Separation of Voronoi Areas

Martin Gebert, Sascha Schreckenbach, Jonathan Sharyari

24th January 2013

Abstract

....

1 Introduction

1.1 Voronoi Diagram

Given a set of points in space, a Voronoi diagram is a partitioning of the space into a set of Voronoi regions or ”cells”. The Voronoi region corresponding to a point p consists of all points that are closer to p than to any other point in space.

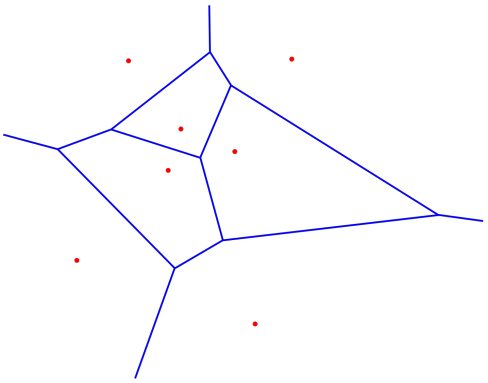

Figure 1: A Voronoi diagram.

1.2 Delaunay triangulation

A Delaunay triangulation for a set of points in space is a triangulation of the points, with the added property that no point is inside the circumcircle of any triangle in the triangulation.
The Delaunay triangulation is dual to the Voronoi diagram.

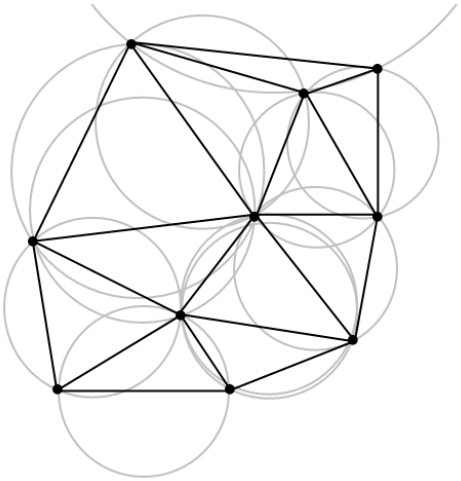

Figure 2: A Delaunay triangulation, with a set of corresponding Delaunay circles.

1.3 Arrangement

Given a set of planar curves, the arrangement is the subdivision of the plane into zero-dimensional, one-dimensional and two-dimensional cells, called vertices, edges and faces, respectively induced by the given curves. 1

1.4 The Set Cover Problem

Given a set $\mathcal{A}$ of sets of numbers, the set cover problem is the problem of finding a minimal subset $\mathcal{B}$ of $\mathcal{A}$, so that the union of the sets in $\mathcal{B}$ is equal to the union of all elements of $\mathcal{A}$.

1.4.1 Example

Let $\mathcal{A} = \{\{1,2\}, \{2,3\}, \{2,5\}, \{3,5\}\}$. The union of all sets of $\mathcal{A}$ is $\{1,2,3,5\}$.
The subset $\mathcal{B}_1 = \{\{1,2\}, \{2,3\}\}$ has the union $\{1,2,3\}$ and does not cover the set $\mathcal{A}$.
The subset $\mathcal{B}_2 = \{\{1,2\}, \{2,3\}, \{2,5\}\}$ has the union $\{1,2,3,5\}$ that covers $\mathcal{A}$, but it is not minimal.
A minimal solution is $\mathcal{B}_3 = \{\{1,2\}, \{3,5\}\}$.
The set cover problem is known to be NP-complete.4

2 Problem formulation

Given a set $\mathcal{R}$ of red points, find a set $\mathcal{B}$ of blue points such that in the Voronoi diagram of $\mathcal{R} \cup \mathcal{B}$, no two regions corresponding to points in $\mathcal{R}$ are incident to each other. For the Delaunay triangulation of $\mathcal{R} \cup \mathcal{B}$, this means that there is no edge connecting two red points.

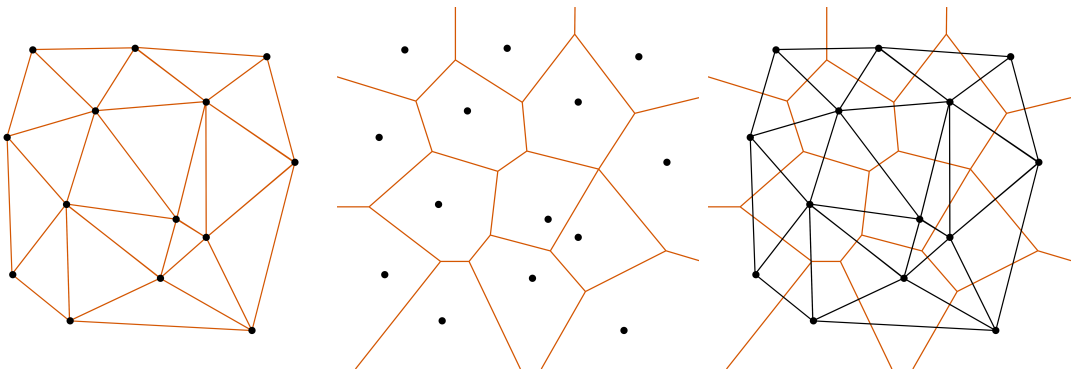

Figure 3: A triangulation of a set of points (left), the corresponding Voronoi diagram (middle) and the overlapping of the triangulation and the Voronoi diagram, emphasizing the dual relationship (right).

3 Background

3.1 Upper and lower bounds

Given a set $\mathcal{R}$ of red points with $N = |\mathcal{R}|$, it has been shown 2 that at least N-1 points are needed to solve this problem. This is exemplified in the picture below, and always applies to the special case when all points lie in row. In the general case, it has been shown that for $N > 2$ there are cases where at least N blue points are required to solve the problem.

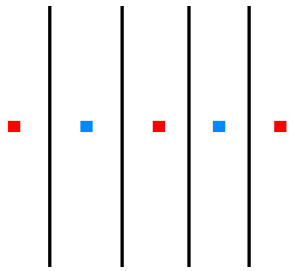

Figure 4: Example of points not in general position, having a solution with N-1 blue points.

In the general case, it has been proved 2 that at most $3N/2$ blue points are needed to solve the problem, and in the case where the points in $\mathcal{R}$ are in convex position, $5N/2$ points are sufficient. It has been conjectured that N points are sufficient to solve the problem in the general case.

3.2 complexity

Although not yet shown, the problem is suspected to be NP-hard 2, as a similar problem has been proven to be NP-hard. 3

4 Algorithms

4.1 General outline

1. For the set $\mathcal{P}$ of points (both red and blue) find the delaunay triangulation.
2. For each edge connecting two red points in the triangulation, find a circle with the two points on its perimeter and add these to the (possibly empty) set of circles $\mathcal{C}$.
3. Calculate the arrangement $\mathcal{A}$ of the circles $\mathcal{C}$.
4. Find a set of points $\mathcal{P}^*$, so that for every face in the arrangement $\mathcal{A}$, exactly one point is located in the interior. For every point $\rho \in \mathcal{P}^*$ and every circle ς in $\mathcal{C}$, determine whether ρ is in the interior of ς.
5. Find a minimum subset of $\mathcal{P}^*$, so that every circle in $\mathcal{C}$ is covered by at least one point.

6. Calculate a triangulation for the set of points $\mathcal{P}^* \cup Red(\mathcal{P})$, where $Red(\mathcal{P})$ denotes the set of red points in $\mathcal{P}$.

7. For every red edge still in the triangulation, calculate the length of its dual in the voronoi graph. If no red edges exists, terminate. If the length is sufficiently small, use random search to find an optimal solution. Otherwise, restart from 2.

4.2 Finding circles corresponding to unsatisfied edges

Following the terminology used by the CGAL-project, a **line** is a line unbounded on both sides, whereas an line that is bounded on one side is called a **ray** and when bounded on both sides it is called a **segment**. The approach for choosing circles depends on which type of voronoi edge is to be blocked. Given a voronoi edge, corresponding to the neighbouring points p_1 and p_2, there are the following cases:

1. For a **line**, the smallest possible circle is chosen. This is the circle with its centre in the middle of p_1 and p_2, and a radius such that p_1 and p_2 are on its perimeter. A line only occurs in the voronoi diagram in the trivial case where all points lie in a straight line.

2. Note that a **ray** in the voronoi diagram is always associated with two points on the convex hull, for which infinitely large circles exist with the points p_1 and p_2 on its perimeter. In this case, a circle is randomly chosen.

3. For a **segment**, the circle with p_1 and p_2 on its perimeter, and its centre on the midpoint of the segment is chosen.

4.3 Finding a point in the interior of a face

A relatively difficult task proved to be that of finding a point in the interior of an arbitrary face, given two point p_1 and p_2 known to be on the boundary of the face. The following algorithm was used.

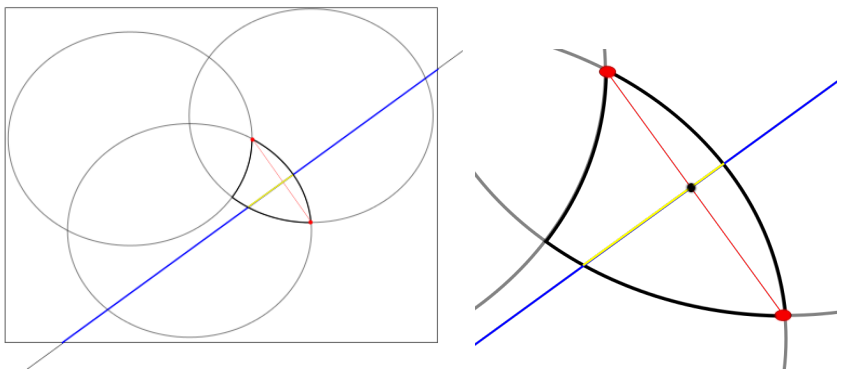

Figure 5: ARE WE USING THIS SECTION AT ALL?.

- Find the point p_m in the middle of p_1 and p_2. If this point is inside the face, chose this point. (WE DON'T ACTUALLY DO THIS YET)

- Find the line l going through p_m and is perpendicular to the edge between p_1 and p_2.
- Find the segment s that is the intersection of l and the smallest rectangle large enough to contain all circles of the arrangement. This step is needed because a CGAL function needs an object of type segment as input. We describe this step here for the sake of completeness.
- Find the segment that is the intersection of the segment s and the face itself.
- Any point except the endpoints of this segment are inside the face. The midpoint of the segment is chosen.

4.4 Finding a minimal subset of points

The problem of finding a minimal subset of a set of points $\mathcal{P}^*$, so that at least one point in $\mathcal{P}$ is in the interior of each circle in $\mathcal{C}$, is exactly that of the set covering problem. The problem is easily stated in terms of an integer programming problem, and can thus be solved relatively efficiently with an integer programming solver, such as Gurobi.

4.4.1 Pseudo code

For every point $\rho_i \in \mathcal{P}^*$, let $x(\rho_i)$ be a boolean, set to 1 if the point ρ_i appears in a solution and 0 otherwise. Also, for every circle $\varsigma_j \in \mathcal{C}$, let $\varsigma_j(\rho_i)$ be 1 if point ρ_i lies in the interior of c_j and 0 otherwise.

```
1  Minimize: Σ_{ρ∈P*} x(ρ_i)
2  Constraint: Σ_{ρ∈P*} ς_j(ρ_i) ≥ 1        for each ς ∈ C
```

4.5 Refining a near-optimum through random search

Random search is a direct search method that does not require the derivative of the function to be minimized. It is algorithmically simple, and in general terms work as follows:

To minimize a function $f : \mathbb{R}^n \to \mathbb{R}$, start with a (potentially random) solution $x : \mathbb{R}^n$.

```
1  while (f(x) > 0
2    Generate a new candidate solution y, in the neighbourhood of x
3    if f(y) < f(x)
4      replace the current solution x with y.
```

There are several candidates for choosing the objective function *f*, for example the number of red delaunay circles or red voronoi edges. For this problem, we have chosen the sum the squared lengths of all red voronoi edges. (SHOULD I EXPLAIN "WHY" HERE, OR LATER?)

5 Results

- Some problems, and the resulting solution.
- How well does it perform?

5.1 Graphical interface

In order to easily specify new problems, and to visualize the process of solving the problems, a minimalistic graphical interface is used. It has methods for adding, moving and removing points. It automatically updates and draws the voronoi diagram corresponding to the set of points, and also a set of delaunay circles corresponding to the red edges in the voronoi diagram.

Additionally, methods for saving and loading configurations were added to simplify testing and debugging.

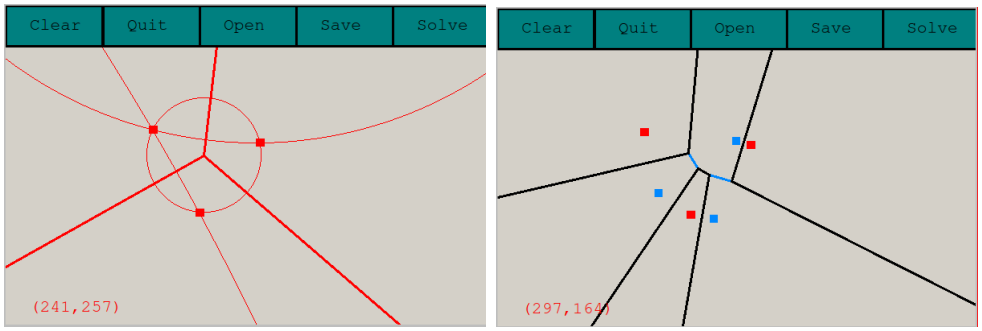

Figure 6: An example of a problem configuration (left) and a found solution to the same problem (right).

6 Discussion

- Point in face - even if we don't include the above section, we must note that by not using the point-in-face algorithm, there is a risk that the problem is unsolvable.
- Random search has a few things to note. Local minima is always a problem. Then two things that can be optimized, but we don't have time; first the threshhold, where we decide to start doing an RA, second the ammount of randomization used. Tabu search would probably work awesome.
- Complexity of our algorithm?

7 Tools

In this project, a set of tools and libraries were used. Most notably, the Computational Geometry Algorithms Library (CGAL) provides a vast number of efficient algorithms for

different areas within computational geometry. In this project, we have had extensive use of the methods for calculating voronoi diagrams, triangulations and arrangements.

The Gurobi optimzer was used for solving the integer programming problem. It provides an easy-to-use C++-API.

We used git for revision control and to simplify our collaboration. The source code for the project is available at `https://github.com/martin-mfg/voronoi-thesis-tester`

8 references

1. Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin

   CGAL manual chapter 20, 2D Arrangements

   http://www.cgal.org/Manual/3.3/doc_html/cgal_manual/Arrangement_2/Chapter_main.html

2. O. Aichholzer, R. Fabila-Monroy, T. Hackl, M. van Kreveld, A. Pilz, P. Ramos, und B. Vogtenhuber

   Blocking Delaunay Triangulations.

   Proc. Canadian Conference on Computational Geometry, CCCG 2010, Winnipeg, August 9/11, 2010.

3. M. de Berg, D. Gerrits, A. Khosravi, I. Rutter, C. Tsirogiannis and A. Wolff.

   How Alexander the Great Brought the Greeks Together while Inflicting Minimal Damage to the Barbarians.

   Proc. 26th European Workshop on Computational Geometry, pages 73-76, 2010.

4. Richard M. Karp Reducibility Among Combinatorial Problems. In: R. E. Miller, J. W. Thatcher (Hrsg.): Complexity of Computer Computations. Plenum Press, New York 1972, S. 85-103.

8.1 images

- Delaunay circumcircles, GNU Free Documenation Licence, Nü es

  http://commons.wikimedia.org/wiki/File:Delaunay_circumcircles.png

- Delaunay triangulation 1-3, public domain, user Capheiden

  http://upload.wikimedia.org/wikipedia/de/1/17/Voronoi-Delaunay.svg

  http://upload.wikimedia.org/wikipedia/de/4/48/Voronoi-Diagramm.svg

  http://upload.wikimedia.org/wikipedia/commons/1/1f/Delaunay-Triangulation.svg