Recent research achievements

Boris Zolotov, МКН СПбГУ

Advanced Mathematics

Friday, December 20

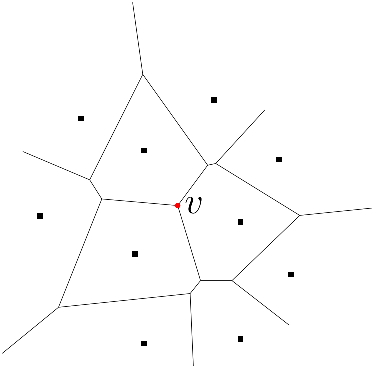
Part 1

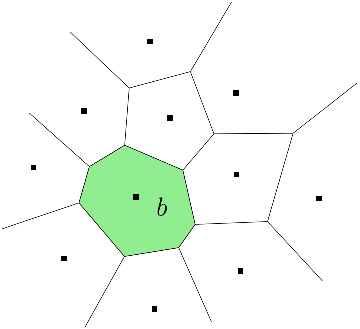
Sublinear Explicit Incremental Voronoi Diagrams

Voronoi Diagram: basics

Voronoi diagram of $S = \{s_1, \ldots, s_N\}$: subdivision of the Euclidean plane, for each q inside the *cell* of s_i

$$\mathrm{dist}(q, s_i) < \mathrm{dist}(q, s_j), \quad j \neq i.$$

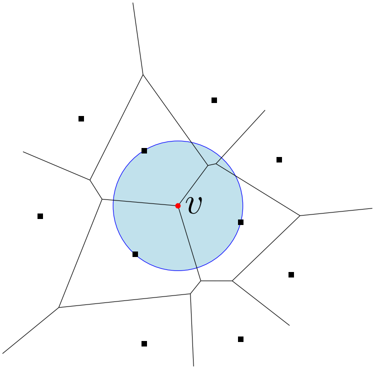


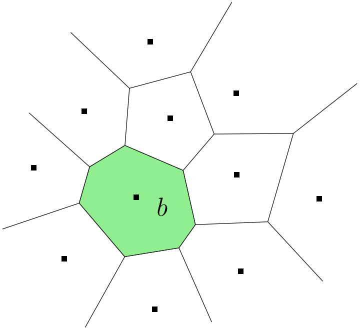


Voronoi Diagram: basics

Voronoi diagram of $S = \{s_1, \ldots, s_N\}$: subdivision of the Euclidean plane, for each q inside the *cell* of s_i

$$\mathrm{dist}(q, s_i) < \mathrm{dist}(q, s_j), \quad j \neq i.$$

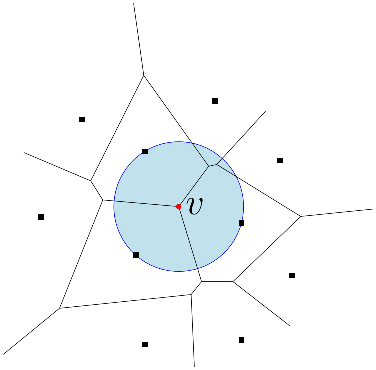


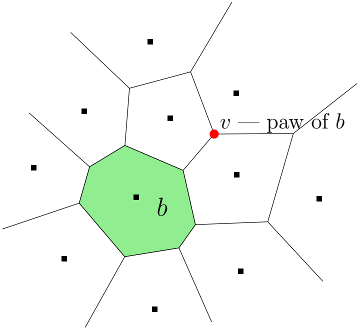


Voronoi Diagram: basics

Voronoi diagram of $S = \{s_1, \ldots, s_N\}$: subdivision of the Euclidean plane, for each q inside the *cell* of s_i

$$\mathrm{dist}(q, s_i) < \mathrm{dist}(q, s_j), \quad j \neq i.$$





Dynamic VD: setting

Incremental Voronoi Diagram

Maintain initially empty Voronoi diagram under insertion of new sites

There is a linear-time solution, no faster solutions possible: there may be $O(N)$ changes per insertion.

Dynamic VD: setting

Incremental Voronoi Diagram

Maintain initially empty Voronoi diagram under insertion of new sites

There is a linear-time solution, no faster solutions possible: there may be $O(N)$ changes per insertion.

Incremental Combinatorial Voronoi Diagram

Maintain the graph of initially empty Voronoi diagram under insertion of new sites

There is a naive linear-time algorithm. Can we find a faster solution?

Dynamic VD: combinatorial changes

Theorem (Allen et al. 2017)

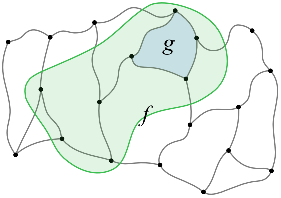
The number of cells with combinatorial changes is $O(N^{\frac{1}{2}})$ *amortized, there are a constant number of combinatorial changes per cell, cells with changes are connected.*

Dynamic VD: combinatorial changes

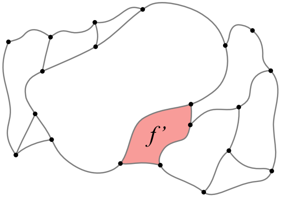
Theorem (Allen et al. 2017)

The number of cells with combinatorial changes is $O(N^{\frac{1}{2}})$ amortized, there are a constant number of combinatorial changes per cell, cells with changes are connected.

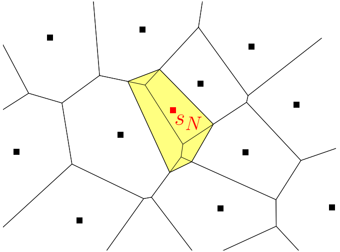
When we are inserting new site s_N, the graph of the VD undergoes operation called *flarb*.



(a) Graph before flarb



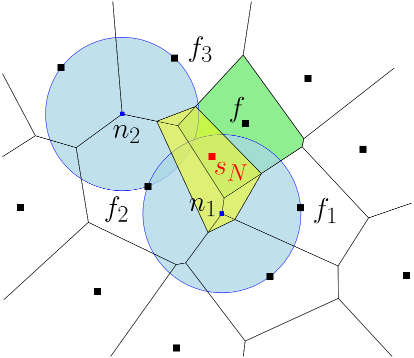
(b) Graph after flarb



(c) Flarb on a VD

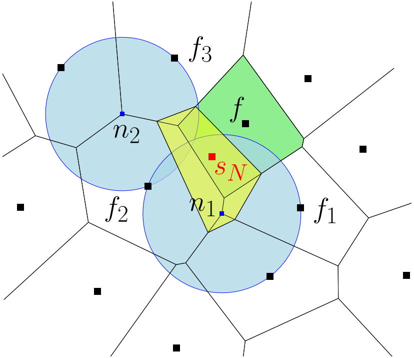
Identifying changes

Theorem: Let g be a cell adjacent to f. Cell g needs to undergo changes $\iff$ circle of either of its vertices that are paws of f encloses s_N. Allen et al. 2017



Identifying changes

Theorem: Let g be a cell adjacent to f. Cell g needs to undergo changes $\Longleftrightarrow$ circle of either of its vertices that are paws of f encloses s_N. Allen et al. 2017



Dynamic circle reporting structure (Chan 2010; Allen et al. 2017)

Returns all k circles enclosing given point in $\tilde{O}(k)$. Addition and deletion of a circle in $\tilde{O}(1)$.

Description of data structure

Definition

A cell is *big* if it has size at least $N^{\frac{1}{4}}$. Otherwise it is *small*.

Description of data structure

Definition

A cell is *big* if it has size at least $N^{\frac{1}{4}}$. Otherwise it is *small*.

(1) Graph of VD in a form of adjacency list,

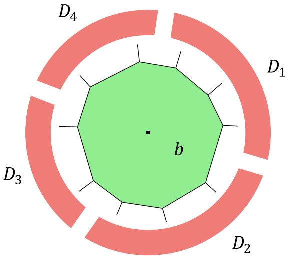
(2) *Dynamic nearest neighbor* structure for sites,

(3) Graph of big cells stored as adjacency list,

(4) For each big cell:

– linked list of DCRs,

– binary search tree of vertices in circular order.



Handling insertion

Initialize queue; look at cells one-by-one, on each step add to it unprocessed cells with changes, then implement changes.

Handling insertion

Initialize queue; look at cells one-by-one, on each step add to it unprocessed cells with changes, then implement changes.

- Small cell: look at each paw to find neighboring cells needing changes, add them to the queue.
- Big cell: ask DCRs to return Voronoi circles that enclose s_N, add corresponding cells to the queue.

Handling insertion

Initialize queue; look at cells one-by-one, on each step add to it unprocessed cells with changes, then implement changes.

- Small cell: look at each paw to find neighboring cells needing changes, add them to the queue.
- Big cell: ask DCRs to return Voronoi circles that enclose s_N, add corresponding cells to the queue.

$$\text{Time complexity:} \quad \tilde{O}\left(sN^{\frac{1}{4}} + \sum_{i=1}^{|B|}\left(\left\lceil \frac{|b_i|}{N^{\frac{1}{4}}} \right\rceil + \ell_i\right) + N^{\frac{3}{4}} + sN^{\frac{1}{4}}\right).$$

Handling insertion

Initialize queue; look at cells one-by-one, on each step add to it unprocessed cells with changes, then implement changes.

- Small cell: look at each paw to find neighboring cells needing changes, add them to the queue.
- Big cell: ask DCRs to return Voronoi circles that enclose s_N, add corresponding cells to the queue.

$$\text{Time complexity:} \quad \tilde{O}\left(sN^{\frac{1}{4}} + \sum_{i=1}^{|B|}\left(\left\lceil \frac{|b_i|}{N^{\frac{1}{4}}} \right\rceil + \ell_i\right) + N^{\frac{3}{4}} + sN^{\frac{1}{4}}\right).$$

This is $\tilde{O}(N^{\frac{3}{4}})$ amortized.

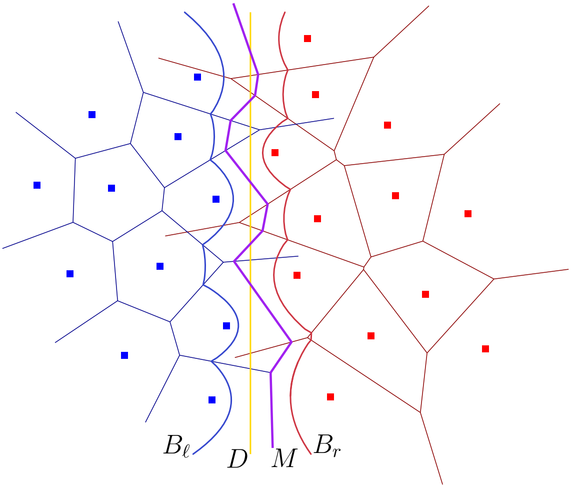
Ongoing research:

Õptimal-Time Incremental Voronoi Diagrams

Divide-and-conquer

Time complexity $\tilde{O}(N^{\frac{3}{4}})$ is not bad, but the lower bound is $N^{\frac{1}{2}}$. Can we meet it?

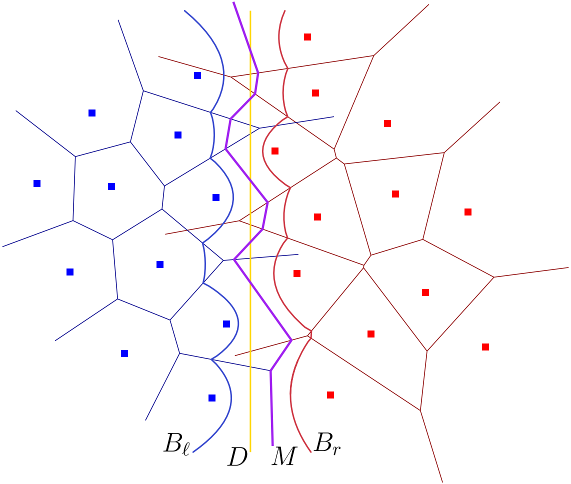
The idea is to use divide-and-conquer approach. It can allow us to omit doing unnecessary work.



Divide-and-conquer

Time complexity $\tilde{O}(N^{\frac{3}{4}})$ is not bad, but the lower bound is $N^{\frac{1}{2}}$. Can we meet it?

The idea is to use divide-and-conquer approach. It can allow us to omit doing unnecessary work.



(1) Dividing line,

(2) Beach lines,

(3) Blue and red forests,

(4) Merge curve.

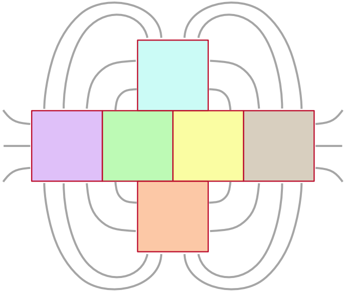
Previous research:

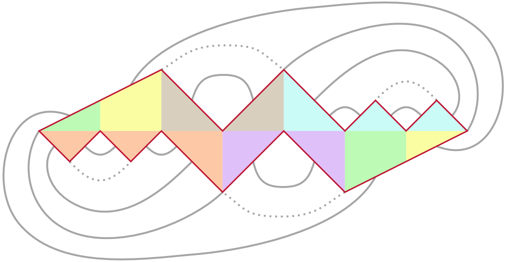
Polyhedra glued from regular pentagons

Gluings

Definition (Alexandrov 1950)

A *gluing* is a set of polygons equipped with a number of rules describing the way edges of these polygons must be glued to each other.

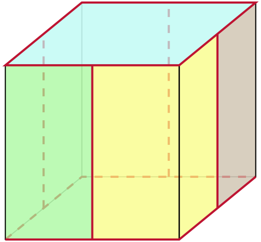


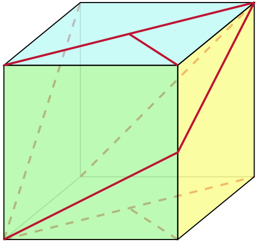


Alexandrov's Uniqueness Theorem

Theorem (Alexandrov 1950)

If a gluing is homeomorphic to a sphere and the sum of angles at each of its vertices is $\leq 360^\circ$*, there* $\exists\,!$ *convex polyhedron that corresponds to this gluing.*





Polyhedron Reconstruction

The proof of Alexandrov's theorem is non-constructive.

The algorithmic question remains open: how to find the polyhedron corresponding to a net?

Polyhedron Reconstruction

The proof of Alexandrov's theorem is non-constructive.

The algorithmic question remains open: how to find the polyhedron corresponding to a net?

Each of the following problems is still open:

Cauchy Rigidity Problem (Demaine and O'Rourke 2007, 23.22)

Given edge lengths of a triangulated convex polyhedron, output approximate coordinates of its vertices *in polynomial time.*

Polyhedron Reconstruction

The proof of Alexandrov's theorem is non-constructive.

The algorithmic question remains open: how to find the polyhedron corresponding to a net?

Each of the following problems is still open:

Cauchy Rigidity Problem (Demaine and O'Rourke 2007, 23.22)

Given edge lengths of a triangulated convex polyhedron, output approximate coordinates of its vertices *in polynomial time.*

Skeleton Reconstruction Problem

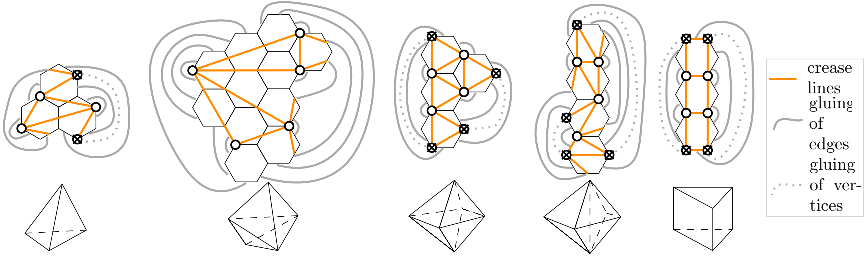
Given a net satisfying the conditions of Alexandrov's Theorem, find the skeleton of the convex polyhedron corresponding to it *in polynomial time.*

Edge-to-Edge Gluings of Regular Polygons

Maybe Alexandrov's Problem becomes easier, if we start building the gluings from similar simple blocks?

Definition

A gluing is *edge-to-edge* if every edge is glued to another entire edge.

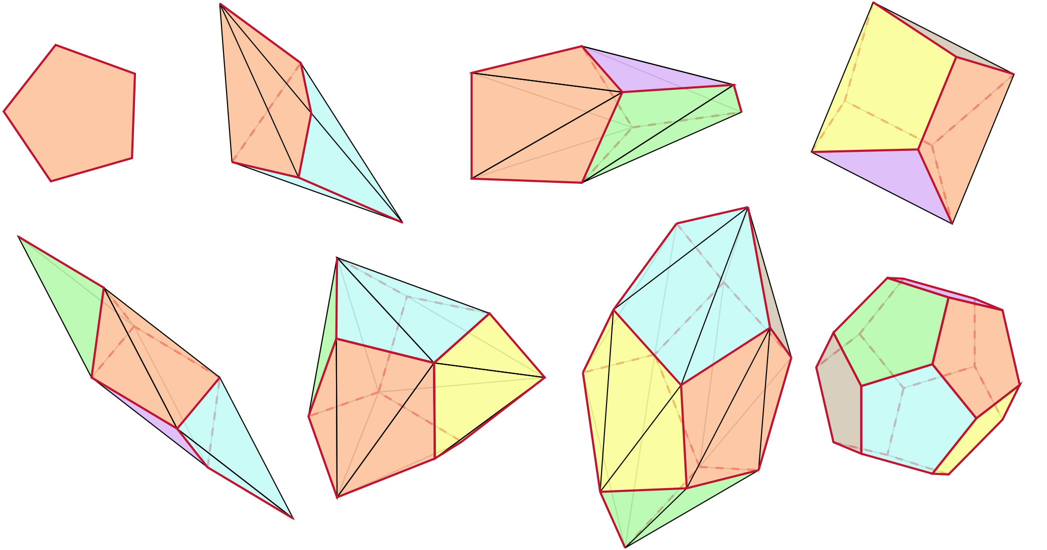


Khramtcova and Langerman 2017

Polyhedra glued from regular pentagons

One can enumerate all gluings of pentagons. Here are all the polyhedra that can be obtained:

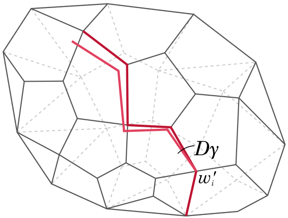
Arseneva, Langerman, **Zolotov,** 2019

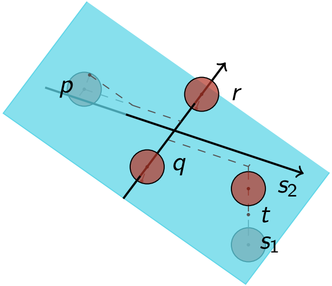


Determining the shape from the gluing

Theorem

Each vertex of $\mathcal{P}$ lies within an r-ball centered at the corresponding vertex of P, where $r = E^2 \cdot L \cdot 2\sin(\mathcal{D}\gamma/2) + E\mu$.



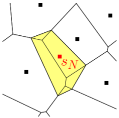


This theorem allows to create a procedure for checking whether there is a given edge in $\mathcal{P}$. The procedure was implemented.

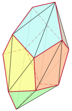
Conclusion

Results achieved:

(1) First sublinear algorithm for maintaining the graph of a Voronoi diagram

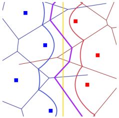


(2) Full list of the polyhedra that are made by edge-to-edge gluings of regular pentagons



Ongoing research:

(3) Algorithm for maintaining the graph of a Voronoi Diagram with the optimal time complexity



Thank you for your attention!

Allen, S. R., Barba, L., Iacono, J., and Langerman, S. (2017). Incremental voronoi diagrams. *Discrete & Computational Geometry*, 58(4):822–848.

Chan, T. M. (2010). A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15.

Demaine, E. and O'Rourke, J. (2007). *Geometric folding algorithms*. Cambridge University Press.

Khramtcova, E. and Langerman, S. (2017). Which convex polyhedra can be made by gluing regular hexagons? In *JCDCG*^{3}.