Recent research achievements

Boris Zolotov, МКН СПбГУ

Advanced Mathematics

Friday, December 20

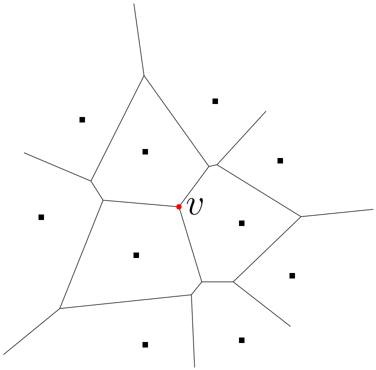
Part 1

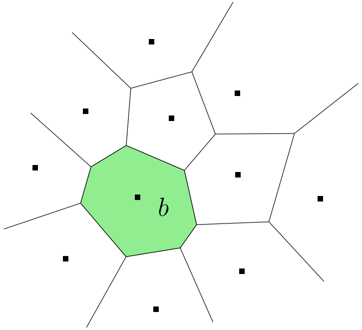
Sublinear Explicit Incremental Voronoi Diagrams

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

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

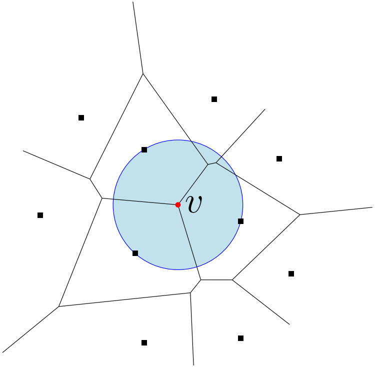


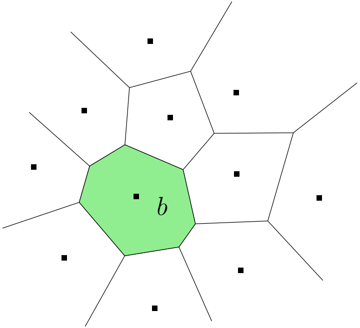


Voronoi Diagram: basics

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

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

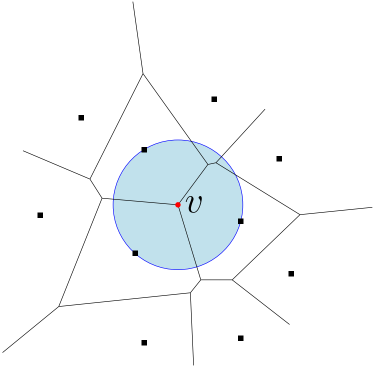


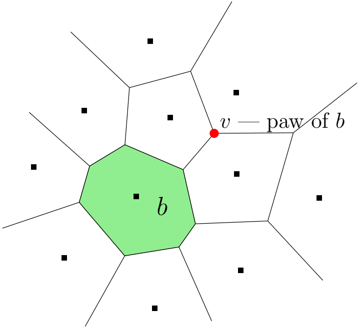


Voronoi Diagram: basics

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)

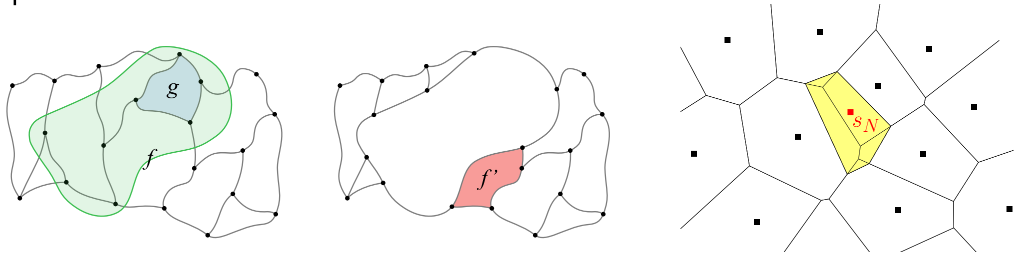
The number of cells with combinatorial changes is $O(N^{\frac{1}{2}})$ *amortized, there are a constant number of combinatorial changes per cell, cells with changes are connected.*

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.

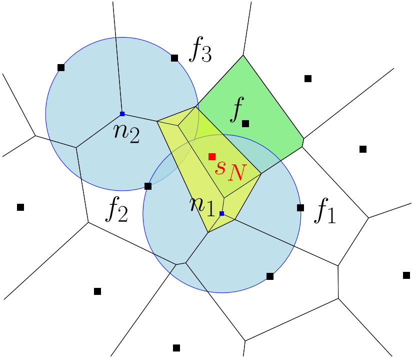
When we are inserting new site s_N, the graph of the VD undergoes operation called *flarb*.



(a) Graph before flarb (b) Graph after flarb (c) Flarb on a VD

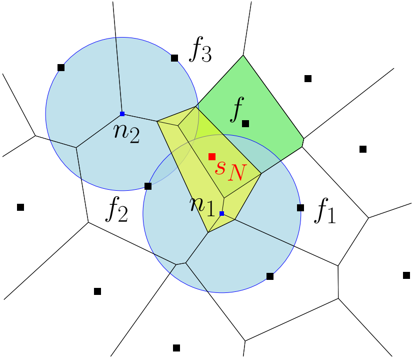
Identifying changes

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 $\iff$ 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)

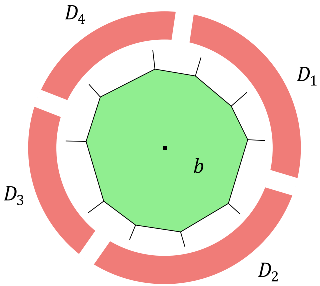
Returns all k circles enclosing given point in $\tilde{O}(k)$. Addition and deletion of a circle in $\tilde{O}(1)$.

Big cells and small cells

Definition

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

Small cells can be processed by brute force. For big cells we will need DCRs.



Description of data structure

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 containing all the big cells and the cell of site closest to s_N. Look at cells one-by-one, on each step add to the queue unprocessed cells with changes. After all cells needing changes are in the queue, implement changes in them.

- 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. When implementing changes use graph of big cells to identify edges that have to be deleted. Use BSTs to find the vertices that now belong to the cell of s_N.

Amortized expected runtime

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

- s is $O(N^{\frac{1}{2}})$ amortized — Allen et al. 2017,
- $\sum_{i=1}^{|B|} |b_i| \leq N$, $|B| \leq N^{\frac{3}{4}}$,
- $\sum_{i=1}^{|B|} \ell_i \leq sN^{\frac{1}{4}}$.

So this is simply

$$\tilde{O}\left(N^{\frac{3}{4}}\right) \quad \text{amortized.}$$

Ongoing research:

Õptimal-Time Incremental Voronoi Diagrams

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.