Blockcourse on High-Dimensional Computational Geometry Spring 2013

Lecture 3: Voronoi Diagrams & VCP Preliminaries — July 24, 2013

Wolfgang Mulzer, Helmut Alt *Scribes: Simon Tippenhauer, Terese Haimberger*

1 Voronoi Diagrams

First of all we want to define a Voronoi diagram in arbitrary dimension. Therefore let $P = \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^d$ be a subset of n points in d-dimensional euclidean space. The points in P are also called sites of the Voronoi diagram. Then the Voronoi diagram $VD(P)$ of this point set is defined as follows:

Definition 1. *Let* $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^d$. *Then*

$$VD(P) = \{q \in \mathbb{R}^d \mid \exists p_i, p_j \in P : d(q, p_i) = d(q, p_j), i \neq j \text{ and } d(q, p) \geq d(q, p_i) \text{ for all } p \in P\}$$

and the faces of VD(P) are:

$$j - face : \{q \in \mathbb{R}^d \mid \exists \text{ fixed set of } d - j + 1 \text{ points in } P \text{ that have the smallest distance to } q\}$$

or in other words:

$$F \subseteq \mathbb{R}^d \text{ is a } j\text{-face of } VD(P) :\Leftrightarrow \exists \text{ subset } Q \subseteq P \text{ with } |Q| = d - j + 1$$
$$\text{s.t. } q \in F \Leftrightarrow \text{all } d(q, p) \text{ are the same for } p \in Q \text{ and}$$
$$\text{the distances from } Q \text{ to all } p \in P - Q \text{ are at least as large.}$$

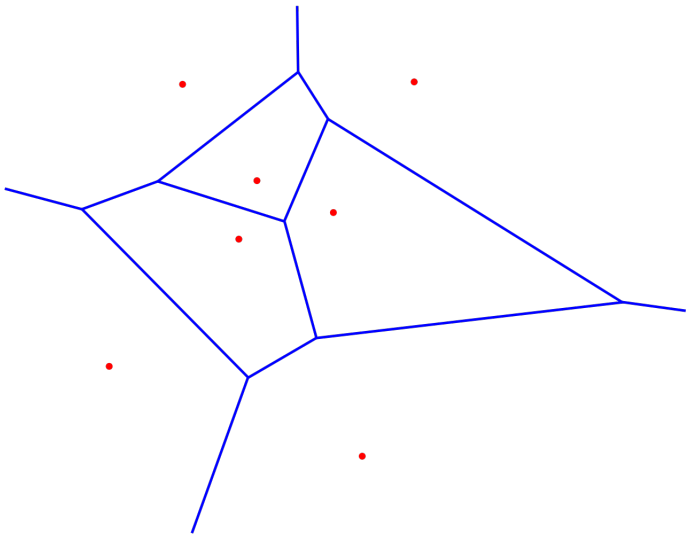


Figure 1: 2-dimensional Voronoi diagram.

The Voronoi diagram is the union of all j–faces for $j = 0, 1, \ldots, d-1$. A d–face of the Voronoi diagram is an intersection of halfspaces bounded by bisectors. Hence it is a convex polytope.

Voronoi diagrams $\leftrightarrow$ halfspace intersection

Let $P = \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^d$ be a set of points. For a given point $q \in \mathbb{R}^d$ we want to compute the point $p \in P$ that has the minimal distance to this point compared to any other point of P. Therefore we do a lifting to the unit paraboloid one dimension higher which helps us to find p.

Lifting to the unit paraboloid is the following function:

$$p = (z_1, z_2, \ldots, z_d) \mapsto \hat{p} = (z_1, z_2, \ldots, z_d, \sum_{i=1}^{d} z_i^2) \in \mathbb{R}^{d+1}$$

Whereas the unit paraboloid is the graph of

$$f(z_1, z_2, \ldots, z_d) \mapsto (z_1^2, z_2^2, \ldots, z_d^2)$$

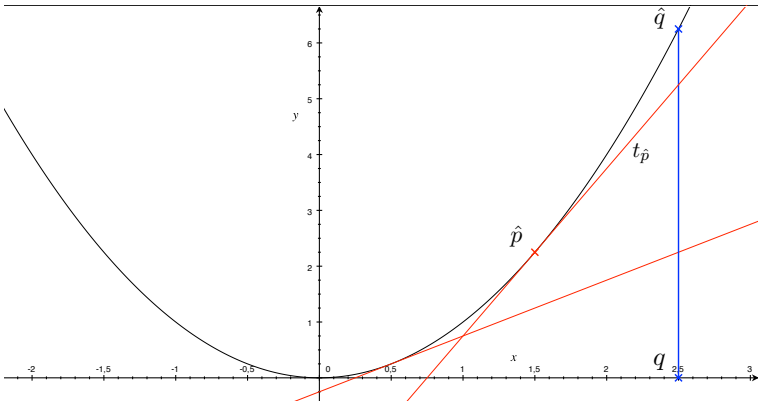


Figure 2: Lifting to unit paraboloid of a point set, its corresponding tangents, and one additional point in two dimensions.

Take $\hat{P} = \{\hat{p_1}, \hat{p_2}, \ldots, \hat{p_n}\} \subseteq \mathbb{R}^{d+1}$ and let $t_{\hat{p}}$ be the tangent on the unit paraboloid at $\hat{p}$. Then

$$t_{\hat{p}} : x_{d+1} = z_{d+1} + \nabla f(p) \cdot (\vec{x} - p) = \sum_{i=1}^{d} z_i^2 + \begin{pmatrix} \frac{\partial f(p)}{\partial z_1} \\ \frac{\partial f(p)}{\partial z_2} \\ \vdots \\ \frac{\partial f(p)}{\partial z_d} \end{pmatrix} \cdot (\vec{x} - p)$$

$$= \sum_{i=1}^{d} z_i^2 + \begin{pmatrix} 2z_1 \\ 2z_2 \\ \vdots \\ 2z_d \end{pmatrix} \cdot (\vec{x} - p) = \sum_{i=1}^{d} z_i^2 + \sum_{i=1}^{d} 2z_i(x_i - z_i)$$

$$= \sum_{i=1}^{d} (z_i^2 + 2z_i x_i - 2z_i^2) = \sum_{i=1}^{d} (2z_i x_i - z_i^2)$$

Now let $q = (q_1, q_2, \ldots, q_d)$ be some other point in $\mathbb{R}^d$. Consider $\hat{q}$ and take the tangent hyperplane $t_{\hat{p}}$ of some point $p \in P$. What is the vertical distance between $\hat{q}$ and $t_{\hat{p}}$?

$$\hat{q}_{d+1} - t_{\hat{p}}(q_1, q_2, \ldots, q_d) = \sum_{i=1}^{d} q_i^2 - \sum_{i=1}^{d}(2z_i q_i - z_i^2) = \sum_{i=1}^{d}(q_i^2 - 2z_i q_i + z_i^2) = \sum_{i=1}^{d}(q_i - z_i)^2 = d(d,p)^2$$

So, to find the closest site for q, take $\hat{q}$ and find the highest tangent $t_{\hat{p}}$ that intersects the vertical ray through q in $\mathbb{R}^{d+1}$.

Take home message: The faces of $VD(P)$ are exactly the faces of $\cap t_{\hat{p}}^+$ (projected onto $\mathbb{R}^d$).

2 Volume of Convex Polytopes

2.1 Introduction

This chapter deals with the VCP problem:

Given: a convex polytope of dimension d

Compute: the volume of the polytope

For $d = 2$, where our input is a polygon and our output is its area (the two-dimensional "volume"), the VCP problem is easy to solve, even if the polygon isn't convex:

Let $(x_1, y_1), \ldots, (x_n, y_n)$ be the vertices, in counterclockwise order, of a simple polygon (where a simple polygon is any non-intersecting polygon, which may or may not be convex). Then we can compute the area of the polygon in linear time using the formula

$$a = \frac{1}{2}\sum_{i=1}^{n}(x_i y_{i+1} - x_{i+1} y_i),$$

where $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.

For arbitrary dimensions, we will see that the problem becomes considerably more difficult. Before we begin analyzing the complexity, however, we will (re-)acquaint ourselves with some useful concepts and definitions.

2.2 Counting Problems

Definition 2. *A function $f : \{0,1\}^* \to \mathbb{N}$ is in the complexity class **#P** iff there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M such that*

$$\forall x \in \{0,1\}^* : f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}|,$$

where $M(x,y)$ is the output of M, which is written on the tape when M terminates.

This definition resembles the usual definition of the complexity class NP, except that we count the number of witnesses y (hence "counting problem") rather than just deciding whether one exists or not ("decision problem").

An equivalent definition of #P can be formulated using non-deterministic Turing machines:

Definition 3. *A function $f : \{0,1\}^* \to \mathbb{N}$ is in the complexity class **#P** iff there exists a non-deterministic Turing machine M with polynomial runtime such that*

$$\forall x \in \{0,1\}^* : f(x) = \textit{number of accepting computations for } x.$$

Some examples of counting problems are #SAT, which counts the number of satisfying assignments for a given Boolean formula, #TSP, which counts the number of feasible solutions to the travelling salesman problem, and #PMATCH, which counts the number of perfect matchings for a bipartite graph.

Observation 4. *Suppose that $L \subseteq \{0,1\}^*$ is NP-complete and that the corresponding counting function f_L is computable in polynomial time. Then, by checking whether $f_L(x) \geq 1$ or not, we can determine in polynomial time whether $x \in L$ or not. In other words, $L \in \mathrm{P}$, from which (given the NP-completeness of L) we can conclude that $\mathrm{P} = \mathrm{NP}$.*

Since, however, this conclusion contradicts the general belief that $\mathrm{P} \neq \mathrm{NP}$, we presume that our initial supposition was incorrect; i. e. the counting function for an NP-complete problem cannot be computed in polynomial time.

In any case, we observe that #P is at least as difficult as NP.

In order to define when a function is #P-complete, we need the complexity class FP (analogous to P), which is defined as follows:

Definition 5. *A function $f : \{0,1\}^* \to \mathbb{N}$ is in the complexity class **FP** iff it can be computed in polynomial time, i. e., there exists a polynomial-time Turing machine M such that*

$$\forall x \in \{0,1\}^* : M(x) = f(x)$$

if we interpret $M(x)$ as the binary representation of a natural number.

We also need to define what we mean when we say that a counting problem can be reduced to another. There are several different possible approaches, one of which is Cook reducibility (reduction by oracle).

Definition 6. *Let $f : \{0,1\}^* \to \mathbb{N}$ be some function. Then FP^f is the class of all functions g that can be computed in polynomial time with a Turing machine using an oracle for f. The Turing machine for g may write queries $x \in \{0,1\}^*$ on an extra "oracle tape" at any time (not necessarily just once, as would be the case in a many-one or Karp reduction), and the oracle will calculate the result $f(x)$ and write it on the oracle tape in one step.*

*We say that the functions $g \in \mathrm{FP}^f$ are **Cook reducible** to f.*

Based on this definition of reducibility, we define #P-completeness as follows:

Definition 7. *A function $f : \{0,1\}^* \to \mathbb{N}$ is **#P-hard** iff $\#\mathrm{P} \subseteq \mathrm{FP}^f$ (in other words, every problem in #P is Cook reducible to f). It is **#P-complete**, iff it is both #P-hard and an element of #P.*

Another useful approach is parsimonious reducibility:

Definition 8. *Let $f, g : \{0,1\}^* \to \mathbb{N}$ be two counting problems. A parsimonious reduction from f to g is a polynomial time function $\sigma : \{0,1\}^* \to \{0,1\}^*$ such that*

$$\forall x \in \{0,1\}^* : \ f(x) = g(\sigma(x)),$$

in other words, the transformation σ preserves the number of witnesses.

2.2.1 Examples

SAT The reduction in Cook's theorem of any problem in NP to the Boolean satisfiability problem is a parsimonious reduction; the number of satisfying assignments of the constructed formula is the same as the number of witnesses of the original problem.

3SAT Although the standard reduction from SAT to 3SAT is not parsimonious, #3SAT is #P-complete, as was shown in the third problem set.

2SAT #2SAT is also #P-complete, although 2SAT itself can be computed in polynomial time.

PERM Let $A = (a_{ij})$ be an $n \times n$ matrix and let $S(n)$ be the set of all permutations over $\{1, \ldots, n\}$. Then we can compute the determinant

$$\det(A) = \sum_{\sigma \in S(n)} \operatorname{sgn}(\sigma) \cdot a_{1\sigma(1)} \cdots a_{n\sigma(n)}$$

in polynomial time. The similar-looking problem PERM of computing the permanent

$$\operatorname{perm}(A) = \sum_{\sigma \in S(n)} a_{1\sigma(1)} \cdots a_{n\sigma(n)},$$

however, is #P-complete, even if we only consider matrices A with entries from $\{0,1\}$ in $\mathbb{Q}$.

Perfect Matchings Suppose that $A = (a_{ij})$ is an $n \times n$ matrix with entries from $\{0,1\}$ in $\mathbb{Q}$. Then we can use this matrix to construct an instance for the perfect matching counting problem #PMATCH:

Let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ be two disjoint sets of vertices and let $E \subseteq U \times V$ be a set of edges, such that

$$\forall u_i \in U \ \forall v_j \in V : \ (u_i, v_j) \in E \ \Leftrightarrow \ a_{ij} = 1.$$

Then the number of perfect matchings for this bipartite graph $G = (U, V, E)$ yields the permanent of the original matrix A.

Since the above construction is a parsimonious reduction of the #P-complete problem PERM to #PMATCH, it follows that #PMATCH is itself #P-hard.

2.3 Linear Orders

Definition 9. *A **partial order** on a set X is a binary relation $\leq$ over X that is reflexive, anti-symmetric and transitive.*

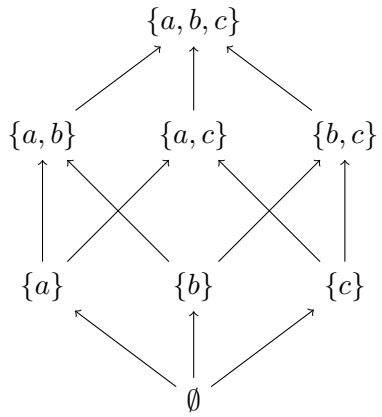
We can represent partial orders using Hasse diagrams.

Definition 10. *A **Hasse diagram** for a partial order $\leq$ on a set X is a directed acyclic graph*

$$H_{\leq} = (X, \{(x, y) \in X \times X \mid x \leq y \ \wedge \ \nexists z \in X : x \leq z \leq y\}),$$

with a node for each element of X and an edge for every pair (x, y) in the transitive reduction of $\leq$. When drawing a Hasse diagram, the nodes are typically arranged in such a manner that all the edges point upward.

Example. Let A be some set and $X := 2^A$ (all subsets of A). Then the subset relation $\subseteq$ is a partial order on X. For $A = \{a, b, c\}$, the Hasse diagram would be:



Definition 11. *A **linear order** on a set X is a partial order with the following additional property:*

$$\forall x, y \in X : \ x \leq y \ \vee \ y \leq x.$$

Definition 12. *A **linear extension** of a partial order $\mathcal{P} \subseteq X^2$ on a set X is a linear order $\mathcal{L} \subseteq X^2$ with $\mathcal{P} \subseteq \mathcal{L}$ (i. e. $\mathcal{L}$ preserves the partial order $\mathcal{P}$).*

In the next lecture, we will show that #LEXT, the counting problem that computes the number of linear extensions for a given partial order, is #P-hard.