Edge-flipping

`Topics:` empty circles, Delaunay's lemma, edge-flip algorithm, running time, maxmin angle property.

Empty circles. Recall from Claim 1 that three points $a, b, c \in S$ are vertices of a Delaunay triangle iff the circle that passes through a, b, c is empty. A Delaunay edge, ab, belongs to one or two Delaunay triangles. In either case, there is a pencil of empty circles passing through a and b. The centers of these circles are the points on the Voronoi edge $V_a \cap V_b$, see Figure 7. Similar to Claim 1 we have

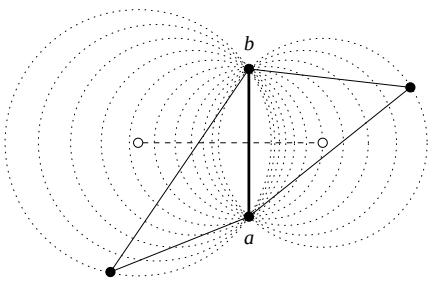

Figure 7: The Voronoi edge is the dashed line segment of centers of circles that pass through the endpoints of ab.

CLAIM 3. Let $S \subseteq \mathbb{R}^2$ be finite and in general position and $a, b \in S$. Then ab is a Delaunay edge iff there is an empty circle that passes through a and b.

Delaunay's lemma. By a *triangulation* we mean a collection of triangles together with their edges and vertices. A triangulation K *triangulates* S if the triangles decompose the convex hull of S and the set of vertices is S. If we start with K we can test an edge by considering only one circle and decide for only one point whether or not it is enclosed by that circle. We start with a definition. An edge $ab \in K$ is *locally delone* if

(i) it belongs to only one triangle and therefore bounds the convex hull of S, or

(ii) it belongs to two triangles, abc and abd, and d lies outside the circumcircle of abc.

The definition is illustrated in Figure 8. A locally delone edge is not necessarily an edge of the Delaunay triangulation, and it is fairly easy to construct such an example. However, if *every* edge is locally delone then we can show that all are Delaunay edges.

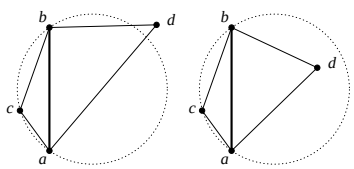

Figure 8: To the left ab is locally delone and to the right it is not.

LEMMA 4. If every edge of K is locally delone then K is the Delaunay triangulation of S.

PROOF. Consider a triangle $abc \in K$ and a vertex $p \in K$ different from a, b, c. We show that p lies outside the circumcircle of abc. Because this is then true for every p, the circumcircle of abc is empty, and because this is then true for every triangle abc, K is the Delaunay triangulation of S. Choose a point x inside abc such

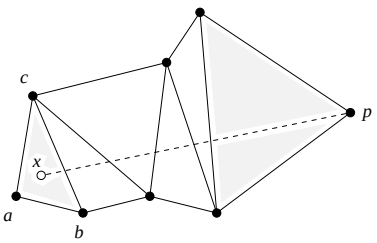

Figure 9: Sequence of triangles in K that intersect xp.

that the line segment from x to p contains no vertex other than p. Let $abc = \tau_0, \tau_1, \ldots, \tau_k$ be the sequence of triangles that intersect xp, as in Figure 9. We write

$\pi_i(p)$ for the power of p to the circumcircle of τ_i, as before. Since the edges along xp are all locally delone, we have $\pi_0(p) > \pi_1(p) > \ldots > \pi_k(p)$. Since p is one of the vertices of the last triangle we have $\pi_k(p) = 0$. Therefore $\pi_0(p) > 0$, which is equivalent to p lying outside the circumcircle of abc. □

Edge-flip algorithm. If ab belongs to two triangles, abc and abd, whose union is a convex quadrangle then we can *flip* ab to cd. Formally this means we remove ab, abc, abd from the triangulation and we add cd, acd, bcd to the triangulation, see Figure 10. The picture of a flip looks like a tetrahedron with front and back superimposed. We can use edge-flips as elementary operations to convert an arbitrary triangulation K to the Delaunay triangulation. Initially, we mark every edge of K and we push every edge on a stack.

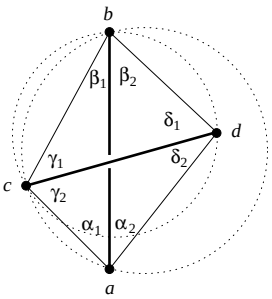

Figure 10: Flipping ab to cd. If ab is not locally delone then the union of the two triangles is convex and cd is locally delone.

```
while the stack is non-empty do
  pop ab from the stack and unmark ab;
  if ab is not locally delone then
    flip ab to cd;
    for xy ∈ {ac, cb, bd, da} do
      if xy is not marked then
        mark xy and push it on the stack
      endif
    endfor
  endif
endwhile.
```

Let n be the number of points. The amount of memory used by the algorithm is $O(n)$ because there are at most $3n-6$ edges, and the stack contains at most one copy of each edge. At the time the algorithm terminates every edge is locally delone. By Lemma 4 the triangulation is therefore the Delaunay triangulation of the point set.

Running time. We show that running time of the algorithm is at most quadratic in the number of points. This also implies that the algorithm terminates. We prove this by interpreting a flip as a tetrahedron in 3-dimensional space. Let $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ be the vertical projections of points a, b, c, d in the x_1x_2-plane to the paraboloid $\Pi : x_3 = x_1^2 + x_2^2$, see Figure 11.

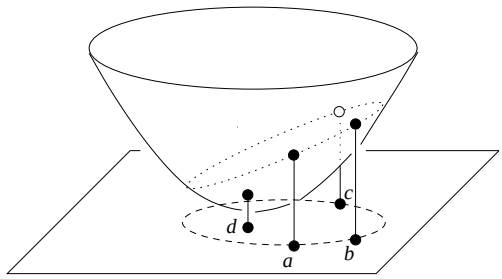

Figure 11: Points a, b, c lie on the dashed circle in the x_1x_2-plane and d lies inside that circle. The dotted curve is the intersection of the paraboloid with the plane that passes through $\hat{a}, \hat{b}, \hat{c}$. It is an ellipse whose projection is the dashed circle.

CLAIM 5. Point d lies inside the circumcircle of abc iff point $\hat{d}$ lies vertically below the plane passing through $\hat{a}, \hat{b}, \hat{c}$.

PROOF. Let U be the circumcircle of abc and H the plane passing through $\hat{a}, \hat{b}, \hat{c}$. We first show that U is the vertical projection of $H \cap \Pi$. Transform the entire space by mapping every point (x_1, x_2, x_3) to $(x_1, x_2, x_3 - x_1^2 - x_2^2)$. Points $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ are mapped back to a, b, c, d and the paraboloid Π becomes the x_1x_2-plane. The plane H becomes an upside down paraboloid that passes through a, b, c. It intersects the x_1x_2-plane in the circumcircle of abc. Plane H partitions Π into a patch below H, a curve in H, and a patch above H. The curve in H projects to the circumcircle of abc, and the patch below H projects to the open disk inside the circle. It follows that $\hat{d}$ belongs to the patch below H iff d lies inside the circumcircle of abc. □

Flipping ab to cd is like gluing the tetrahedron $\hat{a}\hat{b}\hat{c}\hat{d}$ from below to the two triangles next to $\hat{a}\hat{b}$. The algorithm can be understood as gluing a sequence of tetrahedra. Once we glue $\hat{a}\hat{b}\hat{c}\hat{d}$ we cannot glue another tetrahedron right below $\hat{a}\hat{b}$. In other words, once we flip ab we cannot introduce ab again by some other flip that substitutes a locally delone edge for an edge that is not locally delone. This implies there are at most as many flips as there are edges connecting n points, namely $\binom{n}{2}$. Each flip takes constant time, hence the total running time is $O(n^2)$.

There are cases where the algorithm takes $\Theta(n^2)$ flips to change an initial triangulation to the Delaunay triangulation, and one such case is illustrated in Figure 12. Take a convex upper and a concave lower curve and

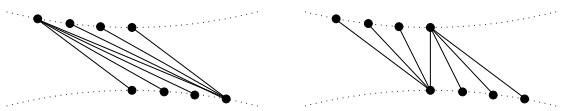

Figure 12: To the left we see about one third of the edges in the initial triangulation, and to the right we see the same number of edges in the final Delaunay triangulation.

place m points on each such that the upper points lie to the left of the lower points. The edges connecting the two curves in the initial and the Delaunay triangulation are shown in Figure 12. For each point count the positions it is away from the middle, and for each edge charge the maximum of the two numbers obtained for its endpoints. In the initial triangulation the total charge is about m^2, and in the Delaunay triangulation the total charge is zero. Each flip moves an endpoint by at most one position and therefore decreases the charge by at most one. A lower bound of about m^2 for the number of flips follows.

Maxmin angle property. A flip substitutes two new triangles for two old triangles. It therefore changes six of the angles. In Figure 10 the new angles are $\gamma_1, \delta_1, \beta_1 + \beta_2, \gamma_2, \delta_2, \alpha_1 + \alpha_2$ and the old angles are $\alpha_1, \beta_1, \gamma_1 + \gamma_2, \alpha_2, \beta_2, \delta_1 + \delta_2$. We claim that for each of the six new angles there is an old angle that is at least as small. Indeed, $\gamma_1 \geq \alpha_2$ because both angles are opposite the same edge, namely bd, and a lies outside the circle passing through b, c, d. Similarly, $\delta_1 \geq \alpha_1$, $\gamma_2 \geq \beta_2$, $\delta_2 \geq \beta_1$, and for trivial reasons $\beta_1 + \beta_2 \geq \beta_1$ and $\alpha_1 + \alpha_2 \geq \alpha_1$. It follows that a flip does not decrease the smallest angle in a triangulation. Since we can go from any triangulation K of S to the Delaunay triangulation this implies that the smallest angle in K is no larger than the smallest angle in the Delaunay triangulation.

LEMMA 6. Among all triangulations of a finite set $S \subseteq \mathbb{R}^2$ the Delaunay triangulation maximizes the minimum angle.

Figure 13 illustrated the above proof of Lemma 6 by sketching what we call the *flip graph* of S. Each triangulation is a node, and there is a directed arc from node μ to node ν if there is a flip that changes the triangulation μ to ν. The direction of the edge corresponds to our requirement that the flip substitutes a locally delone edge for one that is not locally delone. The running time analysis implies that the flip graph is

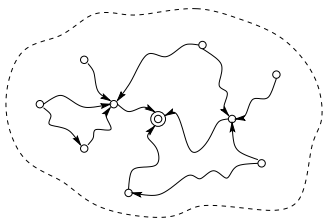

Figure 13: Sketch of a flip graph. The sink is the Delaunay triangulation. There is a directed path from every node to the Delaunay triangulation.

acyclic and that its undirected version is connected. If we allow flips in either direction we can go from any triangulation of S to any other triangulation in less than n^2 flips.

Bibliographic notes. A proof of Lemma 4 and its generalization to arbitrary finite dimensions is contained in the original paper by Boris Delaunay [1]. The edge-flip algorithm is due to Charles Lawson [3]. The algorithm does not generalize to three or higher dimensions. For planar triangulations the edge-flip operation is widely used to improve local quality measures, see e.g. Schumaker [4]. For almost all interesting measures the algorithm gets caught in local optima. The observation that the Delaunay triangulation maximizes the smallest angle was first made by Robin Sibson [5]. Minimizing the largest angle seems more difficult and the only polynomial time algorithm uses edge-insertions, which are somewhat more powerful than edge-flips [2].

[1] B. DELAUNAY. Sur la sphère vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* **7** (1934), 793–800.

[2] H. EDELSBRUNNER, T. S. TAN AND R. WAUPOTITSCH. An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. *SIAM J. Sci. Stat. Comput.* **13** (1992), 994–1008.

[3] C. L. LAWSON. Software for C^1 surface interpolation. In *Mathematical Software III*, Academic Press, New York, 161–194.

[4] L. L. SCHUMAKER. Triangulation methods. In *Topics in Multivariate Approximation*, C. K. Choi, L. L. Schumaker and F. I. Utreras (eds.), Academic Press, 1987, 219–232.

[5] R. SIBSON. Locally equiangular triangulations. *Comput. J.* **21** (1978), 243–245.