ELSEVIER

Computational Geometry 23 (2002) 85–98

Computational Geometry
Theory and Applications

www.elsevier.com/locate/comgeo

Higher order Delaunay triangulations ☆

Joachim Gudmundsson [a], Mikael Hammar [b], Marc van Kreveld [a,*]

[a] *Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands*
[b] *Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden*

Received 21 February 2001; received in revised form 27 April 2001; accepted 18 May 2001

Communicated by K. Mehlhorn

Abstract

For a set P of points in the plane, we introduce a class of triangulations that is an extension of the Delaunay triangulation. Instead of requiring that for each triangle the circle through its vertices contains no points of P inside, we require that at most k points are inside the circle. Since there are many different *higher-order Delaunay triangulations* for a point set, other useful criteria for triangulations can be incorporated without sacrificing the well-shapedness too much. Applications include realistic terrain modeling and mesh generation.

Keywords: Delaunay triangulation; Terrain modeling

1. Introduction

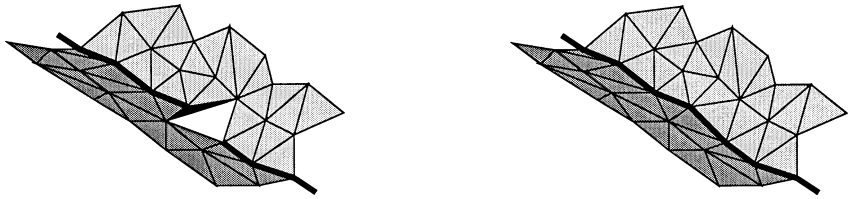
One of the most well-known and useful structures studied in computational geometry is the Delaunay triangulation [8,12,21]. It has applications in spatial interpolation between points with measurements, because it defines a piecewise linear interpolation function. The Delaunay triangulation also has applications in mesh generation for finite element methods. In both cases, the usefulness of the Delaunay triangulation as opposed to other triangulations is the fact that the triangles are well-shaped. It is well known that the Delaunay triangulation of a set P of points maximizes the smallest angle, over all triangulations of P.

One specific use of the Delaunay triangulation for interpolation is to model elevation in Geographic Information Systems. The so-called *Triangulated Irregular Network*, or TIN, is one of the most common

☆ This research is partially supported by the ESPRIT IV LTR Project No. 21957 (CGAL).

* Corresponding author.
E-mail addresses: joachim@cs.uu.nl (J. Gudmundsson), mikael@cs.lth.se (M. Hammar), marc@cs.uu.nl (M. van Kreveld).

0925-7721/02/$ – see front matter
PII: S0925-7721(01)00027-X



(a) (b)

Fig. 1. Artificial dam that interrupts a valley line (a), and a correct version obtained after one flip (b).

ways to model elevation. Elevation is used for hydrological and geomorphological studies, for site planning, for visibility impact studies, for natural hazard modeling, and more.

Because a TIN is a piecewise linear, continuous function which is generally not differentiable at the edges, these edges play a special role. In elevation modeling, one usually tries to make the edges of the TIN coincide with the ridges and valleys of the terrain. Then the rivers that can be predicted from the elevation model are a subset of the edges of the TIN. When one obtains a TIN using the Delaunay triangulation of a set of points, the ridges and valleys in the actual terrain will not always be as they appear in the TIN. The so-called 'artificial dam' in valleys is a well-known artifact in elevation models, see Fig. 1. It appears when a Delaunay edge crosses a valley from the one hillside to the other hillside, creating a local minimum in the terrain model slightly higher up in the valley. It is known that in real terrains such local minima are quite rare [14]. These artifacts need to be corrected, if the TIN is to be used for realistic terrain modeling [23], in particular for hydrological purposes [19,20,24]. If the valley and ridge lines are known, these can be incorporated by using the constrained Delaunay triangulation [7,10,18]. The cause of problems like the one mentioned above may be that the Delaunay triangulation is a structure defined for a planar set of points, and does not take into account the third dimension at all. One would like to define a triangulation that is both well-shaped and has some other properties as well, like avoiding artificial dams. This leads us to define *higher-order Delaunay* (*HOD*) *triangulations,* a class of triangulations for any point set P that allows some flexibility in which triangles are actually used. The Delaunay triangulation of P has the property that for each triangle, the circle through its vertices has no points of P inside. A k-order Delaunay (k-OD) triangulation has the relaxed property that at most k points are inside the circle. The idea is then to develop algorithms that compute some HOD triangulation that optimizes some other criterion as well. Such criteria could, for example, be minimizing the number of local minima, or minimizing the number of local extrema. The former criterion deals with the artificial dam problem, and the latter criterion also deals with interrupted ridge lines. For finite element method applications, criteria like minimizing the maximum angle, maximum area triangle, and maximum degree of any vertex may be of use [3–5].

In Section 2 we define HOD triangulations and show some basic properties. In Section 3 we give an algorithm to compute which edges can be included in a k-OD triangulation. The algorithm runs in $O(nk^2 + n \log n)$ expected time. In Section 4 we consider 1-OD triangulations, and prove more specific, useful results in this case. In Section 5 we give the applications. We show that for 1-OD triangulations, most of the criteria we study can be optimized in $O(n \log n)$ time. We also give an approximation algorithm for the case of k-OD triangulations. Directions for further research are given in Section 6.

2. Higher-order Delaunay triangulations

We first define higher-order Delaunay edges, higher-order Delaunay triangles, and higher-order Delaunay triangulations. Given two vertices u and v we will denote by $\overline{uv}$ the edge between u and v and by $\overrightarrow{uv}$ the directed line segment from u to v. Furthermore, the unique circle through three vertices u, v and w is denoted $C(u, v, w)$, and the triangle defined by u, v and w is denoted $\triangle uvw$. We will assume throughout this paper that P is non-degenerate, that is, no three points of P lie on a line and no four points of P lie on a circle.

Definition 1. Let P be a set of points in the plane. For $u, v, w \in P$:

- An edge $\overline{uv}$ is a *k-order Delaunay edge* (or *k-OD edge*) if there exists a circle through u and v that has at most k points of P inside, Fig. 2.
- A triangle $\triangle uvw$ is a *k-order Delaunay triangle* (or *k-OD triangle*) if the circle through u, v and w has at most k points of P inside.
- A triangulation of P is a *k-order Delaunay triangulation* (or *k-OD triangulation*) of P if every triangle of the triangulation is a k-OD triangle.

For $k = 0$, the definitions above match the usual Delaunay edge and triangle definitions.

Lemma 1. *Let P be a set of points in the plane.*

(a) *Every edge of a k-OD triangle is a k-OD edge.*
(b) *Every edge of a k-OD triangulation is a k-OD edge.*
(c) *Every k-OD edge with $k > 0$ that is not a 0-OD edge intersects a Delaunay edge.*

Proof. For the first part, consider the circle through the three vertices of the k-OD triangle. This circle contains at most k other points of P. Since no four points are co-circular, we can slightly change the circle by letting one of the vertices loose and leaving it to the outside, while acquiring no other point of P. This shows that the edge connecting the two other vertices of the triangle is a k-OD edge. The second part of the lemma follows immediately from the first part, and the third part is trivial. □

Note that the converse of Lemma 1(a) is not true. Not every triangle consisting of three k-OD edges is a k-OD triangle. Fig. 3(a) shows an example where three 1-OD edges form a 3-OD triangle.

The following observation is a reformulation of Lemma 9.4 in [8].

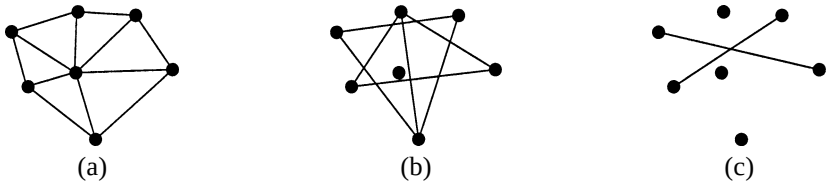


Fig. 2. (a) The 0-OD edges (b) extended with the new 1-OD edges, and (c) the new 2-OD edges.

(a) (b)

Fig. 3. (a) Not every triangle with three 1-OD edges is a 1-OD triangle. (b) Not every 1-OD triangle ($\triangle uvx$) can be included in a 1-OD triangulation.

Observation 1. *For any k-OD edge $\overline{uv}$ and any Delaunay edge $\overline{sp}$ that intersects $\overline{uv}$, the circle $C(u, v, s)$ contains p.*

The following corollary is a direct consequence of Observation 1.

Corollary 1. *Consider a k-OD edge $\overline{uv}$. Any circle through u and v that does not contain any vertices to the left* (*right*) *of $\overrightarrow{vu}$ contains all vertices to the right* (*left*) *of $\overrightarrow{vu}$ that are incident to Delaunay edges that intersect $\overline{uv}$.*

There is a close connection between k-OD edges and higher-order Voronoi diagrams.

Lemma 2. *Let P be a set of n points in the plane, let $k \leqslant n/2 - 2$, and let $u, v \in P$. The edge $\overline{uv}$ is a k-OD edge if and only if there are two incident faces, F_1 and F_2, in the order-$(k+1)$ Voronoi diagram such that u is in the set of points that induces F_1 and v is in the set of points that induces F_2.*

Proof. For two points u and v in P, let m be the smallest integer such that the bisector of u and v appears in the order-m Voronoi diagram. Since on the one side of the line through u and v there are at least $n/2 - 1$ points of P, the bisector of u and v will appear in all higher-order Voronoi diagrams from order-m up to order-$(n/2 - 1)$. This bisector separates two faces representing subsets of P as stated in the lemma. □

Since the worst case complexity of order-$(k+1)$ Voronoi diagrams is $O((k+1)(n-k-1))$ [17], it follows that $O(n + nk)$ pairs of points can give rise to a k-OD edge. These pairs can be computed in $O(nk2^{c\log^* k} + n\log n)$ expected time [22].

A natural question to ask is whether any k-OD edge or any k-OD triangle can be part of some k-OD triangulation. Put differently, can k-OD edges exist that cannot be used in any k-OD triangulation? Indeed, such edges (and triangles) exist. In the next section we will give a method to test for any k-OD edge if it can be extended to a k-OD triangulation. Fig. 3(b) shows an example where $\triangle uvx$ is a 1-OD triangle that cannot be included in a 1-OD triangulation since $\triangle uvy$ is not a 1-OD triangle. To distinguish between "useful" and "non-useful" k-OD edges we use the following definition.

Definition 2. Let P be a set of points in the plane. A k-OD edge $\overline{uv}$ with $u, v \in P$ is *useful* if there exists a k-OD triangulation that includes $\overline{uv}$. A k-OD triangle $\triangle uvw$ with $u, v, w \in P$ is *valid* if it does not contain any point of P inside and its three edges are useful.

3. Finding all useful k-OD edges

In this section we give an efficient way to compute all useful k-OD edges of a point set P. We also show how a useful k-OD edge can be included in a k-OD triangulation.

To decide if a k-OD edge, $\overline{uv}$ can be included in some k-OD triangulation one has to check if the point set can be k-OD triangulated with the edge $\overline{uv}$. We will show that it suffices to check only two circles through the points u, v, and count how many points these contain. For simplicity, we will assume, without loss of generality, that $\overline{uv}$ is vertical and that u is above v.

Lemma 3. *Let $\overline{uv}$ be a k-OD edge and let s_1 be the point to the left (right) of $\overrightarrow{vu}$, such that the circle $C(u, s_1, v)$ contains no points to the left (right) of $\overrightarrow{vu}$. If $\triangle us_1v$ is not a k-OD triangle then $\overline{uv}$ is not useful.*

Proof. Assume that $\triangle us_1v$ is not a k-OD triangle. It follows that the circle $C(u, s_1, v)$ contains more than k points to the right of $\overrightarrow{vu}$. Suppose that still a k-OD triangulation T exists that includes $\overline{uv}$. Let $\triangle us_iv$ be the triangle in T to the left of $\overrightarrow{vu}$. Then point s_i must lie such that $\overline{vs_i}$ intersects $\overline{us_1}$ or such that $\overline{us_i}$ intersects $\overline{vs_1}$. By symmetry we may assume the latter to be the case, see Fig. 4(a). Let p_1 and p_2 be two points in P such that $\triangle s_1p_1p_2$ is in T and it intersects the triangle $\triangle uvs_1$. Possibly, $p_1 = u$, or $p_2 = s_i$, or both. The circle $C(s_1, p_1, p_2)$ includes the whole part of the circle $C(u, v, s_1)$ to the right of $\overrightarrow{vu}$ since p_1 and p_2 lie outside $C(u, v, s_1)$ (one of them may lie on the circle). Hence, $C(s_1, p_1, p_2)$ contains at least $k + 2$ points: $k + 1$ points that are also inside $C(u, s_1, v)$, and furthermore the point v. This implies that $\triangle us_iv$ cannot be a k-OD triangle either, contradicting the assumption that a k-OD triangulation exists. □

We would like to go a step further than the result of Lemma 3, namely, prove that if the "first" triangle on the left side of $\overrightarrow{uv}$, $\triangle us_1v$, is a k-OD triangle, and the symmetrically defined triangle on the right side of $\overrightarrow{vu}$ is also a k-OD triangle, then $\overline{uv}$ is useful. We show this result constructively, by giving a method that gives a k-OD triangulation that includes $\overline{uv}$. It appears that we only have to compute a triangulation of a region near $\overline{uv}$, called the hull of $\overline{uv}$. The hull is defined as follows.

Definition 3. The *hull* of a k-OD edge $\overline{uv}$ is the closure of the union of all Delaunay triangles whose interior intersects $\overline{uv}$, Fig. 4(b).

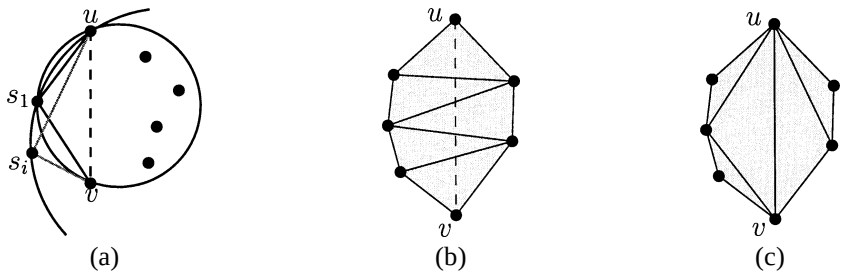


Fig. 4. (a) If $\overline{uv}$ is useful then $\triangle us_1v$ is a useful k-OD triangle. (b) The hull of k-OD edge $\overline{uv}$ (shaded), and (c) the completion of the hull of $\overline{uv}$.

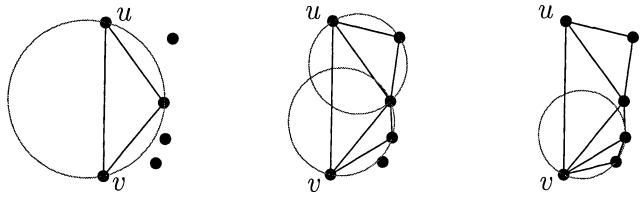


Fig. 5. Greedy triangulation to the right of $\overrightarrow{uv}$.

The following algorithm computes a triangulation of the hull. Let $\overline{uv}$ be a k-OD edge. Let p_1 be the point to the right of $\overrightarrow{vu}$ such that the part of $C(u, v, p_1)$ to the right of $\overrightarrow{vu}$ is empty. Note that p_1 must be a vertex on the boundary of the hull of $\overline{uv}$. Add the two edges $\overline{up_1}$ and $\overline{vp_1}$ to the graph. Continue like this recursively for the two edges $\overline{up_1}$ and $\overline{vp_1}$ until the hull of $\overline{uv}$ to the right of $\overrightarrow{vu}$ is completely triangulated. The same procedure is then performed on the left side of $\overrightarrow{vu}$. The triangulation obtained is called the *greedy triangulation* of the hull of $\overline{uv}$, see Figs. 4(c) and 5. The next corollary, which is a direct consequence of Corollary 1, shows that the hull is a simple polygon consisting of at most $2k + 2$ vertices.

Corollary 2. *The Delaunay edges intersecting one useful k-OD edge $\overline{uv}$ are connected to at most k vertices on each side of the k-OD edge.*

Proof. Assume that there are more than k vertices determining the hull of $\overline{uv}$, say, to the right of $\overrightarrow{vu}$. Let s_1 be as in Lemma 3. Then $C(u, v, s_1)$ must contain all those points to the right of $\overrightarrow{vu}$ by Corollary 1, so $\overline{uv}$ is not useful, a contradiction. □

Lemma 4. *Let $\overline{uv}$ be a k-OD edge, let s_1 be the point to the left of $\overrightarrow{vu}$, such that the circle $C(u, s_1, v)$ contains no points to the left of $\overrightarrow{vu}$. Let s_2 be defined similarly but to the right of $\overrightarrow{vu}$. Edge $\overline{uv}$ is a useful k-OD edge if and only if $\triangle uvs_1$ and $\triangle uvs_2$ are k-OD triangles.*

Proof. The "only if" part is given in Lemma 3. For the "if" part, consider the greedy triangulation of $\overline{uv}$. It is given that $\triangle uvs_1$ and $\triangle uvs_2$ are k-OD triangles. Since the recursive continuation for the edges $\overline{vs_1}$, $\overline{vs_2}$, $\overline{us_1}$ and $\overline{us_2}$ yields circles to be tested that can only contain fewer points than $C(u, s_1, v)$ or $C(u, s_2, v)$, the corresponding triangles must be k-OD triangles too. □

After preprocessing, we can test efficiently whether an edge $\overline{uv}$ is a useful k-OD edge. First, locate u and v in the Delaunay triangulation, and traverse it from u to v along $\overline{uv}$ from Delaunay triangle to Delaunay triangle. Collect all intersected Delaunay edges. If there are more than $2k - 1$ of them, stop and report that $\overline{uv}$ is not a useful k-OD edge. Otherwise, find the points that determines the hull of $\overline{uv}$. If there are more than k points on one side of $\overline{uv}$, stop with the same result. Next, determine s_1 and s_2 as in the lemma just given. Now we must test how many points—at most k—lie in the circles $C(u, s_1, v)$ and $C(u, s_2, v)$ to determine usefulness. To this end, we use a data structure on a set of points that can report the kth furthest point from a given query point. The query points will be the centers of the circles $C(u, s_1, v)$ and $C(u, s_2, v)$. If the kth furthest point we find is further from the center of $C(u, s_1, v)$ than the radius of $C(u, s_1, v)$, and the similar test is true for $C(u, s_2, v)$, then we can conclude that $\overline{uv}$ is useful. The data structure that we need is the kth order Voronoi diagram preprocessed for efficient point

location. It answers queries in $O(\log n)$ time. This makes the total test for usefulness run in $O(\log n + k)$ time.

Theorem 1. *Let P be a set of n points in the plane. In $O(nk^2 + n\log n)$ expected time one can compute all useful k-OD edges of P.*

Proof. In Section 2 we showed that all k-OD edges can be determined in $O(nk2^{c\log^* k} + n\log n)$ expected time using an algorithm for higher-order Voronoi diagrams [22]. There are $O(nk)$ edges to be tested, so it takes $O(nk^2 + n\log n)$ expected time overall to determine all useful k-OD edges. □

If k is constant or small, the algorithm just given is efficient, but for larger values of k there are better algorithms based on partition trees. The idea is to build a data structure that determines whether a query k-OD edge is useful or not, without spending $\Omega(k)$ time for the query. We store all points of P in a 2-dimensional partition tree such that the points inside a query halfplane are represented by a small number of canonical subsets [2]. If we test an edge $\overline{uv}$, the halfplane will be bounded by the line through u and v. The canonical subsets are preprocessed into associated structures for further querying. To this end, we translate the problem of pushing a circle (initially, infinitely large, that is, a halfplane) through u and v to find the first point hit, to a 3-dimensional ray shooting problem. Any point $p = (p_x, p_y)$ in a canonical subset is mapped to $\widehat{p} = (p_x, p_y, p_x^2 + p_y^2)$, a point on the unit paraboloid $z = x^2 + y^2$. The circle squeezing through u and v becomes a plane rotating about $\widehat{u}$ and $\widehat{v}$ from vertical position until a third point is contained in it. This problem is dualized to ray shooting from infinity in a set of planes in 3-dimensional space. Since the total complexity of the unbounded cells in the 3-dimensional arrangement is only quadratic, we can solve the problem in an associated structure on m planes with $O(m^2)$ storage, $O(m^2\log m)$ preprocessing time, and $O(\log m)$ query time (find the cell in logarithmic time, and do ray shooting in that cell in logarithmic time) [1]. For the main tree, the 2-dimensional partition tree on the points of P, we can use one that has quadratic size and close to quadratic preprocessing time, and gives the points in a logarithmic number of canonical subsets [2]. To test all k-OD edges for usefulness, we obtain an $O'(n^2)$ time solution overall (we use the notation $O'(f(n))$ as a shorthand for $O(f(n)\cdot\log^c n)$ for some constant c). When k is considerably larger than $\sqrt{n}$, this solution is more efficient than the previous one.

Alternatively, we can use for the main tree one that has $O'(n^{3/2})$ size and preprocessing time, and yields $O'(n^{1/4})$ canonical subsets to be examined at each query [2]. For the associated structure, we take a partition tree for ray shooting in sets of planes in 3-dimensional space [2]. The query problem in the associated structure can also be solved in $O'(n/m^{1/3})$ query time with a data structure that uses $O'(m)$ storage and preprocessing time, where $n \leqslant m \leqslant n^3$. Since we know that $O(nk)$ queries are performed, we can balance the query time for the associated structures and their construction time. This occurs when $O(nk)\cdot n/m^{1/3} = m$, yielding an $O'(n^{3/2}k^{3/4})$ time solution overall.

Theorem 2. *Let P be a set of n points in the plane. In $O(n^2\log^c n)$ time one can compute all useful k-OD edges of P, where c is some constant. Alternatively, the problem can be solved in $O(n^{3/2}k^{3/4}\log^c n)$ time.*

Which solution is the most efficient one depends on the given value of k. It should be noted, however, that one can expect constant values of k to be most useful in practice, in which case the more complex solutions based on partition trees are not necessary.

To conclude this section we observe that the greedy triangulation of the hull of a useful k-OD edge can be computed in $O(k^2)$ time with the straightforward algorithm given earlier. Alternatively, for non-constant k, we can preprocess the hull of a useful k-OD edge for *arc pushing queries* [6]. This yields an $O(k \log^3 k)$ time algorithm for the greedy triangulation of the hull.

4. First order Delaunay triangulations

We examine the special structure of 1-OD triangulations in this section. We already observed in Corollary 2 that if $\overline{uv}$ is a useful 1-OD edge, and not a 0-OD edge, then it intersects exactly one Delaunay edge. We again assume without loss of generality that $\overline{uv}$ is vertical and that u is above v.

Lemma 5. *Every useful 1-OD edge intersects at most one useful 1-OD edge.*

Proof. From Corollary 2 we know that any Delaunay edge intersects at most one useful 1-OD edge. It remains to prove the lemma for any non-Delaunay, useful 1-OD edge $\overline{uv}$. Let $\overline{sy}$ be the Delaunay edge that intersects $\overline{uv}$. Then $\overline{us}$, $\overline{vs}$, $\overline{uy}$ and $\overline{vy}$ are Delaunay edges. If there exists a useful 1-OD edge that intersects the Delaunay edge $\overline{us}$, then it must be connected to y, according to Corollary 2; see Fig. 6. Denote this edge by $\overline{xy}$. Since $\overline{xy}$ is a useful 1-OD edge, x must be connected to u and s by Delaunay edges. Consider the circle $C(u, v, y)$. This circle contains only s, by Observation 1 and from the fact that $\overline{uv}$ is useful. The circle $C(u, y, x)$ can be obtained by expanding $C(u, v, y)$ until it hits x while releasing v. Since we let go of v, both v and s are contained in $C(u, y, x)$. Thus $\triangle xuy$ cannot be a 1-OD triangle and hence $\overline{xy}$ cannot be a useful 1-OD edge, which contradicts our assumption. By symmetry this holds for any edge intersecting $\overline{uy}$, $\overline{vs}$ or $\overline{vy}$. □

The lemma just given also shows that if $\overline{uv}$ is a useful 1-OD edge that is not Delaunay, then the four Delaunay edges $\overline{us}$, $\overline{uy}$, $\overline{sv}$ and $\overline{vy}$ must be in every 1-OD triangulation. Given a triangulation $\mathcal{T}$ and two edges e_1 and e_2 in $\mathcal{T}$, we say that e_1 and e_2 are independent if they are not incident to the same triangle in $\mathcal{T}$. From Corollary 2 and Lemma 5 we obtain the main result of this section.

Corollary 3. *Every 1-OD triangulation can be obtained from a Delaunay triangulation by flips of independent Delaunay edges.*

It is easy to see that—given the Delaunay triangulation—all 1-OD edges can be determined in linear time. In $O(n \log n)$ time, we can find out which ones are useful.

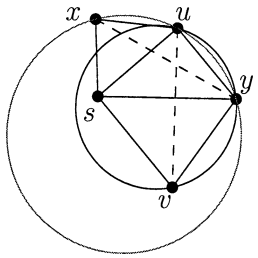


Fig. 6. There are no non-Delaunay, useful 1-OD edges that intersect.

5. Triangulations with additional criteria

Recall from the introduction that Delaunay triangulations are often used in terrain modeling, because they give well-shaped triangles. However, artifacts like artificial dams may arise. Since the Delaunay triangulation is completely specified by the input points (in non-degenerate cases), there is no flexibility to incorporate other criteria into the triangulation, which is why HOD triangulations were introduced. In this section we try to minimize the number of artificial dams in HOD triangulations, and deal with a number of other criteria as well. Many of these criteria can be optimized for 1-OD triangulations, which is what we will show first. Then we give an approximation algorithm to incorporate some of the criteria in k-OD triangulations. The approximation factor is not a constant, but a function of k, unfortunately.

5.1. Applications for 1-OD triangulations

5.1.1. Minimizing the number of local minima

To minimize the number of local minima is straightforward if the domain is the class of 1-OD triangulations. If the number of local minima is minimized, so is the number of artificial dams. Assume that the Delaunay triangulation of a point set is given, and for each vertex the height is known.

Lemma 6. *The insertion of a useful 1-OD edge while deleting a Delaunay edge to remove a local minimum cannot prevent any other local minimum from being removed.*

Proof. Consider a convex quadrilateral with vertices v, y, u and s, such that $\overline{vy}$, $\overline{yu}$, $\overline{us}$, $\overline{sv}$ and $\overline{ys}$ are Delaunay edges. Assume that $\overline{uv}$ is a useful 1-OD edge, otherwise we have no choice but to include $\overline{ys}$. If a local minimum in u or v can be removed by the flip that makes u and v connected, then u or v must be that local minimum and the other of u and v must even be lower. In particular, u and v are both lower than y and s. The only two vertices that lose a neighbor by the flip are y and s—they lose each other as neighbor—but neither can become a local minimum because they remain connected to the vertices u and v. □

Theorem 3. *An optimal 1-OD triangulation with respect to minimizing the number of local minima can be obtained by flips of independent Delaunay edges in* $O(n \log n)$ *time.*

5.1.2. Minimizing the number of local extrema

The number of local extrema—minima and maxima—can also be efficiently minimized over all 1-OD triangulations. In the previous subsection we could choose the edge in any quadrilateral that connects to the lowest of the four points. But if we want to minimize local minima and maxima we get conflicts: it can be that one diagonal of a convex quadrilateral gives an additional local minimum and the other diagonal gives a local maximum. Consider the subdivision S consisting of all edges that must be in any 1-OD triangulation, so S contains triangular and convex quadrilateral faces only. Consider the set of points that either have no lower neighbors or no higher neighbors; they are extremal in S. Consider the graph $G = (M, A)$, where M is the set of nodes representing the local extrema, and two nodes x and y are connected if they represent points on the same quadrilateral face and one diagonal makes x not a local extremum and the other diagonal makes y not a local extremum; see Fig. 7. Since nodes in the graph G correspond uniquely to vertices in S, we will call the nodes minimal, maximal or extremal if the vertices they represent are minimal, maximal or extremal.

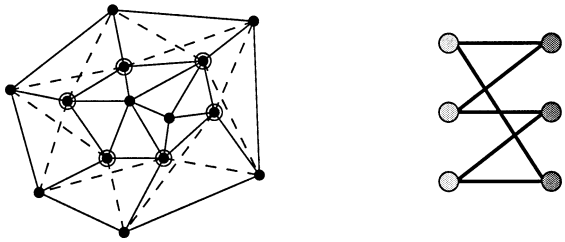


Fig. 7. Local extrema appearing in an even cycle.

Lemma 7. *Any quadrilateral face of S defines at most one arc in G, and this arc connects a local minimum to a local maximum.*

Proof. Any triangulating edge of a quadrilateral face can only avoid a local extremum if the highest two points are opposite and the lowest two points are opposite in the quadrilateral. One triangulating edge can only make the second-highest point non-maximal, and the other triangulating edge can only make the second-lowest point non-minimal. □

From the lemma it follows that G is bipartite, because every arc connects a local minimum to a local maximum. G may contain many isolated nodes; these extreme points can be removed. For every arc one can choose to make one of its nodes non-extremal by choosing the appropriate triangulation of the quadrilateral represented by the arc. For any node incident to only one arc, we can choose to make that node non-extremal without giving up optimality (minimum number of local extrema). If there are no nodes connected to only one other node, all nodes appear in cycles. Since the graph is bipartite, every cycle has even length; see Fig. 7. Take any cycle (of even length). Now all nodes in the cycle can be made non-extremal: we assign one quadrilateral (represented by the arc) to one incident extremum of S and choose the triangulating edge to make it non-extremal. We can repeat to treat nodes with only one incident arc, and even cycles, until no more extrema can be removed by triangulating edges. Then we complete the triangulation of S in any manner. This greedy, incremental method completes the subdivision S to a 1-OD triangulation that minimizes the number of local minima and maxima. The algorithm can be implemented to run in linear time when S is given.

Theorem 4. *An optimal 1-OD triangulation with respect to minimizing the number of local extrema can be determined in* $O(n \log n)$ *time.*

5.1.3. Other criteria

In visualization applications it is sometimes important to construct planar drawings with small degree and large angles between the edges. Thus, a natural optimization criterion for a 1-OD triangulation would be to minimize the maximum degree, since the Delaunay triangulation already maximizes the minimum angle. Besides visualization applications [9], constructing drawings with high angular resolution is important in the design of optical communication networks [11]. The problem of minimizing the maximum degree has been studied in several papers [13,15,16]. We know of no polynomial-time algorithm that gives an optimal solution to this optimization problem. The problem of completing the triangulation of a biconnected planar graph while minimizing the maximum degree is known to be

NP-hard. There is an efficient approximation algorithm though, as the following result shows. Let $\Delta(G)$ be the maximum degree of a graph G.

Theorem 5 (Kant and Bodlander [16]). *There is a linear time algorithm to triangulate a biconnected planar graph G such that, for the triangulation G' of G, $\Delta(G') \leqslant \lceil \frac{3}{2}\Delta(G) \rceil + 11$.*

We cannot use the result directly since it does not work on biconnected embedded straight-line graphs such as the embedded graph G formed by all Delaunay edges not intersecting useful 1-OD edges. But every quadrilateral face of G is convex, hence the bounded faces of G can always be triangulated using the above result. The problem is the unbounded face of G, the convex hull of P, which will be triangulated by edges that are not straight lines. Before using the theorem above, we will extend the graph G by adding points outside the convex hull and making the unbounded face triangulated. Let h be the number of points in P on the convex hull of P. Recursively we add points until the convex hull contains three points as follows: Construct an exterior hull H with $\lceil h/2 \rceil$ points that entirely includes P. Connect each point in H with three points in the convex hull of P, keeping the resulting graph planar. Note that this recursive procedure gives a triangulation of the region outside P, where each point originally not in P has degree at most 7, and each point on the convex hull of P is connected to at most two additional edges. This gives us the following theorem.

Theorem 6. *There is an $O(n \log n)$ time algorithm to compute a 1-OD triangulation $\mathcal{T}$ of a set P of n points, such that the maximum degree of a vertex in $\mathcal{T}$ is at most $\lceil \frac{3}{2}\Delta \rceil + 13$, where Δ is the degree of the maximum degree vertex in the 1-OD triangulation that minimizes the maximum degree.*

Note that the Delaunay triangulation itself is a 2-approximation of the optimal 1-OD triangulation.

As was pointed out in the introduction, criteria such as minimizing the maximum angle and minimizing the maximum area triangle may be of use for finite element method applications. These criteria (together with a number of other criteria not mentioned above) are trivial to optimize for 1-OD triangulations. This follows from the fact that these are all local optimization criteria, thanks to the nice properties of 1-OD triangulations.

Theorem 7. *For a Delaunay triangulation of a set P of n points in the plane, an optimal 1-OD triangulation can be obtained by flips of independent Delaunay edges in $O(n \log n)$ time for each one of the following criteria*: (i) *minimizing the maximal area triangle,* (ii) *minimizing the maximal angle,* (iii) *maximizing the minimum radius of a circumcircle,* (iv) *maximizing the minimum radius of an enclosing circle,* (v) *minimizing the sum of inscribed circle radii, and* (vi) *minimizing the total edge length.*

5.2. Applications for k-OD triangulations

It appears to be difficult to obtain general optimization results for all of the criteria listed before, given a value of $k \geqslant 2$. When k is so large that every pair of points gives a useful edge (like $k = n - 3$), then certain criteria can be optimized. For example, when minimizing the number of local minima, we can choose an edge from every point to the global minimum (in non-degenerate cases), so that there is only one local minimum. For minimizing the maximum angle and some other criteria, various optimal results are known [3].

To develop approximation algorithms for k-OD triangulations, we need to determine how many hulls a single hull can intersect. To this end, we first prove an upper bound on the maximum number of useful k-OD edges that intersect a given Delaunay edge. Fig. 8 shows that $\Omega(n)$ 2-OD edges can intersect a given Delaunay edge. But these 2-OD edges cannot all be useful. The next lemma shows that the maximum number of useful k-OD edges intersecting a given Delaunay edge does not depend on n, but only on k.

Lemma 8. *Let $\overline{uv}$ be any Delaunay edge. The number of useful k-OD edges in a triangulation $\mathcal{T}$ that intersect $\overline{uv}$ is* $O(k)$.

Proof. For simplicity we assume that u is vertically above v. Let Q_l and Q_r be the set of points that determines the hull of $\overline{uv}$ to the left, respectively to the right of $\overrightarrow{uv}$. Assume without loss of generality that $|Q_l| \leqslant |Q_r|$. Let p_r be the point in Q_r such that $C(u, v, p_r)$ includes all points in Q_r, except p_r itself. Let $p_l \in Q_l$ be such that $\overline{p_l p_r} \in \mathcal{T}$ and it intersects $\overline{uv}$.

Since the interior of $C(u, v, p_r)$ is completely contained in the union of the interiors of $C(u, p_r, p_l)$ and $C(v, p_r, p_l)$, it follows that $C(u, p_l, p_r)$ together with $C(v, p_l, p_r)$ contain all the points in $Q_r \setminus \{p_r\}$; see Fig. 9. Now we apply Lemma 3. If $\overline{p_l p_r}$ is useful then the "first" triangles left, denoted $\triangle p_l p_r x$, and right, denoted $\triangle p_l p_r y$, of $\overrightarrow{p_r p_l}$ must be k-OD triangles, that is, the two circles $C(p_l, p_r, x)$ and $C(p_l, p_r, y)$ each include at most k points. The union of the interiors of $C(p_l, p_r, x)$ and $C(p_l, p_r, y)$ includes the union of the interiors of $C(u, p_r, p_l)$ and $C(v, p_r, p_l)$, and therefore it also includes the interior of $C(u, v, p_r)$, which contains $|Q_r| - 1$ points. Hence, it follows that $|Q_r| - 1 \leqslant 2k$, and $|Q_l| \leqslant |Q_r| \leqslant 2k+1$. Since $\mathcal{T}$ is a planar triangulation the total number of useful k-OD edges intersecting $\overline{uv}$ is $O(k)$. □

Lemma 9. *Let $\overline{uv}$ be a useful k-OD edge and let H be its hull. The number of hulls of useful k-OD edges included in a triangulation $\mathcal{T}$ that intersect the interior of H is* $O(k^2)$.

Proof. By Corollary 2, the hull H of $\overline{uv}$ contains $O(k)$ Delaunay edges of P in its interior. Any useful k-OD edge intersecting H must intersect at least one of these Delaunay edges, or itself be the Delaunay edge. By Lemma 8, $O(k)$ useful k-OD edges included in a triangulation $\mathcal{T}$ can intersect a common Delaunay edge. The bound of $O(k^2)$ follows. □

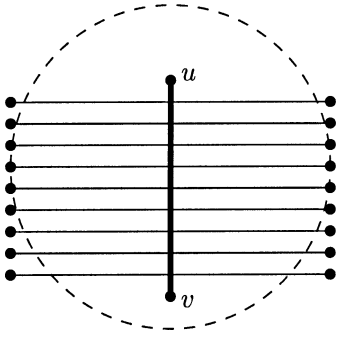


Fig. 8. Many 2-OD edges can intersect a Delaunay edge $\overline{uv}$.

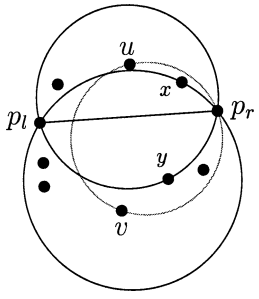


Fig. 9. Illustration of the proof of Lemma 8.

To keep track of which hulls intersect we define a hull intersection graph G as follows. There is a node for the hull of every useful k-OD edge. Two nodes are connected by an arc if their hulls intersect, that is, there exists a point that is interior to both hulls. The choice of one useful k-OD edge e possibly prohibits the choice of any other useful k-OD edge whose hull intersects the hull of e. In any case, if we choose an independent set of nodes in graph G, we get a set of hulls of useful k-OD edges that can be used together in a k-OD triangulation.

5.2.1. Minimizing the number of local extrema (revisited)

Suppose that all useful k-OD edges and their hulls have been computed, and the hull intersection graph G as well. Choose any useful k-OD edge e that removes a local minimum. Mark the node for the hull of e in G, and also the adjacent nodes. Choose the greedy triangulation of the hull of e. The choice of e can prevent other useful k-OD edges to be chosen in the triangulation. Let $\mathcal{T}$ be the optimal triangulation with respect to minimizing the number of local mimima. Then at most $O(k^2)$ useful k-OD edges of $\mathcal{T}$ are removed from consideration according to Lemma 9. Therefore, $O(k^2)$ points can be prevented from being non-local minimas. Continue to choose a useful k-OD edge that avoids another local minimum, provided its node in G is unmarked, until no such choice exists.

The same approach can be used to minimize the number of extrema.

Theorem 8. *Let m be the smallest number of local minima (or extrema) in any k-OD triangulation of a set P of points. There is an $O(n \log n + nk^3)$ expected time algorithm that computes a k-OD triangulation of P with at most $O(m \cdot k^2)$ local minima (or extrema).*

6. Conclusions and directions for further research

This paper introduced a class of triangulations that generalizes the Delaunay triangulation: the empty-circle property for the triangles in the Delaunay triangulation is replaced by requiring that the circle of each triangle contains at most some given number k of points. Such a triangulation is called a k-OD triangulation. For any point set, there may be several different k-OD triangulations. Therefore, one can study the optimization of some geometric criterion over all possible k-OD triangulations of a given point set. Such optimizations have applications in terrain modeling for GIS and in mesh generation for finite element methods.

The class of 1-OD triangulations was studied in more detail, and because of its special properties, most of the criteria could be optimized efficiently. For minimizing the maximum degree we obtained an approximation result. For the case when $k \geqslant 2$ we just gave some initial results. Obviously, optimization or approximation results are a topic of future research. Another issue we would like to address is experimental. With the application to realistic terrain modeling in mind, how many fewer local minima will a k-OD triangulation have in practice, when compared to the Delaunay triangulation? More fundamental is the question whether k-OD triangulations that minimize the number of local minima generally are more realistic as terrain models. This is a topic of further research.

References

[1] P.K. Agarwal, Range searching, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, FL, 1997, Chapter 31, pp. 575–598.

[2] P.K. Agarwal, J. Erickson, Geometric range searching and its relatives, in: B. Chazelle, J.E. Goodman, R. Pollack (Eds.), Advances in Discrete and Computational Geometry, Contemporary Mathematics, Vol. 223, American Mathematical Society, Providence, RI, 1999, pp. 1–56.
[3] M. Bern, Triangulations, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, FL, 1997, Chapter 22, pp. 413–428.
[4] M. Bern, D. Dobkin, D. Eppstein, Triangulating polygons without large angles, Internat. J. Comput. Geom. Appl. 5 (1995) 171–192.
[5] M. Bern, D. Eppstein, Mesh generation and optimal triangulation, in: D.-Z. Du, F.K. Hwang (Eds.), Computing in Euclidean Geometry, 2nd Edition, Lecture Notes Series on Computing, Vol. 4, World Scientific, Singapore, 1995, pp. 47–123.
[6] S.-W. Cheng, O. Cheong, H. Everett, R. van Oostrum, Hierarchical vertical decompositions, ray shooting, and circular arc queries in simple polygons, in: Proc. 15th Annu. ACM Symp. on Computational Geometry, 1999, pp. 227–236.
[7] L.P. Chew, Constrained Delaunay triangulations, Algorithmica 4 (1989) 97–108.
[8] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer, Berlin, 1997.
[9] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Algorithms for drawing graphs: an annotated bibliography, Computational Geometry 4 (1994) 235–282.
[10] B. Falcidieno, C. Pienovi, Natural surface approximation by constrained stochastic interpolation, Comput. Aided Design 22 (3) (1990) 167–172.
[11] M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Simvonis, E. Welzl, G. Woeginger, Drawing graphs in the plane with high resolution, SIAM J. Comput. 22 (1993) 1035–1052.
[12] S. Fortune, Voronoi diagrams and Delaunay triangulations, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, FL, 1997, Chapter 20, pp. 377–388.
[13] A. Garg, R. Tamassia, Planar drawings and angular resolution: Algorithms and bounds, in: Proc. 2nd Annu. European Sympos. Algorithms, Lecture Notes in Computer Science, Vol. 855, Springer, New York, 1994, pp. 12–23.
[14] M.F. Hutchinson, Calculation of hydrologically sound digital elevation models, in: Proc. 3th Int. Symp. on Spatial Data Handling, 1988, pp. 117–133.
[15] K. Jansen, One strike against the min-max degree triangulation problem, Computational Geometry 3 (1993) 107–120.
[16] G. Kant, H.L. Bodlaender, Triangulating planar graphs while minimizing the maximum degree, Inform. Comput. 135 (1997) 1–14.
[17] D.T. Lee, On k-nearest neighbor Voronoi diagrams in the plane, IEEE Trans. Comput. C-31 (1982) 478–487.
[18] J.J. Little, P. Shi, Structural lines, TINs, and DEMs, in: T.K. Poiker, N. Chrisman (Eds.), Proc. 8th Int. Symp. on Spatial Data Handling, 1998, pp. 627–636.
[19] D.R. Maidment, GIS and hydrologic modeling, in: M.F. Goodchild, B.O. Parks, L.T. Steyaert (Eds.), Environmental Modeling with GIS, Oxford University Press, New York, 1993, pp. 147–167.
[20] D.M. Mark, Automated detection of drainage networks from digital elevation models, Cartographica 21 (1984) 168–178.
[21] A. Okabe, B. Boots, K. Sugihara, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Wiley, Chichester, UK, 1992.
[22] E.A. Ramos, On range reporting, ray shooting and k-level construction, in: Proc. 15th Annu. ACM Symp. on Computational Geometry, 1999, pp. 390–399.
[23] B. Schneider, Geomorphologically sound reconstruction of digital terrain surfaces from contours, in: T.K. Poiker, N. Chrisman (Eds.), Proc. 8th Int. Symp. on Spatial Data Handling, 1998, pp. 657–667.
[24] D.M. Theobald, M.F. Goodchild, Artifacts of TIN-based surface flow modeling, in: Proc. GIS/LIS, 1990, pp. 955–964.