Algorithms for Incremental Voronoi Diagrams

Zolotov Boris

June 3, 2022

1 Introduction

Computational geometry studies computations on discrete geometric objects such as arrangements, diagrams, foldings and drawings. In particular one of its main interests is to design algorithms to construct such objects and data structures that store them efficiently.

The most crucial and defining concept applied in all those constructions is *distance:* whatever is studied, there is always some underlying metric that describes the object in question and defines its properties. It can be just the Euclidean metric, or some polyhedral metric, or any other metric that corresponds to a surface or a folding of a polygon.

We will be considering three different types of distance metrics. The Euclidean metric in $\mathbb{R}^2$ gives rise to the ordinary Voronoi diagrams, and we are interested in algorithms that allow for fast updates of a Voronoi diagram that is stored explicitly. We also consider polyhedral metrics of the spaces induced by gluing n congruent regular polygons edge to edge, and we propose a way to classify all such spaces in time polynomial in n. Finally, PCB-s are of our interest, which are unions of disjoint planes each equipped with a certain metric.

2 Literature Review

Even though there are many well-known algorithms that construct the Voronoi diagram for a given family of sites, the problem of making a Voronoi diagram dynamic, i. e. implementing changes to it when a new point site is inserted, has only drawn attention in the implicit case: maintaining implicit Voronoi diagram can be done in very little time. Maintaining a Voronoi diagram explicitly has not yet been considered.

The worst one can expect is that when a new site is inserted, the number of updates in a Voronoi diagram (i. e. vertices and edges that change) is linear. This can happen for every insertion if we consider an embedded diagram and store the coordinates of all the vertices.

The situation improves if we consider *the graph* of the Voronoi diagram that is subject to combinatorial changes: no coordinates matter, it is just deletion or addition of edges that is performed. In Allen et al. it was proved that when a new site is inserted to a Voronoi diagram, only $O(\sqrt{n})$ combinatorial changes happen to the graph of the diagram. This opened a possibility to find a sublinear algorithm that finds and implements combinatorial changes in an explicitly stored graph of the Voronoi diagram.

There is a known algorithm that is sublinear in n (which is the number of sites) that does exactly that. The algorithm has running time of $\tilde{O}(n^{3/4})$, which still does not achieve the $O(\sqrt{n})$ bound on the number of changes, that is why the quest for a better algorithm is still open.

Another type of metrics and distances that can be often seen in computational geometry is polyhedral metrics and geodesic distances. A question concerning them that has been standing for a long time already is the Alexandrov's problem of finding a convex polyhedron corresponding to a given polyhedral metric. There is almost no hope of solving this problem exactly for an arbitrary polyhedral metric, that is why several special cases are considered in literature.

2.1 Voronoi diagrams

We begin with standard definitions related to Voronoi diagrams and their basic properties. A detailed treatment of Voronoi diagrams and their applications can be found in. Let $S := \{s_1, s_2, \ldots, s_N\}$ be a set of N distinct points in $\mathbb{R}^2$; these points are called *sites*. Let $\text{dist}(\cdot, \cdot)$ denote the Euclidean distance between two points in $\mathbb{R}^2$. We assume that the sites in S are in *general position*, that is, no four sites lie on a common circle.

Definition 1. The *Voronoi diagram* of S is the subdivision of $\mathbb{R}^2$ into N cells, called *Voronoi cells*, one cell for each site in S, such that a point q lies in the Voronoi cell of a site s_i if and only if $\text{dist}(q, s_i) < \text{dist}(q, s_j)$ for each $s_j \in S$ with $j \neq i$.

Let f_i denote the Voronoi cell of a site s_i. Edges of the Voronoi diagram, called *Voronoi edges,* are portions of bisectors between two sites which are the common boundary of the corresponding Voronoi cells. *Voronoi vertices* are points where at least three Voronoi cells meet. The *Voronoi circle* of a Voronoi vertex v is the circle passing through the sites whose cells are incident to v. Vertex v is the center of its Voronoi circle.

Since the sites are in the general position, each Voronoi vertex has degree three. Each Voronoi edge is either a segment or a ray and the graph of the Voronoi diagram formed by its edges and vertices is planar and connected.

2.1.1 Combinatorial Changes to the Voronoi Diagram and the Flarb Operation

We now overview the definitions and results from Allen et al. that we need to present our approach. In order to prove the $\Theta(N^{\frac{1}{2}})$ bound on the number of combinatorial changes caused by insertion of a site, a graph operation called *flarb* is introduced.

Let G be a planar 3-regular graph embedded in $\mathbb{R}^2$ without edge crossings (edges are not necessarily straight-line). Let $\mathcal{C}$ be a simple closed Jordan curve in $\mathbb{R}^2$.

Definition 2. Curve $\mathcal{C}$ is called *flarbable* for G if:

- the graph induced by vertices inside the interior of $\mathcal{C}$ is connected,
- $\mathcal{C}$ intersects each edge of G either at a single point or not at all,
- $\mathcal{C}$ passes through no vertex of G, and
- the intersection of $\mathcal{C}$ with each face of G is path-connected.

Given a graph G and a curve $\mathcal{C}$ flarbable for G, the *flarb* operation is, informally, removing part of G that is inside $\mathcal{C}$ and replacing it with $\mathcal{C}$. Formally, the flarb operation for G and $\mathcal{C}$ is defined as follows:

- For each edge $e_i \in G$ that intersects $\mathcal{C}$ let u_i be its vertex lying inside $\mathcal{C}$ and v_i its vertex outside $\mathcal{C}$. Create a new vertex $w_i = \mathcal{C} \cap e_i$ and connect it to v_i along e_i.
- Connect consecutive vertices w_i along $\mathcal{C}$.
- Delete all the vertices and edges inside $\mathcal{C}$.

Let $\mathcal{G}(G, \mathcal{C})$ denote the graph obtained by applying the flarb operation to graph G and curve $\mathcal{C}$.

Lemma 1. *The following holds for graph $\mathcal{G}(G, \mathcal{C})$:*

(a) $\mathcal{G}(G, \mathcal{C})$ has at most two more vertices than G does;
(b) $\mathcal{G}(G, \mathcal{C})$ is a 3-regular planar graph;
(c) $\mathcal{G}(G, \mathcal{C})$ has at most one more face than G does.

Proof. Items (a) and (b) have already been proved in literature. To prove (c) note that there is one new face bounded by the cycle added along $\mathcal{C}$ while performing the flarb. All the other faces of G are either deleted, left intact, or cropped by $\mathcal{C}$; these operations obviously do not increase the number of faces. □

Theorem 2. *Let G be a graph of the Voronoi diagram of a set of $N-1$ sites $s_1 \dots s_{N-1}$. For any new site s_N there exists a flarbable curve $\mathcal{C}$ such that the graph of the Voronoi diagram of sites $s_1 \dots s_N$ is $\mathcal{G}(G, \mathcal{C})$.*

2.1.2 Cost of the Flarb

We want to analyze the number of structural changes that a graph undergoes when we apply the flarb operation to it. There are two basic combinatorial operations on graphs:

- *Link* is the addition of an edge between two non-adjacent vertices.
- *Cut* is the removal of an existing edge.

Other combinatorial operations, for example insertion of vertex of degree 2, are assumed to have no cost.

Definition 3. $\text{cost}(G, \mathcal{C})$ is the minimum number of links and cuts needed to transform G into $\mathcal{G}(G, \mathcal{C})$.

Note that sometimes there are less combinatorial changes needed than the number of edges intersected by $\mathcal{C}$. Consider edges e_1, e_2 of G crossed consecutively by $\mathcal{C}$ and edge n adjacent to them that reappears in $\mathcal{G}(G, \mathcal{C})$ as a part n^* of $\mathcal{C}$. Then n^* can be obtained without any links or cuts by lifting n along e_1 and e_2 until it coincides with n^* or (which is the same) shrinking e_1 and e_2 until their endpoints coincide with their intersections with $\mathcal{C}$. We will call it *preserving operation.*

Theorem 3. *For a flarbable curve $\mathcal{C}$, it holds that*

$$\text{cost}(G, \mathcal{C}) \leq 12|\mathcal{S}(G, \mathcal{C})| + 3|\mathcal{B}(G, \mathcal{C})| + O(1).$$

Where

- *$|\mathcal{B}(G, \mathcal{C})|$ is the number of faces of G wholly contained inside $\mathcal{C}$.*
- *$|\mathcal{S}(G, \mathcal{C})|$ is the number of* shrinking *faces — i.e., the faces whose number of edges decreases when flarb operation is applied.*

The following upper bound can be used to evaluate the number of combinatorial changes needed to update the graph of a Voronoi diagram when a new site is inserted.

Theorem 4. *Consider one insertion of a new site to a Voronoi diagram V.*

- *The number of cells of V undergoing combinatorial changes is $O(N^{\frac{1}{2}})$ amortized in a sequence of insertions;*
- *There are a constant number of combinatorial changes per cell;*
- *The cells of V with combinatorial changes form a connected region.*

By *a change in cell* we always mean a combinatorial change, that is a *link* or a *cut.*

2.2 The coin problem

A detailed introduction into amortized analysis, including the definition of potential function and examples of estimates of it is given in [**?**]. Here we focus on one classical result concerning amortized analysis. It is rather folklore and is given as an exercise in several university courses; we show it here to establish its formal proof, because it is crucial for understanding of what we prove further.

Given n coins stored in n piles (some piles may be empty). A series of operations is executed on these piles. One operation consists of selecting a pile and distributing all its coins equally among other piles, one coin per pile. Piles that receive a coin and piles that do not can be chosen freely. An example of such operations can be seen in Figure **??**. We assume the number of piles is equal to the number of coins so that these operations can be executed on any configuration of coins.

Clearly during one operation at most n coins are distributed. However, during k consecutive operations the average number of coins disrtibuted per operation is $o(n)$. Informally, each operation makes the heights of the piles more uniform, making it impossible to have a large pile to distribute after each of the operations. This observation is formalized by the following theorem:

Theorem 5. *For $k \geq \sqrt{n}$, if k consecutive operations are executed, then the average number of coins distributed per operation is at most $3\sqrt{n}$.*

Proof. Denote piles by $p_1, \ldots, p_n$. We introduce a potential function that describes the configuration of piles and helps estimate the number of coins distributed during an operation executed on a pile:

$$\Phi = \sum_{j=1}^{n} \min\left\{\text{size}(p_j), \sqrt{n}\right\}.$$

Note that Φ is always at most n, since n is the sum of actual sizes of all the piles. Denote by T_i the number of coins distributed during the i-th operation, and by Φ_i the value of the potential after the i-th operation. Therefore, Φ_0 and Φ_k are the values of the potential before and after the execution of all the operations respectively.

We will now estimate $T_i + \Phi_{i-1} - \Phi_i$. To do so, note that the pile that is being distributed (without loss of generality, p_1) decreases in size to zero, and several other piles increase in size by 1. Distributing p_1 makes Φ decrease by at most $\sqrt{n}$, regardless of size(p_1) being greater or less than $\sqrt{n}$, by definition of Φ. Figure **??** illustrates it: a change in size of a pile does not affect the potential if the size of the pile is greater than $\sqrt{n}$.

Several other piles receive one coin, the number of such piles is equal to T_i. However, some of these piles may not contribute to the change in the potential, again because their size is greater than $\sqrt{n}$. Note that the number of such large piles is at most $\sqrt{n}$. Thus, when the coins are being put into piles, there is at least $T_i - \sqrt{n}$ increase in potential. Combining these observations,

$$T_i + \Phi_{i-1} - \Phi_i \;\leq\; T_i + \sqrt{n} - \left(T_i - \sqrt{n}\right) \;=\; 2\sqrt{n}.$$

We can now sum up $T_i + \Phi_{i-1} - \Phi_i$ for each of the consecutive operations. Note that $\Phi_0 - \Phi_k$ is between $-n$ and n.

$$\sum_{i=1}^{k} \left(T_i + \Phi_{i-1} - \Phi_i\right) \;=\; \sum_{i=1}^{k} T_i + \Phi_0 - \Phi_k \;\leq\; 2\sqrt{n} \cdot k + n.$$

This means average T_i per operation is at most $2\sqrt{n} + \frac{n}{k} = 3\sqrt{n}$. □

Informally, this theorem means that if during an operation one pile loses many coins, and many distinct piles get at most one coin each, there can only be so many of such operations. We will rely on this observation further on.

2.3 Gluings of squares

Given a collection of 2D polygons, a *gluing* describes a closed surface by specifying how to glue each edge of these polygons onto another edge. We consider only proper gluings, where only segments of equal lengths can be glued together. The following theorem is crucial in that it establishes the connection between gluings and convex polyhedra:

Theorem 6. *If a gluing is homeomorphic to a sphere and the sum of angles at each of its vertices is at most* 360° *then there is a single convex polyhedron* P *that can be glued from this net.*

Note that the polygons of the gluing may be folded in order to glue the polyhedron.

There is no known exact algorithm for reconstructing the 3D polyhedron. It is known that the problem of reconstructing the polyhedron can be reduced to a system of partial differential equations. Still this method does not produce the exact answer, and there is no known algorithm for it that works faster than pseudopolynomial in n, n being the number of vertices. Sometimes the coordinates of the polyhedron, even if its general shape is known, can not be expressed as closed formulas.

Enumerating all possible valid gluings is also not an easy task. Demaine et al. showed that for any even n there is a polygon with n vertices that has $2^{\Omega(n)}$ gluings: that is a star with two additional vertices on midpoints of edges half perimeter from one another. So it is also important to estimate the number of gluings of the collection of polygons under consideration.

Complete enumerations of gluings and the resulting polyhedra are only known for very specific cases such as the Latin cross, a single regular convex polygon, and a collection of regular pentagons glued edge-to-edge.

The case when the polygons to be glued together are all congruent regular k-gons, and the gluing is edge-to-edge, was studied recently for $k \geq 6$. Our aim is to study the case of $k = 4$: namely, to *enumerate* all valid gluings of squares and *classify* them up to isomorphism.

2.3.1 Chen—Han algorithm for gluings of squares

It is shown that polyhedra are isomorphic if the lengths of shortest geodesic paths between their vertices of nonzero curvature coincide. Thus, the problem of finding out if two gluings are isomorphic can be reduced to calculating the pairwise geodesic distances between vertices of a gluing. Algorithm we are using for this is the Chen—Han algorithm.

The idea of the algorithm is to project a cone of all possible paths from the source onto the polygons of the gluing. For n faces, this algorithm runs in $O(n^2)$ time. To apply it for arbitrary edge-to-edge gluings of squares, it has to be proven that the running time is preserved. We make use of a Lemma that was proved in the Bachelor's thesis.

Lemma 7. *If* T *is a square of the gluing and* π *is a geodesic shortest path between two vertices of the gluing then the intersection between* π *and* T *is of at most 5 segments.*

This lemma implies the following theorem.

Theorem 8. *The isomorphism between two edge-to-edge gluings of at most n squares can be tested in $O(n^2)$ time.*

2.4 PCB routing

A printed circuit board that consists of several layers can be thought of as a disjoint union of several metric spaces, distance between points of distinct spaces being defined additionally. The main problem is as follows. Given several layers of a printed circuit board *(PCB)*, route all the wires so that certain design rules are respected and wires have small length, small number of bends, and small number of transitions between layers. There is of course a general problem of connecting points $a_1, \ldots, a_n$ with $b_1, \ldots, b_n$ pairwisely and optimally, but it is too difficult to solve practically, and probably is **NP**-hard. Most routing algorithms these days rely on empirical data only.

3 Purpose of the study

The purpose of this study is to find answers to the following open problems concerning metric spaces:

1) Find out if there exist algorithms for explicit incremental Voronoi diagrams whose running time is between $O(n^{3/4})$ and $O(n^{1/2})$.

2) Find a suitable variant of Voronoi diagrams for multilayer PCB boards and study their properties.

3) Find an appropriate data structure for maintaining and answering queries about divisions of space by arbitrary surfaces.

4) Estimate the number of convex polyhedra that can be glued using at most n congruent regular triangles, hexagons, or quadrilaterals.

5) Find a polynomial time approximation algorithm for the PCB routing problem.

6) Find weights for the parameters such that the corresponding multicriterion optimisation problem outputs results that are applicable in practice.

4 Methodology

4.1 Voronoi diagrams

One can try to employ the well-known «divide and conquer» strategy to devise an algorithm for the incremental diagram: divide all the sites into two halves by a line and update only the half of the diagram that receives the new site. It can be shown that adding a line as a site to the diagram still allows for a $O(\sqrt{n})$ upper bound on the number of combinatorial changes per insertion. This means that, theoretically, «divide and conquer» is a desirable approach.

One possible direction of research is to look into generalisation of algorithms and techniques for both implicit and explicit diagrams to various non-Euclidean metrics, for example those implied by the PCB problem. There are some hints that the algorithms will still work, but there can be certain underwater rocks in the process of translating the algorithm from one metric to another.

Another generalization to look at is considering not Voronoi cells, but any regions of space whose borders are arbitrary surfaces that intersect rarely enough, for example instance surfaces. One can find out what properties hold for such setups and what algorithms and data structures can be developed to work with them.

4.2 Nets

A net consisting of several regular polygons can be associated with a set of polygons drawn on a corresponding grid, edges of those polygons divided into pairs. The set of polygons should satisfy certain conditions: each of them being convex, the sum of angles of all the polygons at any vertex not exceeding 360°, the edges in a pair having equal projections on coordinate axes. Then the estimate of the number of valid nets really boils down to the estimate of the number of drawings, which is a simple combinatorial problem.

4.3 PCB-s

The problem of finding a shortest path between two points in this setting can be thought of as multicriterion optimisation problem: we are trying to optimize not only overall length of the path, but also the number of turns and the number of transitions between layers (also called *vias,* those are especially expensive to make).

There are certain limitations that apply to the path that we can construct. First of all, only turns of 45° (that leave an internal angle of 135°) are allowed. Second, there is minimal distance required between two adjacent wires. To some extent, we can think that there is a grid that the wires align to, but we can not use the size of this grid in the asymptotic estimations.

There are of course obstacles possible to the path we are constructing. To start with, one can assume that they admit some simple shape (convex, polygonal or round). To do proper preprocessing, however, one needs to count in dynamic obstacles posed by previously routed wires.

5 Contribution

If those problems are solved, the corresponding results will be the cornerstones of the corresponding fields, since they will be the answers for well posed questions that are important and understandable. They make us closer to finding constructive solutions to the Alexandrov's problem and help plan routes and process geodesic data faster and better.

The problem of PCB routing is tightly connected with several industrial projects that need feasible routing algorithms to work with.