MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

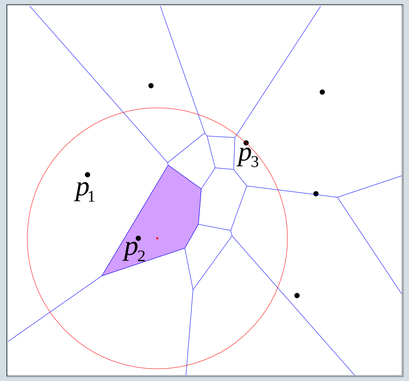

Casper Kejlberg-Rasmussen
Aarhus University

K-order Voronoi Diagrams in the I/O Model

Introduction

Problem [3]

If we are given a set of sites $S \subseteq R^d$ which are points, we want to compute the cell-complex where we in each cell have the same k nearest sites.



This can be illustrated in the plane, $d = 2$, for the seven points to the right, here $k = 2$. Notice that all interior points in the purple cell have the same points p_1, p_2, p_3 as the k nearest sites.

Below we see another example of a k'th order Voronoi diagram in the plane, here $k = 5$ and therefore all points in the same cell have the same k nearest sites. Here the sites could be restaurants, and we want to know the k closest restaurants at any given point in the plane.

Motivation

Voronoi diagrams is used in many different places in the natural sciences, among some of the countless examples here are some which also benefits from having the k'th order Voronoi diagram

Computer science [5]

- Computational geometry
- Associative file search

Natual sciences [5]

- Thiessen polygons in metrology (the post office problem)

See also [5] for many more examples.

Reduction to Arrangements of Hyperplanes

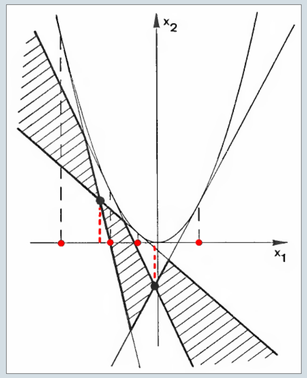
We can reduce the k'th order Voronoi diagram into the problem of computing arrangements of hyperplanes [3]. We do this by transforming every site $s = (\sigma_1, \ldots, \sigma_d)$ into a hyperplane according to the transformation

$$\varepsilon(s): x_{d+1} = 2\sigma_1 x_1 + \ldots + 2\sigma_d x_d - (\sigma_1^2 + \ldots + \sigma_d^2)$$

The hyperplane $\varepsilon(s)$ is the tangent plane to the point $U(s)$ on the paraboloid

$$U(s): x_{d+1} = x_1^2 + \ldots + x_d^2$$

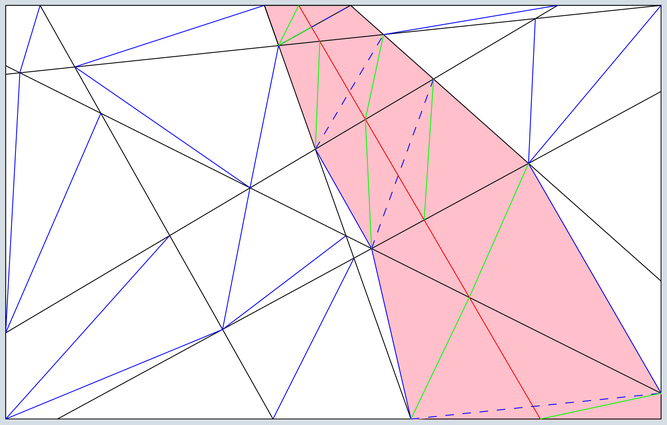
Now every point in the $(d+1)$-dimensional space which have k hyperplanes above it and $n-k$ hyperplanes below it, are said to belong to the k'th level. See the figure to the right for an example of how to construct the second order Voronoi diagram in one dimension. Here the red points are the sites, and the red lines are the Voronoi cells. The second level is show in shaded on the figure, notice that if we project these areas down onto the line we will get the second-order Voronoi diagram of the four points in one dimension.



Arrangements in the RAM model

For the I/O model there are currently no algorithms to compute $\leq k$-level, and in the RAM model the current best bounds can be seen below, the bound by Mulmuley is optimal. All running times are expected, as all algorithms uses a random incremental construction (RIC) to compute the $\leq k$–level.

Author	Dimension	RAM model	I/O model
Agarwal et al. [2]	$d = 2$	$O(nk + \alpha(n)\log(n))$	
Agarwal et al. [2]	$d = 3$	$O(nk^2 + n\log^3(n))$	
Mulmuley [2]*	$d \geq 4$	$O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$	My Research



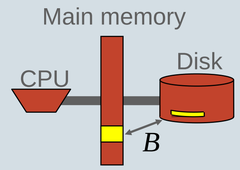
We keep the canonical triangulation (bottom, left corner triangulation) of each convex cell of the arrangement, see the above figure (blue lines), and we maintain:

- for each simplex of the triangulation we have a conflict list of all hyperplanes intersecting it,
- and likewise we have a conflict list of all simplices a hyperplane intersect.

Now to insert a new hyperplane we look at all simplices it intersects (pink areas). The convex cell the simplex lies in must be cut into two, and we have to retriangulate both parts, and remove the old simplices (dashed blue lines).

RIC in the I/O model/Future work

If we try to use the previous algorithm on large inputs it will be very ineffective as there will be a hard disk access each time a new hyper-planes is inserted. To get a efficient algorithm we need to consider another model than the RAM model, we should instead consider the I/O model.
In the I/O model we only count accesses to the hard disk, and we are here allowed to transfer B elements, between disk and RAM, at the same time-cost as to transfer one element, this is illustrated on the figure below.



The previous RAM algorithm used a framework for RIC [4], this framework can also be made to work in the I/O model as described in [1]. But here we need to handle a larger amount of hyperplanes at each step of the algorithm, that is we need graduations, where we divide the input hyperplanes into groups of hyperplanes. Now we have to insert all hyperplanes of one group at the same time. This gives more complex procedures for maintaining the convex cells.

References

[1] A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer and E. A. Ramos. *Randomized external-memory algorithms for line segment intersection and some other geometric problems,* 2001 Int. Jou. of CG & App.

[2] P. K. Agarwal, M. de Berg, J. Matousek and O. Schwarzkopf. *Constructing levels in arrangements and higher order Voronoi diagrams,* 1998 SIAM.

[3] H. Edelsbrunner. *Algorithms in combinatorial geometry,* 1987Springer.

[4] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars. *Computational geometry - algorithms and applications,* 1997 Springer.

[5] F. Aurenhammer. *Voronoi diagrams - A survey of a fundamental geometric data structure,* 1991 ACM.