Extended Abstract-Randomized Incremental Construction of Delaunay Triangulation

Talk by Loreta Vrapi

Based the book "Computational Geometry: Algorithms and Applications", Chapter: "Delaunay Triangulation" by M. Berg, O. Cheong, M. Kreveld and M. Overmars

Supervised by Professor Stefan Felsner

Abstract

This talk focuses on the development and analysis of the randomized incremental algorithm for Delaunay triangulation and its probabilistic behaviors. This algorithm efficiently constructs the Delaunay triangulation by inserting points incrementally while maintaining the Delaunay property. It leverages randomization techniques to achieve efficient average-case performance.

Keywords: Delaunay Triangulation, Randomized Incremental Construction (RIC), Flip.

1. Introduction

The randomized incremental algorithm is a method used in computational geometry for constructing geometric structures, such as triangulations by incrementally adding points in a random order. The random order, in which points are added, introduces an element of probabilistic behavior to the algorithm. This randomization helps to distribute the points more evenly and reduces the likelihood of worst-case scenarios.

Delaunay triangulation is defined as a geometric structure that partitions a set of points into a set of non-overlapping triangles, satisfying the Delaunay criterion. The Delaunay criterion states that no point lies inside the circumcircle of any triangle in the triangulation.

2. Computing the Delaunay Triangulation using the Randomized Incremental approach

The algorithm proceeds as follows:
1- Add two new points, p_{-1} and p_{-2}, to the existing point set P. See Figure 1
2- Initialize a triangulation by forming the triangle $p_0p_{-1}p_{-2}$, where p_0 is the highest point of the set P, ensuring that this triangle encompasses all the other points within its interior.
3- Incrementally add the remaining points to the existing triangle, maintaining a random order of insertion.
4- Perform necessary flips, if required, to preserve the Delaunay condition after each iteration.
5- Discard the points p_{-1} and p_{-2}, as they were solely introduced to create the initial enclosing triangle and are no longer needed for the final Delaunay triangulation.

The points p_{-1} and p_{-2} are chosen "far" away to ensure the preservation of the Delaunay condition when they are removed.

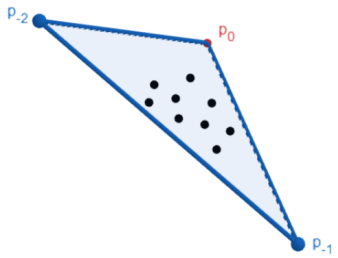

Figure 1: The Initial Big Triangle: Encompassing All Points Within Its Bounds

3. The Analysis

3.1. Flip - Legalize edge:

An edge that violates the Delaunay criterion is referred to as 'illegal'. We can maintain the Delaunay condition by legalizing/flipping all the edges of the new triangles. See Figure 2.

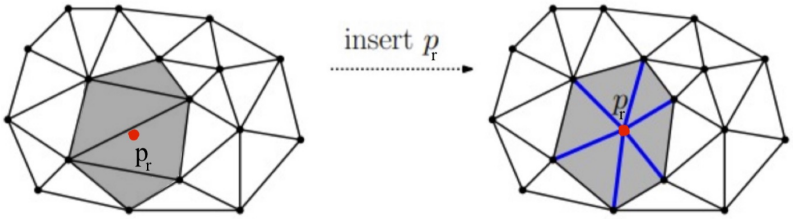

Figure 2: Legalizing the edges

3.2. Point-Location:

While building the triangulation we build a point location structure. See Figure 3.

4. Probabilistic behaviors

Lemma 1. *The expected number of triangles created by the algorithm is at most* $9n + 1$.

Beweis 1. *Idea: By inserting a new point* p_r *at iteration* r *we split one or two triangles, which create the same number of edges in the triangulation. For every edge we legalize (flip), we create two new triangles. The flip creates an edge incident to* p_r
So: If after inserting p_r *there are* k *edges incident to* p_r*, then we have at most* $2(k-3)+3 = 2k-3$ *new triangles. The number* k *is also the degree of* p_r*. We want to bound the expected degree of* p_r*, which is a random number of the inserted points. Delaunay has at most* $3(r+3)-6$ *points (Euler's formula). So the total degree is less than* $2(3(r+3)-9) = 6r$*. This means, the expected degree of a randomly inserted point is at most* 6*. To summarize, we can bound the number of triangles created in iteration* r*:*

$$E[\ nr.\ of\ triangles\ in\ step\ r\ ] \leq E[2deg(p_r) - 3] = 2E[deg(p_r)] - 3 \leq 2 * 6 - 3 = 9 \qquad (1)$$

In total, we have 9 *triangles for each step and the big triangle* $p_0p_{-1}p_{-2}$*. Using linearity of expectation, we have at most* $1 + 9n$ *triangles.*

□

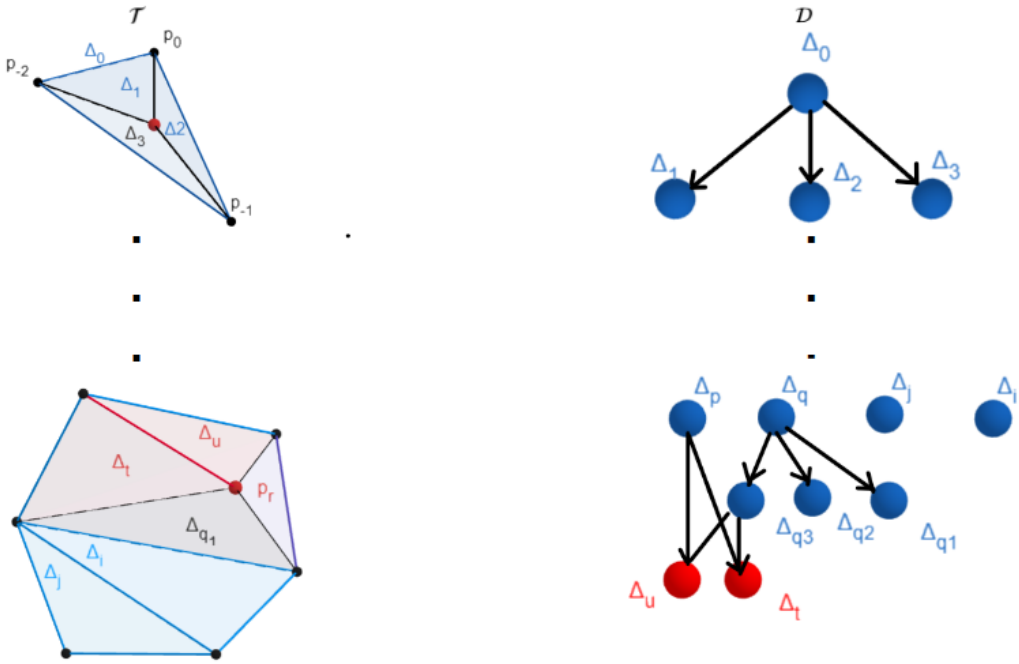

Figure 3: Point-Location: Locate p_r inserted at iteration r

Lemma 2. *The Delaunay triangulation of a set P of n points in the plane can be computed in $O(n \log n)$ expected time, using $O(n)$ expected storage.*

Beweis 2. *We suppose that P is a point set in general position.*
a- The time spent by the algorithm (without the Point-Location step) is proportional to the number of triangles created. By Lemma 1 it is $O(n)$.
b- The time to locate p_r in the current triangulation is linear in the number of nodes to be visited. Counting the triangles at current triangulation to locate p_r is $O(1)$ adding the linear time in the number of the destroyed triangles that contain p_r.
Let $K(\triangle)$ be the set of points lying in the circumcircle of $\triangle$. To locate p_r, the visit is charged to the triangle $\triangle$ with $p_r \in K(\triangle)$. So is each $\triangle$ charged at most once for every point in $K(\triangle)$.
In total we have $O(n + \sum_{\triangle} card(K(\triangle)))$ for the location step.
For T_r, the set of triangles of the triangulation in the r iteration, rewrite:

$$\sum_{\triangle} card(K(\triangle)) = \sum_{r=1}^{n} (\sum_{\triangle \in T_r - T_{r-1}} card(K(\triangle)))$$

Denote k_q the number of triangles $\triangle \in T_r$ with q a point in $K(\triangle)$ and k_{q,p_r} the number of triangles $\triangle \in T_r$, with $q \in K(\triangle)$ and p_r incident to $\triangle$.

Then $\sum_{\triangle \in T_r - T_{r-1}} card(K(\triangle)) = \sum_{q \in P - P_r} k_{q,p_r}$.

Suppose we fix P_r for a moment. A triangle $\triangle \in T_r$ is incident to a random point $p \in P_r$ with probability at most $3/r$ and consequently $E[k_{q,p_r}] \leq (3 \times k_q)/r$. Now we try to find an upper-bound for $E[\sum_{\triangle \in T_r - T_{r-1}} card(K(\triangle))]$, which depents on the number n and r. Then by summing over r we will get the wanted result.

□