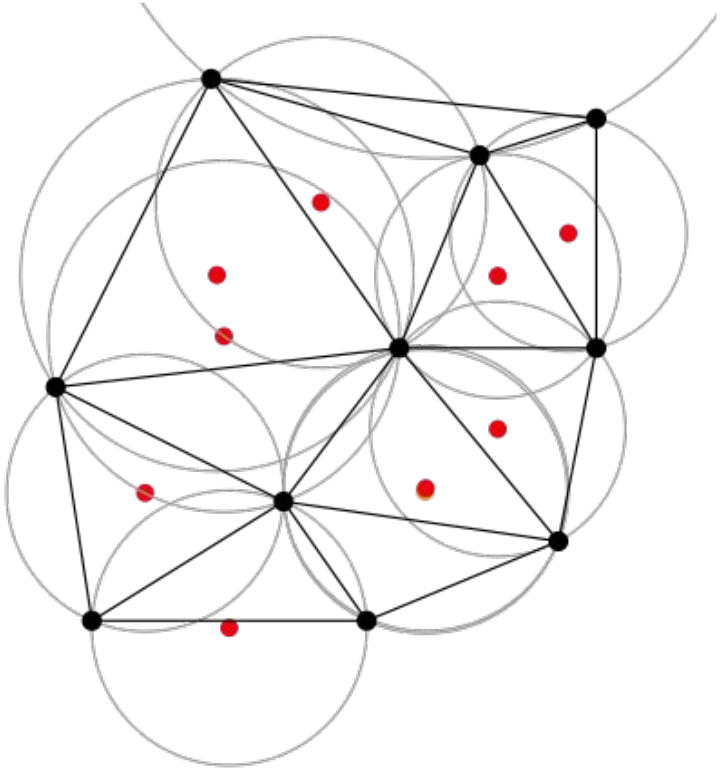
Delaunay Triangulations in MPI

Isys Johnson
CSE 633

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

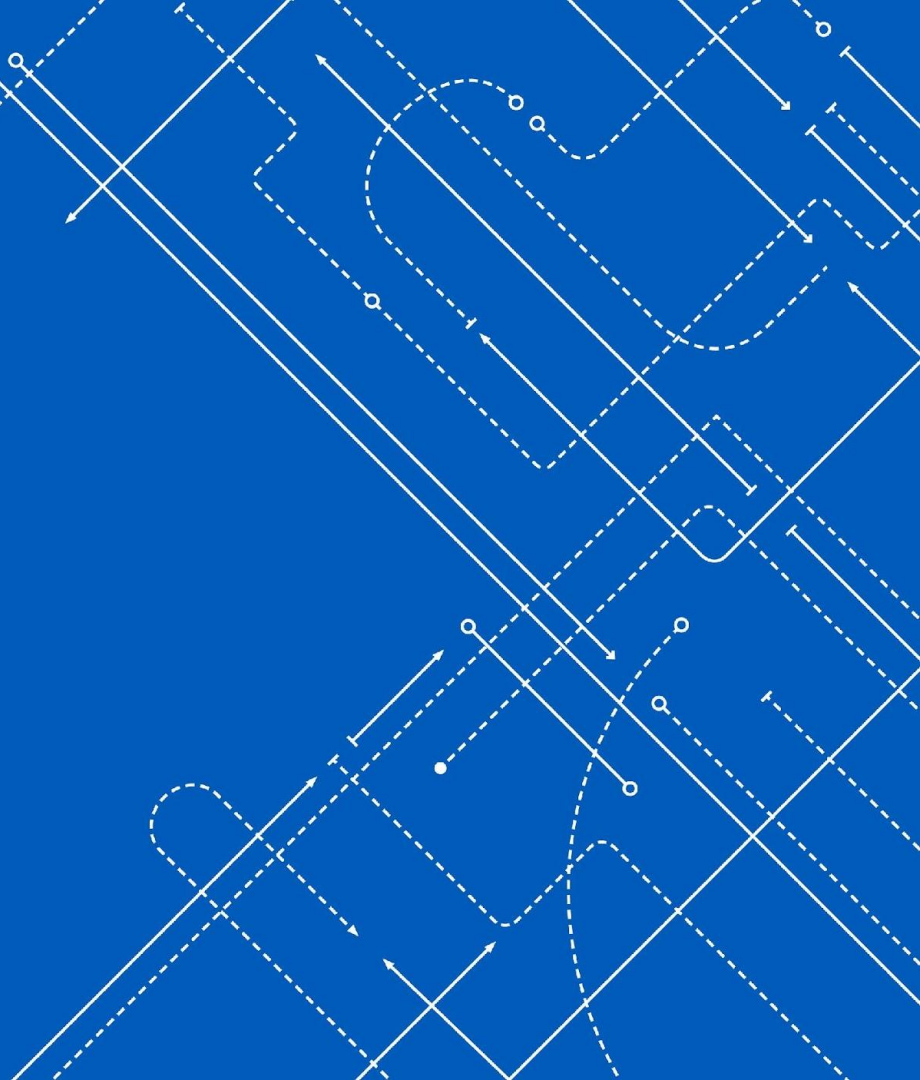
Delaunay Triangulation

The Delaunay triangulation for a given set of points P is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P).



University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Parallel Divide and Conquer scheme for 2D Delaunay triangulation



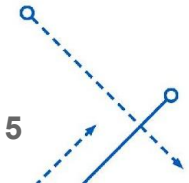
Outline of Parallel Implementation

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Outline of Triangulation Scheme

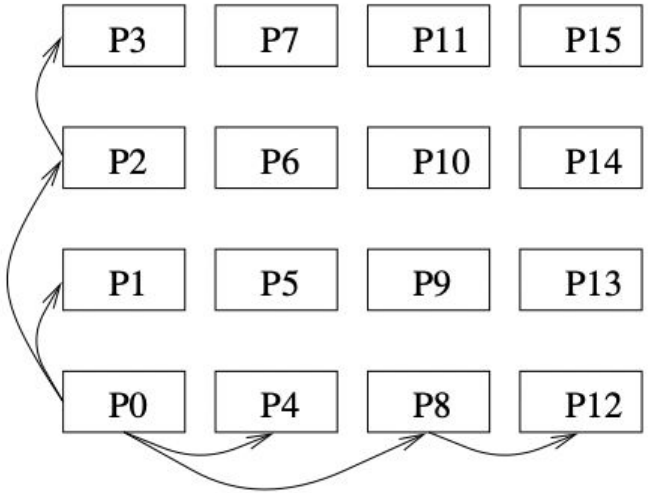
We follow the scheme and implementation described in “Parallel Divide and Conquer scheme for 2D Delaunay triangulation by Chen et al.

1. Distribute data to 2D grid of processors
2. Triangulate each block by applying the Rex A. Dwyer’s divide and conquer algorithm on individual processors
3. Create the affected zone on individual processors
4. Merge the affected zones in the reverse of partitioning the point set and transmit the modified triangles back to the associated processes.



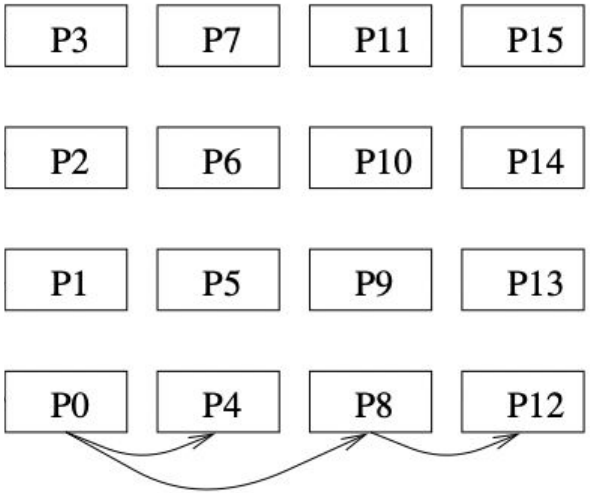
Overview of Parallel Algorithm

1. Configure the P processors as a 2D Array



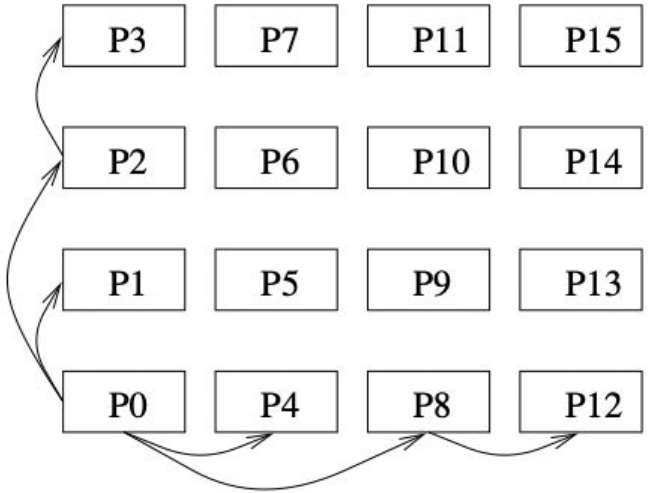
Overview of Parallel Algorithm

2. Partition the whole set to the first processors of each column



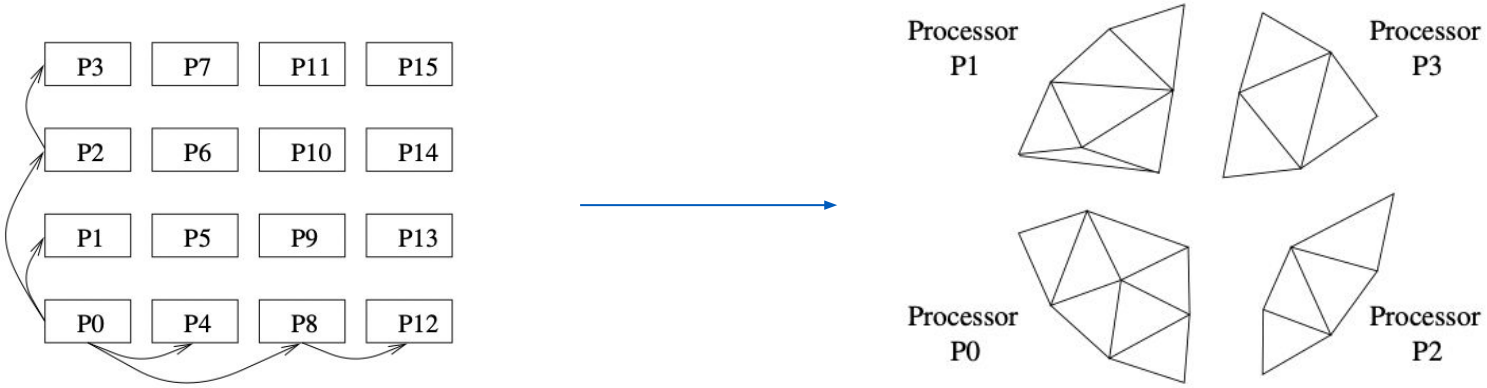
Overview of Parallel Algorithm

3. Partition the point set on the first processor to the other processors on its column.



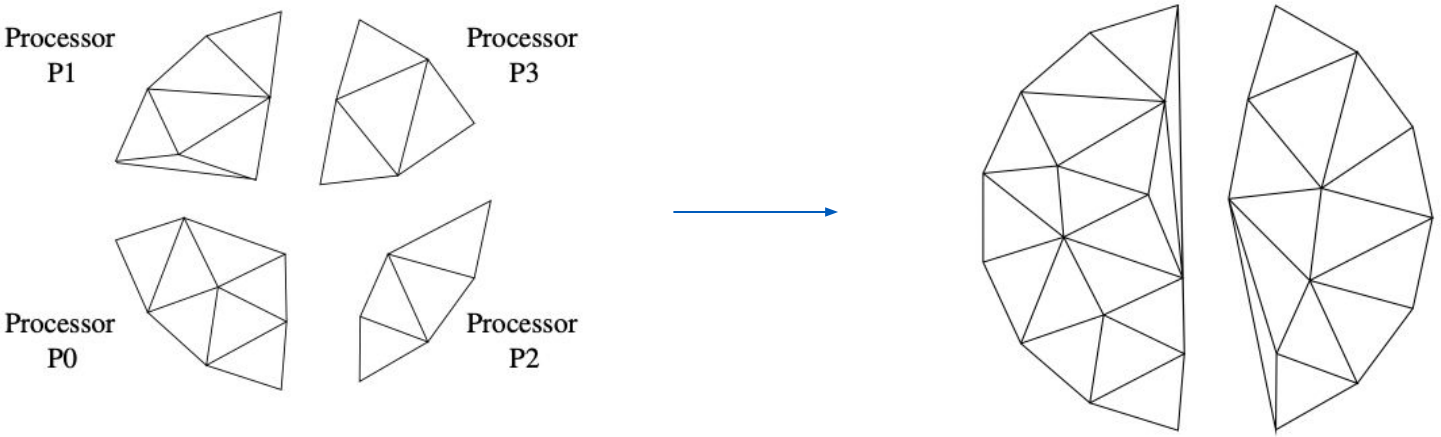
Overview of Parallel Algorithm

4. Triangulate each block by applying the divide and conquer algorithm on individual processors



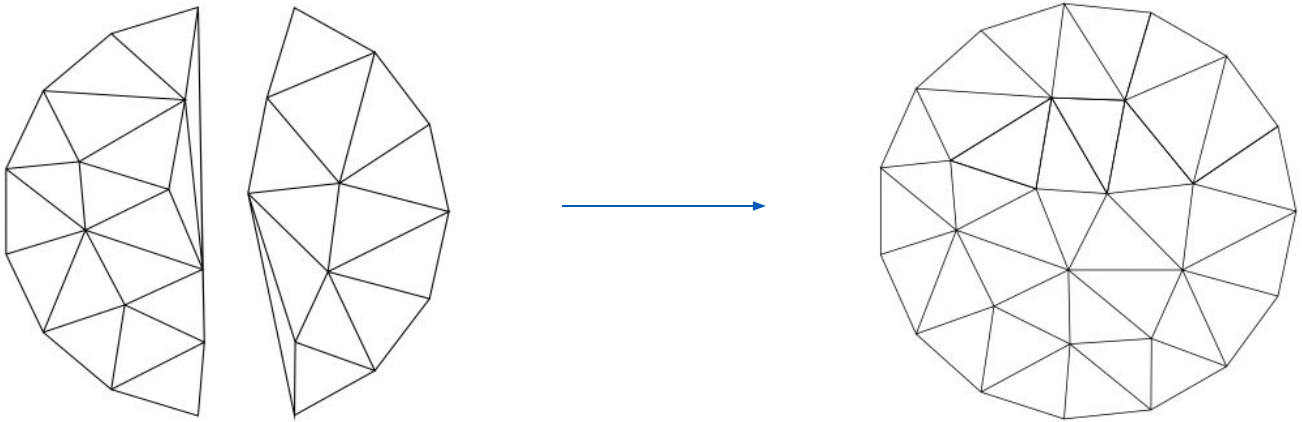
Overview of Parallel Algorithm

5. Merge triangulated blocks, checking and adjusting to maintain the circumcircle property



Overview of Parallel Algorithm

5. Merge triangulated blocks, checking and adjusting to maintain the circumcircle property



University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Implementing the *Affected Zone*

Implementing the *Affected Zone*

In most parallel implementations of the Delaunay triangulation, most of the time is spent at the merge step.

To improve efficiency, Chen et. al narrows the scope of the this merge step to only the triangles that would be affected by the triangulations. This is called the **affected zone**

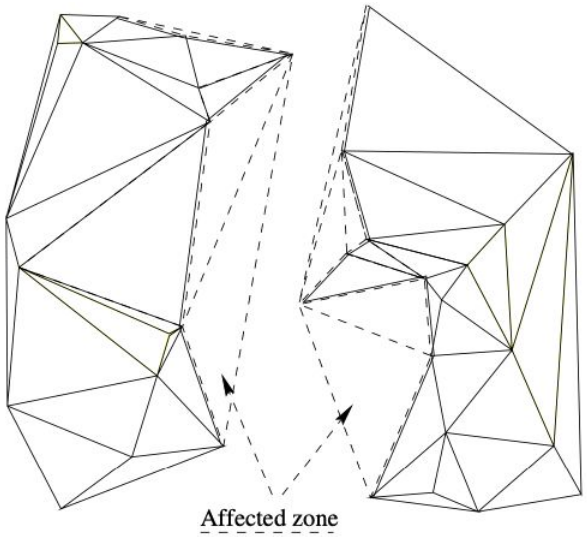


Figure 5. The affected zones of two triangulations.

Implementing the Affected Zone

This leads to the question:

What triangles make up the affected zone?

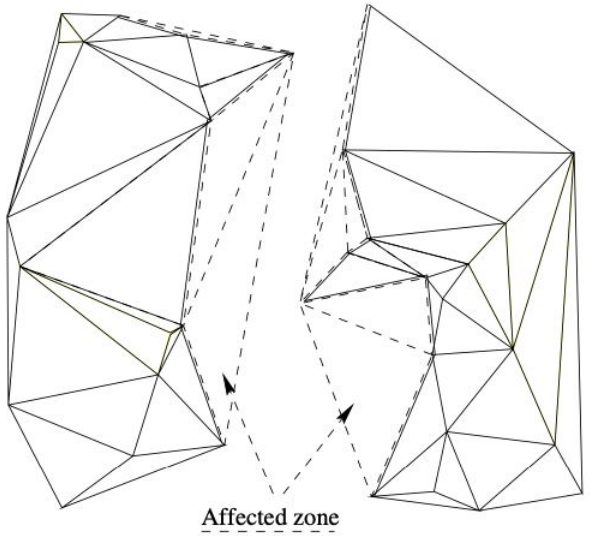
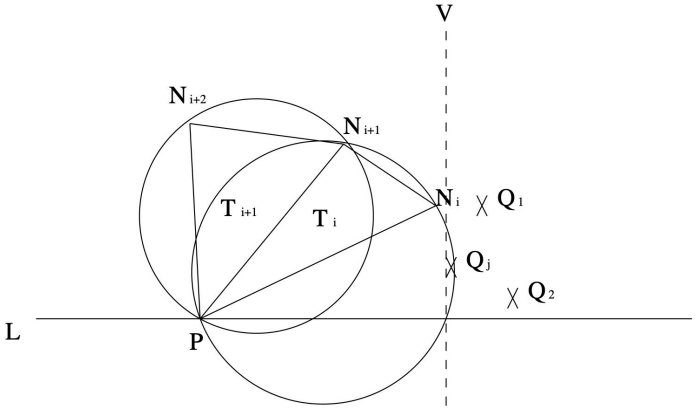


Figure 5. The affected zones of two triangulations.

Implementing the Affected Zone

For a given Delaunay triangulation, new points are added to form a new Delaunay triangulation. Assuming a specified line outside the given triangulation, the newly added points are on different sides of the line.

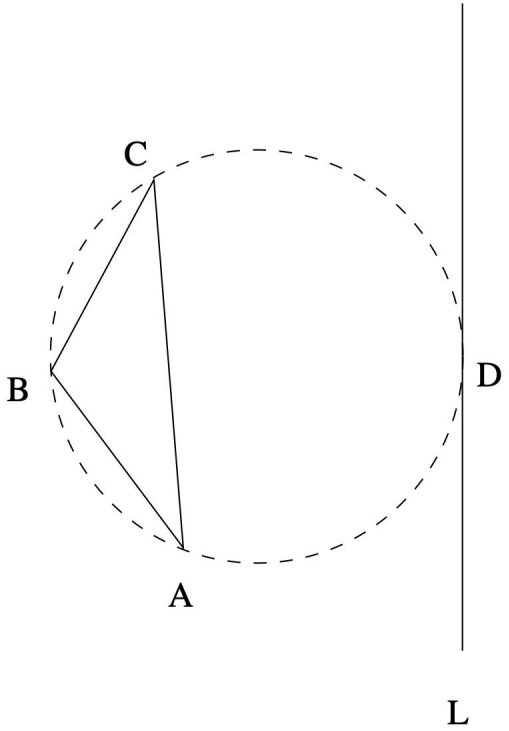
In the given triangulation, the interior triangles whose circumcircles touch the line may not be Delaunay triangles for the new point set. That is, these triangles are indeterminate.



The Line-incircle Test

For a triangulation to be Delaunay, all triangles must pass the line-incircle test.

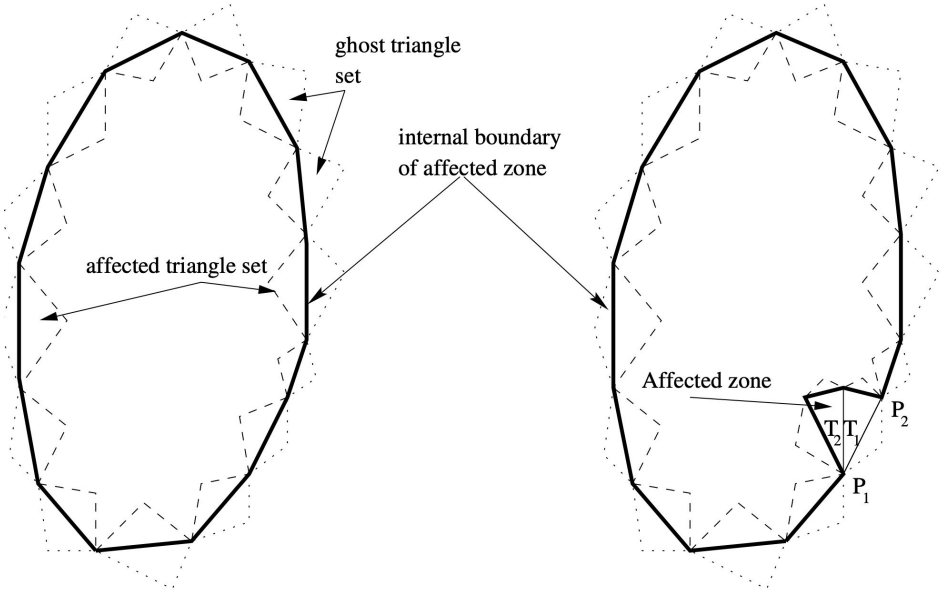
If the line is inside the circumcircle of the triangle, then the triangle passes the line-incircle test.



Implementing the Affected Zone

These indeterminate triangles make up the candidates for the affected region.

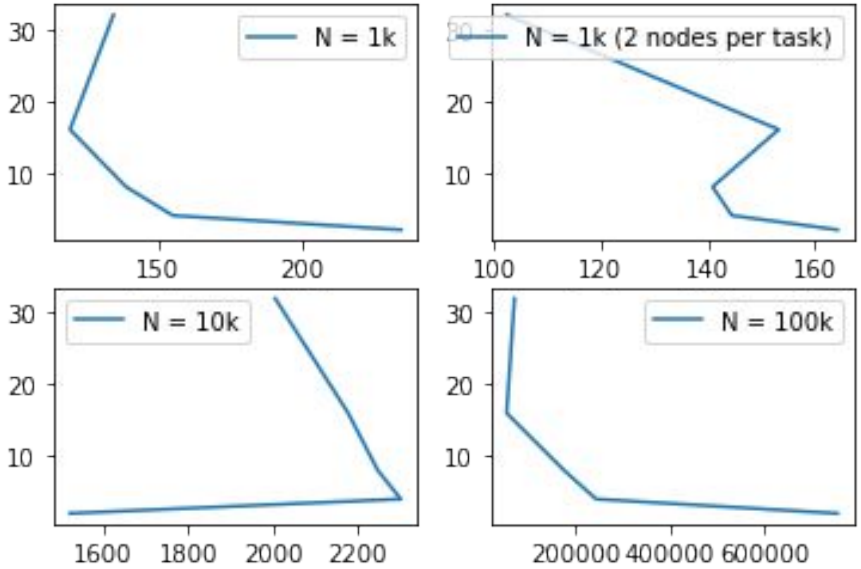
Now, the indeterminate triangles can be searched inwardly from the triangles on the convex hull. The affected region can be constructed by discovering all the triangles that pass the line-incircle test.



University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Experiments and Results

Results - Runtimes on Different Grid Sizes and Processors



Results - Runtimes on Different Grid Sizes and Processors

PEs/Nodes	Avg Runtime Size = 1000	Avg Runtime Size = 10k	Avg Runtime Size = 100k
1	234.82	521.32	753699.71
1	155.11	2301.81	244035.51
2	138.75	2247.64	176626.32
4	19.16	53.33	55223.45
8	134.35	102.34	71598.76
16		2177.68	
32		2005.68	

Bibliography

- *Parallel divide-and-conquer scheme for 2D Delaunay triangulation.* Min-Bin Chen, Tyng-Ruey Chuang, Jan-Jan Wu. 2006
- *A Simple Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations in O(n loglogn) Expected Time.* Dex A. Dwyer
- *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams*. Leonidas Guibas and Jorge Stolfi. 1985
- *Python + MPI implementation of Guibas & Stolfi's D&C Delaunay Triangulation Algorithm: https://github.com/v-hill/delaunay-triangulation/*
- *Dywer DT algorithm implementation in C by Dwyer: https://github.com/rexdwyer/DelaunayTriangulation*

Future Work

- Comparing performance with different point distributions
- Parallelizing the sort procedure
- Adding Visualizations