# Separators of Delaunay triangulations

Jonathan MacKenzie* Evan Thorpe†

March 2, 2019

## 1 Introduction

Our chosen problem is a combination of two problems and their algorithms, namely *Delaunay triangulation* in $\mathbb{R}^2$ and the *Planar Separator theorem* of Lipton and Tarjan [5]. The former triangulates (connects into triangles) points in $\mathbb{R}^2$ such that no point is contained in the circumscribed circle of any triangle in the triangulation.[1] Given a set of points in $\mathbb{R}^2$, a Delaunay triangulation can be found in $O(n\log n)$ time. Lipton and Tarjan's separator theorem states that for any planar graph, one can always partition the graph into two disjoint connected subgraphs, each of size no more than $2n/3$, by removing at most $2\sqrt{2}\sqrt{n}$ vertices. These vertices to remove can be found in $O(n)$ time [5].

The question, as posed by Stefan Langerman in [7], is whether or not such an $O(\sqrt{n})$ separator can be found for a Delaunay triangulation in $O(n)$ time given only the set of points in $\mathbb{R}^2$.

Practical applications of such an algorithm are not immediately obvious, but Delaunay triangulation is heavily used in graphics because it avoids creating "sliver triangles" (triangles with one or two very acute angles) and the ability to split a graph into roughly equal-sized partitions in $O(n)$ time at the cost of $O(\sqrt{n})$ vertices can be very useful for recursive divide-and-conquer algorithms.

## 2 Related Work

Delaunay triangulations are very well-known constructs in computational geometry. Two main approaches exist to find them, both of which are $O(n\log n)$ in time. The first is an incremental approach, as described in [2], the second is a divide-and-conquer approach, as described in [3].

It should be noted that both of these papers refer to Voronoi diagrams in their titles, and not Delaunay triangulations. A Voronoi diagram partitions the plane around points in the plane such that each section contains one point

*Department of Math and Computing Science, Saint Mary's University

†Department of Math and Computing Science, Saint Mary's University

[1] A delaunay triangulation can also be defined as a triangulation such that for any triangles sharing an edge, their opposing angles add to at least $\pi$

and is exactly the region consisting of all points closest to that point than any other. The Voronoi diagram of a set of points is dual to the Delaunay triangulation of those points, and in fact can be constructed by connecting the centres of the circumcircles of each triangle in the triangulation. Thus, the aforementioned papers give algorithms to find Voronoi diagrams by first finding Delaunay triangulations.

Lipton and Tarjan's separator theorem, as previously mentioned, provides an algorithm to separate a graph into disjoint connected subgraphs, each containing no more than 2/3 of the original vertices, at the cost of $O(\sqrt{n})$ vertices.

The problem of finding a $O(\sqrt{n})$ separator of the Delaunay triangulation of a set of points in linear time is currently an open problem, posed in [7].

# 3 Algorithm/Theoretical Analysis

Most of our work done so far has been combing through papers of different separators, such as in [5] and [6], as well as papers on Delaunay triangulation. We have also been searching for packages that implement these algorithms, breaking apart and trying to understand the algorithms implementation.

Our initial algorithm will be the concatenation of a graph separator algorithm and a Delaunay triangulation. As graph separation runs in linear time and Delaunay triangulation is $O(n\log n)$, this first "concatenation" algorithm will be $O(n\log n)$ in time.

The graph separation algorithm is somewhat more complex than we anticipated, which may make it more difficult to dissect it and integrate the Delaunay aspect. There are, however, some parallels: one part of the Lipton-Tarjan algorithm involves triangulating the graph (turning all faces into triangles by adding edges) but this is after some other steps so it is unclear whether or not this will be an advantage.

We've also been examining other separation algorithms, especially [6] which is an algorithm for separating sphere packings. As graphs can be represented by sphere packings,[2] it follows that any graph's corresponding sphere packing can be separated by removing at most $O(\sqrt{n})$ spheres. This spatial representation could prove very useful, as our problem input with be spatial data.

The order in which we've done things differs slightly from our plan in the Gantt chart. Instead of researching one topic and then looking for/ writing code for it, we've been researching both topics first and will begin implementing the algorithms soon.

The biggest difficulties we've faced would be either finding the time to work on it, or finding time to work together. Thankfully, for the research and papers section of this assignment, working together isn't that important. As stated before, most of our time has been individually reading papers. Each time one of us found the time to work on this project, we were able to get a good chunk done.

[2] Vertices are represented as spheres who touch if and only if the corresponding vertices are connected by an edge

# 4 Results and Discussion

For the Lipton-Tarjan algorithm, we found existing packages in both Python and C++. For testing, both packages contain tools that may prove useful. The python package has a random planar graph generator, as well as a way to test if the algorithm worked correctly. The C++ package has multiple pre-defined graphs that we can test on. If we decide to use either of these packages, our plan would be to learn how they work and modify them to our needs.

As for code of Delaunay triangulation, we have found two different algorithms, both of them using python with numpy. The two differ in the fact that they implement different algorithms.

The implementation in [4] uses the Bowyer-Watson algorithm, which has an average time complexity of $O(n\log n)$, however, its worse case running time is $O(n^2)$[1]. As the author of the code notes, they did not write it to be efficient, but rather wrote it for readability.

The other implementation, seen in [8], uses the s-hull method. This method also has a time complexity of $O(n\log n)$. The author of this code admits that it could be improved with support for holes, and also by having it automatically add Steiner points.

When testing either of these implementations, they are both capable of generating random points. Both implementations also are very well documented, which should help when modifying them to our needs.

If we are not able to develop a linear-time algorithm for Delaunay separation, we will aim to make an optimized algorithm and run some empirical tests, comparing it to the simple concatenation algorithm and examining how it reacts to various inputs.

# References


[1] C. Arens. The bowyer-watson algorithm; an efficient implementation in a database environment. *Case study report TU Delft, July 2002, 52 p.*, 2002.

[2] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7, 06 1992.

[3] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, 1983.

[4] V. Likhosherstov. lipton_tarjan. `https://github.com/ValeryTyumen/lipton_tarjan`, 11 2018.

[5] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36, 1979.

[6] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44, 01 1997.

[7] J. ORourke. Open problems from CCCG 2017. *Canadian Conference on Computational Geometry*, 2018.

[8] A. Pletzer. Delaunay triangulation (python recipe). https://code.activestate.com/recipes/579021-delaunay-triangulation/, 02 2015.