DCG, Wintersemester 2024/25 Lecture 10 (6 pages)

Computing Delaunay Triangulations II

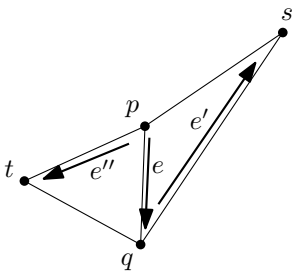
Anne Driemel and Herman Haverkort updated: November 13, 2024

In this lecture we want to continue the analysis of the randomized incremental algorithm to compute the Delaunay triangulation, which we discussed in the previous lecture (Algorithm 9.1).

1 Analysis of Algorithm 9.1

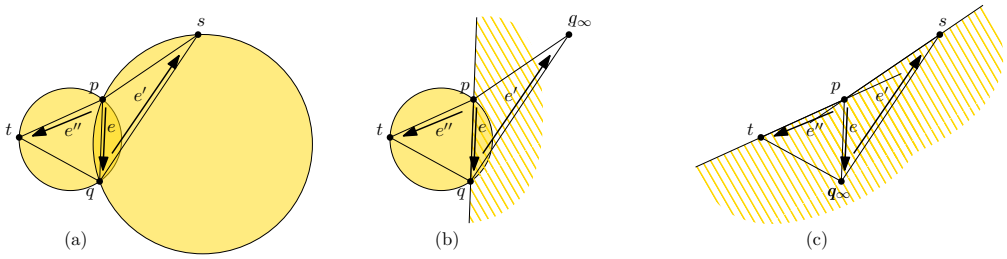
We define configurations and killing sets for the analysis of the algorithm,

Definition 10.1 (Configuration)**.** *A configuration Δ is an ordered tuple (s, q, p, t), modulo reversal*[1]*, of four different points from $P \cup \{q_\infty\}$, where q_∞ is the vertex at infinity. A configuration is present in the triangulation T, if the following edges are present in T: there is a half-edge e from p to q, and there is a half-edge e' from q to s with $e' = Next(e)$, and there is a half-edge e'' from p to t with $e'' = Next(Twin(e))$.*



Definition 10.2 (Killing-Set)**.** *For a triangle (p, q, s) with $p, q, s \in P$, let $D(p, q, s)$ be the unique disk that has the points p,q, and s on its boundary. For an infinite triangle (p, q, q_∞), with $p, q \in P$ and an infinite vertex q_∞, let $D(p, q, q_\infty)$ be the halfspace that lies to the left of the edge (p, q). The killing-set $K(\Delta)$ of a configuration $\Delta = (s, q, p, t)$ is defined as follows*

$$K(\Delta) = (D(p, q, s) \cup D(q, p, t)) \cap (P \setminus \{s, p, q, t\})$$



[1] With "modulo reversal" we mean that (s, q, p, t) and (t, p, q, s) are considered to be the same tuple.

Example 10.3. *Three different examples of killing sets are shown in the figure above. In (a) neither of the two triangles is an infinite triangle, in (b) one triangle is an infinite triangle, in (c) both triangles are infinite triangles.*

Theorem 10.4. *Algorithm 9.1 computes the Delaunay triangulation of n points in $\mathbb{R}^2$ in general position in $O(n \log n)$ expected time, where the expectation is with respect to the random permutation used by the algorithm.*

Proof. The correctness of the algorithm follows from Theorem 9.5. It remains to show the running time. Computing a random permutation in line 2 can be done in time $O(n)$. Initializing the triangulation T in lines 3 and 4 can be done in constant time. Computing the conflict graph of the triangles in T and the points $p_4, \ldots, p_n$ can be done in time in $O(n)$. Removing the infinite vertex and its incident edges on line 39 can be done in time in $O(n)$. Thus, the work done before and after the main for-loop can be done in $O(n)$ time.

It remains to analyse the main for-loop. During each iteration, the algorithm first computes the cycle of horizon edges. This can be done in time in $O(|\mathit{Conflicts}(p_i)|)$ in one of several ways. For example, one can check all three edges of all faces in $\mathit{Conflicts}(p_i)$ until one horizon edge is found. From there, one can trace the horizon by navigating the DCEL, at every vertex following pointers to edges that lie inside of the horizon until the next horizon edge is found. Next, on lines 13–36, the algorithm updates the DCEL (lines 14–16, 25–29, 31, and 35) and the conflict graph (lines 17–24 and 34).

We first discuss the updates to the DCEL. These take time in $O(|\mathit{Conflicts}(p_i)|)$ (note that the horizon H also has size $O(|\mathit{Conflicts}(p_i)|)$). Note that all faces of $\mathit{Conflicts}(p_i)$ are removed in this iteration. Therefore, the total time spent on updating the DCEL in the entire algorithm is bounded by the total number of faces removed, which in turn is upper bounded by the total number of faces created. Let F_i be the number of faces created when adding p_i. By linearity of expectation we have that

$$\mathbf{E}\left[\sum_{i=4}^{n} F_i\right] = \sum_{i=4}^{n} \mathbf{E}\left[F_i\right]$$

where the expectation is over the random permutation of P used by the algorithm. This is the total expected number of faces created after initialization. (We also have to take into account the faces created in the triangulation of $\{p_1, p_2, p_3, q_\infty\}$. This number is equal to 4.)

For fixed i, with $4 \leq i \leq n$, the expected number of faces created, when adding p_i is equal to the vertex degree of p_i in the Delaunay triangulation T of $q_\infty, p_1, \ldots, p_i$. This triangulation T is affected by the random permutation of P, but we can argue as follows. Suppose we fix a subset $S \subset P$ of size $(n - i)$ and assume S is chosen as the set $\{p_{i+1}, \ldots, p_n\}$ in the random permutation. Now, the triangulation T is fixed, even though, we do not know in which order the points of $P \setminus S$ were added to the triangulation when it was built. Let d_j denote the vertex degree of p_j in T. Since, in the permutations that have a fixed set S for $\{p_{i+1}, \ldots, p_n\}$, all permutations of the first i points $p_1, ..., p_i$ occur equally often, p_i is distributed uniformly at random in the set $\{P \setminus S\}$. Therefore, we can estimate the expected degree of p_i, and thus, the expected number of faces created when adding p_i, by:

$$\mathbf{E}\left[F_i\right] = \sum_{j=1}^{i} \frac{d_j}{i} = \frac{1}{i} \sum_{j=1}^{i} d_j \leq \frac{1}{i} 6(i-1) \leq 6$$

where the last two inequalities follow from Observation 9.4 and the fact that the total sum of vertex degrees is equal to twice the number of edges of a graph. (Since T is fixed, the total number of edges in T is fixed.) Since this bound on the expectation holds for any choice of

S, it holds as a bound on the expectation of F_i over all random permutations. Therefore, the expected total time spent on updating the triangulation is in $O(n)$.

It remains to analyse the time spent on updating the conflict graph (lines 17–24 and 34). Assume we maintain the conflict lists as doubly-linked lists, such that:

- the entry of any face f in the list of a point p also contains a pointer to the entry of p in the list of f;
- the entry of any point p in the list of a face f also stores the index of p in the permuted set P;
- for each face f, the list of points that conflict with f is ordered by index of the points.

On lines 6–7, for each face f, the list of points that are in conflict with f is constructed in order by index.

On line 18, we merge two ordered conflict lists into one ordered list while removing duplicates; this can be done in time linear in the size of the conflict lists by using the sorted order and the stored indices of the points. Note that each of these conflict lists is at most as large as the killing set of the configuration formed by the two faces on either side of e. The merged list is subsequently processed in linear time on lines 19–24.

Removing a face f from conflict lists on line 34 takes $O(1)$ time per point in $\mathit{Conflicts}(f)$, using the pointers mentioned above; thus, this takes time that is at most linear in the size of the killing set of any configuration that involves f.

Note that the configurations whose killing set sizes determine the amount of work on lines 18–24 and 34 are subsequently destroyed, so they will not cost any time ever again. Thus, the total time spent on updating the conflict graph is bounded by the expected total size of the killing sets of the configurations which were created in the course of the algorithm. By Lemma 10.5 below this is in $O(n \log n)$. □

Lemma 10.5. *Consider the execution of Algorithm 9.1 for computing the Delaunay triangulation of n points in $\mathbb{R}^2$. Let $\mathcal{D}_i$ be the set of configurations newly created in T when adding point p_i, for $4 \leq i \leq n$, and let $\mathcal{D}_3$ be the set configurations in T created in the beginning. It holds that*

$$\mathbf{E}\left[\sum_{i=3}^{n} \sum_{\Delta \in \mathcal{D}_i} \big|K(\Delta)\big|\right] \in O(n \log n)$$

Proof. We adapt the proof of Lemma 7.4 from Lecture 7 (respectively, the proof of Lemma 6.6 from Lecture 6). Let $\mathcal{W}_i$ denote the set of configurations that are destroyed in T during the addition of point p_i. We estimate the expectation

$$\mathbf{E}\left[\big|\mathcal{D}_i \cap \mathcal{W}_{i+1}\big|\right]$$

for fixed i. That is, we determine the expected number of configurations that are created when adding a point p_i and are destroyed in the next round, when the next point p_{i+1} is added.

Fix a configuration $\Delta \in \mathcal{D}_i$ and consider the probability that $\Delta \in \mathcal{W}_{i+1}$. This happens if and only if the next point p_{i+1} is contained in the killing set $K(\Delta)$. The probability of this event is therefore $\big|K(\Delta)\big|/(n-i)$, since we can think of the point p_{i+1} being chosen uniformly at random from the $n-i$ remaining points in P that have not been processed, yet. Therefore,

$$\mathbf{E}\left[\big|\mathcal{D}_i \cap \mathcal{W}_{i+1}\big|\right] = \mathbf{E}\left[\sum_{\Delta \in \mathcal{D}_i} \frac{\big|K(\Delta)\big|}{n-i}\right] = \frac{1}{(n-i)} \cdot \mathbf{E}\left[\sum_{\Delta \in \mathcal{D}_i} \big|K(\Delta)\big|\right] \tag{1}$$

where the last inequality follows from the linearity of expectation.

Now, fix a configuration $\Delta \in \mathcal{W}_{i+1}$ and look "backwards": What is the probability that Δ was only created in round i? Here there are two cases, if the infinite vertex p_∞ is not in Δ, then this happens with probability $4/i$. On the other hand, if p_∞ is part of the configuration Δ, then this happens with probability $3/i$.

This implies

$$\mathbf{E}\left[\left|\mathcal{D}_i \cap \mathcal{W}_{i+1}\right|\right] \leq \mathbf{E}\left[\left|\mathcal{W}_{i+1}\right|\frac{4}{i}\right] = \frac{4}{i} \cdot \mathbf{E}\left[\left|\mathcal{W}_{i+1}\right|\right] \tag{2}$$

where, again, the last inequality follows from the linearity of expectation.

Since the left hand side is the same in the above equations, we obtain

$$\mathbf{E}\left[\sum_{\Delta \in \mathcal{D}_i} \left|K(\Delta)\right|\right] \leq \frac{4(n-i)}{i} \cdot \mathbf{E}\left[\left|\mathcal{W}_{i+1}\right|\right] \leq \frac{4n}{i} \cdot \mathbf{E}\left[\left|\mathcal{W}_{i+1}\right|\right] \tag{3}$$

From here the proof is the same as in Lemma 7.4.

Just like in the proof of Lemma 7.4, we can show that the *expected* size of $\mathcal{D}_i$ is constant for any fixed i. Here we use Observation 9.4 for bounding the number of configurations present in a fixed triangulation T. The remainder of the proof is the same and we obtain the claimed bound of $O(n \log n)$. □

2 Delaunay triangulations and polytopes $\mathbb{R}^3$

The analysis above indicates a strong relationship between convex hulls in $\mathbb{R}^3$ and the Delaunay triangulations in $\mathbb{R}^2$, since we showed that both can be computed by using almost the same algorithm in almost the same (expected) time. We now want to have a closer look at this relationship.

We will show that for any finite set of points $P \subset \mathbb{R}^2$ in general position we can construct a set of points $P' \subset \mathbb{R}^3$ so that any Delaunay triangle of P corresponds to a facet of the convex hull of P'. This indicates that an algorithm for computing the convex hull of points in $\mathbb{R}^3$ could be used to compute the set of Delaunay triangles of a set of points in $\mathbb{R}^2$ in general position.

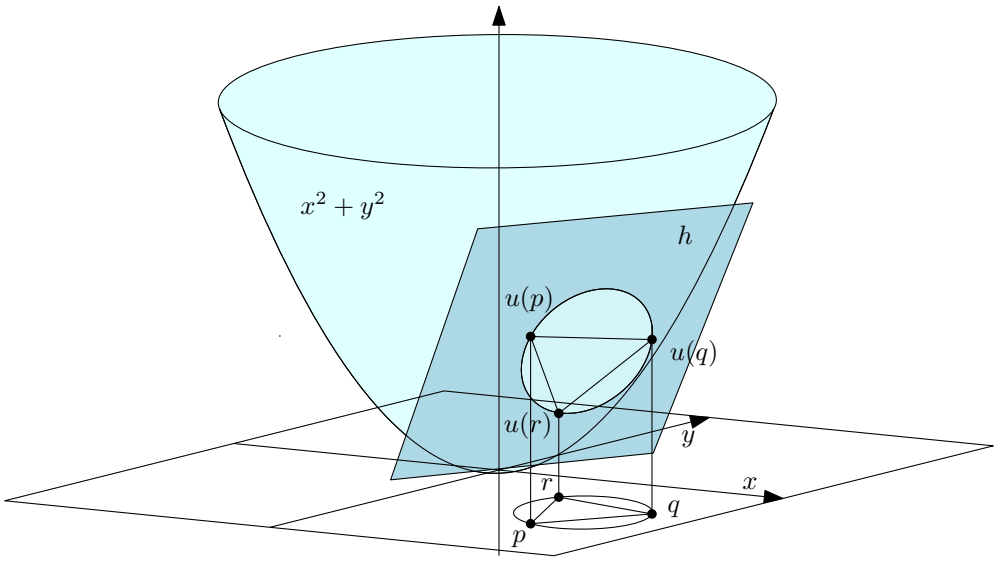
To make this relationship more precise, we define a lifting to the unit paraboloid in $\mathbb{R}^3$. The paraboloid is defined by:

$$U = \left\{ (x, y, z) \in \mathbb{R}^3 \;\middle|\; z = x^2 + y^2 \right\}.$$

Define the mapping $u : \mathbb{R}^2 \mapsto \mathbb{R}^3$ as

$$u(x, y) = (x, y, x^2 + y^2)$$

Intuitively, the function u lifts points on the hyperplane with equation $z = 0$ up onto the paraboloid in $\mathbb{R}^3$.



Lemma 10.6. *Let D be a disk in $\mathbb{R}^2$, there exists a closed halfspace h in $\mathbb{R}^3$, such that for any point $(x, y) \in \mathbb{R}^2$ it holds that*

$$(x, y) \in D \quad \Leftrightarrow \quad u(x, y) \in h$$

Moreover, if the point (x, y) lies on the boundary of D, then $u(x, y)$ lies on the boundary of h.

Proof. Any disk $D \in \mathbb{R}^2$ is of the form

$$D = \left\{ (x, y) \in \mathbb{R}^2 \;\middle|\; (c_1 - x)^2 + (c_2 - y)^2 \leq r^2 \right\}$$

where $(c_1, c_2) \in \mathbb{R}^2$ is the center of the disk and $r \in \mathbb{R}$ is the radius of the disk. A closed halfspace h in $\mathbb{R}^3$ has the form

$$h = \left\{ (x, y, z) \in \mathbb{R}^2 \;\middle|\; a_1 x + a_2 y + a_3 z \leq b \right\}$$

for some $(a_1, a_2, a_3) \in \mathbb{R}^3$ and $b \in \mathbb{R}$. Given a disk, specified by c_1, c_2 and r, we derive the closed halfspace h that satisfies the conditions of the theorem.

$$\begin{aligned}(x,y) \in D \Leftrightarrow \quad & (c_1 - x)^2 + (c_2 - y)^2 && \leq r^2 \\ \Leftrightarrow \quad & -2c_1x + -2c_2y + (x^2 + y^2) && \leq r^2 - c_1^2 - c_2^2 \\ \Leftrightarrow \quad & u(x) \in h\end{aligned}$$

where h is defined by the parameters $a_1 = -2c_1$, $a_2 = -2c_2, a_3 = 1$ and $b = r^2 - c_1^2 - c_2^2$. □

Theorem 10.7. *Let P be a finite set of points in $\mathbb{R}^2$ in general position, and let $P' \subset \mathbb{R}^3$ be defined as*

$$P' = \{\ u(p)\ \mid\ p \in P\ \}$$

If the triangle (p, q, r) is a Delaunay triangle with respect to P, then $u(p), u(q), u(r)$ are contained in the same facet of the convex hull of P'.

Proof. Recall that a facet of $\text{conv}(P')$ can be obtained by intersecting $\text{conv}(P')$ with a hyperplane that has all of P' to one side of the hyperplane. Let (p, q, r) be a Delaunay triangle. Let D be the disk that has p, q, r on its boundary. Let h be the corresponding halfspace of Lemma 10.6. D contains no points of P in its interior, by the Delaunay property. This implies that h contains no points of P', except on the boundary. Moreover, h has $u(p), u(q)$, and $u(r)$ on its boundary. Therefore, $u(p), u(q)$, and $u(r)$ are contained in the same facet of $\text{conv}(P')$. □