulations of points and segments with steiner points. *Int. J. Comput. Geom. Ap.*, 20(01):89–104, 2010.

[3] B. Aronov and M. V. Yagnatinsky. How to place a point to maximize angles. In *Proc. of the 25th Canadian Conference on Computational Geometry*, 2013.

[4] B. Aronov and M. V. Yagnatinsky. Quickly placing a point to maximize angles. In *Proc. of the 26th Canadian Conference on Computational Geometry*, 2014.

[5] K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, Mar. 1995.

[6] L. A. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *Int. J. Numer. Meth. Engrg.*, 49(1-2):109–125, 2000.

[7] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, July 1979.

[8] A. P. Ruszczynski. *Nonlinear optimization*. Princeton University Press, 2006.

[9] S. M. Shontz and P. Knupp. The effect of vertex reordering on 2D local mesh optimization efficiency. In *Proc. of the 17th International Meshing Roundtable*, pages 107–124. Springer Berlin Heidelberg, 2008.