Towards a Definition of Higher Order Constrained Delaunay Triangulations *

Rodrigo I. Silveira† Marc van Kreveld†

Abstract

When a triangulation of a set of points and edges is required, the concept of higher order Delaunay triangulations does not have a proper equivalent of constrained Delaunay triangulations. A single edge may cause that all triangulations with that edge have high order. This paper studies several possible definitions that assure that an order-k constrained Delaunay triangulation exists for any $k \geq 0$, while maintaining the character of higher order Delaunay triangulations of point sets. Properties of these definitions and algorithms to support computations on them are also discussed.

1 Introduction

Higher order Delaunay triangulations [4] are a generalization of the Delaunay triangulation. They provide a class of triangulations that are reasonably well-shaped, depending on a parameter k. A triangulation is *order-k Delaunay* if the circumcircle of the three vertices of any triangle contains at most k other points. For $k = 0$, if no four points are cocircular, there is only one higher order Delaunay triangulation, equal to the Delaunay triangulation. As k is increased, the shape quality of the triangles may decrease, but the number of triangulations may increase, hence there is more flexibility to optimize some other criterion. The concept of higher order Delaunay triangulations has been successfully applied to several areas, including terrain modeling [3], minimum interference networks [1] and multivariate splines [6].

When working with triangulations it is often the case that a given set of edges must be included in the triangulation. We refer to them as constraints, or constraining edges. For example, in mesh generation, the mesh must respect the boundary edges of the components. When working with polyhedral terrains for hydrologic applications, it is common to augment the terrain with the edges representing the drainage network.

Regardless of the reason for including a set of edges, it is important for a triangulation that includes them to have nicely-shaped triangles. The *constrained Delaunay triangulation* (CDT) [2] includes a given set of edges and is as close to the Delaunay triangulation as possible. This is achieved by relaxing the empty-circle property of the Delaunay triangulation: points are allowed inside the circumcircle C of a triangle t if they are separated inside C from t by a constraining edge.

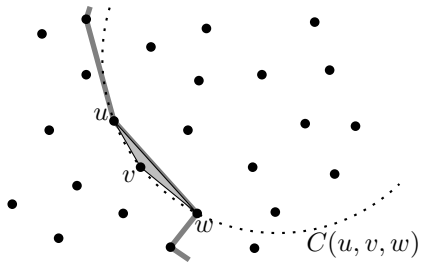

Figure 1: A point set is augmented with some constraining edges (in gray). Any triangulation of this point set that includes the gray edges must include triangle $\triangle uvw$, which has a very high order.

The only previous work on order-k Delaunay triangulations with constraints focuses on finding a lowest-order Delaunay triangulation that includes a given set of edges [5], but the definition of order used does not consider the constraints. Until now, there was no concept equivalent to the CDT for higher order Delaunay triangulations. This implies that if one of the constraints causes the inclusion of a triangle of very high order, then the whole triangulation will have at least that order (see Figure 1). Therefore, all triangles, even ones far away from these high-order constraints, will be allowed to have that very high order, and, more importantly, a bad shape.

In this paper we address the problem of defining higher order constrained Delaunay triangulations. We achieve this by proposing several definitions of the *constrained order* of a triangle, which take the constraints into account when counting the number of points inside the circumcircles of the triangles. The new definitions of order try to reflect that some triangles may have a bad shape because of the constraints, thus their order should be computed in a different way. These triangles will have a lower order than in the standard definition, and therefore a low order of the whole triangulation can be obtained.

In this paper we assume non-degeneracy of the input set P: no four points are cocircular. For brevity, we will sometimes write *order* instead of *constrained order*.

*This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.065.503 (GADGET) and under the project GOGO.

†Department of Information and Computing Science, Utrecht University, the Netherlands, {rodrigo, marc}@cs.uu.nl

2 Proposed definitions

Any suitable definition of order-k constrained Delaunay (k-OCD) triangulations must be in line with the idea of the CDT and, at the same time, be consistent with the spirit of higher order Delaunay triangulations of point sets. We summarize this by establishing a list of properties that a suitable definition should satisfy:

1. For $k = 0$ there is only one constrained triangulation, which is the CDT.
2. If there are no constraining edges, any k-OCD triangulation is a k-OD triangulation.
3. As k increases, the number of k-OCD triangulations also increases.
4. If a point or endpoint of a constraint moves slightly, the constrained order changes only slightly (this is made more precise below).
5. The definition is *intuitive* for triangulations of polygons.
6. The definition is *intuitive* for triangulations of points with constraining edges.

The important part in any definition of higher order constrained Delaunay triangulations is defining when a point inside the circumcircle of a triangle must be counted. We propose seven different definitions, where the varying aspect is when a point is counted in the constrained order. Let $C = C(u, v, w)$ be the circle through u, v, and w. Suppose we want to compute the order of $\triangle uvw$. No other point can be in $\triangle uvw$ and no constraint intersects it, otherwise it cannot be a triangle in a triangulation. In the definitions below, p and r lie in $C(u, v, w)$. Note that $\overline{uv}$, $\overline{vw}$ and $\overline{wu}$ can be constraints.

PATHCON (path connected). A point p is counted if and only if there is a constraint-free path contained in C that connects p to some point interior to $\triangle uvw$.

SEESTRIANG (sees triangle). A point is counted if and only if it can see some point in the interior of the triangle.

SEESVTX (sees vertex of triangle). A point p is counted if and only if it can see some vertex of the triangle and some point in its interior.

CONFEDGE (conflicting edge). A point p is counted if and only if there is a point r inside C such that $\overline{pr}$ intersects $\triangle uvw$ and does not intersect any constraint.

SEESOPP (sees opposite). A point p is counted if and only if it sees the *opposite* vertex of the triangle, that is, the vertex $x \in \{u, v, w\}$ such that $\overline{px}$ intersects $\triangle uvw$.

SEES3VTX (sees 3 vertices). A point p is counted if and only if it can see u, v, and w.

EMPTYQUAD (empty quadrilateral). A point p is counted if and only if the quadrilateral formed by the three vertices of the triangle and p is empty, and the edge of $\triangle uvw$ that is a diagonal of the quadrilateral is not a constraint. This corresponds to the idea of being able to flip one edge of $\triangle uvw$ to connect p.

Figure 2 shows an example where the different definitions can be compared.

We now make Property 4 more precise. Assume that some point or endpoint p moves, without changing the structure of the triangulation. Then a change in the order of the triangulation in all definitions can only occur if p becomes collinear with two points or cocircular with three points. We call this a criticality during the move of p. Property 4 should be interpreted such that if any point moves through only one criticality, then the order of the triangulation changes by at most one.

We make the following simple observations.

Observation 1 *All the previous definitions satisfy Properties 1 to 6, except for PATHCON, which does not satisfy Property 4.*

Observation 2 *For higher order constrained Delaunay triangulations of a point set with constraints or of a polygon, the following inclusion relations hold*[1]*: EMPTYQUAD* $\subseteq$ *SEES3VTX* $\subseteq$ *SEESOPP* $\subseteq$ *SEESVTX* $\subseteq$ *CONFEDGE* $\subseteq$ *SEESTRIANG* $\subseteq$ *PATHCON.*

Note that for triangulations of simple polygons, SEES3VTX = EMPTYQUAD.

3 Computing the order of a triangle

A basic operation when dealing with higher order Delaunay triangulations is determining the order of a triangle. In this section we analyze how efficiently this can be done for each of the proposed definitions.

Let $\triangle uvw$ be the triangle whose order is being computed, let C be its circumcircle, let P_C be the set of points and endpoints inside C, and let E_C be the set of constraining edges that are at least partly inside C.

PATHCON We build a point location data structure for the subdivision induced by $E_C \cup C$, which allows us to know for each point of P_C if it lies in the same face as the interior of $\triangle uvw$. These are the points that can be reached from the triangle, and must be counted. The subdivision is made of parts of constraining edges and circular arcs of C. The point location data structure can be built in $O(n \log n)$ time, and querying each point takes $O(\log n)$ time, therefore the running time is $O(n \log n)$.

[1] With a slight abuse of notation.

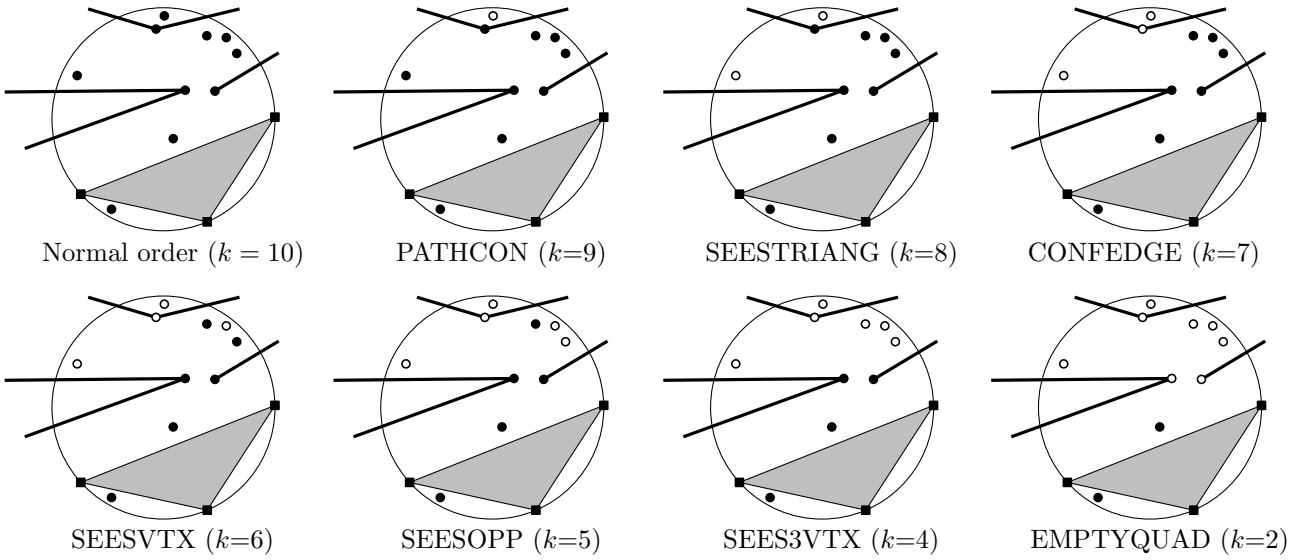

Figure 2: What is the order (k) of the gray triangle? For each definition, the points that are counted are drawn as discs and the ones that are not counted as empty circles. Constraints are drawn with thick edges.

SEESTRIANG The points in P_C can be in one of three regions, bounded by C and the three edges of $\triangle uvw$. For a point in a given region, seeing $\triangle uvw$ is equivalent to seeing one of the edges of $\triangle uvw$. We process each region separately. Assume the current region is the one bounded by $\overline{uv}$. If $\overline{uv}$ is a constraint, no point is counted. Otherwise, let $S \subseteq P_C$ be the subset of (end)points that lie inside that region. For each point in $S \cup \{u, v\}$, we sort the other points around it by angle. This can be done for all the points in $O(n^2)$ time [7]. Then for each point, we go through the sorted list of points around it and check if at any moment $\overline{uv}$ is visible. We can do this in linear time because we do not need to keep track of the order in which the constraints become visible, we only need to know whether there is some constraint between the point and $\overline{uv}$. The running time is $O(n^2)$.

CONFEDGE We compute the visibility graph of P_C in $O(n^2)$ time [7]. To determine if a point must be counted we check if it has a visible point in one of the other two regions of C. The running time is $O(n^2)$.

SEESVTX For each of the vertices of the triangle we compute the visibility polygon, where the edges in E_C and the points in P_C are the obstacles. This can be done in $O(n \log n)$ time. Then we count the points visible to some vertex; they can also see the interior of $\triangle uvw$ due to non-degeneracy. The running time is $O(n \log n)$.

SEESOPP We proceed as before, but only for the opposite vertex. The running time is $O(n \log n)$.

SEES3VTX Same as for SEESVTX, but we count only the points that see the three vertices of the triangle. The running time is $O(n \log n)$.

EMPTYQUAD First we compute the points that see u, v and w. They can be in one of three regions of C. For each region there is a vertex of $\triangle uvw$ that is the opposite vertex. We show how to proceed for the region where that vertex is w. We need to discard the points p such that $\triangle uvp$ is not empty. Let $\triangle uvp$ and $\triangle uvq$ be two triangles, and let α_u (α_v) denote the angle of $\triangle uvp$ at u (at v), and β_u (β_v) the same for $\triangle uvq$. It is easy to see that $\triangle uvp$ contains q if and only if $\beta_u < \alpha_u$ and $\beta_v < \alpha_v$. Each triangle can be represented by a point in the plane using its angles at u and at v. The empty triangles, which define an empty quadrilateral, are the ones lying on the lower-left staircase of the point set. They can be computed in $O(n \log n)$ time by a sweep line algorithm. The running time is $O(n \log n)$.

4 Computing all the k-OCD triangles

Another useful operation related to HODT is computing all the order-k triangles. For example, this is a fundamental step when triangulating polygons optimally for order-k Delaunay triangulations [9]. In what follows we always proceed edge by edge, for each of the $O(n^2)$ possible edges. For each edge, we will find all the order-k triangles adjacent to it. We explain how to proceed for one edge $\overline{uv}$ and assume it is not a constraint (otherwise the algorithms are simpler).

PATHCON We give only a sketch of the algorithm. We sweep a circle C through u and v, starting as the halfplane to the left of $\overrightarrow{uv}$ and we slide it, always touching u and v, until it becomes the halfplane to the right of $\overrightarrow{uv}$. The event points will be the points and the endpoints of the constraints. At every event during the

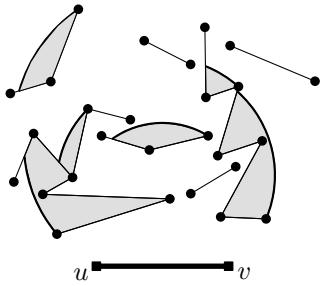

Figure 3: Computing all the order-k triangles for PATHCON. Regions identified by the sweep algorithm.

sweep, a region is found, defined by some connected component of constraints and a part of the boundary of the current sweep circle (see Figure 3). The region is such that the points in it were counted until that moment, but from that moment on will not be counted anymore, because they became disconnected from the region in C with $\triangle uvw$. The regions are disjoint and made of line segments and circular arcs. The time needed to identify all the regions is $O(n \log n)$.

Next all points inside each region must be identified. This can be done in $O(n \log n)$ time by using a point location data structure that can handle circular arcs (e.g. [8]). Once the number of points inside each region is known, it is possible to sweep the circle through u and v again and compute the order of $\triangle uvw$ for all w to the right of $\overrightarrow{uv}$ exactly. The overall running time for one edge $\overline{uv}$ is $O(n \log n)$, and $O(n^3 \log n)$ time is needed to identify all the order-k triangles.

SEESTRIANG, SEESVTX, SEESOPP, SEES3VTX and EMPTYQUAD For these definitions a simple circle sweep, where every point and endpoint of a constraint defines an event, is enough to determine the order of $\triangle uvw$ for each point w to the right of $\overrightarrow{uv}$. Since the definitions are based on visibility between the points and some elements of $\triangle uvw$ (an edge, a vertex, etc.), once a vertex is counted (that is, it sees the part of $\triangle uvw$ in question), it will be counted until it stops to be inside the sweep circle. Given the visibility graph, we can determine if a point must be counted in $O(1)$ time. In the case of EMPTYQUAD we can first discard all the points that create a non-empty triangle with $\overline{uv}$, as explained in the previous section. After obtaining the possible third points, only the ones that define empty triangles must be selected. Again, this can be done in $O(n \log n)$ time. Therefore the running time for one edge $\overline{uv}$ is $O(n \log n)$ (assuming the visibility graph is given). It follows that all order-k triangles can be found in $O(n^3 \log n)$ time.

CONFEDGE For this definition we can apply the algorithm used for the previous definitions, but in this case, checking if a point must be counted takes more time. This is because every time a third point w to the right of $\overrightarrow{uv}$ is processed, many of the points inside C that can see w will be counted from the next step on. Hence, $O(n)$ time is required to find these points. The total running time per edge is $O(n^2)$, so it takes $O(n^4)$ time to find all order-k triangles.

5 Discussion

In the context of higher order Delaunay triangulations, we proposed seven different definitions of the order of a triangle that take into account a set of constraining edges. This constitutes an attempt to combine the concepts of *constrained Delaunay triangulations* and *higher order Delaunay triangulations*. The proposed definitions can be seen as natural generalizations of the idea of *order* of a triangle. They define a hierarchy that goes from the normal, restrictive, order definition, to a very permissive definition that counts much fewer points than the original one. In general it cannot be stated which definition is the best one, and which one to choose may depend on the application.

For each definition we presented algorithms to compute the order of one triangle and to find all order-k triangles of a point set with constraining edges. These are basic problems that need to be solved for most implementations of higher order Delaunay triangulations.

References

[1] M. Benkert, J. Gudmundsson, H. J. Haverkort, and A. Wolff. Constructing interference-minimal networks. In *Proc. 32nd Int. Conf. on Current Trends in Th. and Prac. of Comp. Sci. (SOFSEM'06)*, pages 166–176, 2006.

[2] L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[3] T. de Kok, M. van Kreveld, and M. Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 36:52–65, 2007.

[4] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[5] J. Gudmundsson, H. Haverkort, and M. van Kreveld. Constrained higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 30:271–277, 2005.

[6] M. Neamtu. Delaunay configurations and multivariate splines: a generalization of a result of B. N. Delaunay. *Trans. Amer. Math. Soc.*, 359(7):2993–3004, 2007.

[7] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 164–171, 1988.

[8] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Comm. ACM*, 29:669–679, 1986.

[9] R. I. Silveira and M. van Kreveld. Optimal higher order Delaunay triangulations of polygons. In *Proc. 23rd Eur. Workshop on Comput. Geom.*, pages 194–197, 2007.