**Figure 1.** Left: The feasible polytope is defined by multiple half-planes; Right: Goal is to find optimal vertex in the feasible polytope that is furthest in the direction of the objective function vector $c$.

*Note: These lecture notes are based on the textbook "Computational Geometry" by Berg et al.; lecture notes from [1]; and lecture notes from MIT 6.854J Advanced Algorithms class by M. Goemans*

# 1   The Linear Programming Problem

In linear programming (LP), we want to find a point in $d$ dimensional space that minimizes a given linear *objective function* subject to a set of linear *constraints*. Frequently, LP is done in 100's or 1,000's of dimensions, but many applications occur in low-dimensional spaces.

Formally, we're given a set of linear inequalities, called constraints in $\mathbb{R}^d$. Given a point $(x_1, \ldots x_d) \in \mathbb{R}^d$, we can express a constraint as $a_1 x_1 + \ldots a_d x_d \leq b$, by specifying coefficients $a_i$, $b \in \mathbb{R}$.[1] Each constraint defines a halfspace in $\mathbb{R}^d$ and the intersection of halfspaces defines a (possibly empty or unbounded) polytope called the *feasible polytope*.

Next we're given a linear *objective function* to be maximized. Given a point $x \in \mathbb{R}^d$, we express the objective function as $c_1 x_1 + \ldots c_d x_d$, for coefficients $c_i$.[2] We can think of the coefficients as a vector $c \in \mathbb{R}^d$, and then the value of the objective function for $x \in \mathbb{R}^d$ is just $x \cdot c$. Assuming general position, it's not hard to see that if a solution exists, it'll be achieved by a vertex of the feasible polytope. See Figure 1.

In general, a $d$-dimensional LP can be expressed as.

Maximize: $c_1 x_1 + c_2 x_2 + \ldots c_d x_d$.
Subject to:

$$a_{1,1} x_1 + \ldots a_{1,d} x_d \leq b_1$$
$$a_{2,1} x_1 + \ldots a_{2,d} x_d \leq b_2$$
$$\ldots$$
$$a_{n,1} x_1 + \ldots a_{n,d} x_d \leq b_n$$

---

[1] Note that there is no loss in generality by assuming $\leq$ constrains, since we can convert a $\geq$ to this form by simply multiplying the inequality by $-1$.

[2] Again there is no difference between minimization or maximization since we can negate the coefficients to go from one to the other.
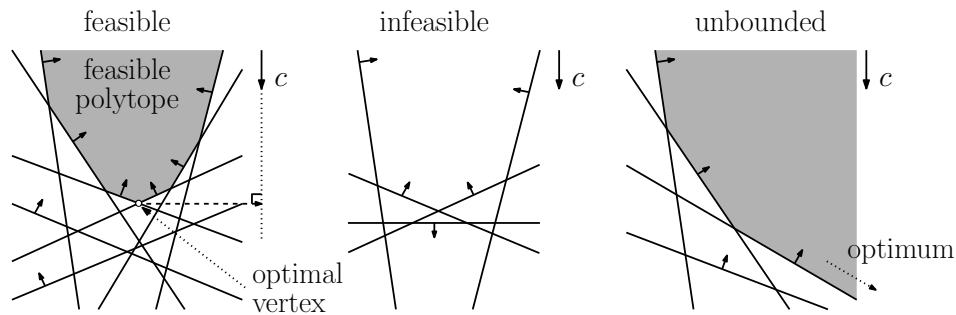
**Figure 2.** Possible Outcomes of a LP

where $x_1, \ldots x_d$ are the $d$ variables, and $a_{i,j}, c_i$ and $b_i$ are given real numbers. This can be written in matrix notation as

$$\text{Maximize:} \quad c^T x,$$
$$\text{Subject to:} \quad Ax \leq b$$

Here $c$ and $x$ are $d$-vectors, $b$ is an $n$ and $A$ is a $n$ by $d$ matrix, where $n$ is the number of constraints. Note that $n$ should be at least as large as $d$.

There are three possible outcomes for a given LP problem. See Figure 2

**Feasible:** An optimal point exists and (assuming general position) is a unique vertex of the feasible polytope.

**InFeasible:** The feasible polytope is empty and there is no solution

**Unbounded:** The feasible polytope is unbounded in the direction of $c$ and so no finite optimal solution exists.

## 2 Solving LP in Constant Dimensions

We now discuss the incremental construction method for efficiently solving LP in constant dimensions. There are other methods for general LP (such as the interior point method). These algorithms have *weakly polynomial time* in that they are polynomial in the number of bits of the input. (This is in contrast to *strongly polynomial time* algorithms that are polynomial in the number of values (i.e. numbers) in the input, not in their size).

Incremental construction is a technique that is frequently used in computational geometry.

### 2.1 Initialization

Recall that we are given $n$ halfspaces $\{h_1, \ldots h_d\}$ in $\mathbb{R}^d$, and an objective vector $c$, and we want to compute the vertex of the feasible polytope that is the most extreme in the direction of $c$.

We will initially assume that the LP is bounded and that we have $d$ halfspaces that provide us with an initial feasible point. Our approach will be to add halfspaces one at a time and successively update this feasible point.

Finding a set of initial $d$ bounding halfplanes is non-trivial. Assume that there is some maximum value $M$ that any variable can take on. The we add $d$ constraints of the following form for all variables

$1 \leq i \leq d \; x_i \leq M$ if $c_i > 0$, and $-x_i \leq M$ otherwise. These will be our initial $d$ constraints. See Figure 3 left.
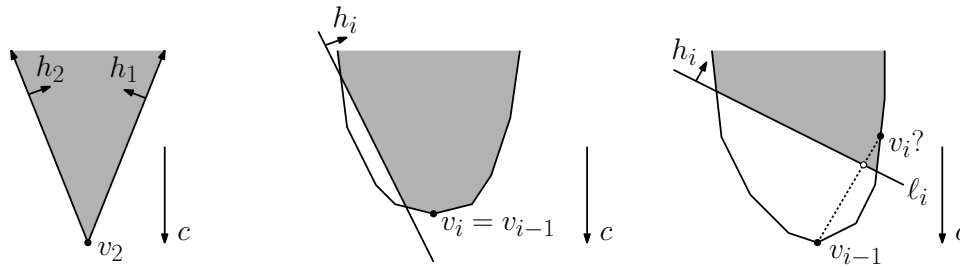
**Figure 3.** Left: Starting the incremental construction; Right: Proof that new optimum lies on $\ell_i$

Note that if one of these "max value" constraints turns out to intersect the point that is eventually output, then we know that the original LP is unbounded. In this way, we can detect unbounded LPs also.

Additionally, we'll assume that there is a unique solution, which we call the optimal vertex. This follows via the general position assumption (or by rotating the plane slightly).

## 2.2 Incremental Algorithm

We can imagine adding the halfspaces $h_{d+1}, h_{d+2}, \ldots$ and with each addition, update the current optimum vertex if necessary. Note that the feasible polytope gets smaller with each halfplane addition and so the value of the objective function can only decrease. In Figure 4, the $y$-coordinate of the feasible vertex decreases.

Let $v_i$ be the current optimum vertex after halfplane $h_i$ is added. There are two cases that can occur when $h_i$ is added. In the easy case, $v_{i-1}$ lies in the halfspace $h_i$, and so already satisfies the constraint. Thus $v_i = v_{i-1}$.

In the hard case, $v_{i-1}$ is not in the halfplane $h_i$, i.e. it violates the constraint. In this case, the following lemma shows that $v_i$ must lie on the hyperplane that bounds $h_i$.

**Lemma 1.** *After addition of halfplane $h_i$, if the LP is still feasible but $v_i \neq v_{i-1}$, then $v_i$ lies on the hyperplane bounding $h_i$.*

**Proof:** Let $\ell_i$ denote the bounding hyperplane for $h_i$. By way of contradiction, suppose $v_i$ does not lie on $\ell_i$ (see Figure 3). Now consider the line segment between $v_{i-1}$ and $v_i$. First note that this line segment must cross $\ell_i$ since $v_i$ is in $h_i$ and $v_{i-1}$ is not. Further, the entire segment is in the region bounded by the first $i-1$ hyperplanes, and so by convexity, the part of the segment that is in $h_i$ is in the region bounded by the first $i$ hyperplanes.

Next, note that the objective function is maximized on this line segment at the point $v_{i-1}$. Also, since the objective function is linear, it must be non-decreasing as we move from $v_i$ to $v_{i-1}$. Thus, there is a point on $\ell_i$ with objective function at least equal to $v_i$. But this contradicts the uniqueness property, so $v_i$ must be on $\ell_i$. $\qquad\qquad\square$

## 2.3 Recursively updating $v_i$

Consider the case where $v_{i-1}$ does not lie in $h_i$ (Figure 4, left). Again, let $\ell_i$ denote the hyperplane bounding $h_i$. We basically project everything onto that hyperplane and solve a $d-1$ dimensional LP. In particular, we first project $c$ onto $\ell_i$ to get the vector $c'$ (Figure 4, right). Next intersect each of the halfspaces $h_1, \ldots h_{i-1}$ with $\ell_i$. Each projection in a $d-1$ dimensional halfspace that lies on $\ell_i$. Finally, since $\ell_i$ is a $d-1$ dimensional hyperplane, we can project $\ell_i$ onto $\mathbb{R}^{d-1}$ with a
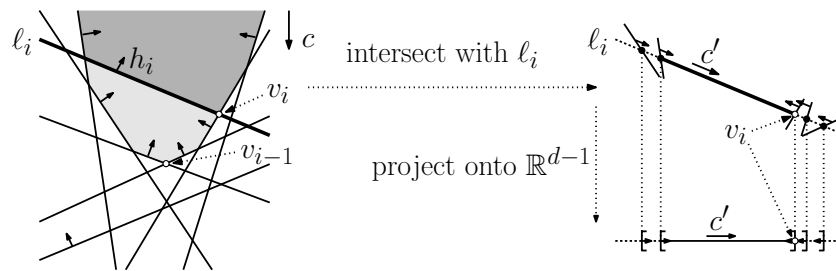
**Figure 4.** Projection during the incremental construction.

1-to-1 mapping. Then we apply this mapping to all the other vectors to get a LP in $\mathbb{R}^{d-1}$ with $i-1$ constraints.

Algebraically, the way we can do this is simply to set the constraint associated with $\ell_i$ to equality and then remove a variable and a constraint from the LP. For example, if the constraint associated with $\ell_i$ is $x_1 + 2x_2 - 3x_3 \le 5$. Then we set $x_1 = 5 - 2x_2 + 3x_3$, do a substitution in the LP using this equation wherever we see the variable $x_1$ and then remove the variable $x_1$ from the LP. We can do all this in $O(di)$ time.

## 2.4   Base Case

The recursion ends when we get an LP in 1-dimensional space. Then the projected objective vector just points one way or the other on the real line; the intersection of each half-space with $\ell_i$ is a ray. Computing the intersection of a collection of rays on the line can be done in linear time. (This is the heavy solid line in Figure 4, right). The new optimum is whichever endpoint of this interval is most extreme in the direction of $c'$. If the interval is empty, then the feasible polytope is also empty. So when $d = 1$, we can solve the LP over $i$ halfplanes in $O(i)$ time.

## 2.5   Worst-Case Analysis

Let $T(d, n)$ be the runtime for the LP with $n$ constraints in $d$ dimensional space. What is $T(d, n)$? If $x_{n-1}$ satisfies the removed constraint (which takes $O(d)$ time to check), we're done. If not, we reduce the LP to only $d-1$ variables in $O(dn)$ time ($O(d)$ time to eliminate the variable in each constraint). So, in the worst case, we get $T(d, n) = \sum_{i=d+1}^{n} \max(d, di + T(d-1, i-1))$. Since $d$ is constant, can simplify to $T(d, n) = \sum_{i=d+1}^{n} (i + T(d-1, i-1))$. Unfortunately, the solution to this recurrence is $\Theta(n^d)$.

## 2.6   Randomization to the Rescue

Note that the above analysis assumes we *always* require a projection, and that we never get the lucky case where $v_{i-1}$ is in $h_i$. If we first randomly permute the hyperplanes, we can calculate the probability of the "lucky" and "unlucky" cases to get an expected runtime. Let $p_i$ be the probability that there is no change to $v_{i-1}$. Then the expected runtime, up to constant factors that don't depend on $d$, is given by the recurrence relation:
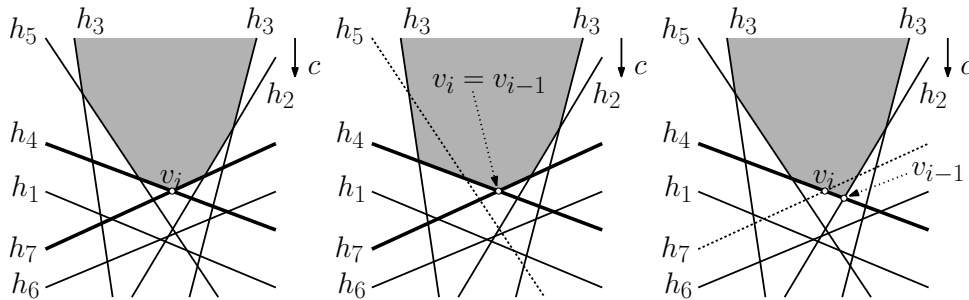
**Figure 5.** Backwards analysis for Randomized LP

$$T(d,n) \leq \left( \sum_{i=d+1}^{n} (1-p_i)d + p_i(di + T(d-1,i)) \right)$$

$$\leq \sum_{i=d+1}^{n} (d + p_i(di + T(d-1,i)))$$

So what is $p_i$? Assuming general position, there are exactly $d$ halfspaces whose intersection defines the point $v_i$. At step $i$, there have been $i$ total halfspaces inserted, exactly $d$ of which define the point $v_i$. Since the halfspaces are randomly permuted, this means that

$$p_i = \frac{d}{i}$$

For example, in Figure 5, $h_7$ and $h_4$ define the point $v_i$, so $v_i$ changes iff one of these two is the last of the 7 halfspaces inserted. Note that in this analysis, we have denoted $d$ halfspaces as special (those that define $v_i$) and only then revealed the permutation order of the first $i$ halfspaces. This technique is sometimes called *backwards analysis* or *principle of deferred decision*.

We can now prove the following.

**Lemma 2.** $T(d,n) \leq \gamma_d d! n$, where $\gamma_d = \frac{d+d^2}{d!} + \gamma_{d-1}$; $\gamma_1 = 1$

**Proof:** We will prove this by induction on $d$. The base case, $d = 1$, is trivial since $T(1,n) = n$.
For the inductive step, we have:

$$T(d,n) \leq \sum_{i=d+1}^{n} (d + p_i(di + T(d-1,i)))$$

$$\leq \sum_{i=d+1}^{n} (d + (d/i)(di + \gamma_{d_1}(d-1)!i))$$

$$\leq \sum_{i=d+1}^{n} (d + d^2 + \gamma_{d-1}d!)$$

$$\leq (d + d^2 + \gamma_{d-1}d!)n$$

$$\leq \gamma_d d! n$$

Step 2 holds by plugging $p_i$ into the recurrence and invoking the inductive hypothesis.
The last step holds provided that $\gamma_d = \frac{d+d^2}{d!} + \gamma_{d-1}$.      □

**Lemma 3.** *There is a constant $\gamma$ such that $\gamma_d \leq \gamma$ for all $d \geq 1$.*

**Proof:** Idea: Show that $(d + d^2)/d! \leq 2^{-d}$ for $d$ sufficiently large. Then bound the geometric sum. $\square$

Putting these last two lemmas together, we get that:

**Lemma 4.** $T(d, n) = O(d!n)$.

# 3   Higher Dimension Convex Hull Algorithms

*Note: These lecture notes are based on lecture notes from MIT 6.854J Advanced Algorithms class by M. Goemans*

## 3.1   Definitions

A *polytope* is informally a geometric object with "flat" sides. More formally, it is the convex hull of a finite number of points. Another recursive definition is:

- A 0-polytope is a point

- A 1-polytope is a line segment (edge)

- The sides (faces) of a k-polytope are (k-1)-polytopes that may have (k-2)-polytopes in common. (For example a 2-polytope has sides that are line segments, which may meet at points.

A *simplex* is $k$-polytope that is the convex hull of its $k + 1$ vertices. Informally, it is the generalization of the idea of a triangle or tetrahedron.

For any $0 \leq k < d$, a *k-face* of a d-polytope, $P$ is a face of $P$ with dimension $k$. A (d-1)-face is called a facet. A (d-2)-face is called a ridge. A 1-face is a edge, and a 0-face is a vertex.

A *simplicial polytope* is a polytope where every face is a simplex. It is the convex hull of a set of points in general positions.

Every facet of a *d*-polytope has a *supporting hyperplane*, which is the hyperplane in dimension $d$ that intersects the entire facet.

## 3.2   Number of Facets

Even outputting a convex hull in high dimensions can be a challenge. In particular, the worst case number of facets of a set of $n$ points in $d$ dimensions is $\Theta(n^{\lfloor d/2 \rfloor})$.

We show this in Section 5.

## 3.3   Facet Graph

A simplex in $d$ dimensions is defined by $d + 1$ points, and has $d + 1$ facets. But in general, things may not be this simple. A convex hull over $n$ points can have many more facets. For an arbitrary polytope $P$, the facet graph $\mathcal{F}(P)$ has the following properties:

- Vertices of $\mathcal{F}(P)$ are the facets of conv$(P)$. Each vertex is associated with the $d$ points that define the facet.
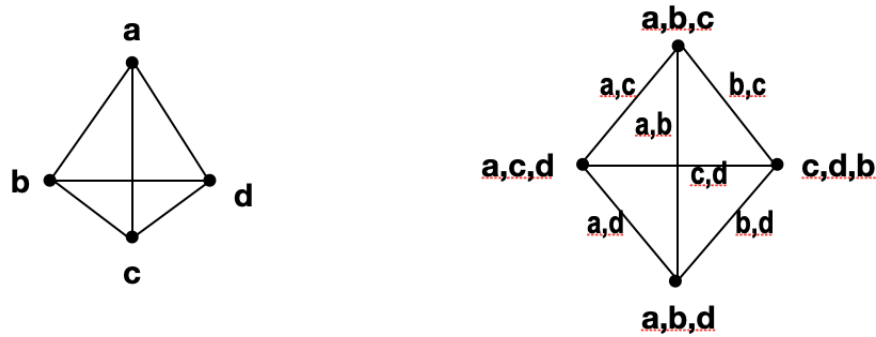
**Figure 6.** Left: 3D polytope, with four vertices labelled $a, b, c, d$; Right: The corresponding facet graph with vertices for each of the 4 facets on the left. Each facet vertex is labelled with the 3 points from the set $\{a, b, c, d\}$ supporting that facet. There is an edge for every ridge that is labelled with the 2 points supporting that ridge. Note that each facet is connected to $d$ ridges, and each ridge is connected to 2 facets.

- Edges of $\mathcal{F}(P)$ are the ridges of $\mathrm{conv}(P)$. Each ridge connect two facets, whose intersection is the ridge. Each ridge is defined by $d - 1$ points

Assuming general position, there are $d$ ridges bordering each facet. To see this, note that each facet is uniquely determined by $d$ points. And each ridge bordering that facet is uniquely determined by $d - 1$ points. This implies that each facet borders $d$ ridges. For example, if we have the facet $v_1, v_5, v_7, v_8, v_9$. Then this facet borders the 5 ridges: $(v_5, v_7, v_8, v_9)$; $(v_1, v_7, v_8, v_9)$; $(v_1, v_5, v_8, v_9)$; $(v_1, v_5, v_7, v_9)$; $(v_1, v_5, v_7, v_8)$.

An example facet graph is give in Figure 6.

# 4   Convex Hull Algorithm for High Dimensions

Seidel's algorithm has runtime $O(n + n^{\lfloor d/2 \rfloor})$ and assumes points are in general position. For $d \geq 3$, it is optimal. Take a random permutation $x_1, x_2, \ldots x_n$ of the points Let $P_i$ be the convex hull of $x_1, \ldots x_i$. We incrementally compute $P_{d+2}, \ldots, P_n$, using notions of visibility.

## 4.1   Visibility

We make use of the following definitions about visibility.

- A facet $F$ is *visible* from a point $x$, if the supporting hyperplane of $F$ separates $x$ from $P$ (the set of points in the convex hull). Otherwise $F$ is called *obscured*.

- From the vantage of a point $x$, a ridge of $P$ is called

  - *visible:* if both facets it connects are visible
  - *obscured:* if both facets are obscured
  - *a horizon ridge:* if one facet is visible and the other obscured.

### 4.1.1   Algorithm Overview

Our algorithm will incremental, keeping track of the convex hull of points $x_1, \ldots, x_{i-1}$. It adds vertex $x_i$ in step $i$, removing all facets visible from $x_i$ and adding in all the new facets induced by $x_i$. See Figure 7.
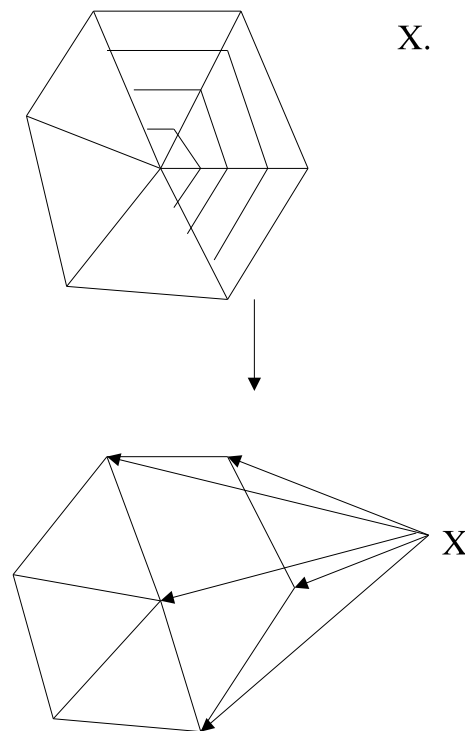
7

X.

X

**Figure 7.** Top: Shaded regions are the facets visible from the point $X$. Bottom: Visible facets are removed and new facets are added.

### 4.1.2 The Ridge Search Tree

The main data structure in our algorithm will be the *ridge search tree*. This stores the ridges of the polytope, and each of the stored ridges maintains doubly-linked pointers with the two facets bordering that ridge. By successively finding ridges in the ridge search tree, and then finding the corresponding facets, we can easily traverse the facet graph.

This ridge search tree enables lookups of any "key" consisting of a set of $d-1$ vertices to either find the ridge supported by these vertices if it exists in the polytope, or return NIL if that set of vertices is not a ridge in the polytope.

The search tree for the ridges is height $O(d)$, enabling lookups and insertions in $O(d)$ time. The idea is that each internal node has branching factor up to $n$, the number of total possible points we are finding the convex hull of. To look up a possible ridge, we sort the points in lexicographical order and then hash in the root node on the first point in the key to get the correct child node. We continue in this way till we reach a leaf node, and find the ridge if it is in tree, or find the correct place that it should be inserted if it is not in the tree.

### 4.2 The algorithm

The algorithm is incremental, keeping track of the convex hull of points $x_1, \ldots, x_{i-1}$. It adds vertex $x_i$ in step $i$, removing all facets visible from $x_i$ and adding in all the new facets induced by $x_i$. See Figure 7.

First, randomly permutes all the points in $P$. Then, start out with the convex hull formed by

the first $d$ points. Let $C_{i-1}$ be the convex hull of points $x_1, \ldots x_{i-1}$.

The algorithm adds points $x_{d+1}, \ldots x_n$ as follows.

1. Find one facet $F$ of $C_{i-1}$ that is visible from $x_i$. If there is no visible facet, skip all steps below. A visible facet can be found via linear programming in expected $O(d!i)$ time as follows. We want to find a hyperplane $a^T x = b$ (where unknowns are $a, b \in \mathbb{R}^d$) such that $a^T x_i = b$ and either (1) $a^T x_j \leq b$ for all $j = 1, \ldots, i-1$; or (2) $a^T x_j \geq b$ for all $j = 1, \ldots, i-1$. This can be found by a linear program in $O(d)$ dimensions, with $O(i)$ constraints. Any solution to this LP will correspond to a hyperplane supporting a new facet in $C_i$. Vertices in the facet can be found by finding the $d$ points on the hyperplane in $O(di)$ time; all incident ridges can be found in $O(d)$ time. One of the two facets incident to the horizon ridge is a visible facet. To find which, determine which supporting plane separates $x_i$ from $x_1, x_2, \ldots, x_{i-1}$.

2. Find all visible facets; determine all horizon ridges. Then, delete all visible facets and all visible ridges. We can do this via depth first search since visible facets and invisible facets are separated by horizon ridges. We can determine if a facet is visible in $O(d)$ time by seeing if its supporting hyperplane separates $v_i$ from points on the neighboring facet.

3. Construct all new facets. Each horizon ridge corresponds to a new facet combining $x_i$ and the points in the ridge.

4. Each new facet contains $d$ ridges. Find each of these ridges in the ridge search tree, or insert if new. Update one of the two doubly-linked pointers between each of these ridges and the newly created facet.

## 4.3   Example for Step 1

The equation $(2,1)^T(x,y) = (3,4)$ defines a line in $\mathbb{R}^2$. In general, $a^T(x,y) = b$ defines a line in $\mathbb{R}^2$. In Step 1, we want to find a vector $a$ and a vector $b$ to ensure that the point $x_i$ is on the hyperplane, and that all other points $x_1, \ldots, x_{i-1}$ are on the same side of the halfspace. For example, if $i = 3$ and $x_1 = (1,0)$, $x_1 = (0,1)$, and $x_2 = (1,2)$, we want to solve the following linear program:

Find variables $a_1, a_2, b_1, b_2$, such that:

$$(a_1, a_2)^T(1,2) = (b_1, b_2)$$
$$(a_1, a_2)^T(1,0) \leq (b_1, b_2)$$
$$(a_1, a_2)^T(0,1) \leq (b_1, b_2)$$

Note that technically, each of these lines expands to two linear inequalities. Also, there is nothing to maximize or minimize, we just want to find *any* feasible point (which is easier than a usual LP). Finally, we also can check feasibility for a LP where the last two lines have $\geq$ instead of $\leq$.

## 4.4   Runtime

The time to add point $x_i$ is $O(i + N_i)$ where $N_i$ is a random variable giving the number of new facets created at step $i$. To see this, first note that step (1) takes expected $O(d!i)$ time to solve the linear program this is $O(i)$ time assuming $d$ is fixed. In step 2, we delete all visible facets and

ridges, but the time to do this is charged to when they were created. In step 3, we create $N_i$ new facets, taking time $O(N_i)$. In step 4, there are at most $O(d \cdot N_i)$ new ridges, each of these can be processed in the ridge tree in $O(d)$ time, so this step takes $O(d^2 N_i)$ time. So the total time to process $x_i$ is $O(i + N_i)$.

To get the expected runtime, we can compute $E(N_i)$ using the principle of deferred decision (aka backward analysis). Recall our claim that if we have a polytope with $i$ vertices in $\mathbb{R}^d$, then the number of facets is $O(i^{\lfloor d/2 \rfloor})$. First, we fix one of the $O(i^{\lfloor d/2 \rfloor})$ facets of $C_i$ (see Section 5 for how we can show there are at most $O(i^{\lfloor d/2 \rfloor})$ facets). The probability that the $i$-th point ($x_i$) participates in this facet is $d/i$.

Hence using linearity of expectation over all $O(i^{\lfloor d/2 \rfloor})$ facets, $E(N_i) = O(\frac{d}{i} i^{\lfloor d/2 \rfloor}) = O(i^{\lfloor d/2 \rfloor - 1})$. Thus, the expected runtime of Seidel's algorithm is:

$$\sum_{i=1}^{n} O(i + N_i) = \sum_{i=1}^{n} O(i + i^{\lfloor d/2 \rfloor - 1})$$
$$= O(n^{\lfloor d/2 \rfloor})$$

The second step above holds since, for any value $d$, $\sum_{i=1}^{n} i^d = O(n^{d+1})$. This is true since

$$\sum_{i=1}^{n} i^d \le \sum_{i=1}^{n} n^d$$
$$= n^{d+1}$$

# 5   Bounding the Number of Facets

## 5.1   Polarity

There are two ways to create polytopes: (1) convex hull of a set of points; and (2) intersection of a collection of closed halfspaces. We show that these are essentially identical through a concept caller *polar transformation*. A polar transformation maps points to hyperplanes and vice versa. This transformation is another example of duality.

Let $\mathcal{O}$ be the origin in $\mathbb{R}^d$ for some number $d$. Then we can view any point $p \in \mathbb{R}^d$ as a $d$-element vector. (If $\mathcal{O}$ is not the origin then $p$ can be identified with the vector $p - \mathcal{O}$.) Given two vectors $p$ and $x$, recall that $p \cdot x$ is the dot-product of $p$ and $x$. Then the *polar hyperplane* of $p$ is denoted :

$$p^* = \{x \in \mathbb{R}^d, \ p \cdot x = 1\}.$$

Clearly this is linear in the coordinates of $x$, and so $p^*$ is a hyperplane in $\mathbb{R}^d$. If $p$ is on the unit sphere centered at $\mathcal{O}$, then $p^*$ is a hyperplane that passes through $p$ and is orthogonal to the vector $\overrightarrow{\mathcal{O}p}$.

As $p$ moves away from the origin along this vector, the dual hyperplane move closer to the origin, and vice versa, so that the product of their distances from the origin is always 1. See Figure 8(left).

This transformation also has an inverse. Let $h$ be any hyperplane that does not contain $\mathcal{O}$. Then the *polar point* of $h$, denoted $h^*$ is the point that satisfies $h^* \cdot x = 1$ for all $x \in h$.

## 5.2   Properties

Like with point-line duality, the polar transformation satisfies certain incidence and inclusion properties between points and hyperplanes.
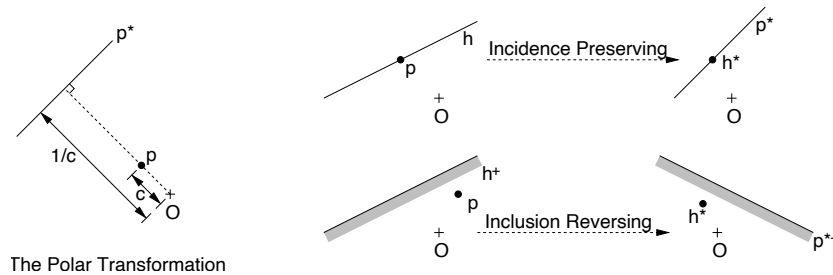
**Figure 8.**

Let $p$ be any point in $\mathbb{R}^d$ and let $h$ be any hyperplane in $\mathbb{R}^d$. The polar transformation satisfies the following properties. For a hyperplane $h$, let $h^+$ be the halfspace containing the origin and $h^-$ be the other halfspace for $h$. Figure 8(right) illustrates the following facts.

- **Incidence Preserving:** Point $p$ belongs to hyperplane $h$ iff $h^*$ belongs to $p^*$

- **Inclusion Reversing:** Point $p$ belongs to halfspace $h^+$ iff point $h^*$ belongs to halfspace $(p^*)^+$. This implies that point $p$ belongs to halfspace $h^-$ iff point $h^*$ belongs to halfspace $(p^*)^-$. Intuitively, this means that the polarity transform reverses relative positions.

Note that a bijective transformation that preserves incidence relations is called a *duality*. So the above claim shows that the polarity transform is another dualtiy.

## 5.3 Convex Hulls and Halfspace Intersection

We now want to transform a polytope defined as the convex hull of a finite set of points to a polytope defined as the intersection of a finite set of closed halfspaces. To do this, we need a mapping from a point to a *halfspace*. For any point $p \in \mathbb{R}^d$, define

$$p^\# = \overline{(p^*)^-} = \{x \in \mathbb{R}^d \mid x \cdot p \leq 1\}$$

This just first finds the polar hyperplane of $p$, and then takes the *closed* halfspace containing the origin.

Now for any set of points $P \subseteq \mathbb{R}^d$, define its *polar image* to be the intersection of these halfspaces.

$$P^\# = \{x \in \mathbb{R}^d \mid x \cdot p \leq 1, \forall p \in P\}$$

Thus, $P^\#$ is the intersection of a finite set of closed halfspaces, one for each $p \in P$. Is $P^\#$ convex? Yes, since each halfspace is convex, and the intersection of any set of convex spaces is convex.

The following lemma shows that $P$ and $P^\#$ are essentially equivalent via polarity.

**Lemma 5.** *Let $S = \{p_1, \ldots p_n\}$ be a set of points in $\mathbb{R}^d$ and let $P = \text{conv}(S)$. Then:*

$$P^\# = S^\#$$

*Furthermore:*

1. *A point $a \in \mathbb{R}^d$ is on the boundary of $P$ iff the hyperplane $a^*$ supports $P^\#$.*

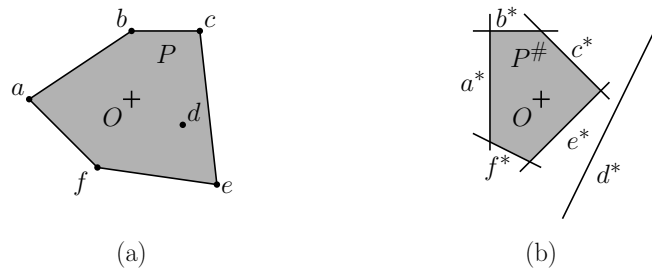2. *Each $k$-face of $P$ corresponds to a $(d-1-k)$-face of $P^\#$*

11

(a)                           (b)

**Figure 9.**

**Proof:** Assume that $\mathcal{O}$ is contained within $P$. We can guarantee this by, e.g., translating $P$ so that its center of mass coincides with the origin.

Consider some point $v_1 \in P$ that is on some hyperplane $h$ supported by a facet containing points $v_1, v_2, \ldots, v_d \in P$. [**Jared:**  *EXERCISE: What can be said about hyperplanes $v_1^*, \ldots, v_d^*$ and point $h^*$.* ]

Then, hyperplanes $v_1^*, \ldots, v_d^*$ contain point $h^*$ in the polar plane, by the incidence preservation property. This shows parts (1) and (2) of the lemma.

Now let $h_1, \ldots h_\ell$ be all the hyperplanes that support facets of $\mathrm{conv}(S)$. Consider some point $v \in S - P$ (i.e. $v$ is not in $\mathrm{conv}(S)$) and consider all the hyperplanes $h_1, \ldots h_\ell$ that bound facets of $\mathrm{conv}(S)$. Note that $v \in h_i^+$ for all $i = 1, \ldots, \ell$. [**Jared:**  *EXERCISE: What can be said about hyperplane $v^*$? What about the points $h_i^*$ and the halfspace $v^{*+}$? What does this say about $P^{\#}$ and $S^{\#}$*]

Then, hyperplane $v^*$ does not intersect any of the points $h_1^*, \ldots h_\ell^*$, and each point $h_i^*$ is in the open halfspace $v^{*+}$ by the inclusion reversing property. This shows that $P^{\#} = S^{\#}$ in the polar plane. (See Figure 9 for an example with point $d \in S - P$. Note that $d^*$ is a redundant halfspace in the intersection of planes in the polar space.)                                          □

Thus, the polar image $P^{\#}$ of a polytope is structurally isomorphic to $P$. Also, this proves that the convex hull problem is equivalent to the halfspace intersection problem in any dimension. In fact, once we have the incidence graph output from one of the problems, we can just flip that graph upside down to get the output of the other problem.

As an aside, note that we can talk about polytopes being polar duals of each other. For example, the cube and the octahedron are polar duals; the dodecahedron and icosohedron are polar duals, and the tetrahedron is self-dual.

## 5.4   Some Observations

**Incidence Graphs.**   Figure 10 illustrates an incidence graph for a simplex over 4 vertices in 3 dimensions. Each vertex in the top row is a 3-face (facet), defined by 3 of the 4 vertices. Each vertex in the next row is a 2-face, defined by 2 of the 4 vertices. Each vertex in the bottom row is a 1-face (i.e. point), defined by one vertex. An edge in the incidence graph connects two faces if one if the faces is included in the other.

Two observations. First, the incidence graph of the simplex in the polar plane, can be read bottom up by just taking the polar halfplane $v^*$ for each vertex $v$ in the incidence graph, and thinking of each face as the intersection of a collection of these halfplanes. Second, for a simplex, there are exactly $d + 1$ facets. But for an arbitrary polytope, that is the convex hull of $n$ points, there may be many more facets.
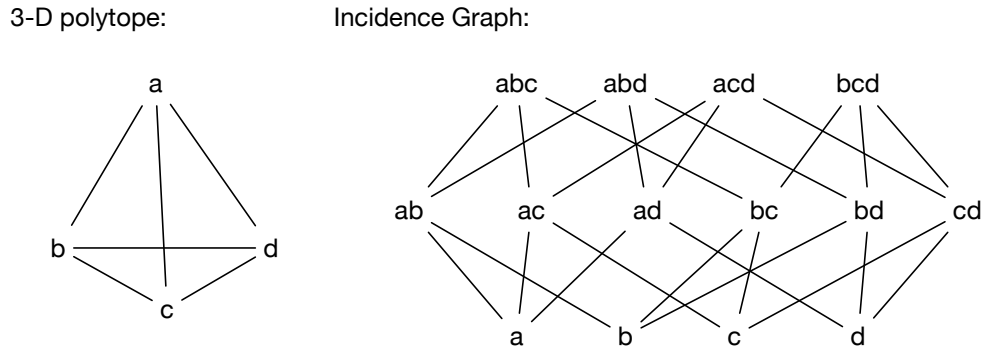
3-D polytope:             Incidence Graph:



**Figure 10.** Left: Polytope; Right: Incidence graph for all faces



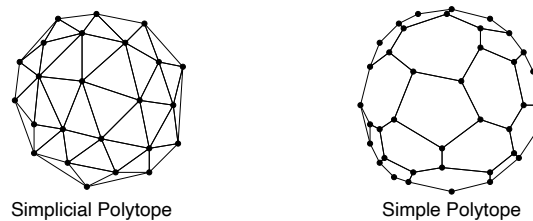Simplicial Polytope                Simple Polytope

**Figure 11.** The Simplicial (Simple) Polytope is the convex hull (intersection) of points (halfspaces) in general position.

**Simple and Simplicial Polytopes.** If a polytope is the convex hull of a set of points in $\mathbb{R}^d$ in general position, then for all $0 \leq j \leq d - 1$, each $j$-face is a $j$-simplex. Such a polytope is called *simplicial* (see Figure 11.)

In the dual view, consider a polytope that is the intersection of $n$ halfspaces in general position. Each $j$-face for $0 \leq j \leq d - 1$ is the intersection of exactly $d - j$ hyperplanes. Such a polytope is said to be *simple*. Note that in simple polytopes, each vertex is incident to exactly $d$ facets. Thus, the local region around any vertex is equivalent to a simplex.

Among all polytopes with a fixed number of vertices, simplicial polytopes maximize the number of facets. To see this, note that if there is a degeneracy (i.e. d+1 points on one facet), perturbing some point on this facet will break it into multiple facets. Dually, among all polytopes with a fixed number of facets, simple polytopes maximize the number of vertices.

## 5.5   How Many Facets?

So, how many facets are in a convex hull defined by $n$ points in $d$ dimensional space? The following theorem has a remarkably beautiful proof (also due to Seidel) that uses polar duality.

**Theorem 1.** *A polytope in $\mathbb{R}^d$ that is the convex hull of $n$ points has $O(n^{\lfloor d/2 \rfloor})$ facets. A polytope in $\mathbb{R}^d$ that is the intersection of $n$ halfspaces has $O(n^{\lfloor d/2 \rfloor})$ vertices.*

**Proof:** We will prove the polar form of the theorem. Consider a polytope defined by intersection of $n$ halfspaces in general position. By the discussion in the last section, this gives rise to a simple polytope, which maximizes the number of vertices. Suppose by convention that $x_d$ is the vertical axis. Then given a face, its *highest* and *lowest* vertices are defined as those having the maximum and minimum $x_d$ coordinates, respectively. (Note that there are no ties, assuming symbolic perturbation). Our proof is based on a charging argument. We place a charge at each
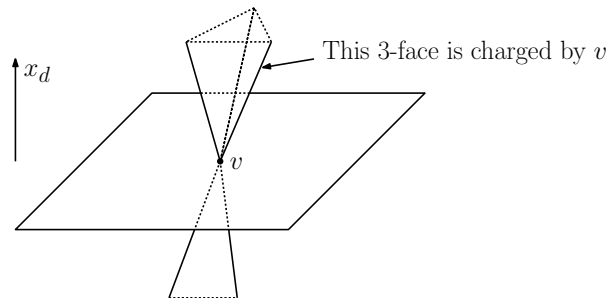
**Figure 12.**

vertex. We then move the charge at each vertex to a specially chosen incident face so that no face receives more than 2 charges.

Consider some vertex $v$. Note that there are $d$ edges (1-faces) that are incident to $v$ (See Figure 12 for example in $\mathbb{R}^5$). Consider a horizontal (i.e. orthogonal to $x_d$) hyperplane that passes through $v$. Note that at least $\lceil d/2 \rceil$ of the edges must lie on the same side of this hyperplane (again, use symbolic perturbation to ensure no two points have exactly the same $x_d$ coordinate, so none of these edges lie on this hyperplane).

Hence, there is a face of dimension at least $\lceil d/2 \rceil$ that spans these edges and is incident to $v$ (e.g. the 3-face above $v$ in Figure 12). So $v$ is either the highest or lowest vertex on this face. We move $v$'s charge to this face. Thus, we move the charge of every vertex to a face of dimension at least $\lceil d/2 \rceil$. Also, every such face will receive two such charges - one from its highest and one from its lowest vertex.

So how many charges are there in total? The number of $j$-faces is at most $\binom{n}{d-j}$, since each $j$ face is the intersection of $d - j$ halfspaces. Thus, the total number of charges is at most:

$$2 \sum_{j=\lceil d/2 \rceil}^{d-1} \binom{n}{d-j} = 2 \sum_{i=1}^{\lfloor d/2 \rfloor} \binom{n}{i}$$

$$\leq 2 \sum_{i=1}^{\lfloor d/2 \rfloor} n^i$$

$$= O(n^{\lfloor d/2 \rfloor})$$

The second step holds since $\binom{n}{x} \leq n^x$, because $\binom{n}{x}$ is the number of ways to choose x items from a set of size $n$ without replacement, and $n^x$ is the number of ways to choose $x$ items from a set of $n$ **with** replacement.

The last step holds since a geometric summation is asymptotically equal to its largest term. Finally, this argument shows that the number of vertices is $O(n^{\lfloor d/2 \rfloor})$, since each vertex initially had exactly one charge. Then the fact that there are $O(n^{\lfloor d/2 \rfloor})$ facets for a convex hull of $n$ points in $\mathbb{R}^d$ holds by polar duality, i.e. Lemma 5.     □

Is this bound tight? Yes. There is a family of polytopes called cyclic polytopes which match this asymptotic bound.

# References

[1] David Mount. Computational Geometry. http://www.cs.umd.edu/class/fall2016/cmsc754/Lects/cmsc754-fall16-lects.pdf, 2016.