Delaunay Triangulation in Parallel

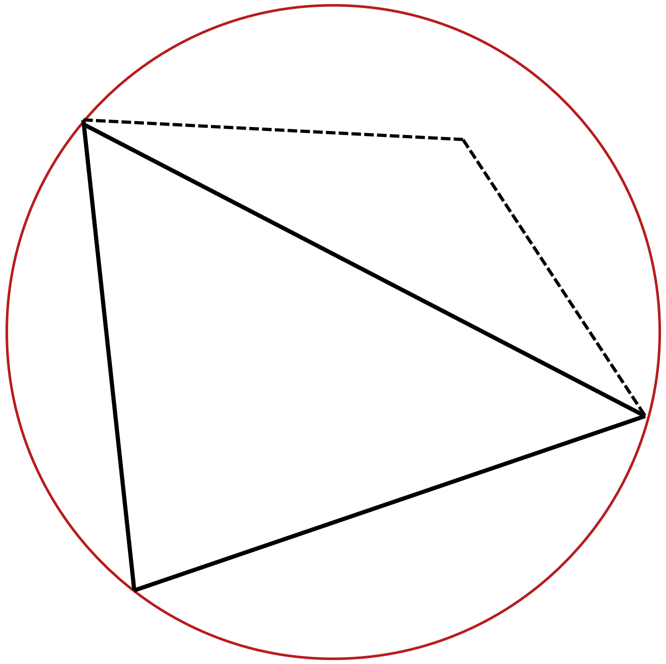
Adarsh Prakash

CSE 633 : Parallel Algorithms

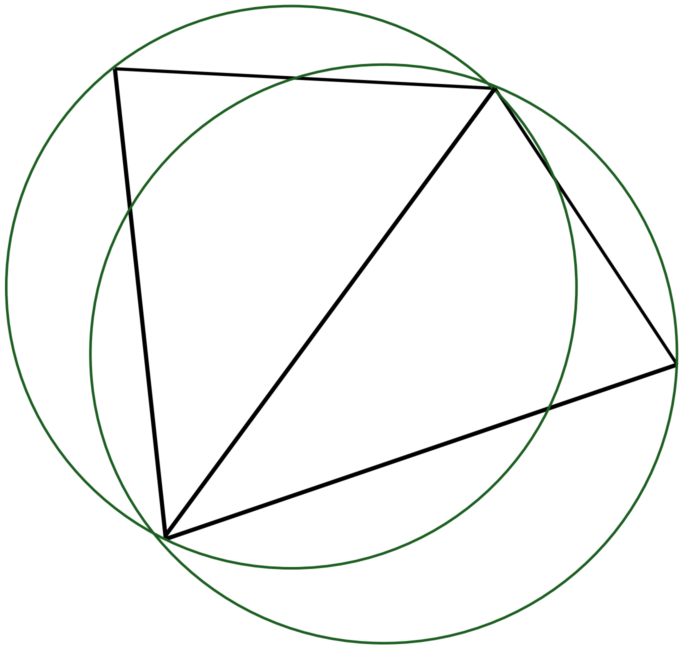
University at Buffalo

Definition

Triangle formed by points A, B & C (ΔABC) is **Delaunay Triangle**, if no other points lie in the circumcircle of ΔABC.



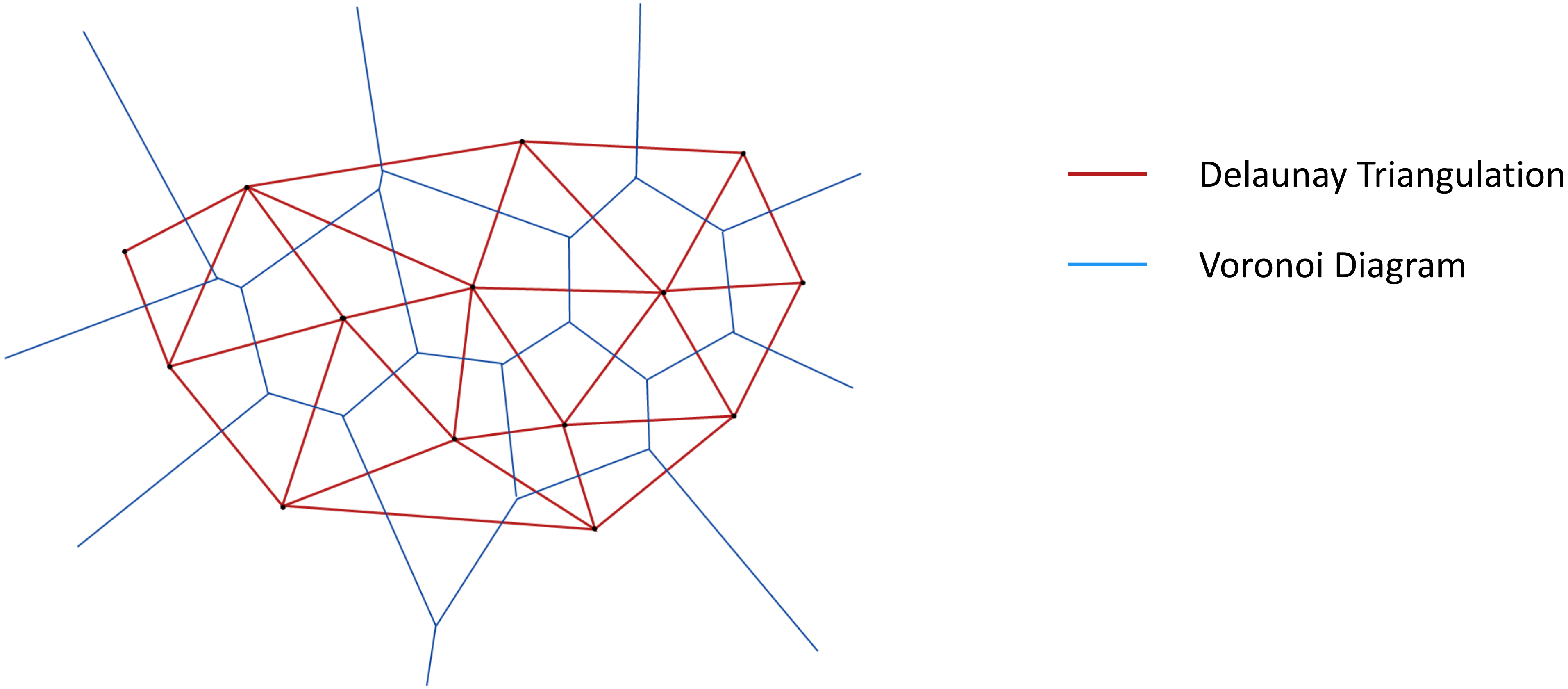
Invalid Delaunay Δ



Valid Delaunay Δs

Delaunay v/s Voronoi: Duality

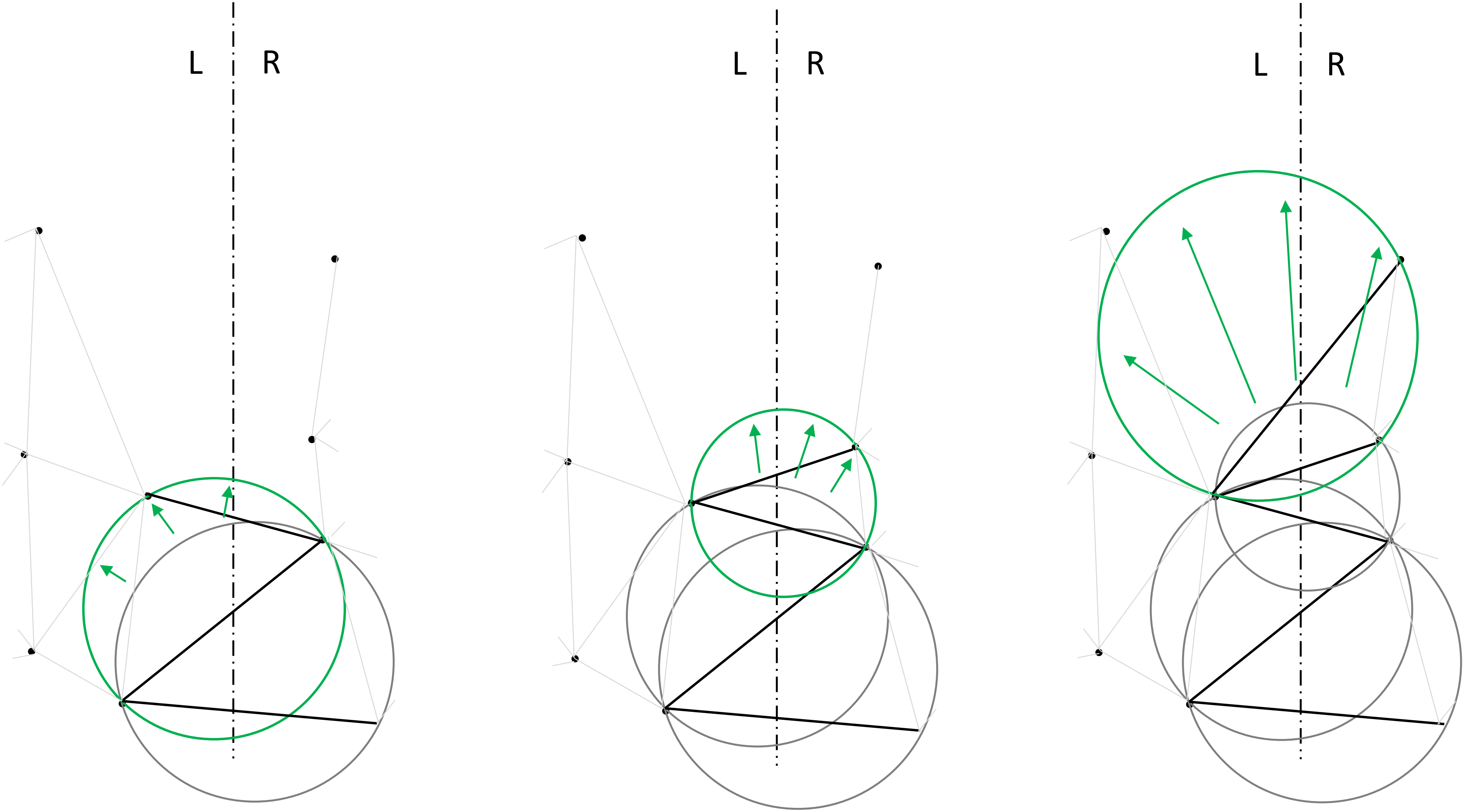
A Voronoi diagram is constructed by connecting the centers of all the circumcircles formed by the Delaunay Triangles in a graph.



Algorithm

- **Divide-and-conquer** algorithm proposed by **Leonidas Guibas** and **Jorge Stolfi** [1].
- Follows closely the Voronoi construction algorithm from **Shamos** and **Hoey** [2].
- Difference is it clearly describes how to make use of quad-edge data structure to avoid computation of complete hull.
- Properties:
  - A quad-edge know their direction (origin-destination NOT point-point)
  - A quad-edges maintains pointers to all edges leaving from and terminating at their origin and destination. (4-8 pointers depending on implementation)
- Plan is to parallelize this algorithm.

Algorithm: Merge Step



Analysis

- Sequential runtime: ***O(n * logn)*** [*T(n) = 2 * T(n/2) + O(n)*]
- "Heavy" merge step with *O(n)*. Parallelization possible?!!
- Analysis with ***p*** processors:
  - Each processor locally and simultaneously computes DT on $\frac{n}{p}$ points → $O\{\frac{n}{p} * log\ (\frac{n}{p})\}$
  - DTs from each processor is stitched together (happens *logp* times) → *O(n * logp)*
  - So, total runtime = $O\ \{\frac{n}{p} * log\ (\frac{n}{p})\ \ + n * logp\}$
- If *p = logn*, runtime = ***O(n log(logn))***
- Let's see if we can reach that theoretical target.

Plan

- Need for scaling rules out OpenMP.
- Implementation in MPI.
- Input is randomly generated points following Uniform Distribution.
- Preprocess input to sort by x-coordinate.
- Timeline: Finish serial code by this weekend and start parallelizing.

References

- *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams* – **Guibas, L.** and **Stolfi, J.**
- *Closest-Point Problems* – **Shamos, M.I.** and **Hoey, D.**
- *On computing Voronoi diagrams by divide-prune-and-conquer* – **Amato, N.M.** and **Ramos, E.A.**
- *Chapter 10: Computational Geometry, Algorithms – Sequential and Parallel* **– Miller, R.** and **Boxer, L.**