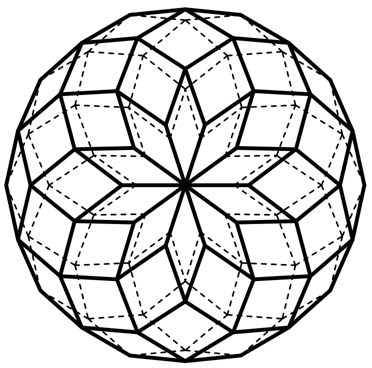
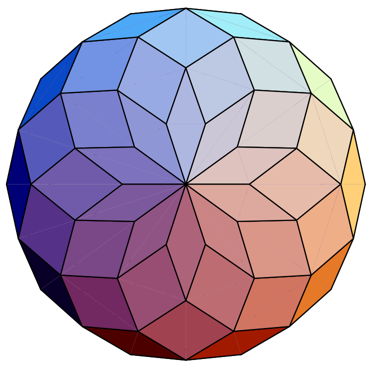
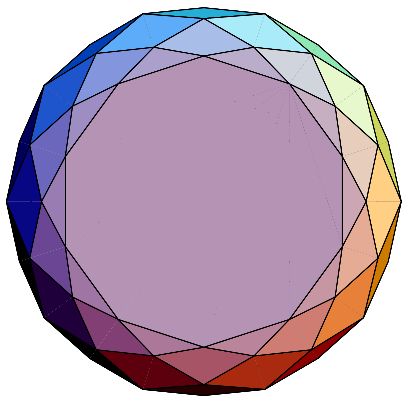
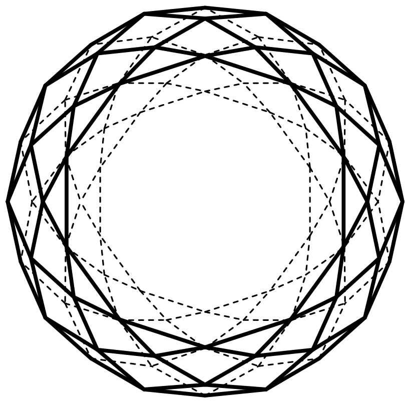
Lecture: Polyhedral Computation, Spring 2014

Komei Fukuda
Department of Mathematics, and
Institute of Theoretical Computer Science
ETH Zurich, Switzerland

February 18, 2014

Contents

1 Overview

Polyhedral computation deals with various computational problems associated with convex polyhedra in general dimension. Typical problems include the representation conversion problem (between halfspace and generator representations), the redundancy removal from representations, the construction of hyperplane arrangements and zonotopes, the Minkowski addition of convex polytopes, etc.

In this lecture, we study basic and advanced techniques for polyhedral computation in general dimension. We review some classical results on convexity and convex polyhedra such as polyhedral duality, Euler's relation, shellability, McMullen's upper bound theorem, the Minkowski-Weyl theorem, face counting formulas for arrangements. Our main goal is to investigate fundamental problems in polyhedral computation from both the complexity theory and the viewpoint of algorithmic design. Optimization methods, in particular, linear programming algorithms, will be used as essential building blocks of advanced algorithms in polyhedral computation. Various research problems, both theoretical and algorithmic, in polyhedral computation will be presented. The lecture consist of the following sections which are ordered in a way that the reader can follow naturally from top to bottom.

Lectures

(* planned.)

Case Studies

Matching Polytopes, Zonotopes and Hyperplane Arrangements, Bimatrix Games, Order Polytopes, Cyclic Polytopes, etc. Note that this part is not yet integrated into the main text. See the supplementary notes, "Case Study on Polytopes (for Polyhedral Computation)."

2 Integers, Linear Equations and Complexity

2.1 Sizes of Rational Numbers

Whenever we evaluate the complexity of an algorithm, we use the **binary encoding length of input data** as input size and we bound the number of arithmetic operations necessary to solve the worst-case problem instance of the same input size. Also, in order to claim a **polynomial complexity**, it is not enough that the number of required arithmetic operations is bounded by a polynomial function in the input size, but also, the largest size of numbers generated by the algorithm must be bounded by a polynomial function in the input size. Here we formally define the sizes of a rational number, a rational vector and a rational matrix.

Let $r = p/q$ be a rational number with canonical (i.e. relatively prime) representation with $p \in \mathbb{Z}$ and $q \in \mathbb{N}$. We define the *binary encoding size* of r as

$$\text{size}(r) := 1 + \lceil \log_2(|p| + 1) \rceil + \lceil \log_2(q + 1) \rceil. \tag{2.1}$$

The *binary encoding size* of a rational vector $v \in \mathbb{Q}^n$ and that of a rational matrix $A \in \mathbb{Q}^{m \times n}$ are defined by

$$\text{size}(v) := n + \sum_{j=1}^{n} \text{size}(v_i), \tag{2.2}$$

$$\text{size}(A) := mn + \sum_{i=1}^{m} \sum_{j=1}^{n} \text{size}(a_{ij}). \tag{2.3}$$

Exercise 2.1 For any two rational numbers r and s, show that

$$\text{size}(r \times s) \le \text{size}(r) + \text{size}(s),$$
$$\text{size}(r + s) \le 2(\text{size}(r) + \text{size}(s)).$$

Can one replace the constant 2 by 1 in the second inequality?

2.2 Linear Equations and Gaussian Elimination

Theorem 2.1 *Let A be a rational square matrix. Then the size of its determinant is polynomially bounded, and more specifically,* $\text{size}(\det(A)) < 2\,\text{size}(A)$.

Proof. Let p/q be the canonical representation of $\det(A)$, let p_{ij}/q_{ij} denote that of each entry a_{ij} of A, and let δ denote size(A).

First, we observe

$$q \le \prod_{i,j} q_{ij} < 2^{\delta - 1}, \tag{2.4}$$

where the last inequality can be verified by taking $\log_2$ of the both sides. By the definition of determinant, we have

$$|\det(A)| \le \prod_{i,j} (|p_{ij}| + 1). \tag{2.5}$$

Combining (2.4) and (2.5),

$$|p| = |\det(A)|q \le \prod_{i,j}(|p_{ij}|+1)q_{ij} < 2^{\delta-1}, \tag{2.6}$$

where the last inequality again is easily verifiable by taking $\log_2$ of both sides. Then it follows from (2.4) and (2.6) that

$$\mathrm{size}(\det(A)) = 1 + \lceil \log_2(|p|+1)\rceil + \lceil \log_2(q+1)\rceil \le 1 + (\delta-1) + (\delta-1) < 2\delta. \tag{2.7}$$

∎

Corollary 2.2 *Let A be a rational square matrix. Then the size of its inverse is polynomially bounded by its size* $\mathrm{size}(A)$.

Corollary 2.3 *If $Ax = b$, a system of rational linear equations, has a solution, it has one polynomially bounded by the sizes of $[A, b]$.*

Here is a theorem due to Jack Edmonds (1967).

Theorem 2.4 *Let $Ax = b$ be a system of rational linear equations. Then, there is a polynomial-time algorithm (based on the Gaussian or the Gauss-Jordan elimination) to find either a solution x or a certificate of infeasibility, namely, λ such that $\lambda^T A = \mathbf{0}$ and $\lambda^T b \neq 0$.*

Proof. Let Δ be the size of the matrix $[A, b]$. We need to show that the size of any number appearing in the Gaussian elimination is polynomially bounded by the size of input. Here we use the Gauss-Jordan elimination without normalization. Let $\widehat{A}$ be the matrix after applying it for k times, that is, we have (after applying possible row and column permutations),

$$\widehat{A} = \begin{array}{c|ccc|c|}
 & 1 & \cdots & k & s \\ \hline
1 & \hat{a}_{11} & 0\cdots & 0 & \\
\vdots & 0 & \ddots & 0 & \\
k & 0 & \cdots 0 & \hat{a}_{kk} & \\ \hline
 & 0 & \cdots & 0 & \\
r & 0 & \ddots & 0 & \hat{a}_{rs} \\
 & 0 & \cdots & 0 & \\ \hline
\end{array}. \tag{2.8}$$

Since we do not apply any normalization to nonzero diagonals, we have $\hat{a}_{ii} = a_{ii} \neq 0$ for all $i = 1, \dots, k$. Let $K = \{1, \dots, k\}$. We need to evaluate the sizes of all entries $\hat{a}_{rs}$ with $s > k$. For $r > k$, one can easily see

$$\hat{a}_{rs} = \frac{\det(\widehat{A}_{K\cup\{r\},K\cup\{s\}})}{\det(\widehat{A}_{K,K})} = \frac{\det(A_{K\cup\{r\},K\cup\{s\}})}{\det(A_{K,K})}, \tag{2.9}$$

where the last equation is valid because the elementary row operation without normalization does not change the value of the determinant of any submatrix of A containing K as row indices. It follows that $\mathrm{size}(\hat{a}_{rs}) < 4\Delta$. A similar argument yields $\mathrm{size}(\hat{a}_{rs}) < 5\Delta$ when $r \le k$. The same bounds for the size of the converted right hand side $\hat{b}_r$ follow exactly the same way. When the system has a solution, there is one defined by $x_i = \hat{b}_i / a_{ii}$ for $i = 1, \dots, k$ and $x_j = 0$ for $j > k$, at the last step k of the algorithm. Clearly, the size of this solution is polynomially bounded. When the system has no solution, there is a totally zero row $\widehat{A}_r$ and $\hat{b}_r \neq 0$. It is left for the reader to find an efficient way to find a succinct certificate λ. ∎

Exercise 2.2 Assuming that A, b are integer, revise the Gauss-Jordan algorithm so that it preserves the integrality of intermediate matrices and is still polynomial. (Hint: each row can be scaled properly.)

2.3 Computing the GCD

For given two positive integers a and b, the following iterative procedure finds the greatest common divisor (GCD):

```
procedure EuclideanAlgorithm($a$, $b$);
begin
    if $a < b$ then swap $a$ and $b$;
    while $b \neq 0$ do
    begin
        $a := a - \lfloor a/b \rfloor \times b$;
        swap $a$ and $b$;
    end;
    output $a$;
end.
```

Note that the algorithm works not only for integers but rational a and b.

Exercise 2.3 Apply the algorithm to $a = 212$ and $b = 288$.

Exercise 2.4 Explain why it is a correct algorithm for GCD. Analyze the complexity of the algorithm in terms of the sizes of inputs a and b. How can one extend the algorithm to work with k positive integers?

This algorithm does a little more than computing the GCD. By looking at the matrix operations associated with this, we will see that the algorithm does much more. Let's look at the two operations the algorithm uses in terms of matrices:

$$\begin{bmatrix} a, & b \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} b, & a \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} a, & b \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\lfloor a/b \rfloor, & 1 \end{bmatrix} = \begin{bmatrix} a - \lfloor a/b \rfloor \times b, & b \end{bmatrix}. \tag{2.11}$$

It is important to note that the two elementary transformation matrices

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{(Swapping)}, \qquad \begin{bmatrix} 1 & 0 \\ -\lfloor a/b \rfloor, & 1 \end{bmatrix} \text{(Remainder)} \tag{2.12}$$

are integer matrices and have determinant equal to $+1$ or -1. This means that the transformations preserve the existence of an integer solution to the following linear equation:

$$\begin{bmatrix} a, & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = c \quad \text{(i.e., } ax + by = c\text{)}. \tag{2.13}$$

Let $T \in \mathbb{Z}^{2\times 2}$ be the product of all transformation matrices occurred during the Euclidean algorithm applied to a and b. This means that $|\det T| = 1$ and

$$\begin{bmatrix} a, & b \end{bmatrix} T = \begin{bmatrix} a', & 0 \end{bmatrix}, \tag{2.14}$$

where a' is the output of the Euclidean algorithm and thus $\text{GCD}(a, b)$.

Now, we see how the algorithm finds the general solution to the linear diophantine equation:

$$\text{find } \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{Z}^2 \text{ satisfying } \begin{bmatrix} a, & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = c. \tag{2.15}$$

Once the transformation matrix T is computed, the rest is rather straightforward. Since T is integral and has determinant -1 or 1, the following equivalence follows.

$$\begin{aligned} &\left\langle \exists \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{Z}^2 : \begin{bmatrix} a, & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = c \right\rangle \Leftrightarrow \left\langle \exists \begin{bmatrix} x' \\ y' \end{bmatrix} \in \mathbb{Z}^2 : \begin{bmatrix} a, & b \end{bmatrix} T \begin{bmatrix} x' \\ y' \end{bmatrix} = c \right\rangle \\ \Leftrightarrow &\left\langle \exists \begin{bmatrix} x' \\ y' \end{bmatrix} \in \mathbb{Z}^2 : \begin{bmatrix} a', & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = c \right\rangle \Leftrightarrow \langle\, a' | c \;\; (a' \text{ divides } c) \rangle . \end{aligned}$$

Finally, when $a'|c$, let $x' := c/a'$ and y' be any integer. Then,

$$\begin{bmatrix} x \\ y \end{bmatrix} := T \begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} c/a' \\ y' \end{bmatrix} \tag{2.16}$$

with $y' \in \mathbb{Z}$ is the general solution to the diophantine equation (2.15).

2.4 Computing the Hermite Normal Form

By extending the Euclidean algorithm (in matrix form) to a system of linear equations in several integer variables, we obtain a procedure to solve the linear diophantine problem:

$$\text{find } x \in \mathbb{Z}^n \text{ satisfying } Ax = b, \tag{2.17}$$

where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$ are given. We assume that A is full row rank. (Otherwise, one can either reduce the problem to satisfy this condition or show that there is no $x \in \mathbb{R}^n$ satisfying $Ax = b$. How?)

Note that a seemingly more general problem of rational inputs A and b can be easily scaled to an equivalent problem with integer inputs.

Theorem 2.5 *There is a finite algorithm to find an $n \times n$ integer matrix T with $|\det T| = 1$ such that $A\,T$ is of form $[B\ \mathbf{0}]$, where $B = [b_{ij}]$ is an $m \times m$ nonnegative nonsingular lower-triangular integer matrix with $b_{ii} > 0$ and $b_{ij} < b_{ii}$ for all $i = 1, \dots, m$ and $j = 1, \dots, i-1$.*

This matrix $[B \ \mathbf{0}]$ is known as the *Hermite normal form*, and it will be shown that it is unique.

Corollary 2.6 *The linear diophantine problem (2.17) has no solution x if and only if there is $z \in \mathbb{Q}^m$ such that $z^T A$ is integer and $z^T b$ is fractional.*

Proof. The "if" part is trivial. To prove the "only if" part, we assume that $\nexists x \in \mathbb{Z}^n : A\,x = b$. Let T be the integer matrix given by Theorem 2.5. Because $|\det T| = 1$, we have the following equivalence:

$$\langle \nexists x \in \mathbb{Z}^n : A\,x = b \rangle \Leftrightarrow \langle \nexists x' \in \mathbb{Z}^n : A\,T\,x' = b \rangle \Leftrightarrow \langle \nexists x' \in \mathbb{Z}^n : [B\ \mathbf{0}]\,x' = b \rangle$$
$$\Leftrightarrow \left\langle B^{-1}b \text{ is not integer } \right\rangle.$$

Since $B^{-1}b$ is not integer, there is a row vector z^T of B^{-1} such that $z^T b$ is fractional. Since $B^{-1}A\,T = [I\ \mathbf{0}]$, we know that $B^{-1}A = [I\ \mathbf{0}]\,T^{-1}$ and it is an integer matrix as $|\det T| = 1$. This implies that $z^T A$ is integer. This completes the proof. ∎

As for the single diophantine equation, one can write the general solution to the linear diophantine problem (2.17).

Corollary 2.7 *Let $A \in \mathbb{Q}^{m\times n}$ be a matrix of full row rank, $b \in \mathbb{Q}^m$, and $A\,T = [B\ \mathbf{0}]$ be an Hermite normal form of A. Then the following statements hold.*

(a) *The linear diophantine problem (2.17) has a solution if and only if $B^{-1}b$ is integer.*

(b) *If $B^{-1}b$ is integer, then the general solution x to (2.17) can be written as*

$$x = T \begin{bmatrix} B^{-1}b \\ z \end{bmatrix}, \tag{2.18}$$

for any $z \in \mathbb{Z}^{n-m}$.

Now, we are going to prove the main theorem, Theorem 2.5.

Proof. (of Theorem 2.5). Extending the operations we used in (2.12), our proof of Theorem 2.5 involves three elementary matrix (column) operations on A:

(c-0) multiplying a column of A by -1;

(c-1) swapping the positions of two columns of A;

(c-2) adding an integer multiple of a column to another column of A.

Both (c-1) and (c-2) were already used and (c-0) is merely to deal with negative entries which are allowed in A. Each operation can be written in form $A\,T$, where T is a *unimodular* matrix, i.e., an integer matrix of determinant equal to -1 or 1.

The algorithm operates on the first row, the second and to the last row. We may assume that we have already transformed the first $k(\geq 0)$ rows properly, namely, we have a sequence of matrices T_1, T_2,..., T_s such that $T = T_1 T_2 \cdots T_s$ and

$$A_k := A\,T = \begin{bmatrix} B' & \mathbf{0} \\ C & A' \end{bmatrix} \tag{2.19}$$

where B' is a $k \times k$ matrix which is already in the form required for the final B, namely, it is a nonnegative nonsigular lower-triangular integer matrix with $b'_{ii} > 0$ and $b'_{ij} < b'_{ii}$ for all $i = 1, \ldots, k$ and $j = 1, \ldots, i-1$. Now we process the kth row of A_k, and essentially the first row of A'. The first row of A' contains $n-k$ integers and the rest is written by A'':

$$A' = \begin{bmatrix} a'_{11}, a'_{12}, \ldots, a'_{1(n-k)} \\ A'' \end{bmatrix} \tag{2.20}$$

Now we apply the Euclidean algorithm to find the GCD, say α, of the $n-k$ integers. For this, we first use (c-0) operations to make the numbers all nonnegative. Then remaining operations are straightforward with (c-1) and (c-2) to convert the first row to $[\alpha, 0, 0, \ldots, 0]$. This in the form of the whole matrix A_k looks like (for some T'),

$$A_k\, T' = \begin{bmatrix} B' & \mathbf{0} \\ C & \begin{matrix} \alpha, 0, 0, \ldots, 0 \\ A''' \end{matrix} \end{bmatrix}. \tag{2.21}$$

Note that α is strictly positive because of the full row rank assumption. Finally, we can reduce the entries in the first row of C by adding some integer multiples of α in the $(k+1)$st column (for some T'') as

$$A_k\, T'\, T'' = \begin{bmatrix} B' & \mathbf{0} \\ c'_{11}, \ldots, c'_{1k} & \alpha, 0, 0, \ldots, 0 \\ C'' & A''' \end{bmatrix}, \tag{2.22}$$

so that all entries $c'_{11}, \ldots, c'_{1k}$ are smaller than α. Now we have made the principal $(k+1) \times (k+1)$ matrix in the form of the final B. This completes the proof. ∎

While the algorithm is finite, it is not clear how good this is. It has been observed that the largest size of numbers appearing during the course of the algorithm grows rapidly. There are ways to make the procedure polynomial. We will discuss this issue later.

Example 2.1 The following small example is simple enough to calculate by hand.

$$A = \begin{bmatrix} -8 & 10 & -4 \\ -4 & -2 & 8 \end{bmatrix},$$
$$A_1 = \begin{bmatrix} 2 & 0 & 0 \\ 14 & 20 & -36 \end{bmatrix},$$
$$A_2 = [B\ \mathbf{0}] = \begin{bmatrix} 2 & 0 & 0 \\ 2 & 4 & 0 \end{bmatrix}.$$

The transformation matrix T with $A_2 = A\,T$ is

$$T = \begin{bmatrix} 6 & -2 & 9 \\ 7 & -2 & 10 \\ 5 & -1 & 7 \end{bmatrix}.$$

Example 2.2 The following small example (randomly generated) shows how numbers grow rapidly. Of course, this example is not meant to be computed by hand.

$$A = \begin{bmatrix} -100 & -32 & 140 & 168 & 147 \\ 68 & -16 & -125 & 168 & 7 \\ -12 & -28 & -50 & 147 & -133 \\ -60 & 64 & -65 & 28 & 28 \end{bmatrix},$$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -523 & 944 & 159 & 320 & 1976 \\ -115 & 604 & 54 & 151 & 388 \\ 976 & -2080 & -565 & -156 & -3652 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -1489619 & -2848 & -495 & 5739 & -1722 \\ -37305137 & -71331 & -6180 & 143636 & -29988 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ -299296004657 & -572624931 & -602688 & 309680 & 1400700 \end{bmatrix},$$

$$A_4 = [B\ \mathbf{0}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 43 & 129 & 12 & 140 & 0 \end{bmatrix}.$$

The transformation matrix T with $A_4 = A\,T$ is

$$T = \begin{bmatrix} -807814365429333 & -1545542680854 & -1626716396 & -377867 & 1101240 \\ -1448925874428057 & -2772142804282 & -2917738997 & -677754 & 1975225 \\ -731120268289411 & -1398808473625 & -1472275536 & -341992 & 996688 \\ -365381147997122 & -699061793372 & -735777339 & -170912 & 498100 \\ 248937455097979 & 476277073276 & 501291704 & 116444 & -339360 \end{bmatrix}.$$

Observation 2.8 *In the example above, the numbers appearing during the course of the algorithm seem to grow. This is a fact commonly observed. Yet, it is not known if our algorithm is exponential. There are ways to modify the algorithm so that it runs in polynomial-time.*

Exercise 2.5 Write a computer program to compute a Hermite normal form of any integer matrix A of full row rank. For this, one needs to use an environment where infinite precision integer arithmetic is supported, e.g., C/C++ with GNU gmp, Mathematica, Maple, and Sage.

2.5 Lattices and the Hermite Normal Form

There is a close relationship between the Hermite normal form of a matrix $A \in \mathbb{Q}^{m \times n}$ and the lattice generated by (the columns of) A. Recall that the lattice $L(A)$ generated by A is defined by

$$L(A) = \{y : y = Ax, x \in \mathbb{Z}^n\}. \tag{2.23}$$

The lattice $L(A)$ is *full dimensional* if it spans the whole space $\mathbb{R}^m$, or equivalently, A is full row rank.

Lemma 2.9 *Let A be rational matrix of full row rank with Hermite normal form $[B\ \mathbf{0}]$ Then, $L(A) = L([B\ \mathbf{0}])$.*

Proof. This follows directly from the fact that $A\,T = [B\ \mathbf{0}]$ for some unimodular matrix T. ∎

Theorem 2.10 *Let A and A' be rational matrices of full row rank with Hermite normal forms $[B\ \mathbf{0}]$ and $[B'\ \mathbf{0}]$, respectively. Then the matrices A and A' generate the same lattice (i.e. $L(A) = L(A')$) if and only if $B = B'$.*

Proof. Clearly, the sufficiency is clear: if $B = B'$, $L(A) = L(A')$.

Assume that $L := L(A) = L(A')$. Then, by Lemma 2.9, $L = L(B) = L(B')$. Now we show that the kth columns $B_{\cdot k}$ and $B'_{\cdot k}$ are equal for $k = 1, \ldots, n$. First, observe that $B_{\cdot k}$ and $B'_{\cdot k}$ are in L with the property (*) that the first $(k-1)$ components are all zero and the kth component is positive. Because B is in Hermite normal form, it follows that $B_{\cdot k}$ is a vector in L satisfying (*) with its kth component being smallest possible. Since the same thing can be said about $B'_{\cdot k}$, $b_{kk} = b'_{kk}$ for $k = 1, \ldots, n$. In addition, because B is in Hermite normal form, $B_{\cdot k}$ is a lexicographically smallest vector of nonnegative components satisfying (*), and so is $B'_{\cdot k}$. Such a vector is unique, and thus $B_{\cdot k} = B'_{\cdot k}$. ∎

Corollary 2.11 *Every rational matrix of full row rank has a unique Hermite normal form.*

A *basis* of a full dimensional lattice L(A) is a nonsingular matrix B such that $L(A) = L(B)$. A direct consequence of Theorem 2.5 is

Corollary 2.12 *Every rational matrix of full row rank has a basis.*

Exercise 2.6 Let A be a rational matrix of full row rank, let B be a bases of $L(A)$ and let B' be a nonsingular $n \times n$ matrix whose column vectors are points in $L(A)$. Show the following statements are valid:

(a) $|\det(B)| \leq |\det(B')|$.

(b) B' is a basis of $L(A)$ if and only if $|\det(B)| = |\det(B')|$.

Using the fact (a) above, we show that the size of the Hermite normal form is small.

Theorem 2.13 *The size of the Hermite normal form of a rational matrix A of full row rank is polynomially bounded by* size(A).

Proof. Let $[B\ \mathbf{0}]$ be the Hermite normal form of A, and let B' be any basis (i.e. nonsingular $m \times m$ submatrix) of A (which is not necessarily a basis of $L(A)$). First of all, $\det(B) > 0$. By Exercise 2.6 (a), we have $\det(B) \le |\det(B')|$. By Theorem 2.1, this inequality implies that the size of $\det(B)$ is polynomially bounded by size(A).

Since B is lower triangular, $\det(B)$ is the product of diagonal entries b_{ii}, $(i = 1, ..., n)$. It follows that the size of each entry b_{ii} is less than the size of $\det(B)$, and thus is polynomially bounded by size(A). Since B is in Hermite normal form, each nondiagonal entry b_{ij} is less than or equal to b_{ii} and has the same property. ∎

The theorem above suggests that the Hermite Normal Form might be computable in polynomial time. In fact, there are methods to control the largest size of numbers generated during the course of the algorithm given in the previous section.

One such algorithm is as follows. First of all, a given matrix A of full row rank, is enlarged to

$$\widehat{A} := \left[\begin{array}{c|cccc} & M & 0 & \cdots & 0 \\ A & 0 & & \ddots & 0 \\ & 0 & \cdots & 0 & M \end{array} \right], \tag{2.24}$$

where M is set to be the positive integer $|\det(B')|$ for an arbitrarily chosen basis B' of A. The first observation is that this new matrix generates the same lattice as A.

Exercise 2.7 Show that $L(\widehat{A}) = L(A)$.

Thus, computing the Hermite normal form of $\widehat{A}$ is equivalent to that of A. Now, since we have the added columns, it is possible to reduce the entries appearing in the first n columns by adding proper multiples of the last m columns, so that all entries are nonnegative and at most M. This reduction should be applied before the Euclidean algorithm is applied to each row. Since size(M) is polynomially bounded by size(A), one can control the number of arithmetic operations and the size of numbers appearing in the application of Euclidean algorithm applied to each row of $\widehat{A}$.

The sources of the ideas and more detailed discussion can be found in Shrijver's book [47, Section 5.3]. One important consequence of Theorem 2.13 , Corollary 2.3 and Theorem 2.4 is the following.

Corollary 2.14 *For any rational matrix A of full row rank, there is a transformation matrix T such that AT is the Hermite normal form of A and* size(T) *is polynomially bounded by the size of A. Furthermore, such a matrix can be computed in polynomial time.*

2.6 Dual Lattices

For a lattice L in $\mathbb{R}^m$, its *dual lattice* L^* is defined by

$$L^* := \{y \in \mathbb{Q}^m : y^T z \in \mathbb{Z}, \forall z \in L\}. \tag{2.25}$$

If L is generated by a matrix A,

$$L^* = (L(A))^* = \{y \in \mathbb{Q}^m : y^T A \in \mathbb{Z}^n\}. \tag{2.26}$$

In terms of the Hermite normal form $[B\ \mathbf{0}]$ of A (assuming A is full row rank), the dual lattice is generated by the transpose (and the rows) of B^{-1}, that is,

$$L^* = L((B^{-1})^T). \tag{2.27}$$

Why is it so? To see that, set $L' = L((B^{-1})^T)$ and we will show $L' = L^*$. Since $B^{-1}B = I$ and L is generated by (the columns of) B, each row of B^{-1} has an integer inner product with any vector in L. This shows $L' \subseteq L^*$. For the converse, take any vector y in L^*. Let $s^T = y^T B$. By definition of L^*, $s \in \mathbb{Z}^m$. Observing that $y^T = s^T B^{-1}$, y is an integer linear combination of the rows of B^{-1}. This proves $L' \supseteq L^*$ and completes the proof.

Example 2.3 Figure 2.1 left depicts the lattice generated by $A = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix}$. The Hermite normal form is $B = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$ and its inverse is $B^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{2}{3} & \frac{1}{3} \end{bmatrix}$. While the primal lattice $L(A)$ is generated by the columns of B, the dual lattice $(L(A))^*$ is generated by the rows of B^{-1}. Figure 2.1 right depicts the dual lattices (partially) on the same scale.

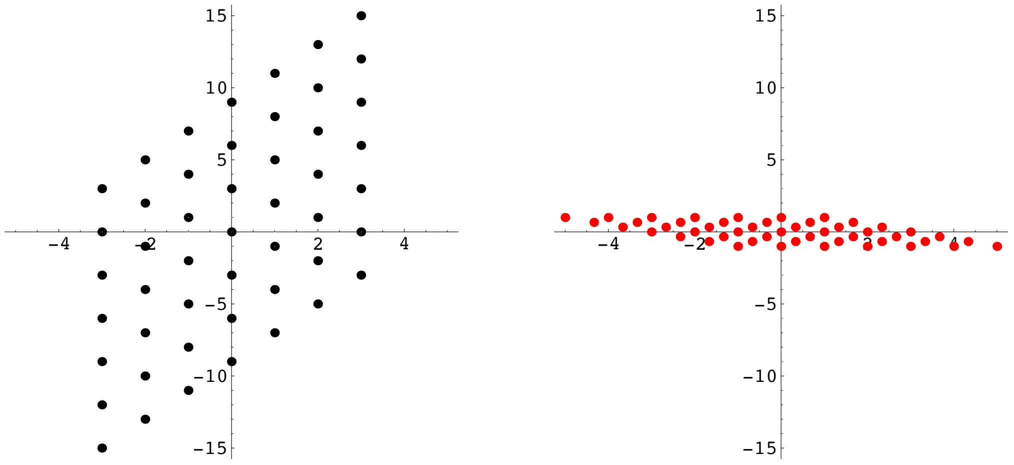


Figure 2.1: Lattice $L(A)$ and its Dual Lattice.

3 Linear Inequalities, Convexity and Polyhedra

3.1 Systems of Linear Inequalities

Consider a system of rational linear inequalities

$$Ax \le b, \tag{3.1}$$

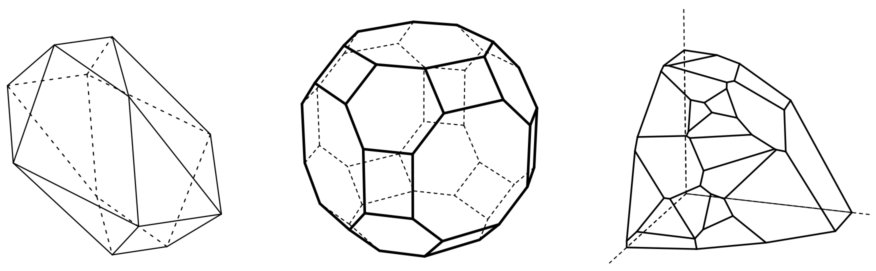
where $A \in \mathbb{Q}^{m\times d}$ and $b \in \mathbb{Q}^m$ are given. The set

$$P = P(A,b) := \{x \in \mathbb{R}^d \ : \ Ax \le b\} \tag{3.2}$$

of solutions to the system is a subset of $\mathbb{R}^d$, known as a *convex polyhedron*. It is in fact a convex set: a subset C of $\mathbb{R}^d$ is said to be *convex* if the line segment $[u, v] := \{x : x = \alpha u + (1-\alpha)v, 0 \le \alpha \le 1\}$ between any two points u and v in C is entirely contained in C. A bounded convex polyhedron is called a *convex polytope*.

A point x is called an *extreme* point of a convex set C if $x \in C$ and x is not on the line segment between any two points u, v in C different from x. Unlike general convex sets, a convex polyhedron P contains only a finite number of extreme points. This will be shown in a more general form in Theorem 3.14

Below, the first two are centrally symmetric polytopes in $\mathbb{R}^3$, and the third one is randomly generated. One can interpret the third one as a bounded polyhedron (i.e. polytope) contained in the nonnegative orthant or as an unbounded polyhedron having only its nonnegative orthant part drawn.



3.2 The Fourier-Motzkin Elimination

Consider a system (3.1) of m linear inequalities in n variables. Solving such a system means either to find a rational vector x satisfying the system or to detect inconsistency of the system. The latter can be proven by a certificate given by the well-known theorem of Gale:

Theorem 3.1 (Gale's Theorem) *For any $A \in \mathbb{R}^{m\times d}$ and $b \in \mathbb{R}^m$, exactly one of the following statements holds:*

(a) *there exists $x \in \mathbb{R}^d$ such that $Ax \le b$;*

(b) *there exists $z \in \mathbb{R}^m$ such that $z \ge \mathbf{0}$, $z^T A = \mathbf{0}$ and $z^T b < 0$.*

It is easy to see that both statements cannot hold simultaneously as it would mean 0 is less than or equal to some (strictly) negative number. Thus the nontrivial part of the theorem is that one of (a) and (b) is always valid. There are several constructive proofs of the theorem.

Here we present an arguably simplest constructive proof due to Fourier and Motzkin. The main idea is to transform the system (3.1) to an equivalent system of the same form with one less variables.

First we rewrite the system (3.1) by looking at the coefficients a_{id}'s for the last variable x_d. Let us define a partition of row indices into the three sets:

$$I^+ := \{i \mid a_{id} > 0\},\ I^- := \{i \mid a_{id} < 0\} \text{ and } I^0 := \{i \mid a_{id} = 0\}.$$

The system (3.1) can be rewritten by solving each inequality with respect to x_d:

$$\begin{aligned} x_d &\le f_i(x') && \forall i \in I^+ \\ g_j(x') &\le x_d && \forall j \in I^- \\ h_k(x') &\le 0 && \forall k \in I^0, \end{aligned}$$

where x' is the vector x with the last component eliminated, i.e. $x' = (x_1, \ldots, x_{d-1})^T$ and each functions f_i, g_j and h_k denote some affine functions in $d-1$ variables.

It is not difficult to show (Exercise 3.1) that the system (3.1) is equivalent to the new system in $d-1$ variables:

$$\begin{aligned} g_j(x') &\le f_i(x') && \forall (i,j) \in I^+ \times I^- \\ h_k(x') &\le 0 && \forall k \in I^0. \end{aligned}$$

This system can thus be written as

$$A'x' \le b'. \tag{3.3}$$

Exercise 3.1 Prove the equivalence: $Ax \le b \Leftrightarrow A'x' \le b'$.

No reason to stop here. Let's continue to eliminate the variable x_{d-1}, then x_{d-2} and so forth until all variables are eliminated. This generates a sequence of equivalent systems of linear inequalities:

$$\begin{array}{l} A^{(0)}x^{(0)} \le b^{(0)} \text{ (This is the original system } Ax \le b.\text{)} \\ \qquad\quad \Updownarrow \\ A^{(1)}x^{(1)} \le b^{(1)} \text{ (This is } A'x' \le b' \text{ above.)} \\ \qquad\quad \Updownarrow \\ A^{(2)}x^{(2)} \le b^{(2)} \\ \qquad\quad \Updownarrow \\ \qquad\quad \vdots \\ \qquad\quad \Updownarrow \\ A^{(d)}x^{(d)} \le b^{(d)}, \end{array}$$

where $A^{(k)}x^{(k)} \le b^{(k)}$ denotes the kth system where the last k variables have been eliminated.

Exercise 3.2 Show that the elimination step as a matrix transformation. More precisely, there is a matrix T (depending on A) such that the system $A'x' \le b'$ is the same system as $TAx \le Tb$ up to positive scaling. Note that the last column of the product TA is totally zero and thus TAx does not involve the last variable x_d.

By the exercise above, the last system $A^{(d)}x^{(d)} \le b^{(d)}$ can be now written as $T^{(d)}Ax \le T^{(d)}b$. Of course, the left hand side $T^{(d)}Ax$ is a vector of zero's.

Exercise 3.3 Prove Theorem 3.1 by using the matrix $T^{(d)}$.

Exercise 3.4 Prove the following forms of alternative theorem using Gale's Theorem. Below, the statement is read as "exactly one of the two statements (a) or (b) holds."

The Farkas Lemma	**Gordan's Theorem**
(a) $\exists x : A\,x = b$ and $x \ge \mathbf{0}$;	**(a)** $\exists x : A\,x = \mathbf{0}$ and $x \gneq \mathbf{0}$;
(b) $\exists z : z^T A \ge \mathbf{0}$ and $z^T b < 0$.	**(b)** $\exists z : z^T A > \mathbf{0}$.

3.3 LP Duality

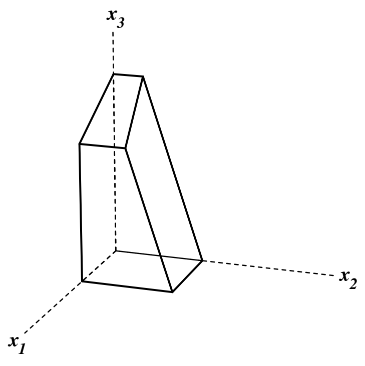
For a given $A \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^d$, the linear programming problem (in canonical form) is

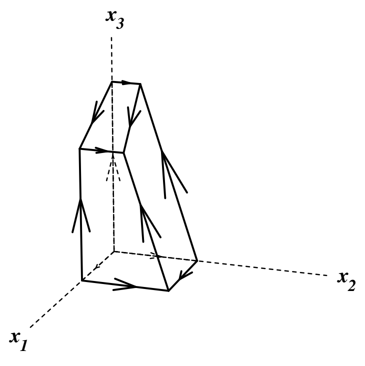
$$\begin{array}{lllcl}
\text{(P):} & \max & c^T x & \Big| & = \sum_{j=1}^d c_j\, x_j \\
 & \text{subject to} & A\,x \le b & \Big| & \sum_{j=1}^d a_{ij}\, x_j \le b_i, \forall i = 1, \ldots, m \\
 & & x \ge \mathbf{0}. & \Big| & x_j \ge 0, \forall j = 1, \ldots, d.
\end{array}$$

We often abbreviate a linear programming problem as an *LP*. A vector x satisfying all the constraints $Ax \le b$ and $x \ge \mathbf{0}$ is called a *feasible solution*. An *optimal solution* is a feasible solution that attains the largest objective value. In the case of minimization problem, an optimal solution attains the smallest value.

The set of feasible solutions $\{x : Ax \le b, x \ge 0\}$ is called the *feasible region*. An LP is called *feasible* if the feasible region is not empty. An LP is called *unbounded* if the objective function $c^T x$ is not bounded above (below for the minimization case) over the feasible region.

Geometrically, the feasible region is a *convex polyhedron*.





In general, maximizing or minimizing a linear function subject to a system of linear inequality constraints in d variables can be reduced to an optimization in the form above. Also, note that no strict inequality constraint such as $x_1 > 0$ is allowed in linear programming.

There are two fundamental theorems, the weak duality theorem and the strong duality theorem. To state these theorems, we need to define the *dual problem*

$$\begin{array}{lll} \text{(D):} & \min & b^T y \\ & \text{subject to} & A^T y \geq c \\ & & y \geq \mathbf{0} \end{array}$$

which is a linear programming problem itself. The original problem (P) is called the *primal problem* when we need to distinguish it from the dual problem.

Theorem 3.2 (LP Weak Duality Theorem) *For any feasible solution x for the primal problem (P) and for any feasible solution y for the dual problem (D), $c^T x \leq b^T y$.*

Theorem 3.3 (LP Strong Duality Theorem) *If both the primal problem (P) and the dual problem (D) are feasible, there exist a dual pair (x^*, y^*) of feasible solutions such that $c^T x^* = b^T y^*$. (By the previous theorem, they are both optimal.)*

The first theorem is very easy to prove. Thus it may not be appropriate to call it a theorem, but since it is widely accepted to be so called. Let's prove it.

Proof. (of the Weak Duality Theorem, Theorem 3.2) Let x and y be a dual pair of feasible solutions. Then,

$$\begin{aligned} c^T x &\leq (A^T y)^T x && \text{(because } A^T y \geq c \text{ and } x \geq \mathbf{0}\text{)} \\ &= y^T A x \\ &\leq y^T b && \text{(because } Ax \leq b \text{ and } y \geq \mathbf{0}\text{)} \\ &= b^T y. \end{aligned}$$

This completes the proof. ∎

Now, we are ready to prove the second theorem which is much harder to prove.

Proof. (of the Strong Duality Theorem, Theorem 3.3) Assume that both the primal problem (P) and the dual problem (D) are feasible. We have to show that

$$\begin{aligned} \exists (x, y) : & Ax \leq b, x \geq \mathbf{0} \\ & A^T y \geq c, y \geq \mathbf{0} \\ & c^T x = b^T y. \end{aligned} \tag{3.4}$$

Now, we verify the statement (3.4) is always valid under the assumption.

$$
\begin{aligned}
(3.4) \Leftrightarrow\ & \left\langle \begin{array}{ll} \exists(x,y): & Ax \le b, x \ge \mathbf{0} \\ & A^T y \ge c, y \ge \mathbf{0} \\ & c^T x \ge b^T y \end{array} \right\rangle \qquad \text{(by the Weak Duality)} \\
\Leftrightarrow\ & \left\langle \exists \begin{bmatrix} x \\ y \end{bmatrix} \ge \mathbf{0} : \begin{bmatrix} A & \mathbf{0} \\ -I & \mathbf{0} \\ \mathbf{0} & -A^T \\ \mathbf{0} & -I \\ -c^T & b^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} b \\ \mathbf{0} \\ -c \\ \mathbf{0} \\ 0 \end{bmatrix} \right\rangle \\
\Leftrightarrow\ & \left\langle \nexists \begin{bmatrix} s \\ t \\ u \\ v \\ w \end{bmatrix} \ge \mathbf{0} : \begin{bmatrix} s \\ t \\ u \\ v \\ w \end{bmatrix}^T \begin{bmatrix} A & \mathbf{0} \\ -I & \mathbf{0} \\ \mathbf{0} & -A^T \\ \mathbf{0} & -I \\ -c^T & b^T \end{bmatrix} = \mathbf{0} \text{ and } \begin{bmatrix} s \\ t \\ u \\ v \\ w \end{bmatrix}^T \begin{bmatrix} b \\ 0 \\ -c \\ 0 \\ 0 \end{bmatrix} < 0 \right\rangle \text{(by Gale's Thm)} \\
\Leftrightarrow\ & \left\langle \nexists \begin{bmatrix} s \\ u \\ w \end{bmatrix} \ge \mathbf{0} : A^T s \ge cw, Au \le bw, b^T s < c^T u \right\rangle \\
\Leftrightarrow\ & \left\langle \nexists \begin{bmatrix} s \\ u \end{bmatrix} \ge \mathbf{0} : A^T s \ge \mathbf{0}, Au \le \mathbf{0}, b^T s < c^T u \right\rangle \qquad \text{(by the Weak Duality)} \\
\Leftrightarrow\ & \left\langle\ A^T s \ge \mathbf{0}, Au \le \mathbf{0}, s \ge \mathbf{0}, u \ge \mathbf{0} \Rightarrow b^T s \ge c^T u\ \right\rangle .
\end{aligned}
$$

Now the last step of the proof is to show the last statement above is always true which implies the theorem. Assume

$$A^T s \ge \mathbf{0}, Au \le \mathbf{0}, s \ge \mathbf{0}, u \ge \mathbf{0}.$$

By the assumption, we have a dual pair (x, y) of feasible solutions. Thus, we have

$$b^T s - c^T u \ge (Ax)^T s - (A^T y)^T u = x^T A^T s - y^T A u \ge 0 - 0 = 0.$$

This completes the proof. ∎

Theorem 3.4 (Complementary Slackness Conditions) *For a dual pair* $(\overline{x}, \overline{y})$ *of feasible solutions for (P) and (D), the following conditions are equivalent:*

(a) *both* $\overline{x}$ *and* $\overline{y}$ *are optimal solutions;*

(b) $c^T \overline{x} = b^T \overline{y}$*;*

(c) $\overline{y}^T (b - A\overline{x}) = 0$ *and* $\overline{x}^T (A^T \overline{y} - c) = 0$.

(c') $\overline{y}_i (b - A\overline{x})_i = 0$ *for all* i *and* $\overline{x}_j (A^T \overline{y} - c)_j = 0$ *for all* j.

(c") $\overline{y}_i > 0$ *implies* $(b - A\overline{x})_i = 0$*, for all* i *and*
$\overline{x}_j > 0$ *implies* $(A^T \overline{y} - c)_j = 0$*, for all* j.

Proof. Left to the reader. ∎

Exercise 3.5 Write the Complementary Slackness Conditions (Theorem 3.4) for the LP of form $\max c^T x$ subject to $Ax \le b$ and its dual LP, and prove the validity.

3.4 Three Theorems on Convexity

Before we discuss the theory of representations and combinatorial structure of convex polyhedron, it is good to mention some basic facts about convexity.

For any subset S of $\mathbb{R}^d$, the *convex hull* $\mathrm{conv}(S)$ of S is the intersection of all convex sets containing S. Since the intersection of two convex sets is convex, it is the smallest convex set containing S.

Proposition 3.5 *Let S be a subset $\mathbb{R}^d$. Then*

$$\begin{aligned} \mathrm{conv}(S) = \{x \ : x = \sum_{i=1}^{k} \lambda_i p_i, \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0 \ \forall i = 1, \ldots, k, \\ \textit{for some finite points } p_1, \ldots, p_k \in S\}. \end{aligned} \tag{3.5}$$

Proof. Let the RHS of (3.5) be T. One has to show both inclusions $\mathrm{conv}(S) \supseteq T$ and $\mathrm{conv}(S) \subseteq T$. Both inclusions are elementary and left to the reader. ∎

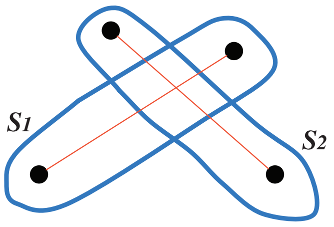
One basic theorem on convexity is Carathéodory's theorem, saying that the finiteness condition on k in (3.5) can be much more restrictive, namely, $k \leq d + 1$.

Theorem 3.6 (Carathéodory's Theorem) *Let a point p be in the convex hull of a set S of k points $p_1, \ldots, p_k$ in $\mathbb{R}^d$. Then p is in the convex hull of at most $d + 1$ points in S.*

Proof. Left to the reader. Hint: When $k \geq d + 2$, the points $p_1, \ldots, p_k$ are *affinely dependent*, i.e., there exist $\alpha_1, \ldots, \alpha_k$ not all zero such that $\sum_i \alpha_i = 0$ and $\sum_i \alpha_i p_i = \mathbf{0}$. Use this to show that at least one point in S is unnecessary to represent x. ∎

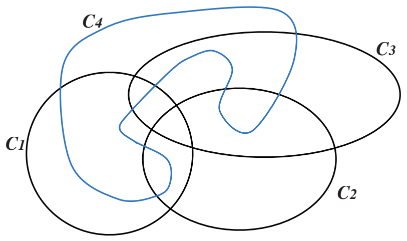
Here are two more basic theorems on convexity.

Theorem 3.7 (Radon's Theorem) *Let S be a subset of of $\mathbb{R}^d$ with $|S| \geq d + 2$. Then S can be partitioned into two sets S_1 and S_2 so that $\mathrm{conv}(S_1) \cap \mathrm{conv}(S_2) \neq \emptyset$.*



Proof. Since $|S| \geq d + 2$, the points in S are affinely dependent. Use this fact to find a natural partition. ∎

Theorem 3.8 (Helly's Theorem) *Let $C_1, C_2, \ldots, C_h$ be convex sets in $\mathbb{R}^d$ such that $C_1 \cap C_2 \cap \cdots \cap C_h = \emptyset$. Then, there are at most $d + 1$ of them whose intersection is empty.*



Proof. (Clearly, the convexity assumption on C_j's is important as the theorem fails with only one nonconvex C_j above.) Use induction on h. If $h \leq d+1$, the theorem is trivial. Assume that the theorem is true for $h < k(\geq d+2)$ and prove (*) the theorem holds for $h = k$. Note that $h \geq d+2$. Suppose the statement (*) does not hold, namely, $S_j := \cap_{j \neq i} C_j \neq \emptyset$ for all $j = 1, \ldots, h$. Apply Radon's theorem to get a contradiction. ∎

3.5 Representations of Polyhedra

For a set $\{v_1, \ldots, v_k\}$ of vectors in $\mathbb{R}^d$, define their *cone* (or *nonnegative*) *hull* as

$$\text{cone}(\{v_1, \ldots, v_k\}) := \{x : x = \sum_i \lambda_i v_i, \lambda_i \geq 0 \; \forall i = 1, \ldots, k\}. \tag{3.6}$$

For subsets P and Q of $\mathbb{R}^d$, their *Minkowski sum* $P + Q$ is defined as

$$P + Q := \{p + q : p \in P \text{ and } q \in Q\}. \tag{3.7}$$

Theorem 3.9 (Minkowski-Weyl's Theorem for Polyhedra) *For $P \subseteq \mathbb{R}^d$, the following statements are equivalent:*

(a) *P is a polyhedron, i.e., there exist $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ for some m such that $P = \{x : Ax \leq b\}$;*

(b) *P is finitely generated, i.e., there exist (finite) vectors v_i's and r_j's in $\mathbb{R}^d$ such that $P = \text{conv}(\{v_1, \ldots, v_s\}) + \text{cone}(\{r_1, \ldots, r_t\})$.*

The statement (b) above can be written in matrix form as follows. Here, $\mathbf{1}$ denotes a vector of all 1s.

(b) *P is finitely generated, i.e., there exist two matrices $V \in \mathbb{R}^{d \times s}$ and $R \in \mathbb{R}^{d \times t}$ for some s and t such that $P = \{x : x = V\mu + R\lambda, \mu \geq \mathbf{0}, \mathbf{1}^T\mu = 1, \lambda \geq \mathbf{0}\}$.*

Theorem 3.9 actually consists of two theorems. The direction from (a) to (b) is Minkowski's Thoerem, while the reverse direction from (b) to (a) is Weyl's Theorem.

When a polyhedron P is bounded (thus a polytope), the minimal representation consists of all extreme points $v_1, \ldots, v_s$ and no rays. Another special case of $b = \mathbf{0}$ leads to a homogeneous version of the theorem. It is a special case but it is actually as powerful as the nonhomogeneous version above (Exercise 3.6).

Theorem 3.10 (Minkowski-Weyl's Theorem for Cones) *For $P \subseteq \mathbb{R}^d$, the following statements are equivalent:*

(a) *P is a polyhedral cone, i.e., there exist $A \in \mathbb{R}^{m\times d}$ for some m such that* $P = \{x : Ax \leq \mathbf{0}\}$;

(b) *P is a finitely generated cone, i.e., there exists a matrix $R \in \mathbb{R}^{d\times t}$ for some t such that* $P = \{x : x = R\lambda, \lambda \geq \mathbf{0}\}$.

We first show one direction which follows almost immediately by the Fourier-Motzkin elimination.

Proof. (for Theorem 3.10 (b) $\Longrightarrow$ (a)). Assume that P is a finitely generated cone and there exists a matrix $R \in \mathbb{R}^{d\times t}$ such that $P = \{x : x = R\lambda, \lambda \geq \mathbf{0}\}$. The conditions $x = R\lambda, \lambda \geq \mathbf{0}$ can be considered a system of linear inequalities in variables x and λ. Thus one can apply the Fourier-Motzkin elimination to eliminate all variables $\lambda_1, \ldots, \lambda_t$ from this system. The result is an equivalent system of inequalities in x variables. This is a representation of form (a). ∎

Let us say that a pair (A, R) of matrices is a *double description pair* or simply a *DD-pair* if they represent the same polyhedron, namely,

$$Ax \leq \mathbf{0} \;\Leftrightarrow\; x = R\lambda, \text{ for some } \lambda \geq \mathbf{0}. \tag{3.8}$$

With this language, the Minkowski theorem says for any matrix A, there exists R such that (A, R) is a DD-pair. The Weyl theorem states that for any R, there exists A such that (A, R) is a DD-pair.

Lemma 3.11 *For two matrices $A \in \mathbb{R}^{m\times d}$ and $R \in \mathbb{R}^{d\times t}$, the pair (A, R) is a DD-pair if and only if (R^T, A^T) is a DD-pair.*

Proof. Because of symmetry, we only need to show one direction. Assume the pair (A, R) is a DD-pair, namely (3.8) is valid. Now we have to show (R^T, A^T) is also a DD-pair. Now we have a sequence of equivalences

$$\begin{aligned}
&R^T y \leq \mathbf{0} \\
&\quad \Leftrightarrow \lambda^T R^T y \leq \mathbf{0}, \forall \lambda \geq \mathbf{0} \\
&\quad \Leftrightarrow (R\lambda)^T y \leq \mathbf{0}, \forall \lambda \geq \mathbf{0} \\
&\quad \Leftrightarrow Ax \leq \mathbf{0} \text{ implies } x^T y \leq 0 && \text{(by the assumption (3.8))} \\
&\quad \Leftrightarrow \nexists x : Ax \leq \mathbf{0} \text{ and } y^T x > 0 \\
&\quad \Leftrightarrow y = A^T\mu, \text{ for some } \mu \geq \mathbf{0} && \text{(by Farkas' Lemma).}
\end{aligned}$$

The equivalence of the first and the last statement is exactly what we needed to prove. ∎

This lemma has a very useful consequence in computation, namely, no one needs to implement both transformations between (a) and (b) but only one.

Proof. (for Theorem 3.10). We have already proved Weyl's theorem. On the other hand, Lemma 3.11 says that showing one direction is sufficient to prove both directions. This completes the proof. ∎

As Lemma 3.11 indicates, there is a polyhedron associated with the pair (R^T, A^T). Namely, if (A, R) is a DD-pair, the polyhedral cone

$$P^* := \{y \in \mathbb{R}^d \;:\; R^T y \leq \mathbf{0}\} \tag{3.9}$$
$$= \{y \in \mathbb{R}^d \;:\; y = A^T \mu, \mu \geq \mathbf{0}\} \tag{3.10}$$

is known as the *dual* or the *dual cone* of $P = \{x : Ax \leq \mathbf{0}\} = \{x : x = R\lambda, \lambda \geq \mathbf{0}\}$.

Exercise 3.6 Derive the nonhomogeneous Theorem 3.9 from the homogeneous Theorem 3.10. Hint: Homogenize a given nonhomogeneous system with an extra dimension, convert it by the homogeneous theorem, and then get a nonhomogeneous representation.

The Fourier-Motzkin elimination is not practical for converting between two representations of polyhedra, due to the explosion of the size of intermediate systems. Methods known as the *double description method* and the *reverse search method* are both much more practical and used in many existing implementations (e.g., LRSLIB, CDDLIB).

3.6 The Structure of Polyhedra

For a nonempty polyhedron P in $\mathbb{R}^d$, we define two sets the *linearity space* and the *recession cone*.

$$\text{lin. space}(P) := \{z : x + \lambda z \in P, \; \forall x \in P \text{ and } \forall \lambda \in \mathbb{R}\} \tag{3.11}$$
$$\text{rec. cone}(P) := \{z : x + \lambda z \in P, \; \forall x \in P \text{ and } \forall \lambda \geq 0\}. \tag{3.12}$$

The recession cone is also known as the *characteristic cone*. Both sets contain the origin, and in general $\text{lin. space}(P) \subseteq \text{rec. cone}(P)$.

A polyhedron P is called *pointed* if it contains an extreme point. Here are some structural properties of polyhedra.

Theorem 3.12 *Let be P be a nonempty polyhedron in $\mathbb{R}^d$, the following statements hold:*

(a) *If P is written as $P = Q + C$ for some polytope Q and some polyhedral cone C, then $C = \text{rec. cone}(P)$.*

(b) *If P is represented as $P = \{x : Ax \leq b\}$, then $\text{rec. cone}(P) = \{z : Az \leq \mathbf{0}\}$ and $\text{lin. space}(P) = \{z : Az = \mathbf{0}\}$;*

(c) *P is pointed if and only if $\text{lin. space}(P)$ is trivial, i.e., $\text{lin. space}(P) = \{\mathbf{0}\}$;*

(d) *P is bounded if and only if $\text{rec. cone}(P)$ is trivial, i.e., $\text{rec. cone}(P) = \{\mathbf{0}\}$.*

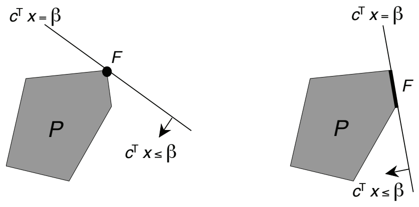
Proof. Left to the reader. ∎

The statement (a) of the theorem above implies that in the generator reporesentation in Minkowski-Weyl's Theorem, Theorem 3.9 (b), the cone part $\text{cone}(\{r_1, \ldots, r_t\})$ is unique while the convex hull part $\text{conv}(\{v_1, \ldots, v_s\})$ is clearly not.

Corollary 3.13 *If P is a cone $\{x : Ax \leq \mathbf{0}\}$ and pointed, then there exists a vector c such that $c^T x > 0$ for all nonzero $x \in P$.*

Proof. Set $c^T = -\mathbf{1}^T A$. Show that this vector satisfies $c^T x > 0$ for all nonzero $x \in P$ (Exercise). ∎

For $c \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$, an inequality $c^T x \leq \beta$ is called *valid* for a polyhedron P if $c^T x \leq \beta$ holds for all $x \in P$. A subset F of a polyhedron P is called a *face* of P if it is represented as $F = P \cap \{x : c^T x = \beta\}$ for some valid inequality $c^T x \leq \beta$.



Note that both $\emptyset$ and P are faces, called *trivial* faces. The faces of dimension 0 are called *vertices* and the faces of dimension $\dim(P) - 1$ are called *facets*. The faces of dimension i are called the *i-faces*.

The first important fact on faces is that there are only finitely many of them. It follows from the following.

Theorem 3.14 *Let $P = \{x \in \mathbb{R}^d : Ax \leq b\}$. Then a nonempty subset F of P is a face of P if and only if F is represented as the set of solutions to an inequality system obtained from $Ax \leq b$ by setting some of the inequalities to equalities, i.e.,*

$$F = \{x : A^1 x = b^1 \text{ and } A^2 x \leq b^2\}, \tag{3.13}$$

where $A = \begin{bmatrix} A^1 \\ A^2 \end{bmatrix}$ *and* $b = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix}$.

Proof. Let F be a nonempty face. Then, $F = P \cap \{x : c^T x = \beta\}$ for some valid inequality $c^T x \leq \beta$. The set F is the set of optimal solutions to the LP: max $c^T x$ subject to $Ax \leq b$. Since the LP has an optimal solution, the dual LP: $\min b^T y$ subject to $A^T y = c, y \geq \mathbf{0}$ has an optimal solution, say y^*, by the strong duality, Theorem 3.3. Then, put $A_i x \leq b_i$ to be in the equality part $A^1 x = b^1$ if and only if $y_i^* > 0$. Then the resulting set $F' = \{x : A^1 x = b^1 \text{ and } A^2 x \leq b^2\}$ coincides with F, since F' is the set of points in P satisfying the complementary slackness conditions, see Theorem 3.4 and Exercise 3.5.

The converse follows immediately by setting $c^T = \mathbf{1}^T A^1$ and $\beta = \mathbf{1}^T b^1$, for a nonempty set F of form (3.13). ∎

Corollary 3.15 *Every minimal nonempty face of $P = \{x \in \mathbb{R}^d : Ax \leq b\}$ is an affine subspace of form $\{x : A^1 x = b^1\}$ where $A^1 x = b^1$ is a subsystem of $Ax = b$.*

Proof. By Theorem 3.14, every nonempty face F has a representation of form

$$F = \{x : A^1 x = b^1 \text{ and } A^2 x \leq b^2\}.$$

Assume F is minimal. Set $F' = \{x : A^1x = b^1\}$. We will show that $F = F'$. We claim that the inequality part $A^2x \le b^2$ must be redundant in the representation of F. Suppose some of the inequalities can be violated by a point in F'. Then, F' is not a minimal nonempty face (why?), a contradiction. ∎

Exercise 3.7 Show that the vertices of a polyhedron are exactly the extreme points.

Corollary 3.16 *Let $P = \{x : Ax \le b\}$ be a rational polyhedron. Then, every nonempty face of P contains a point of size polynomially bounded by the largest size of rows of $[A, b]$.*

Proof. Let δ denote the largest size of rows of $[A, b]$. It is enough to show the claim for every nonempty minimal face of P. By Corollary 3.15, every nonempty minimal face is the set of solutions to a system $A^1x = b^1$, where $A^1x \le b^1$ is a subsystem of $Ax \le b$. Clearly at most d equations in $A^1x = b^1$ are independent, and by Corollary 2.3, the system has a solution whose size is bounded by $d \times \delta$ which is polynomially bounded by δ. ∎

This corollary also implies that every extreme point of a polyhedron has size polynomially bounded by the largest size of rows of $[A, b]$.

Theorem 3.17 *Let $P = \{x : Ax \le b\}$ be a rational polyhedron. Then, it has a generator representation $P = \text{conv}(\{v_1, \ldots, v_s\}) + \text{cone}(\{r_1, \ldots, r_t\})$ such that each generator v_i or r_j is of size polynomially bounded by the largest size of rows of the matrix $[A, b]$.*

Proof. Let $P = \{x : Ax \le b\}$ be a rational polyhedron, and A and b be integer. Let δ denote the largest size of rows of the matrix $[A, b]$.

If P is bounded, the minimal generator representation is the set of extreme points and the size of each extreme point is polynomially bounded by δ. by Corollary 3.16.

If P is a pointed cone, by Corollary 3.13, the intersection of P with the hyperplane $\mathbf{1}^T Ax = -1$ is a polytope. Clearly the extreme points of this polytope constitute a minimal representation of P and have size polynomially bounded by δ. (The halfline generated by each extreme point is called an *extremal ray* of this pointed cone.)

If P is a pointed polyhedron, it is the Minkowski sum of a polytope and a pointed cone, and the first two cases imply the theorem.

If P is not pointed, the linearity space is the null space of A, the Gauss-Jordan algorithm applied to $Ax = 0$ produces a generator representation of this space by linearly independent vectors $\{b_1, \ldots, b_k\}$ of polynomial size, using the proof of Theorem 2.4. Then, setting $Q = P \cap \{x : b_i^T x = 0, i = 1, \ldots, k\}$, $P = Q + \text{lin.space}(P)$. Now Q is pointed and it has a generator representation of polynomially bounded size. ∎

Remark 3.18 *From the proof of Theorem 3.17, a minimal representation of a polyhedron P consists of a set of points each of which is from a minimal nonempty face of P, a set of vectors to generate the pointed cone which is the intersection of P with the linear subspace orthogonal to the linearity space of P, and a set of vectors to generate the linearity space.*

Corollary 3.19 *For $A \in \mathbb{Q}^{m \times d}$, $B \in \mathbb{Q}^{m \times n}$ and $c \in \mathbb{Q}^m$, let P be the polyhedron $\{(x, y) \in \mathbb{R}^{d+n} : Ax + By \le c\}$, and let P_x be the orthogonal projection of P to the x-space, i.e., $x \in P_x$ if and only if $(x, y) \in P$ for some $y \in \mathbb{R}^n$. Then, P_x admits an H-representation $\{x \in \mathbb{R}^d : Dx \le f\}$ such that the size of each $[D_i, f_i]$ is polynomially bounded by the largest* *size of rows of the matrix $[A, B, c]$. Note that in general the number of rows in D may be exponential in one of m, n and d.*

Proof. Left to the reader. Hint: Using the Fourier-Motzkin Algorithm show that $P_x = \{x : z^T Ax \le z^T c, \forall z \in C\}$ for the "projection" cone $C = \{z \in \mathbb{R}^m : z^T B = \mathbf{0}, z \ge \mathbf{0}\}$. Then, apply Theorem 3.17 ∎

3.7 Some Basic Polyhedra

A d-simplex is the convex hull of $d+1$ affinely independent points $v_0, v_1, \dots, v_d$ in $\mathbb{R}^d$. The *standard d-cube* is the convex hull of 2^d 0/1 points in $\mathbb{R}^d$, and a d-cube is any full-rank affine transformation of the standard d-cube. A *zonotope* in $\mathbb{R}^d$ (generated by k generators) is the Minkowski sum of k line segments in $\mathbb{R}^d$. The standard cube is a special zonotope generated by the d line segments $[\mathbf{0}, e_j]$, where e_j denotes the jth unit vector.

The following table gives the number of i-faces of a d-simplex and of a d-cube. It also gives the tight upper bound of the number of i-faces of a zonotope, which will be proved in Chapter 10.

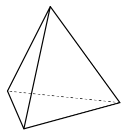
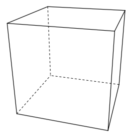
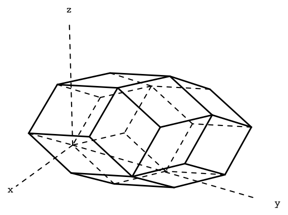
Type	Figure	# Vertices	# Facets	# i-Faces
d-Simplex		$d+1$	$d+1$	$\binom{d+1}{i}$
d-Cube		2^d	$2d$	$\binom{d}{i}2^{d-i}$
Zonotope(d,k)	 $d = 3$ and $k = 5$	$\le 2\sum_{i=0}^{d-1}\binom{k-1}{i}$	$\le 2\binom{k}{d-1}$	$O(k^{d-1})$

Table 3.1: Simplex, Cube and Zonotope

Exercise 3.8 Show that the formula in Table 3.1 for the number of i-faces is correct for d-simplices and for d-cubes.

4 Integer Hull and Complexity

Integer linear programming or simply integer programming (abbreviated by IP) is an extensitvely studied branch of optimizaton. One form of the IP is a linear programming with the additional integer constraints on the variables.

$$\begin{array}{lrl} \text{max} & c^T x & \\ \text{subject to} & Ax & \le b \\ & x & \in \mathbb{Z}^d, \end{array} \tag{4.1}$$

where $A\mathbb{Q}^{m\times d}$ and $b \in \mathbb{Q}^m$ are given. Another form more convenient for the analysis of its complexity is the decision problem

$$\begin{array}{lrl} \text{decide whether there exists} & x & \\ \text{such that} & Ax & \le b \\ & x & \in \mathbb{Z}^d. \end{array} \tag{4.2}$$

In this chapter, we show that IP is NPC, meaning, the decision problem (4.2) is NPC. There are two things in this statement, IP is NP-hard (i.e., at least as any problems in NP), and IP is in NP.

The first part is very easy to see by the standard polynomial reduction from SAT (the satisfiability problem). For this we take an NPC problem *3-SAT*:

$$\begin{array}{l} \text{given a boolean expression in binary } d\text{-variables } B(x) := \bigwedge_{i=1}^m C_i \\ \text{where each clause } C_i \text{ is a disjunction of three literals} \\ \text{decide there exists } x \in \{0,1\}^d \text{ such that B(x)=1.} \end{array} \tag{4.3}$$

The reduction is simple. To reduce this problem to an IP, we will use exactly the same variables x. Each x_j is integer and restricted as $0 \le x_j \le 1$. For each clause, for example, $(x_1 \bigvee \neg x_2 \bigvee x_5)$, we set up the inequality:

$$x_1 + (1 - x_2) + x_5 \ge 1. \tag{4.4}$$

Furthermore, each variable x_j is restricted as $0 \le x_j \le 1$. With all these constraints together, we have an IP of form (4.2) which is equivalent to 3-SAT. Moreover the reduction is polynomial.

Thus, the most critical part of the statement "IP is NPC" is, in fact, **IP is in NP**. More precisely, this means that if an IP (4.2) admits a solution, there is a solution whose size is bounded by a polynomial function of the input size size$[A, b]$.

To see this, one important notion is the integer hull of a polyhedron. For a rational polyhedron $P = \{x : Ax \le b\}$, its *integer hull* is defined by

$$P_I := \text{conv}\{x : x \in P \cap \mathbb{Z}^d\}. \tag{4.5}$$

Section 4.2 analyses the complexity of P_I and shows why IP is in NP. In fact, we prove a stronger statement (in Theorem 4.3) that a feasible IP admits a solution whose size is bounded by a polynomial function of **the largest size of rows of** $[A, b]$.

4.1 Hilbert Basis

The Hermite normal form $[B\ \mathbf{0}]$ of a rational matrix $A \in \mathbb{R}^{m\times d}$ can be considered as a minimal generating set of the lattice $L(A)$ generated by A. Namely, the m columns of B form a minimal set of vectors in $\mathbb{R}^m$ whose integer combinations generate $L(A)$.

In this section, we are going to deal with the lattice points in a polyhedral cone. Is there any similar basis for the integer points in a polyhedral cone C generated by rational vectors $\{a_1, a_2, \ldots, a_t\}$? A *Hilbert basis* of a cone C is a finite set of rational vectors $b_1, b_2, \ldots, b_k$ such that every lattice point in the cone is a nonnegative integer combination of $b_1, b_2, \ldots, b_k$. Here we are mainly concerned with integral Hilbert basis.

Note that a *Hilbert basis* is sometimes called a *Hilbert finite generating set* and then the term *Hilbert basis* is used only for the ones that are (set-inclusion) minimal.

Theorem 4.1 *Every rational cone admits an integral Hilbert basis. Futhermore, if it is pointed, a (set-inclusion) minimal integral Hilbert basis is unique.*

Proof. Without loss of generality, we assume a rational cone C is generated by integral vectors $a_1, a_2, \ldots, a_t$ in $\mathbb{R}^d$, i.e., $C = \text{cone}(\{a_1, a_2, \ldots, a_t\})$. We claim that the finite set $B = \{b_1, \ldots, b_k\}$ of integral vectors contained in the zonotope

$$Z := \{x \in \mathbb{R}^d : x = \sum_{i=1}^{t} \lambda_i a_i, 0 \le \lambda_i \le 1, i = 1, \ldots, t\} \tag{4.6}$$

is a Hilbert basis, see Figure 4.1 for an example with $d = 2$, $t = 2$ and $k = 8$.

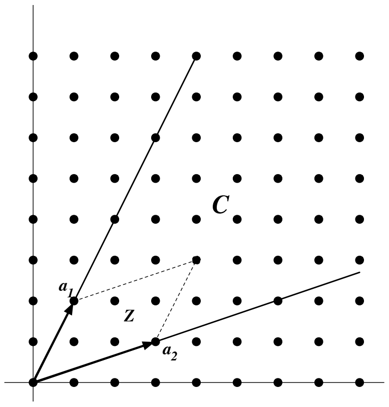


Figure 4.1: The Cone C and the Polytope Z.

Let p be any integral point in C. Then, we have

$$p = \sum_{i=1}^{t} \lambda_i a_i,\ \lambda_i \ge 0, i = 1, \ldots, t, \tag{4.7}$$

for some λ_i (not necessarily integer). Furthermore, we have

$$p - \sum_{i=1}^{t} \lfloor \lambda_i \rfloor a_i = \sum_{i=1}^{t} (\lambda_i - \lfloor \lambda_i \rfloor) a_i. \tag{4.8}$$

First, the LHS is an integer vector. Secondly, the RHS vector, the same vector as the LHS, is in Z, because $0 \le \lambda_i - \lfloor \lambda_i \rfloor < 1$. Therefore, it is an integer vector in Z which is among $b_1, \dots, b_k$. Since $a_1, \dots, a_t$ are contained in $\{b_1, \dots, b_k\}$, p is a nonnegative integer combination of $b_1, \dots, b_k$.

For the second part, assume that the cone C is pointed. We claim that

$$\widehat{B} := \{x \in B \setminus \{\mathbf{0}\} : x \text{ is not the sum of two other vectors in } B\} \tag{4.9}$$

is a unique minimal Hilbert basis. It is easy to see that every vector in $\widehat{B}$ must be in any integral Hilbert basis. Now, we need to show that every vector b in B not in $\widehat{B}$ can be represented as nonnegative integer combination of vectors in $\widehat{B}$. Suppose there is a vector b in B violating this property, and take such a vector b minimizing $c^T b$, where c is a vector such that $c^T x > 0$ for all nonzero $x \in C$. The existence of c is guaranteed because C is pointed, due to Corollary 3.13. Because b is not in $\widehat{B}$, $b = b_i + b_j$ for some nonzero vectors b_i, b_j in B. Now, we have $c^T b = c^T b_i + c^T b_j$, and all terms are positive. This means $c^T b_i < c^T b$ and $c^T b_j < c^T b$. By the assumption that $c^T b$ is minimized under the condition that b is not in $\widehat{B}$, both b_i and b_j must belong to $\widehat{B}$, contradicting b is not a nonnegative integer combination of vectors in $\widehat{B}$. ∎

Exercise 4.1 Show that if a rational cone is not pointed, a minimal integral Hilbert basis is not unique.

Exercise 4.2 In the proof above, assume that $t = d$ and the rational cone is generated by d linearly independent vectors $C = \text{cone}(\{a_1, a_2, \dots, a_d\})$. Derive a tight lower bound of k in terms of d and the absolute value of the determinant $\det([a_1, a_2, \dots, a_d])$. Note that k is the number of lattice points in the zonotope Z and $k \ge 2^d$, because the zonotope Z is combinatorially a cube.

4.2 The Structure of Integer Hull

For a rational polyhedron P in $\mathbb{R}^d$, recall that its integer hull P_I is defined as the convex hull of all integer points in P:

$$P_I := \text{conv}\{x : x \in P \cap \mathbb{Z}^d\}. \tag{4.10}$$

It is not clear from the definition that the integer hull is a polyhedron, and in particular finitely generated. We will show that this is the case and the proof uses an argument similar to those used to prove the existence of a Hilbert basis.

There are some obvious facts on the integer hull. First of all the integer hull of every rational cone C is C itself:

$$C_I := C. \tag{4.11}$$

Theorem 4.2 *The integer hull P_I of a rational polyhedron P is a polyhedron itself, and if it is nonempty, then $P_I = B + C$, where B is an integer polytope and $C = \text{rec. cone}(P)$.*

Proof. Assume that P is a rational polytope with a decomposition $P = Q + C$ into a polytope Q and the recession cone C. Assume that P_I is nonempty. Let $a_1, a_2, \ldots, a_t$ be integral vectors in $\mathbb{R}^d$ with $C = \text{cone}(\{a_1, a_2, \ldots, a_t\})$. Let Z be the zonotope defined by

$$Z := \{x \in \mathbb{R}^d : x = \sum_{i=1}^{t} \lambda_i a_i, 0 \le \lambda_i \le 1, i = 1, \ldots, t\}. \tag{4.12}$$

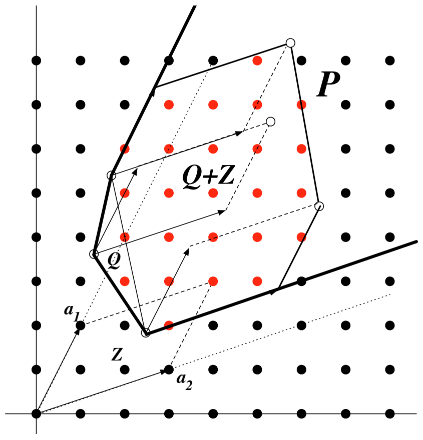


Figure 4.2: The Critical Region $Q + Z$.

For the proof of the theorem, it suffices to show that

$$P_I = (Q + Z)_I + C. \tag{4.13}$$

To see $(Q + Z)_I + C \subseteq P_I$, observe

$$(Q + Z)_I + C \subseteq P_I + C = P_I + C_I \subseteq (P + C)_I = P_I.$$

For the reverse inclusion, take any integer point $p \in P_I$ and we show that $p \in (Q + Z)_I + C$. This is sufficient because $(Q + Z)_I + C$ is convex. Now $p = q + c$, for some $q \in Q$ and $c \in C$. Now, we have $c = \sum_i \lambda_i a_i = \sum_i \lfloor \lambda_i \rfloor a_i + \sum_i (\lambda_i - \lfloor \lambda_i \rfloor) a_i$, where the first term is denoted by c' and the second by z. Clearly, $c' \in C \cap \mathbb{Z}^d$ and $z \in Z$. It follows that $p = q + c' + z = (q + z) + c'$ which implies that $q + z$ is integer and thus $q + z \in (Q + Z)_I$. Since $c' \in C$, we have $p \in (Q + Z)_I + C$. ∎

Theorem 4.3 *The integer hull P_I of a rational polyhedron $P = \{x : Ax \le b\}$ given by an integer matrix A and an integer vector b has a generator representation $P_I = \text{conv}(\{z_1, \ldots, z_k\}) + \text{cone}(\{r_1, \ldots, r_h\})$ such that the size of each generator z_i or r_j is polynomially bounded by the largest size of rows of the matrix $[A, b]$.*

Proof. Let $P = \{x : Ax \leq b\}$ be a rational polyhedron and let δ denote the largest size of rows of $[A, b]$.

By Theorem 3.17, a rational polyhedron P has a generator representation $Q + C$ with $Q = \text{conv}(\{v_1, \ldots, v_s\})$ and $C = \text{cone}(\{r_1, \ldots, r_h\})$, where each of v_i's and r_j's has size polynomially bounded by δ. We may also assume that all r_j's are integer vectors. By Theorem 4.2, $\text{rec.cone}(P_I) = \{r_1, \ldots, r_h\}$. We shall show that

$$P_I = \text{conv}(\{z_1, \ldots, z_k\}) + C, \tag{4.14}$$

where $z_1, \ldots, z_k$ are the integer points in the set $Q + Y$ and

$$\begin{aligned} Y = \{y : y = \sum_{j=1}^{h} \lambda_j r_j, 0 \leq \lambda_j \leq 1, j = 1, \ldots, h, \quad && (4.15)\\ \text{at most } d \text{ of } \lambda_j\text{'s are positive}\}. && (4.16)\end{aligned}$$

Each vector z_i has size polynomially bounded by δ, because all r_j's are integer, polynomially bounded by δ in size, at most d of them are used to represent z_i, and every integer point in Q has size polynomially bounded by δ.

We are left to show the equation (4.14). For this, it is sufficient to show that each minimal nonempty face F of P_I contains at least one point from $\{z_1, \ldots, z_k\}$, see Remark 3.18. Let z be an integer point of F. Because $z \in P$

$$z = q + \sum_{j=1}^{h} \mu_j r_j, \tag{4.17}$$

for some $\mu_j \geq 0$. By Carathéodory's Theorem (Theorem 3.6), one may assume that at most d of μ_j's are positive. Let z' be the vector

$$z' := q + \sum_{j=1}^{h} (\mu_j - \lfloor \mu_j \rfloor) r_j. \tag{4.18}$$

It follows that z' is an integer vector and thus it is one of the vectors from $\{z_1, \ldots, z_k\}$. It is easy to see that $z' \in F$. This completes the proof. ∎

Corollary 4.4 *IP (4.2) is in NP (and thus in NPC).*

4.3 Complexity of Mixed Integer Programming

One natural question is as to whether the mixed integer programming (MIP) is in NP, and thus in NPC. An MIP is to

$$\begin{array}{lrcl} \text{decide whether there exists} & (x, y) & & \\ \text{such that} & A\,x & + & B\,y \leq c \\ & x \in \mathbb{Z}^d, & & y \in \mathbb{R}^n, \end{array} \tag{4.19}$$

where $A \in \mathbb{Q}^{m \times d}$, $B \in \mathbb{Q}^{m \times n}$ and $c \in \mathbb{Q}^m$ are given.

Theorem 4.5 *MIP (4.19) is in NP (and thus in NPC).*

Proof. For the proof, it is sufficient to show that if the MIP admits a feasible solution (x, y) then there is a feasible solution (x, y) such that x has size polynomially bounded by the input size. To see this, we assume that the MIP has a feasible solution, and we look at the projection Q of the nonempty polyhedron $P = \{(x, y) : A\,x + B\,y \leq c\}$ to the x-space. Note that Q_I is nonempty. By Corollary 3.19, Q admits an H-representation $\{x \in \mathbb{R}^d : Dx \leq f\}$ such that the size of each $[D_i, f_i]$ is polynomially bounded by the largest size of rows of the matrix $[A, B, c]$. By Theorem 3.17, it follows that Q has a V-representation in which the size of each generator in the representation is polynomially bounded by the largest size of rows of the matrix $[A, B, c]$. Now, by Theorem 4.3 applied to Q, we have that Q_I has a V-representation in which the size of each generator is polynomially bounded by the largest size of rows of the matrix $[A, B, c]$. This completes the proof. ∎

Notes 4.6 *The author was not aware of a written proof of Theorem 4.5. The proof above is due to François Margot (Carnegie Mellon University) and the author in October 2010. We learned in October 2010 that an independent proof was being written by Michele Conforti (University of Padova) and Gérard Cornuéjols (Carnegie Mellon).*

4.4 Further Results on Lattice Points in Polyhedra

Here, we mention some important results on lattice points in polyhedra without proofs. The original proofs are not particularly difficult but beyond the scope of this lecture notes.

The following is known as an integer analogue of Carathéodory's theorem.

Theorem 4.7 (Cook, Fonlupt and Schrijver (1983)) *Let C be a pointed rational cone and let $\widehat{B}$ be the minimal integral Hilbert basis. Then, every integer point p in $\widehat{B}$ is an integer nonnegative combination of at most $2d - 1$ of the vectors in $\widehat{B}$.*

Later this bound $2d - 1$ was improved to $2d - 2$ by Sebő (1990).

Another interesting result is on the distance between the integer programming solutions and the solutions to the linear programming relaxation.

Theorem 4.8 (Cook, Gerards, Schrijver and Tardos (1986)) *For a given matrix $A \in \mathbb{Z}^{m\times d}$, vectors $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^d$, let (IP) be the integer programming problem $\max c^T x$ subject to $Ax \leq b$ and $x \in \mathbb{Z}^d$, and let (LP) be its LP relaxation (without the $x \in \mathbb{Z}^d$ contraints). Let D denote the largest absolute value of subdeterminants of A. Assume that both problems admit an optimal solution. Then the following statements hold.*

(a) *For any optimal solution x^* of (LP), there exists an optimal solution z^* of (IP) such that $|x_i^* - z_i^*| \leq dD$ for all i.*

(b) *For any optimal solution z^* of (IP), there exists an optimal solution x^* of (LP) such that $|x_i^* - z_i^*| \leq dD$ for all i.*

By Theorem 2.1, the size of D is at most twice the size of the matrix A. The theorem above thus shows that there exists an optimal solution to (IP) in a hypercube of a width of polynomial size centered at a given LP optimal solution, if both admit an optimal solution.

5 Duality of Polyhedra

Duality in convex polyhedra is a very interesting notion not only in the theory of polyhedra but also in polyhedral computation. Duality implies that two basic representation conversions between V-representation and H-representation of a polyhedron are essentially the same thing. Yet, in order to convert one to the other is sometimes tricky because there are certain assumptions under which any specific conversion can work.

5.1 Face Lattice

Let P be a convex polytope in $\mathbb{R}^d$. Each face F of P is a convex polytope again by definition. The *dimension* $\dim(P)$ of P is the maximum number of affinely independent points in P minus one. The number of k-dimensional faces of P is denoted by $f_k(P)$. By Theorem 3.14, $f_k(P)$ is finite. A k-dimensional face (polytope) is called simply *k-face* (*k-polytope*). For a d-polytope P, the vector

$$f(P) := (f_{-1}, f_0, f_1, \dots, f_d) \tag{5.1}$$

is called the *f-vector* of P. Clearly $f_{-1} = f_d = 1$.

The 0-faces of a d-polytope are called the *vertices*, the 1-faces the *edges*, the $(d-1)$-faces the *facets*, and the $(d-2)$-faces the *ridges*.

We denote by $\mathcal{F}(P)$ the finite set of all faces of P ordered by set inclusion. This is called the *face lattice* of P. Recall that a lattice is a partially ordered set (poset in short) where the join (the least upper bound) and the meet (the greatest lower bound) of any two elements a and b exist in the set. The face lattice of a polytope is also known as the *combinatorial structure* of the polytope. In Figure 5.1, the face lattices of 1-, 2- and 3-cubes are depicted, whose f-vectors are $\{1, 2, 1\}$, $\{1, 4, , 4, 1\}$ and $\{1, 8, 12, 6, 1\}$. One can easily show that all 1-polytopes are finite line segments and thus are combinatorially the same *diamond*.

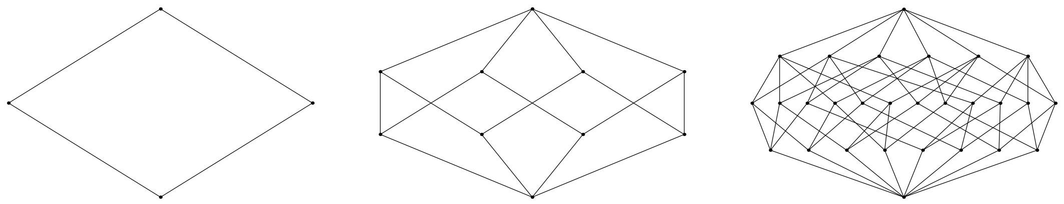


Figure 5.1: The Hasse diagram of the face lattices of 1-, 2- and 3-cubes

A lattice is called *polytopal* if it is isomorphic to the lattice of a polytope. Polytopal lattices are very special. The following proposition summarizes this.

Proposition 5.1 *Every polytopal lattice satisfies the follow properties.*

(a) *It satisfies the* Jordan-Dedekind chain property*, i.e., all maximal chains between any two ordered elements $a < b$ have the same length.*

(a) *Fny two ordered elements $a < b$, the interval $[a, b]$, the set of elements between a and b, is again a polytopal lattice. In particular, this means that every interval of hight 2 is a diamond.*

We shall prove these properties in a later section, as we do not use them to prove the polytopal duality to be described below.

For a polytope P, a polytope P' is called a *dual* of P if $\mathcal{F}(P')$ is anti-isomorhic to $\mathcal{F}(P)$. It follows that a polytope P and a dual polytope P' have the same dimension, and their f-vectors are reversed, $f_i(P) = f_{d-i-1}(P')$ for all $i = -1, 0, \dots, d$. The following is the fundamental theorem of duality.

Theorem 5.2 *Every polytope admits a dual polytope.*

It is easy to see that a dual polytope is not unique. Any d-simplex is a dual of a d-simplex. A 3-cube is a dual of an octahedron but there are many geometrically different polytopes with isomorphic face lattices.

Yet, there is a simple construction of a dual polytope which is extremely useful, both theoretically and computationally. For a convex body C in $\mathbb{R}^d$ containing the origin $\mathbf{0}$ in its interior, define its *polar* denoted by C^* as

$$C^* = \{y \in \mathbb{R}^d : x^T y \leq 1, \forall x \in C\}. \tag{5.2}$$

Theorem 5.3 *Let P be a polytope containing the origin $\mathbf{0}$ in its interior. Then its polar P^* is a dual polytope of P.*

5.2 Active Sets and Face Representations

As we learned from Theorem 3.9, every polyhedron has two representations, H-reprentation and V-representation. These two representations are closely linked to duality. Intuitively, by setting the transpose of an H-representation as a V-reprentation, we obtain a dual. This statement is in general incorrect and can be stated correctly with proper assumptions.

Let P be a polyhedron with an H-representation (A, b) and a V-representation (V, R). Each row of (A, b) is denoted by (A_i, b_i), representing the inequality $A_i x \leq b_i$. Each column of V and R is denoted by v_j and r_k, respectively, the jth vertex generator and the kth ray generator. We employ a little abuse of language here. An H-representation (A, b) is considered as the set of all its rows (A_i, b_i), and similarly V (R) is considered as the set of all its columns v_j's (r_k's).

Let F be a non-empty face of P. An inequality (A_i, b_i) is called *active* at F if the inequality is satisfied with equality at all points in F. The set of all active inequalities is called the *active inequality* set at F.

Similarly, a vertex generator v_j is called *active* at F if $v_j \in F$. A ray generator r_k is called *active* at F if moving from any point on F along the direction r_k won't leave the polyhedron, i.e., $x + \theta r_k \in F$ for any $x \in F$ and $\theta \geq 0$. The pair (V', R') of sets of all active vertices and active rays are called the *active generator* sets at F. We extend the set inclusion for pairs of sets in the natural way, we define $(V'', R'') \subseteq (V', R')$ if and only if $V'' \subseteq V'$ and $R'' \subseteq R'$.

Active inequalities and generators are very important for representation of faces and face lattices.

Theorem 5.4 *Let P be a polyhedron with a V-representation (V,R), and let F be a nonempty face of P. Then the active generator set pair (V',R') at F is a V-representation of F.*

Proof. Let (J,K) be the column index sets of (V',R'), namely, $V' = (v_j : j \in J)$ and $R' = (r_k : k \in K)$. Let

$$\overline{F} = \{x \in \mathbb{R}^d : x = V'\mu' + R'\lambda', \mu' \geq \mathbf{0}, \mathbf{1}^T\mu' = 1, \lambda' \geq \mathbf{0}\}.$$

We need to show $F = \overline{F}$. By definition of active generators, we have $F \supseteq \overline{F}$. For the converse inclusion, let $p \in F$ and suppose $p \notin \overline{F}$. Since $p \in P$,

$$p = V\mu + R\lambda \tag{5.3}$$

for some $\mu \geq \mathbf{0}$, $\mathbf{1}^T\mu = 1$ and some $\lambda \geq \mathbf{0}$. Because $p \notin \overline{F}$, we have $\mu_j > 0$ for $j \notin J$ or $\lambda_k > 0$ for $k \notin K$. Suppose there is $j \notin J$ such that $\mu_j > 0$. Then, $v_j \notin F$. Let (A,b) be an H-representation of P. Since $v_j \notin F$, there is an inequality (A_i, b_i) active at F such that $A_i v_j < b_i$. Since this is active at F, $A_i p = b_i$ and this implies that there is a ray or vertex generator in the RHS representation 5.3 of p which violates this active inequality. This is impossible. The second case is impossible by a similar argument. Thus, $p \in \overline{F}$. This completes the proof. ∎

Theorem 5.5 *Let P be a polyhedron with an H-reprentation (A,b) and a V-representation (V,R). Then*

(a) *the face poset $\mathcal{F}(P) \setminus \{\emptyset\}$ is anti-isomorphic to the set of all active inequality sets ordered by set inclusion;*

(b) *the face poset $\mathcal{F}(P) \setminus \{\emptyset\}$ is isomorphic to the set of all active generator sets ordered by set inclusion.*

Proof.

(a) It is clear that the larger a face is, the smaller its active inequality set is. The main question is if the strictly larger a face is, the strictly smaller its active inequality set is. This follows directly from Theorem 3.14.

(b) Using a similar argument to (a), (b) follows from Theorem 5.4.

∎

5.3 Duality of Cones

Before proving the duality of polytopes, we show the duality of cones is a straightforward consequence of Theorem 5.5, a basic theorem on face lattice representations by active sets.

The notion of dual (polyhedral) cone is essentially the same as that of polytopes, the face lattice of a dual is the polar (upside-down) lattice. There is a small technical difference. Cones are different from polytopes in the sense that every cone has a unique minimal nonempty face (containing the origin), which plays exactly like the empty face of every polytope. For this reason, we define the *face lattice $\mathcal{F}(C)$ of a cone C* as the set of all **nonempty faces** of C ordered by set inclusion. Accordingly, we say that a cone C' is a *dual* cone of a cone C if $\mathcal{F}(C)$ and $\mathcal{F}(C')$ are anti-isomorphic.

Theorem 5.6 *Every (polyhedral) cone admits a dual cone.*

Our proof is by construction. For this, we define two cones. For a real $m \times d$ matrix A, we denote by $C_H(A)$ the cone with A as its H-representation:

$$C_H(A) = \{x : Ax \leq \mathbf{0}\}. \tag{5.4}$$

For a real $d \times s$ matrix R, we denote by $C_V(R)$ the cone with R as its V-representation:

$$C_V(R) = \{x : x = R\lambda, \lambda \geq \mathbf{0}\}. \tag{5.5}$$

Using this notation, Minkowski-Weyl Theorem, Theorem 3.10, says that a set C is of form $C = C_H(A)$ for some matrix A if and only if $C = C_V(A)$ for some matrix R.

The following is a stronger (constructive) version of the cone duality, Theorem 5.6.

Theorem 5.7 *For any real $m \times d$ matrix A, the cone $C_H(A)$ and the cone $C_V(A^T)$ are dual to each other.*

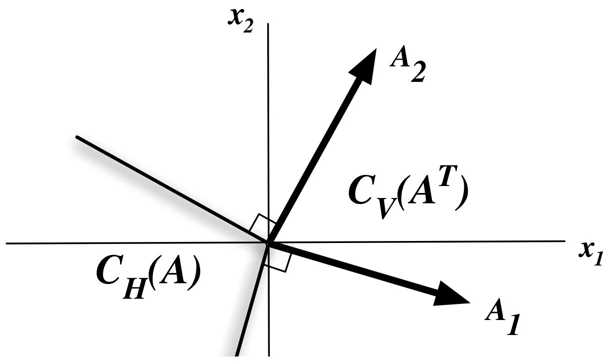


Figure 5.2: Cone Duality

Proof. Let A be a real $m \times d$ matrix. Let F be any nonempty face of $C_H(A)$, and let $I \subseteq [m]$ be the set of active inequality row indices at F, i.e., $F = \{x \in C_H(A) : A_I x = \mathbf{0}\}$ and

$$\begin{aligned} \exists c \in \mathbb{R}^d \text{ such that } A_i c = \mathbf{0}, \forall i \in I, \text{ and} \\ A_j c < \mathbf{0}, \forall j \in [m] \setminus I. \end{aligned}$$

Or equivalently,

$$\begin{aligned} \exists c \in \mathbb{R}^d \text{ such that } c^T (A_i)^T = \mathbf{0}, \forall i \in I, \text{ and} \\ c^T (A_j)^T < \mathbf{0}, \forall j \in [m] \setminus I. \end{aligned}$$

Noting that the vectors $(A_i)^T$ ($i \in [m]$) are the generators of the cone $C_V(A^T)$, the relations above show exactly that $\{(A_i)^T :\in I\}$ is the active generator set at the face of $C_V(A^T)$

determined by the valid inequality $c^T x \le 0$ for the cone $C_V(A^T)$. The reverse direction is obvious.

This provides a one-to-one correspondence between the set of nonempty faces of $C_H(A)$ and the set of nonempty faces of $C_V(A^T)$, reversing the set inclusion. This completes the proof. ∎

5.4 Duality of Polytopes

As we learned in the previous section, the duality of cones arises very naturally, and in fact, an H-representation of a cone immediately gives a V-representation of a dual cone, and vice visa: the cones $C_H(A)$ and $C_V(A^T)$ are dual to each other for any matrix A.

To derive a similar construction for polytopes, one can use the cone duality carefully. The main idea is to express a d-polytope P as the intersection of $(d+1)$-cone C in such a way that P is embedded in C as the intersection of C with hyperplane $x_{d+1} = -1$. This is easy to do if P is a V-polytope. This gives some matrix R and the cone $C_V(R)$ in $\mathbb{R}^{d+1}$. We know how to construct a dual of $C_V(R)$: $C_H(R^T)$. The hard part is the rest: we have to make sure that it is "nicely" intersected by a hyperplane so that the intersection is a polytope and has the same face lattice as $C_H(R^T)$. If this is done, we have the construction of a dual of P.

Let P be a d-polytope with V-reresentation $V = [v_1, \dots, v_m]$. Let

$$\hat{V} := \begin{bmatrix} v_1 & v_2 & \cdots & v_m \\ -1 & -1 & \cdots & -1 \end{bmatrix}. \tag{5.6}$$

By Theorem 5.7, the following cones C and D defined below are dual to each other

$$C := C_V(\hat{V}) = \{x : x = \hat{V}\lambda, \lambda \ge \mathbf{0}\}, \tag{5.7}$$

$$D := C_H(\hat{V}^T) = \{x : \hat{V}^T x \le \mathbf{0}\}. \tag{5.8}$$

Furthermore, by construction, the cone C represents P nicely.

Proposition 5.8 *The face lattices $\mathcal{F}(P)$ and $\mathcal{F}(C)$ are isomorphic.*

Proof. Consider the cut section P' of C with the hyperplane $h^{-1} := \{x \in \mathbb{R}^{d+1} : x_{d+1} = -1\}$. It follows that P and P' are affinely equivalent, and in particular, their face lattices are isomorphic. It is left to show that $\mathcal{F}(C)$ and $\mathcal{F}(P')$ are isomorphic. This follows from the fact that $\hat{V}$ is not only a V-representation of C but also a V-representation of P'. ∎

A proof of Theorem 5.2 is almost complete, because we know the face lattice of D is the target lattice we need to realize as the face lattice of a polytope. The only thing we have to show is that the cone D can be cut nicely by a hyperplane so that the cut section, say Q', has the face lattice isomorphic to D. For this, consider the hyperplane $h^{+1} := \{x \in \mathbb{R}^{d+1} : x_{d+1} = +1\}$. define

$$Q' := D \cap h^{+1}. \tag{5.9}$$

Observe that

$$Q' = \left\{ \begin{bmatrix} x \\ x_{d+1} \end{bmatrix} \in \mathbb{R}^{d+1} : \begin{bmatrix} v_1^T & -1 \\ \vdots & \vdots \\ v_m^T & -1 \end{bmatrix} \begin{bmatrix} x \\ x_{d+1} \end{bmatrix} \leq \mathbf{0} \right\} \cap \{x \in \mathbb{R}^{d+1} : x_{d+1} = +1\} \tag{5.10}$$

$$= \left\{ \begin{bmatrix} x \\ 1 \end{bmatrix} \in \mathbb{R}^{d+1} : v_i^T x \leq 1, \forall i = 1, \ldots, m \right\}. \tag{5.11}$$

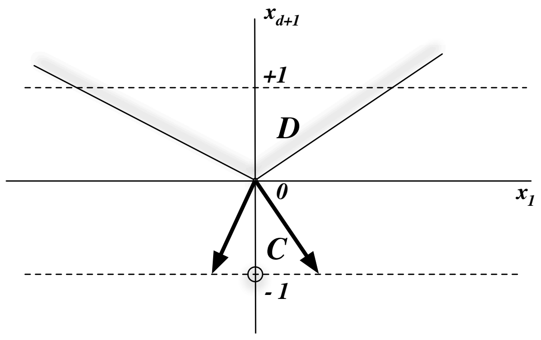
Thus, the polyhedron Q' is affinely equivalent to the polyhedron

$$Q = \{x \in \mathbb{R}^d : v_i^T x \leq 1, \forall i = 1, \ldots, m\} = \{x \in \mathbb{R}^d : V^T x \leq \mathbf{1}\}. \tag{5.12}$$

The polyhedron Q (and Q') may not have the face lattice isomorphic to D in general. Construct a small example to show this fact. The following lemma gives a right assumption for duality to work.

Theorem 5.9 *If P contains the origin in its interior, the polyhedron Q is a polytope dual to P.*

Proof. Assume that P contains the origin in its interior. The only thing left to be shown is that the face lattices of the cone D and the polyhedron Q' are isomorphic. For this, it is sufficient to show that Q' is bounded and a V-representation of Q' is in fact a V-representation of D. (Figure 5.3 shows that the assumption is in fact necessary.)



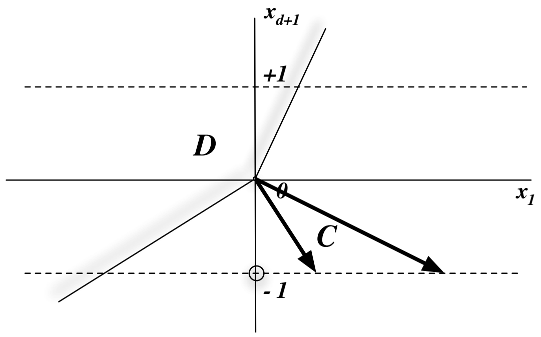


Figure 5.3: Polytope Duality: When it works and When it does not

Observe that the assumption is equivalent to

$$\operatorname{rank} V = d, \text{ and} \tag{5.13}$$
$$\exists \lambda > \mathbf{0} \text{ such that } V\lambda = \mathbf{0}. \tag{5.14}$$

By (a variation of) Gordan's Theorem (Exercise 3.4), the statement (5.14) is equivalent to

$$\nexists x \text{ such that } V^T x \lneq \mathbf{0}. \tag{5.14'}$$

In order to show that Q (and Q') is bounded, suppose the contrary: there is a unbounded direction in Q, i.e., a nonzero vector z such that $V^T z \leq \mathbf{0}$. By (5.14'), this implies $V^T z = \mathbf{0}$ and $z \neq \mathbf{0}$, which is impossible by the assumtion (5.14).

Now, we shall show a V-representation of Q' is a V-representation of D. For this, we take any nonzero vector $(x, x_{d+1})^T \in D$, and show that $x_{d+1} > 0$. This means that the normalized vector $(x/x_{d+1}, 1)^T$ is in Q'. Thus, any V-representation of Q' represents the cone D. To see that $x_{d+1} > 0$, observe that

$$\begin{aligned} & \begin{bmatrix} x \\ x_{d+1} \end{bmatrix} \neq \mathbf{0} \text{ and } \hat{V}^T \begin{bmatrix} x \\ x_{d+1} \end{bmatrix} \leq \mathbf{0} \\ \Longrightarrow & \begin{bmatrix} x \\ x_{d+1} \end{bmatrix} \neq \mathbf{0} \text{ and } V^T x - \mathbf{1} x_{d+1} \leq \mathbf{0} \\ \Longrightarrow & \; x_{d+1} > 0. \end{aligned}$$

The last implication is valid because if $x_{d+1} \leq 0$, $V^T x \leq \mathbf{0}$ for $x \neq \mathbf{0}$ which is impossible by the assumptions (5.13) and (5.14). ∎

5.5 Examples of Dual Pairs

In Section 3.7, we introduced a few examples of polytopes. Let's look at their duals.

First of all, one can easily see that a d-simplex is self-dual.

What is a dual of an d-cube? The simple way to see is to use the centrally symmetric cube Cube(d) whose vertex set is $\{-1, 1\}^d$. Namely,

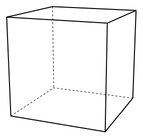
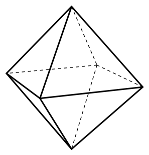
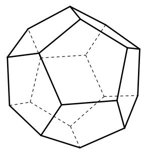
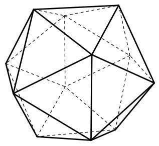
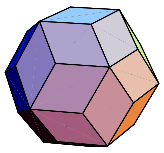
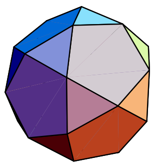
$$\text{Cube}(d) = \text{conv}\{-1, 1\}^d = \{x \in \mathbb{R}^d : \pm(e_i)^T x \leq 1, \forall i = 1, \ldots, d\}. \tag{5.15}$$

The polar of Cube(d) is known as the d-cross polytope:

$$\text{Cross}(d) = \{x : a^T x \leq 1, \forall a \in \{-1, 1\}^d\} = \text{conv}\{\pm e_i : i = 1, \ldots, d\}. \tag{5.16}$$

Among all five regular polytopes in 3 dimension, the remaining duality is between a dodecahedron and an icosahedron.

An *icosa-dodecahedron* is a truncated dodecahedron, obtained from a dodecahedron truncated at each vertex to the midpoint of incident edges. The number of facets is clearly $30 = 12 + 20$, the sum of the numbers of facets of an icosahedron and a dodecahedron. Its dual is known as a *rhombic triacontahedron*, which is a very special zonotope arising as quasicrystal.

Type	Figure	# Vertices	# Facets	# i-Faces
Cube(d)		2^d	$2d$	$\binom{d}{i}2^{d-i}$
Cross(d)		$2d$	2^d	$\binom{d}{i+1}2^{i+1}$
Dodecahedron		20	12	
Icosahedron		12	20	
Rhombic Triacontahedron		32	30	
Icosa-Dodecahedron		30	32	

5.6 Simple and Simplicial Polyhedra

Both a 3-cube and a dodecahedron are *simple* polytopes and their duals are *simplicial* polytopes.

More generally, the *simple* d-polytopes are such that each vertex is contained in exactly d facets, while the *simplicial* d-polytopes are such that each facet contains exactly d vertices.

Proposition 5.10 *For a d-polytope P, the following statements are equivalent:*

(a) *P is simple.*

(b) *Each vertex v of P is incident to exactly d-edges.*

(c) *For each vertex v of P, and for any k distinct edges incident to v, there exists a unique k-face containing the k edges.*

(d) *For each vertex v of P, and for any 2 distinct edges incident to v, there exists a unique 2-face containing the 2 edges.*

Proposition 5.11 *For a d-polytope P, the following statements are equivalent:*

(a) *P is simplicial.*

(b) *Each facet f of P contains exactly d-ridges.*

(c) *For each facet f of P, the intersection of any k distinct ridges contained in f is a $(d-k-1)$-face.*

(d) *For each vertex v of P, the intersection of any 2 distinct ridges contained in f is a $(d-3)$-face.*

5.7 Graphs and Dual Graphs

Proposition 5.1 shows that every interval of hight 2 is a diamond. This means one can define two types of graphs of a polytope. The *graph* of a polytope P is $G(P) = (V(P), E(P))$, where $V(P)$ is the set of vertices of P and $E(P)$ is the set of all edges each of which is represented as the pair of its two vertices. The *dual graph* of a polytope P is $G^D(P) = (F(P), R(P))$, where $F(P)$ is the set of facets of P and $R(P)$ is the set of all ridges each of which is represented as the pair of the two facets containing it. By the definition of duality, if Q is a dual of a polytope P, $G(P)$ is isomorphic to $G^D(Q)$.

6 Line Shellings and Euler's Relation

6.1 Line Shelling

Let $P = \{x \in \mathbb{R}^d : A_i\, x \leq 1,\ i = 1, 2, \ldots, m\}$ be a polytope. P has such a representation iff it contains the origin in its interior.

A *shelling* of the boundary of P is a sequence $F_1,\ F_2,\ \ldots,\ F_m$ of its facets such that $(\cup_{i=1}^{k-1} F_i) \cap F_k$ is a topological $(d-2)$-ball for each $2 \leq k \leq m-1$.

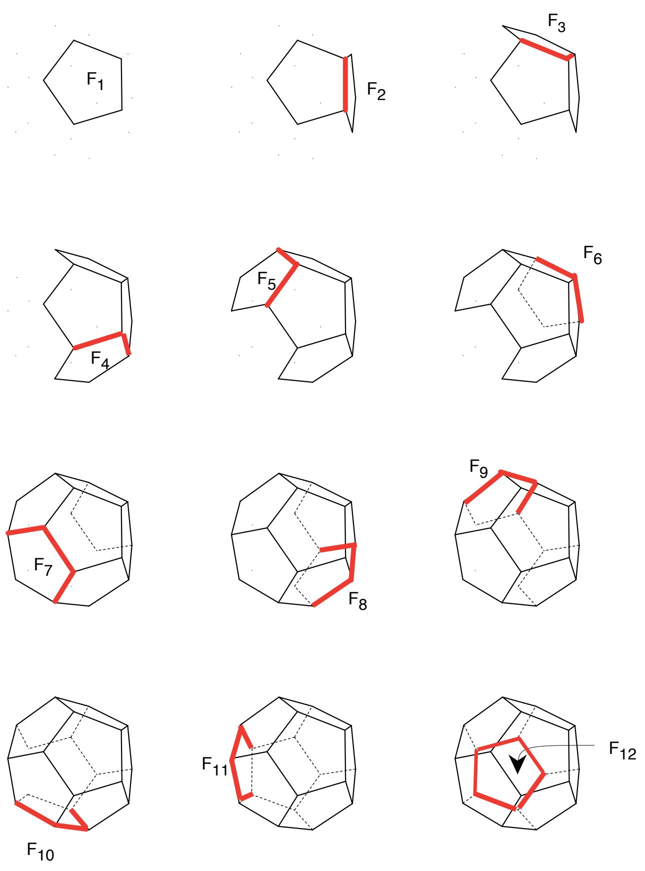


Figure 6.1: A Shelling of a Dodecahedron

The following is a fundamental theorem on polytopes which is extremely useful both theoretically and computationally.

Theorem 6.1 (Bruggesser-Mani [11] (1971)) *The boundary of every polytope admits a shelling.*

This theorem was used without proof by a Swiss mathematician Ludwig Schläfli (1901) to compute the Euler characteristic (to be discussed later) of convex polytopes. Seventy years later, an elegant proof was given. Here is the main idea. Bruggesser-Mani [11] proved a stronger theorem where any line in general position through an interior point of a polytope induces a particular shelling, known as a line shelling. Figure 6.2 illustrates this.

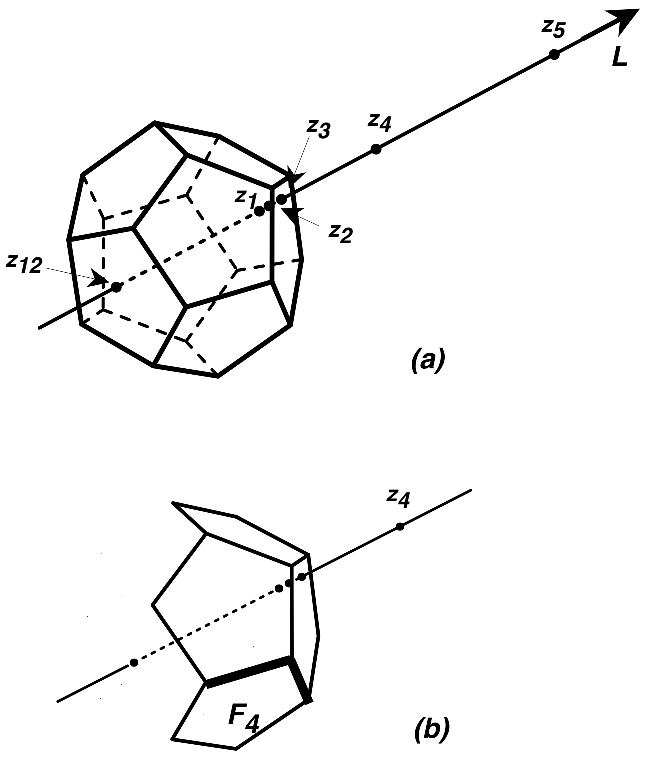


Figure 6.2: A Line Shelling

Imagine that a given polytope P is a planet earth and you are traveling along a generic oriented line L starting from some interior point. The first point z_1 to meet the boundary of P is a point on a facet. Let's call this facet F_1. Then you meet another point z_2 on the boundary of a halfspace spanned by a facet. Let's call this facet F_2. If you move a little forward from z_2, you "see" only two facets F_1 and F_2. This travel induces an ordering of facets as they become visible to you one by one. These facets are not all, and in the figure above, we have a sequence from F_1 up to F_6. The rests are to be ordered in a similar manner but from the opposite side of infinity on L. More precisely, you travel from the other side of infinity and follow the line along the orientation. From a point far from P, you see all facets not yet ordered. Now moving toward P, some facet becomes invisible. Let's call this facet

F_7. Then another facet becomes invisible that is F_8. When we reach the last facet F_{12}, all facets of P are ordered. Such an ordering of facets can be shown to be a shelling of P.

For any point z in $\mathbb{R}^d \setminus P$, the union of all facets visible from z forms a very special subset of the boundary. Let's all it the *visible hemisphere* from z. Similarly, we define the *invisible hemisphere* from z. The proof uses the fact that both the visible and the invisible hemispheres are shellable.

Before giving a formal proof of Theorem 6.1, let us interpret the line shelling in a dual polytope.

Consider a polytope P in $\mathbb{R}^d$ which contains the origin $\mathbf{0}$ in its interior. Thus, its H-representation can be of form

$$P = \{x \in \mathbb{R}^d : Ax \le \mathbf{1}\}.$$

for some $m \times d$ matrix A. The polar dual of P is

$$P^* = \text{conv}\{A_i^T : i = 1, \ldots, m\},$$

where A_i is the ith row of A.

For a generic $c \in \mathbb{R}^d$, sort the vertices A_i^T's of the dual polytope by the linear function $c^T x$. Namely, we suppose

$$A_1\, c > A_2\, c > \cdots > A_m\, c.$$

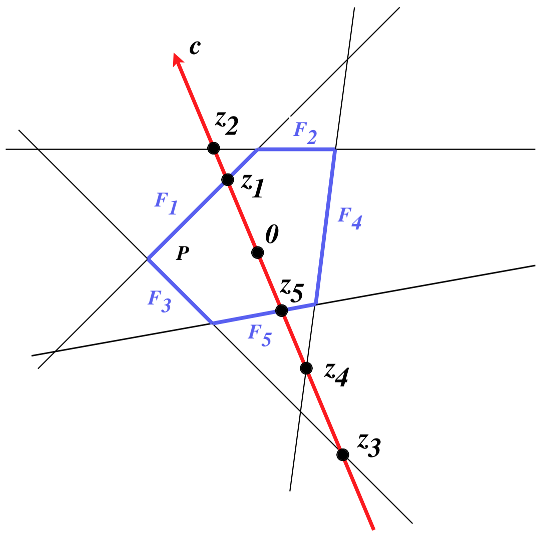
What is the meaning of this sorting for the original polytope P?

Geometrically, the parameterized line $L(\lambda) = \{\lambda\ c \mid \lambda \in \mathbb{R}\}$ meets each hyperplane determined by $A_i\ x = 1$ at a point, say z_i. Let λ_i denotes the parameter value at the intersection. Thus,

$$z_i = \lambda_i\, c \quad \text{and} \quad A_i\, z_i = 1.$$

Consequently:

$$1/\lambda_1 > 1/\lambda_2 > \cdots > 1/\lambda_k > 0 > 1/\lambda_{k+1} > \cdots > 1/\lambda_m.$$



This ordering is exactly the ordering produced by the space travel. (For any positive $1/\lambda_i$, the smaller its value, the farther away the point z_i is from the origin. What about for negative case?)

6.2 Cell Complexes and Visible Hemispheres

A *cell complex* or simply a *complex* K in $\mathbb{R}^d$ is a finite set of polytopes in $\mathbb{R}^d$ such that

(a) If $P \in K$ and F is a face of P, then $F \in K$.

(b) If $P \in K$ and $Q \in K$, then $P \cap Q$ is a common face of P and Q.

The *dimension* $\dim K$ of a complex K is the largest dimension of its members. A complex of dimension d is called a *d-complex*. The *body* $|K|$ of a complex K is the union of all members of K. The *boundary complex* ∂K of a complex K is defined as the subcomplex of K consisting of all elements in K contained in the boundary of its body.

The *complex* $K(P)$ of a polytope P is the set of all faces of P. The boundary complex $\partial K(P)$ is simply the set of all proper faces of P. Both the complex and the boundary complex of a polytope are *pure*, i.e., the maximal members have the same dimension.

A pure complex K is called *B-shellable* if the maximal members can be arranged in a sequence F_1, F_2, …, F_m in such a way that the subcomplex induced by $(\cup_{i=1}^{k-1} F_i) \cap F_k$ is B-shellable for each $2 \le k \le m$. By definition, the complex of a 0-polytope is B-shellable, and those are the only B-shellable 0-complexes.

A pure complex K is called *S-shellable* if the maximal members can be arranged in a sequence F_1, F_2, …, F_m in such a way that the subcomplex induced by $(\cup_{i=1}^{k-1} F_i) \cap F_k$ is B-shellable (S-shellable, respectively) for each $2 \le k \le m-1$ $(k = m)$. By definition, the boundary complex of a 1-polytope is S-shellable, and those are the only S-shellable 0-complexes.

These notions B-shellability and S-shllability are motivated by topological notions of balls and spheres. However, it should be observed that a B-shellable (S-shellable) complex is not necessarily a ball (a sphere). For example, the complex consisting of three 1-polytopes having a single vertex in common is B-shellable but not homeomorphic to a ball. We can add extra conditions to B-shellability (S-shellability) to enforce the resulting complexes to be topologically a ball (a sphere). The following is a combinatorial analogue of Theorem 6.1.

Theorem 6.2 *The boundary complex $\partial K(P)$ of a polytope is S-shellable.*

Before presenting a proof, we will give a nice application of this theorem. We define the *Euler characteristic* of a complex K as

$$\chi(K) = \sum_{i=0}^{\dim K} (-1)^i f_i(K), \tag{6.1}$$

where $f_i(K)$ is the number of i-dimensional members of K. It is easy to see that, for any two subcomplexes A and B of a complex, we have

$$\chi(A \cup B) + \chi(A \cap B) = \chi(A) + \chi(B). \tag{6.2}$$

Theorem 6.3 (Euler's Relation) *The following statements hold.*

(a) *If K is B-shellable, $\chi(K) = 1$.*

(b) *If K is S-shellable, $\chi(K) = 1 + (-1)^{\dim K}$.*

Proof. Both statements are obvious if $\dim K = 0$. By induction, we assume that both statements are true if $\dim K < d$ (≥ 1). First consider a B-shellable d-complex K. Since K is B-shellable, its d-polytopes can be ordered F_1, F_2, …, F_m in such a way that the subcomplex induced by $(\cup_{i=1}^{k-1} F_i) \cap F_k$ is B-shellable for each $2 \leq k \leq m$. When $m = 1$, clearly we have $\chi(K) = \chi(\partial K) + (-1)^d$. Since ∂K is S-shellable by Theorem 6.2 and has dimension $d-1$, the induction hypothesis implies

$$\chi(K) = \chi(\partial K) + 1 = 1 + (-1)^{d-1} + (-1)^d = 1.$$

Now we use the second induction on m. We assume that (a) is valid if $f_d(K) < m$, and then consider the case $f_d(K) = m$. Since the subcomplex A induced by $(\cup_{i=1}^{m-1} F_i)$ is B-shellable by the second induction, it satisfies (a). We denote by B the subcomplex induced by F_m. By using the fact that the subcomplex C induced by $(\cup_{i=1}^{m-1} F_i) \cap F_m$ is a B-shellable $(d-1)$-complex, by the first induction and (6.2), we have

$$\chi(K) = \chi(A) + \chi(B) - \chi(C) = 1 + 1 - 1 = 1.$$

The remaining proof of (b) is straightforward as we already established (a). Let K be a S-shellable d-complex. Then, its d-polytopes can be ordered F_1, F_2, … F_m such that the subcomplex induced by $(\cup_{i=1}^{k-1} F_i) \cap F_k$ is B-shellable (S-shellable, respectively) for each $2 \leq k \leq m-1$ ($k = m$). Note that the subcomplex A induced by $(\cup_{i=1}^{m-1} F_i)$ is B-shellable and satisfies (a). The subcomplex induced by F_m is also B-shellable and satisfies (a). The subcomplex C induced by $(\cup_{i=1}^{m-1} F_i) \cap F_m$ is a S-shellable $(d-1)$-complex, by the first induction and (6.2), we have

$$\chi(K) = \chi(A) + \chi(B) - \chi(C) = 1 + 1 - (1 - (-1)^{d-1}) = 1 + (-1)^d.$$

This completes the proof.

∎

Of special interest are topological properties of visible and invisible hemispheres of a polytope P. Please recall that for any point z in general position in $\mathbb{R}^d \setminus P$, the union of all facets of P is the *visible hemisphere* from z, denoted by $\text{vi}(P, z)$. Similarly, we define the *invisible hemisphere* from z, denoted by $\text{iv}(P, z)$.

Theorem 6.4 *Let P be a d-polytope in $\mathbb{R}^d$ for $d \geq 1$ and let z be any point in general position in $\mathbb{R}^d \setminus P$. Then the two subcomplexes of $K(P)$ induced by the visible hemisphere* $\text{vi}(P, z)$ *and the invisible hemisphere* $\text{iv}(P, z)$ *from z are B-shellable.*

Proof. We use induction on d. By inductive hypothesis, we assume that (*) the visible hemisphere $\text{vi}(P, z)$ and invisible hemisphere $\text{iv}(P, z)$ from z induce B-shellable subcomplexes when $d < k$, with $k \geq 2$. The statement (*) is obvious for $d = 1$.

Now we try to show that (*) is true for $d = k$. We take an oriented line L through z in general position which intersects the interior of P, and let z_1, z_2, …, z_m be the distinct intersections of L and the hyperplanes spanned by the facets F_1, …, F_m. Without loss of generality, the ordering is the one obtained by the space travel on L.

We first show that the visible hemisphere $\text{vi}(P, z)$ induces a B-shellable subcomplex. The point z is between z_i and z_{i+1} for some $1 \geq i < m$. If $i = 1$, $\text{vi}(P, z) = F_1$ and thus obviously $\text{vi}(P, z)$ induces a B-shellable subcomplex. We use induction again, on i, and assume by induction hypothesis that $\text{vi}(P, z)$ induces a B-shellable subcomplex for $i < h$ for some $h \geq 2$. We consider the case $i = h$. Note that $\text{vi}(P, z) = \text{vi}(P, z_i) \cup F_i$, where $\text{vi}(P, z_i)$ induces a B-shellable subcomplex by the inductive hypothesis. Now, we claim that $\text{vi}(P, z_i) \cap F_i$ induces a B-shellable $(d-2)$-subcomplex. This is true because this set is in fact the visible hemisphere $\text{vi}(F_i, z_i)$ from z_i in the $(d-1)$-dimensional space spanned by F_i. Since F_i is a $(d-1)$-ball and $\text{vi}(P, z_i) \cap F_i$ induces a B-shellable subcomplex, $\text{vi}(P, z)$ induces a B-shellable subcomplex. Essentially the same argument shows that the invisible hemisphere $\text{iv}(P, z)$ induces a B-shellable subcomplex.

This completes the double induction proof. ∎

Proof. (of Theorem 6.2) By definition, the boundary complex of any 1-polytope is S-shellable. We assume by induction that $\partial K(P)$ of any polytope of dimension $k - 1$ or less is shellable. Consider any k-polytope P. Let F be a facet of P, and let z be a point from which F is the only visible facet of P. This means that $\text{iv}(P, z)$ is a subcomplex of $\partial K(P)$ induced by all facets of P except F. By Theorem 6.4, $\text{iv}(P, z)$ is B-shellable. We claim that any shelling ordering of $\text{iv}(P, z)$ with F appended at the end is a shelling of $\text{iv}(P, z)$. For this, we only need to show that $\partial K(F)$ is S-shellable. Since F has dimension $k - 1$, this follows from the inductive hypothesis. This completes the proof. ∎

6.3 Many Different Line Shellings

The proof of shellability of polytope boundaries using the notion of line shelling provides many different ways to shell a polytope boundary. The choice of a line is restricted only by the two conditions (1) it has to intersects with the interior of the polytope, (2) it must intersects the hyperplanes spanned by the facets at distinct points.

Proposition 6.5 *The boundary of every polytope admits a shelling F_1, F_2, …, F_m with any one of the following prescribed conditions:*

(a) *both F_1 and F_m can be prefixed arbitrarily.*

(b) *all facets incident to a given vertex can be ordered earlier than any other facets.*

(c) *all facets incident to a given vertex can be ordered later than any other facets.*

7 McMullen's Upper Bound Theorem

7.1 Cyclic Polytops and the Upper Bound Theorem

The *moment curve* in $\mathbb{R}^d$ is the image of the real space $\mathbb{R}$ by the function $m(t)$ defined by

$$m(t) := (t, t^2, t^3, \ldots, t^d)^T. \tag{7.1}$$

The function $m(\cdot)$ is thus a parametric representation of the moment curve.

A *cyclic polytope* is the convex hull of n $(> d)$ distinct points on the moment curve, that is, $\text{conv}\{m(t_1), m(t_2), \ldots, m(t_n)\}$ for some $t_1 < t_2 < \cdots < t_n$. The following is a basic property of the moment curve.

Proposition 7.1 *Any $(d+1)$ distinct points on the moment curve $m(t)$ are affinely independent.*

Proof. Suppose $m(t_1), m(t_2), \ldots, m(t_{d+1})$ are affinely dependent for some $t_1 < t_2 < \cdots < t_{d+1}$. Then they must lie in some hyperplane, and thus there is a linear equation

$$a_0 + a_1x_1 + a_2x_2 \cdots + a_dx_d = 0$$

satisfied by all $m(t_i)$'s. It follows that the polynomial equation

$$a_0 + a_1t^1 + a_2t^2 \cdots + a_dt^d = 0$$

is satisfied by $(d+1)$ distinct values of t, which contradicts to the fundamental theorem of algebra. ∎

Proposition 7.1 implies that the cyclic polytope $\text{c}(d, n)$ is a simplicial polytope and its dual is a simple polytope.

We will see that for any fixed d and n, its combinatorial structure is unique. Thus, we will denote anyone of them by $\text{c}(d, n)$, and their duals by $\text{c}^*(d, n)$.

McMullen's upper bound theorem is one of the most important theorems in the theory of convex polytopes.

Theorem 7.2 (McMullen's Upper Bound Theorem [38] (1970)) *For any fixed d and n, the maximum number of j-faces of a d-polytope with n vertices is attained by the cyclic polytope $\text{c}(d, n)$ for all $j = 0, 1, \ldots, d-1$. Equivalently, for any fixed d and n, the maximum number of j-faces of a d-polytope with n facets is attained by the dual cyclic polytope $\text{c}^*(d, n)$ for all $j = 0, 1, \ldots, d-1$.*

There is an explicit formula for $f_j(\text{c}(d, n))$ for $j = 0, 1, \ldots, d-1$. The following gives essentially a half of these formulas.

Lemma 7.3 *For any $d \geq 0$ and $n \geq d+1$,*

(a) $f_{j-1}(\text{c}(d, n)) = \binom{n}{j}$, *for* $0 \leq j \leq \left\lfloor \frac{d}{2} \right\rfloor$.

(b) $f_k(\text{c}^*(d, n)) = \binom{n}{d-k}$, *for* $\left\lceil \frac{d}{2} \right\rceil \leq k \leq d$.

Proof. By duality, two statements (a) and (b) are equivalent. Let's prove (a).

Consider the cyclic polytope $P = \text{conv}\{m(t_1), m(t_2), \dots, m(t_n)\}$ with $t_1 < t_2 < \cdots < t_n$. Let $0 \le j \le \lfloor \frac{d}{2} \rfloor$. Take the first j points $m(t_1), m(t_2), \dots, m(t_j)$, and consider the hyperplane h determined by

$$a_0 + a_1x_1 + \cdots + a_dx_d = 0, \tag{7.2}$$

where the coefficients a_i's coincide with those in the polynomial

$$p(t) := a_0 + a_1t + \cdots + a_dt^d \equiv \Pi_{i=1}^{j}(t - t_i)^2. \tag{7.3}$$

Note that the assumption $j \le \lfloor \frac{d}{2} \rfloor$ implies that the polynomial $p(t)$ has degree at most d. Observe that h contains all the points $m(t_i)$ for $i = 1, \dots, j$. Furthermore, the remaining points $m(t_i)$ for $i = j+1, \dots, n$ are strictly on the positive side of the hyperplane. This means that $\text{conv}\{m(t_1), \dots, m(t_j)\}$ is a face of P. Since the above discussion works exactly the same way if we take any j points, every j points from $\{m(t_1), m(t_2), \dots, m(t_n)\}$ determine a $(j-1)$-face. ∎

Lemma 7.3 implies an interesting property of the cyclic polytope. Namely, if $d \ge 4$, then every pair of vertices forms an edge. This means that the graph of $\text{c}(d, n)$ is a complete graph for $d \ge 4$. This is not very intuitive because this phenomenon does not occur in the 3-dimensional space.

The proof of Lemma 7.3 gives some ideas on how to determine the facets of $\text{c}(d, n)$. Which d-tuples of points from $\{m(t_1), m(t_2), \dots, m(t_n)\}$ span a facet? Since any d-tuple $\{m(t_{j_1}), m(t_{j_2}), \dots, m(t_{j_d})\}$ is affinely independent and thus it spans a hyperplane. Whether or not it defines a facet is thus equivalent to whether or not all the remaining points are on one side of the hyperplane. This turns out to be quite easy to check through a combinatorial condition, known as *Gale's evenness condition*.

Exercise 7.1 Find a necessary and sufficient condition for a set of d points $m(t_{j_1})$, $m(t_{j_2})$, ..., $m(t_{j_d})\}$ to determine a facet of the cyclic polytope.

Lemma 7.3 gives essentially a half of the f-vector of the cyclic polytope. Yet, by using the fact that it is simplicial, the remaining information on the f-vector will be shown to be determined uniquely.

7.2 Simple Polytopes and h-vectors

To establish the Upper Bound Theorem, Theorem 7.2, we shall prove the dual statement:

> For any fixed d and n, the maximum number of j-faces of a d-polytope with n facets is attained by the dual cyclic polytope $\text{c}^*(d, n)$ for all $j = 0, 1, \dots, d-1$.

We have two basic steps. **The first step** is to show that it is sufficient to consider only simple polytopes as maximizers of the number of j-faces, for a fixed number of facets. More precisely, we have:

Theorem 7.4 *For any d-polytope P with n facets in $\mathbb{R}^d$, there exists a simple d-polytope P' with n facets such that $f_j(P) \leq f_j(P')$ for all $j = 0, 1, \ldots, d-1$.*

Proof. We only have to argue that a small perturbation of each inequality defining P does not decrease the number of j-faces. We leave the proof to the reader. Use Theorem 3.14 and analyze how a face changes as an inequality gets perturbed slightly toward enlarging the polytope. ∎

The second step is to show that among all simple d-polytopes with n facets, the dual cyclic polytope $\mathrm{c}^*(d,n)$ maximizes the number of j-faces for all $j = 0, 1, \ldots, d-1$.

For the rest of this section,

(*) we only consider simple d-polytopes with n facets.

We denote by $\square(d,n)$ the set of all simple d-polytopes in $\mathbb{R}^d$ with n facets.

For any $P \in \square(d,n)$, consider a linear program $c^T x$ subject to $x \in P$. Assume that c is generic and in particular, the LP orientation $\overrightarrow{G}(P)$ of the graph $G(P)$ is well-defined, namely, $\overrightarrow{G}(P)$ is a directed graph with the unique sink (maximizer) vertex and the unique source (minimizer) vertex.

Now, we denote by $h_k(\overrightarrow{G}(P))$ the number of vertices of indegree k, for each $k = 0, 1, \ldots, d$. Clearly, $h_0(\overrightarrow{G}(P)) = h_d(\overrightarrow{G}(P)) = 1$. We shall eventually write $h_k(P)$ instead of $h_k(\overrightarrow{G}(P))$, as we will see below that this number does not depend on c at all.

Lemma 7.5 *For any polytope $P \in \square(d,n)$ and any generic $c \in \mathbb{R}^d$, the value $h_k(\overrightarrow{G}(P))$ depends only on P, and in particular, it does not depend on c. Thus, it can be denoted as $h_k(P)$.*

Proof. Let P be a polytope $P \in \square(d,n)$ and take a generic $c \in \mathbb{R}^d$. We denote by (F, v) a pair of a k-face F of P and a vertex v on F which is the unique sink on F. It is clear that the number of such pairs (F, v) is the number of k-faces, $f_k(P)$.

Now, fix a vertex v of P and fix k. The number of such pairs (F, v) can be counted by using Proposition 5.10. Namely, there are exactly $\binom{r}{k}$ k-faces incident to v whose sink is v, where r is the indegree of v in $\overrightarrow{G}(P)$. Now ranging v over all vertices, we have

$$\sum_{r=0}^{d} h_r(\overrightarrow{G}(P)) \binom{r}{k} = f_k(P), \quad \text{for } k = 0, 1, \ldots, d. \tag{7.4}$$

Now the system of linear equations can be written using a matrix and vectors as

$$\begin{bmatrix} \binom{0}{0} & \binom{1}{0} & & & \cdots & \binom{d}{0} \\ 0 & \binom{1}{1} & \binom{2}{1} & & \cdots & \binom{d}{1} \\ 0 & 0 & \ddots & \cdots & & \vdots \\ 0 & 0 & 0 & \binom{k}{k} & \cdots & \binom{d}{k} \\ 0 & 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 \cdots & \binom{d}{d} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_k \\ \vdots \\ h_d \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_k \\ \vdots \\ f_d \end{bmatrix}. \tag{7.5}$$

The matrix on the LHS is obviously nonsingular, because it is upper triangular and the diagonal entries are all 1's. This means that h_j's are determined uniquely by f_j's, and thus h_j's are independent of c. This completes the proof. ∎

There are different ways to solve the equation (7.5) in terms of the h-vector $h(P) := (h_0, h_1, \ldots, h_d)$. The resulting formula for h in terms of f is given by

$$h_i(P) = \sum_{k=0}^{d} (-1)^{k-i} \binom{k}{i} f_k(P), \quad \text{for } i = 0, 1, \ldots, d. \tag{7.6}$$

This together with Lemma 7.3 provides us with simple explicit formulas for a half of the h-vector of the dual cyclic polytope.

Lemma 7.6

$$h_i(\mathrm{c}^*(d,n)) = \binom{n-i-1}{d-i}, \quad \textit{for } i = \lceil d/2 \rceil, \ldots, d. \tag{7.7}$$

Proof. Substitute $f_k(P)$ in (7.6) with the explicit formula for $f_k(\mathrm{c}^*(d,n))$ in Lemma 7.3 (b). Exercise. ∎

The remaining part of the h vector comes for free, as we observe that the h-vector is symmetric, namely, by the definition of h_i,

$$h_i(P) = h_{d-i}(P), \quad \text{for } i = 0, 1, \ldots, \lfloor d/2 \rfloor, \tag{7.8}$$

where the RHS counts the LHS using the h-vector with the reversed orientation by the vector $-c$. These equations, expressed in terms of f-vector via (7.6), are known as the *Dehn-Sommerville Relations.*

Theorem 7.7 (The Dehn-Sommerville Relations) *Every simple d-polytope P satisfies the following equations.*

$$\sum_{k=i}^{d} (-1)^k \binom{k}{i} f_k(P) = \sum_{k=d-i}^{d} (-1)^{k-d} \binom{k}{d-i} f_k(P), \quad \textit{for } i = 0, 1, \ldots, \lfloor d/2 \rfloor. \tag{7.9}$$

More explicitly, the first two equations are

$$\sum_{k=0}^{d} (-1)^k f_k(P) = f_d(P) = 1, \quad \textit{(i.e. Euler's Relation)}, \tag{7.10}$$

$$-f_1(P) + 2f_2(P) - 3f_3(P) + \cdots + (-1)^d d f_d(P) = -f_{d-1}(P) + d f_d(P). \tag{7.11}$$

The equation (7.5) shows that each f_j is a nonnegative combination of h_j's. Therefore, the following is a strengthening of the Upper Bound Theorem, saying that the h-vector is component-wise maximized by the dual cyclic polytope.

Theorem 7.8 (A Strengthened Upper Bound Theorem) *For any simple d-polytope P with n facets the following inequalities hold.*

$$h_i(P) \le h_i(\mathrm{c}^*(d,n)), \quad \textit{for } i = 0, 1, \ldots, d. \tag{7.12}$$

Proof. Let P be a simple d-polytope with n facets. The claimed inequalities are trivial for $i = 0$ and $i = d$. By the symmetry of the h-vector, we only need to show

$$h_i(P) \leq h_i(\mathrm{c}^*(d,n)) \equiv \binom{n-i-1}{d-i}, \quad \text{for } i = \lceil d/2 \rceil, \ldots, d.$$

We use induction on i but with decreasing values. Suppose the theorem is valid for $i = k+1$ ($k < d$), and consider the case $i = k$.

We claim two inequalities for h-vectors. First we observe that for any facet F of P and for any i,

$$h_i(F) \leq h_{i+1}(P). \tag{7.13}$$

This is valid because we can select a generic c such that all the vertices in F take the object value higher than any other vertices of P. Note that the values $h_i(F)$ and $h_{i+1}(P)$ are invariant over choices of c. Secondly, we have

$$\sum_F h_i(F) = (i+1)h_{i+1}(P) + (d-i)h_i(P). \tag{7.14}$$

The summation in LHS is over all faces F of P. This equation can be verified once we observe that every vertex of a face with indegree i in the face has indegree i or $i+1$ in P. If it has indegree i in P, there are exactly $(d-i)$ facets containing it that preserve the same indegree. If it has indegree $i+1$ in P, there are exactly $(i+1)$ facets containing it that decrease its indegree by one.

Now we look at the inductive step for $i = k$. By the two inequalities (7.14) and (7.13), we have

$$(k+1)h_{k+1}(P) + (d-k)h_k(P) = \sum_F h_k(F) \leq n h_{k+1}(P). \tag{7.15}$$

This implies

$$(d-k)h_k(P) \leq (n-k-1)h_{k+1}(P), \quad \text{or equivalently,} \tag{7.16}$$

$$h_k(P) \leq \frac{n-k-1}{d-k} h_{k+1}(P). \tag{7.17}$$

Now we use the inductive hypothesis for $i = k+1$ to get

$$h_k(P) \leq \frac{n-k-1}{d-k} h_{k+1}(P) \leq \frac{n-k-1}{d-k} \binom{n-k-2}{d-k-1} = \binom{n-k-1}{d-k}. \tag{7.18}$$

This completes the proof. ∎

While the h-vector of the cyclic polytope is extremely simple, its f-vector is rather complicated. The formula can be written explicitly using (7.5), (7.6) and (7.8). We here present a formula for $f_0(\mathrm{c}^*(d,n))$ which is quite simple.

Theorem 7.9 *The maximum number of vertices a d-polytope with n facets can have is realized by the dual cyclic polytope and is*

$$f_0(\mathrm{c}^*(d,n)) = \binom{n-\lceil d/2 \rceil}{n-d} + \binom{n-\lfloor d/2 \rfloor - 1}{n-d}. \tag{7.19}$$

By duality, this number coincides with $f_{d-1}(\mathrm{c}(d,n))$.

Proof. Left to the reader. Hint: use the identity:

$$\binom{n}{0} + \binom{n+1}{1} + \cdots + \binom{n+s}{s} = \binom{n+s+1}{s}.$$

∎

8 Basic Computations with Polyhedra

Consider a system of m linear inequalities in d variables

$$Ax \le b. \tag{8.1}$$

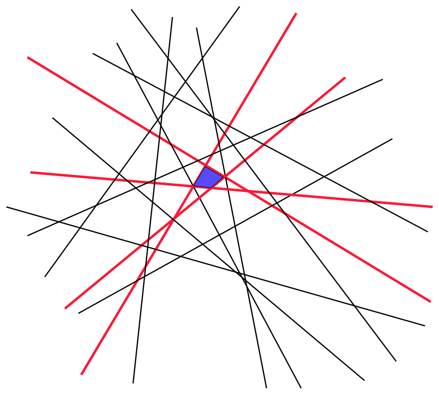
An inequality $A_i x \le b_i$ is called *redundant* in (8.1) if the set of solutions to (8.1) stays unchanged when the inequality is removed from the system. An equivalent condition is that there is no x satisfying $A_i x > b_i$ and $A_j x \le b_j$ for all $j \ne i$.

In this section, we study basic problems in polyhedral computation such as the following two problems:

Problem 8.1 [Single H-Redundancy]
Input: A rational matrix $A \in \mathbb{Q}^{m \times d}$, a rational vector $b \in \mathbb{Q}^m$ and an index $k \in [m] := \{1, \dots, m\}$
Output: **Yes** if $A_k x \le b_k$ is redundant in $Ax \le b$, **No** otherwise.

Problem 8.2 [H-Redundancy Removal]
Input: A rational matrix $A \in \mathbb{Q}^{m \times d}$, a rational vector $b \in \mathbb{Q}^m$
Output: An equivalent subsystem of $Ax \le b$ which is free of redundancies.

The second problem can be solved by solving the first problem for each inequalities, but interestingly, one can do better than that by dynamically selecting the ordering of inequalities to be processed.



The figure above illustrates the H-redundancy problem. The blue region is the feasible region $P = \{x : Ax \le b\}$. The output of the computation is the set of inequalities indicated in red that are essential in the H-representation. Often, the size of output is much smaller than the size of input.

Naturally one can pose the same problems for V-polyhedra. It turns out that those problems can be reduced to the H-redundancy problems. We will see that the H-redundancy problems can be reduced further to the H-redundancy problems for the special case of H-cones. These transformations are discussed in Section 8.4.

Here is a closely related problem that should be solved before the H-Redundancy Removal is solved.

Problem 8.3 [H-Dimension]
Input: A rational matrix $A \in \mathbb{Q}^{m\times d}$, a rational vector $b \in \mathbb{Q}^m$
Output: The dimension of the polyhedron $P = \{x : Ax \le b\}$.

A typical algorithm for this computes not only the dimension of P, but also a relative interior point of P, see Section 8.3. One can then embed the polytope in a lower-dimensional space so that P becomes full-dimensional.

These problems are much easier than other problems in polyhedral computation such as the representation conversion between V- and H-representations and computing the volume of a polytope. In fact, the problems discussed in this section are all polynomially solvable in the size of input.

The main goal of this section is to present many algorithms which are not only polynomial-time but also best possible in terms of the number of LP's that must be solved, or of the size of LP's that must be solved when the number of LP's to be solved is fixed.

For this purpose, we use the notion of LP complexity, where we count the number of LP's and their sizes as a complexity measure. This makes sense only when solving LP's dominates other computations such as solving systems of linear equations of sizes of same order. This applies very well to all problems in this Section.

We denote by $\mathrm{LP}(d, m)$ the time necessary to solve any LP with d variables and m inequality constraints: $\max c^T x$ subject to $Ax \le b$, where A is $m \times d$ rational matrix. We consider $\mathrm{LP}(d, m)$ is an upper bound time measured by big-oh O notation, such as $O(md^3)$ or $O(e^{\sqrt{d \log m}})$. Unlike the usual way to measure the LP complexity using the binary encoding length L of input, we simply ignore L. The main reason is that practically all of implementations of LP algorithms depend hardly on L, but essentially and polynomially on d and m. Further more, we are mostly interested in the case when m is much larger than d and at least as large as $2d$. This practical observation leads to that

Assumption 8.4 *We assume that* $\mathrm{LP}(d, m)$ *satisfy the following assumptions.*

(a) $\mathrm{LP}(d, m) = \mathrm{LP}(d + c_1, m + c_2)$ *for any constants* c_1 *and* c_2.

(b) $\mathrm{LP}(d, m)$ *is at least of order* md^2*, that is,* $\Omega(md^2)$.

The first assumption is based on the fact that LP is solvable in a polynomial time. The second assumption is based on the fact that solving a system of linear inequalities is at least as hard as solving a system of linear equations of the same size (up to constant factor), and the Gaussian elimination has $\Omega(md^2)$ complexity. This second assumption will be used throughout this chapter to argue that the time to solve a linear equality system or to compute a rank of an $m \times d$ matrix is dominated by $\mathrm{LP}(d, m)$.

8.1 Single H-Redundancy Checking

Here we show that Problem 8.1 is linearly equivalent to the linear programming. The one direction is rather obvious, that is, the Single H-redundancy checking can be done by a single LP of the same size.

Proposition 8.5 *Problem 8.1 has No answer if and only if the following LP with $I = [m]$:*

$$\begin{array}{lll} \text{Test}(I,\,k): & \textit{maximize} & A_k x \\ & \textit{subject to} & \\ & & A_i x \le b_i, \quad \forall i \in I \setminus \{k\} \\ & & A_k x \le b_k + 1 \end{array} \tag{8.2}$$

has an optimal solution whose optimal value is strictly greater than b_k.

Note that the reduction is not only polynomial but linear. Surprisingly, there is a linear reduction from the linear programming (the linear feasibility) to the Single H-redundancy.

Proposition 8.6 *The system $Ax \le b$ is consistent if and only if a special case of Problem 8.1:*

$$\textit{is the inequality } x_0 \le 0 \textit{ redundant in } Ax \le b\, x_0 \textit{ and } x_0 \le 0 \tag{8.3}$$

has No answer.

Proof. Suppose the $x_0 \le 0$ is redundant. This is equivalent to the statement that there exists no (x, x_0) such that $Ax \le b\ x_0$ and $x_0 > 0$. This in turn is equivalent to the inconsistency of $Ax \le b$. ∎

We have shown the linear equivalence of the Single H-redundancy checking and the LP. This implies that any redundancy checking algorithm is at least as powerful as an LP algorithm.

In the next section, we will see that removing all H-redundancies can be easier than solving m LP's of size (d, m) that takes time $m \times \text{LP}(d, m)$ if the system is highly redundant.

8.2 H-Redundancy Romoval

Here we discuss the problem of removing all redundancies from an H-representation of a polyhedron, i.e., Problem 8.2.

We shall assume that the input of Problem 8.2 is "clean" in the sense that the underlying polyhedron $P = \{x : Ax \le b\}$ is full-dimensional and no inequality is a positive multiple of another one. This assumption can be met if the preprocessing is done, namely, by embedding the polyhedron in an appropriate subspace. This part will be discussed in Section 8.3.

As we have seen in Section 8.1, removing all redundancies can be done in $m \times \text{LP}(d, m)$ time. Can one do better than this? Here we present an algorithm due to Clarkson [15] which runs much faster than the naive algorithm when the number s of nonredundant inequalities is small relative to m.

Let $Ax \le b$ be an input system. We assume that a point $z \in \mathbb{Q}^d$ is given satisfying $Az < b$, an interior point of the feasible region $P = \{x : Ax \le b\}$. At the general stage of the algorithm, we have already detected a row index set I such that the inequality $A_i x \le b_i$ is nonredundant for $Ax \le b$, for each $i \in I$. Let j be an row index which is not tested yet, i.e. $j \in [m] \setminus I$. Clarkson's algorithm either detects kth inequality is redundant or finds a row index $j \in [m] \setminus I$ such that $A_j x \le b_j$ is nonredundant.

```
     procedure Clarkson($A$,$b$,$z$,$I$,$k$)
     begin
          test whether $A_k x \le b_k$ is redundant in $A_{I\cup\{k\}} x \le b_{I\cup\{k\}}$
          by solving the LP Test($I \cup \{k\}$, $k$) with optimal solution $x^*$
          if nonredundant then
(c1)           return (1, RayShoot($A$,$b$,$z$,$x^* - z$))    //Returns an essential index
          else
(c2)           return (0, $k$)       //Returns an redundant index
          endif
     end
```

Here, the procedure RayShoot(A,b,z,r) returns an index j of a facet-inducing hyperplane $\{x : A_j x = b_j\}$ hit by the ray starting from z along the direction r. It can be easily implemented by analyzing symbolically the ray starting from $z + (\epsilon, \epsilon^2, \dots, \epsilon^d)^T$ along the direction r for sufficiently small $\epsilon > 0$.

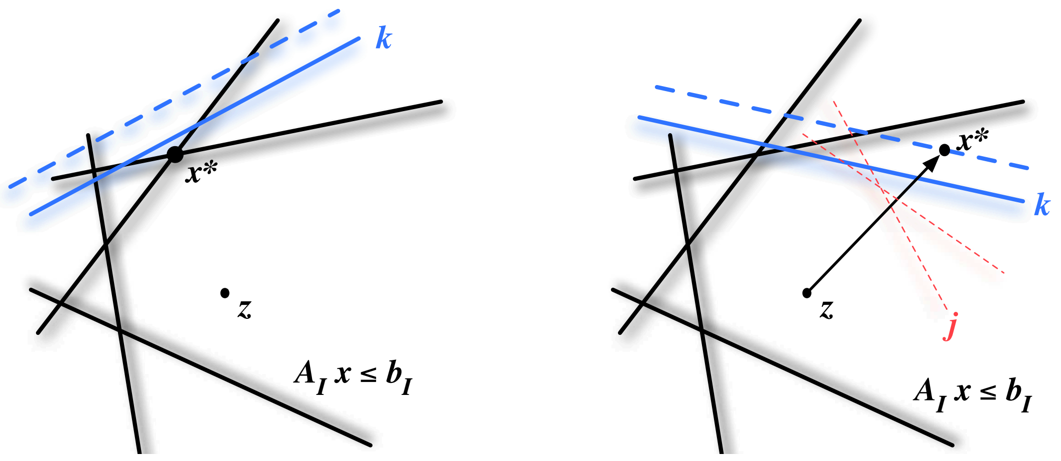


Figure 8.1: Clarkson's Algorithm: Left (k is redundant), Right(An essential j is found)

Exercise 8.1 Write a procedure RayShoot(A,b,z,r) following the specification above. It should be specific enough to be implemented with high level computer languages like C and C++.

Here is the complete algorithm to remove all redundancies. We assume that an interior point z of $P = \{x : Ax \le b\}$ is given.

```
procedure RedundacyRemovalClarkson($A$,$b$,$z$)
begin
     set $I := \emptyset$, $J := [m]$
     repeat
          select an index $k$ from $J$
```

```
            $(\alpha, j)$ =Clarkson($A$,$b$,$z$,$I$,$k$)
            if $\alpha = 1$ **then** $I := I \cup \{j\}$    //Increment the essential set $I$
            $J := J \setminus \{j\}$
        **until** $J = \emptyset$
        return $I$
    **end**
```

Theorem 8.7 *The complexity of Clarkson's algorithm to find a minimal equivalent subsystem of $Ax \leq b$ is $m \times \mathrm{LP}(d, s)$ where s is the number of nonredundant constraints in $Ax \leq b$.*

Proof. At each step, Clarkson's algorithm either finds an row index k to be a redundant inequality row index or discovers a new row index $j \neq k$ for which $A_j x \leq b_j$ is essential. Since the size of an LP solved has d variables and at most $s + 1$ constraints, the complexity follows. Note that the complexity of a ray shooting is $O(md)$. Since the number of ray shooting is at most s, the total time $O(smd)$ of ray shooting is dominated by $m \times \mathrm{LP}(d, s)$. ∎

8.3 Computing H-Dimension

It is often important to know the dimension of a polyhedron. When a polyhedron is a V-polyhedron with representation, it is very easy to compute its dimension. More precisely, if P is a V-polyhedron for some generator matrices $d \times s$ matrix V and $d \times t$ matrix R, i.e.,

$$P = \{x : x = V\lambda + R\mu, \mathbf{1}^T\lambda = 1, \lambda \geq \mathbf{0}, \mu \geq \mathbf{0}\},$$

then the dimension of P is easily computable, namely by the formula,

$$\dim P = \operatorname{rank} \begin{bmatrix} V & R \\ \mathbf{1}^T & \mathbf{0}^T \end{bmatrix} - 1.$$

However, for an H-polyhedron

$$P = \{x : Ax \leq b\}$$

its dimension is nontrivial to compute. Why nontrivial? It is simply because if one knows the dimension, one can decide whether P is empty or not, that is the linear feasibility problem, equivalent to LP. Then, the next question is how many LP's one has to solve to determine the dimension. Obviously, at least one. It is not hard to see that at most m LP's is sufficient.

In this section, we show that one can compute the dimension by solving at most d LP's. As a byproduct, one also finds a point in the relative interior of P.

The first step is to try to find an interior point of P. If it is successful, the dimension is of course d. One can easily see that the following LP will detect the full-dimensionality:

$$\begin{array}{lrcl} \text{maximize} & x_0 & & \\ \text{subject to} & & & \\ & Ax + \mathbf{1}x_0 & \leq & b, \\ & x_0 & \leq & 1. \end{array} \tag{8.4}$$

More precisely, we have three cases, depending on the outcome of the LP. Let x^* be an optimal solution and let x_0^* be the optimal value.

Case 1: $x_0^* > 0$. In this case, an optimal solution x^* is an interior point and $\dim P = d$.

Case 2: $x_0^* < 0$. In this case, the polyhedron P is empty and $\dim P = -1$.

Case 3: $x_0^* = 0$. In this case, the polyhedron P is neither full-dimensional nor empty.

In case 3, we must do more computation. For that, we can make use of a dual optimal solution (s^*, t^*) for the dual LP:

$$\begin{array}{lrcrcl} \text{minimize} & b^T s & & + & t & \\ \text{subject to} & & & & & \\ & A^T s & & & = & \mathbf{0}, \\ & \mathbf{1}^T s & & + & t = & 1, \\ & s & \geq \mathbf{0}, & & t \geq & 0. \end{array} \tag{8.5}$$

By strong duality, the dual optimal value is zero. This means that s^* cannot be totally zero. Let $I = \{i : s_i^* > 0\}$. By the complementary slackness, at any feasible solution (x, x_0) with $x_0 = 0$ (i.e., at any solution x for $Ax \leq b$), every inequality in $A_I x \leq b_I$ must be tight. We might do even further. By Gaussian elimination, we can recognize all other inequalities in $Ax \leq b$ that are forced to be equalities provided $A_I x = b_I$. Let us merge I with these dependent equality indices, and call it $\hat{I}$. Now we are ready to solve another LP to find more implicit equalities in the remaining system. For this, let $C := \hat{I}$, and $D := [m] \setminus C$, and set up an LP:

$$\begin{array}{lrrcl} \text{maximize} & & x_0 & & \\ \text{subject to} & & & & \\ & A_C x & & = & b_C, \\ & A_D x & +\mathbf{1} x_0 & \leq & b_D, \\ & & x_0 & \leq & 1. \end{array} \tag{8.6}$$

At an optimal solution (x^*, x_0^*), there are only two cases because x^* cannot be negative this time. When $x_0^* > 0$, the solution x^* is a relative interior point, and the dimension of P is easily computed. It is d minus the maximum number of independent equalities in $A_I x = b_I$. When $x_0^* = 0$, we do essentially the same thing as we did at the very first stage: use the dual optimal solution to recognize implicit equalities in $A_D x \leq b_D$. Then extend them with possible dependent equalities. Another LP should be solved with extended C and its complement D. Since every time an LP is solved, at least one independent implicit equality is found. This shows that at most d LP's will be solved until a relative interior point is found. Thus we have:

Theorem 8.8 *Problem 8.3 can be solved in $d \times \mathrm{LP}(d, m)$ time.*

Exercise 8.2 (Embedding a Polyhedron) Given a point z in the relative interior of $P = \{Ax \leq b\}$, explain a method to embed P to a lower dimensional space so that it is full-dimensional there.

8.4 Reducing the Nonhomogeneous Case to the Homogeneous Case

We define the *homogenization* of a system $Ax \le b$ as the new system with one extra nonnegative variable x_0,

$$Ax \le b\, x_0 \text{ and } x_0 \ge 0. \tag{8.7}$$

Proposition 8.9 *Let $Ax \le b$ be a consistent system. An inequality $A_i x \le b_i$ is redundant in the system if and only if the corresponding inequality $A_i x \le b_i\, x_0$ is redundant in the homogenization.*

Exercise 8.3 Prove the proposition above. Show that the assumption that $Ax \le b$ being consistent is necessary by providing a small example in which $A_i x \le b_i$ is redundant in $Ax \le b$ but $A_i x \le b_i\, x_0$ is nonredundant in the homogenization.

What we have shown above is that the H-redundancy removal for cones solves the more general problem for polyhedra.

What about for a V-polyhedron? Can we reduce the redundancy removal for V-polyhedra to the one for V-cones? Consider a V-polyhedron with generator pair (V, R) where $V \in \mathbb{Q}^{d\times s}$ and $R \in \mathbb{Q}^{d\times t}$:

$$P_V(V,R) := \{x : x = V\lambda + R\mu, \mathbf{1}^T\lambda = 1, \lambda \ge \mathbf{0}, \mu \ge \mathbf{0}\}.$$

Let v_j denote the jth column of V, and r_k denote the kth column of R. We say a generator v_j (r_k) is *redundant* for $P_V(V,R)$ if removing v_j from V (r_k from R, respectively) does not alter the polyhedron.

Proposition 8.10 *For $V \in \mathbb{Q}^{d\times s}$ and $R \in \mathbb{Q}^{d\times t}$, a generator v_j (r_k) is redundant for $P_V(V,R)$ if and only if the corresponding generator $\begin{bmatrix} v_j \\ 1 \end{bmatrix}$ ($\begin{bmatrix} r_j \\ 0 \end{bmatrix}$, respectively) is redundant in the homogenization*

$$C_V(\hat{R}) := \{x : x = \hat{R}\mu, \mu \ge \mathbf{0}\}, \textit{where } \hat{R} = \begin{bmatrix} V & R \\ \mathbf{1}^T & \mathbf{0}^T \end{bmatrix}.$$

Proof. The proof is straightforward. Left to the reader. ∎

Now, we know that both the H-redundancy removal and the V-redundancy romoval for cones are as powerful as those for polyhedra. Finally, we have the duality of H- and V-redundancy removals which implies that an algorithm for one problem solves both.

Proposition 8.11 *Let $A \in \mathbb{Q}^{m\times d}$. Then, each inequality $A_i x \le 0$ is redundant in $Ax \le \mathbf{0}$ if and only if the corresponding generator A_i^T is redundant for $C_V(A^T)$.*

Proof. Let $A_i x \le 0$ be redundant in $Ax \le \mathbf{0}$. This means there exists no x such that $A_i x > 0$ and $A_j x \le 0$ for all $j \ne i$. By the Farkas Lemma (Exercise 3.4), this is equivalent to the existence of $\mu \ge \mathbf{0}$ such that $A_i^T = \sum_{j\ne i} A_j^T \mu_j$. This is equivalent to saying A_i^T is redundant for $C_V(A^T)$. This completes the proof. ∎

Exercise 8.4 (Dimensionality and Linearity) Given a point z in the relative interior of $C_H(A) := \{Ax \le 0\}$, explain a method to find a basis of the linearity space of $C_V(A^T)$.

9 Polyhedral Representation Conversion

The Minkowski-Weyl Theorem, Theorem 3.9, shows that every convex polyhedron has two representations, an H-representation and a V-representation. The associated problem of computing a (minimal) V-representation from a H-representation or its converse is known as the *representation conversion problem* for polyhedra.

One important characteristic of the representation conversion problem is that the size of output is not easy to measure in terms of the size of input. For example, for a d-cube having $2d$ facets and 2^d vertices, the H-to-V conversion has output whose size is exponential in the input size and the V-to-H conversion has the output size very small relative to the input size.

Given this diversity of output sizes, an ideal algorithm for the conversion problem must be sensitive to the output size, as opposed to optimal for the worst-case output size of a given input size. An algorithm is called *output-polynomial* if its running time is bounded by a polynomial function of both the input size and the output size.

Also, we must take account of the memory footprint. Some algorithms need to store a large amount of data in the memory, while others simply do not store anything except the input data and a few more. We say an algorithm is *compact* if its space is bounded by a polynomial function of the input size only.

One might call an algorithm ideal if it is both compact and output-polynomial. For the representation conversion problem, there is no known output-polynomial algorithm in general. However for the special cases of various nondegeneracy, compact output-polynomial algorithms are known, typically based on the reverse-search paradigm, see Section 9.2.

9.1 Incremental Algorithms

In this section, we present a classical finite algorithm, known as the double description (DD) method [40]. It can be also considered as a constructive proof of Minkowski's Theorem, the implication of (a) $\Longrightarrow$ (b) in the Minkowski-Weyl Theorem, Theorem 3.10. The algorithm is not output-polynomial as it was shown by Bremner [10]. However, it is extremely useful for certain representation conversion problems, in particular, for highly degenerate inputs.

Suppose that an $m \times d$ matrix A is given, and let $C(A) = \{x : Ax \leq 0\}$. We call any vector $r \in C(A)$ a *ray* of $C(A)$. The DD method is an incremental algorithm to construct a $d \times n$ matrix R such that (A, R) is a DD pair.

Let K be a subset of the row indices $\{1, 2, \ldots, m\}$ of A and let A_K denote the submatrix of A consisting of rows indexed by K. Suppose we already found a generating matrix R for $C(A_K)$, or equivalently (A_K, R) is a DD pair. If $A = A_K$, clearly we are done. Otherwise we select any row index i not in K and try to construct a DD pair (A_{K+i}, R') using the information of the DD pair (A_K, R). Note that $K + i$ is a simplified notation for $K \cup \{i\}$.

Once this basic procedure is described, we have an algorithm to construct a generating matrix R for $C(A)$. This procedure can be easily understood geometrically and the reader is strongly encouraged to draw some simple example in the three dimensional space.

The newly introduced inequality $A_i\, x \le 0$ partitions the space $\mathbb{R}^d$ into three parts:

$$\begin{array}{rcl} H_i^+ & = & \{x \in R^d : A_i\, x > 0\} \\ H_i^0 & = & \{x \in R^d : A_i\, x = 0\} \\ H_i^- & = & \{x \in R^d : A_i\, x < 0\}. \end{array} \tag{9.1}$$

Let J be the set of column indices of R and let r_j denote the jth column of R. The rays r_j $(j \in J)$ are then partitioned into three parts:

$$\begin{array}{rcl} J^+ & = & \{j \in J : r_j \in H_i^+\} \\ J^0 & = & \{j \in J : r_j \in H_i^0\} \\ J^- & = & \{j \in J : r_j \in H_i^-\}. \end{array} \tag{9.2}$$

We call the rays indexed by J^+, J^0, J^- the *positive*, *zero*, *negative* rays with respect to i, respectively. To construct a matrix R' from R, we generate new $|J^+| \times |J^-|$ rays lying on the ith hyperplane H_i^0 by taking an appropriate positive combination of each positive ray r_j and each negative ray $r_{j'}$ and by discarding all positive rays.

The following lemma ensures that we have a DD pair (A_{K+i}, R'), and provides the key procedure for the most primitive version of the DD method.

Lemma 9.1 (Main Lemma for Double Description Method) *Let (A_K, R) be a DD pair and let i be a row index of A not in K. Then the pair (A_{K+i}, R') is a DD pair, where R' is the $d \times |J'|$ matrix with column vectors r_j ($j \in J'$) defined by*

$$\begin{array}{rcl} J' & = & J^- \cup J^0 \cup (J^+ \times J^-), \text{ and} \\ r_{jj'} & = & (A_i\, r_j) r_{j'} - (A_i\, r_{j'}) r_j \text{ for each } (j, j') \in J^+ \times J^-. \end{array}$$

Proof. Let $C = C(A_{K+i})$ and let C' be the cone generated by the matrix R'. We must prove that $C = C'$. By the construction, we have $r_{jj'} \in C$ for all $(j, j') \in J^+ \times J^-$ and $C' \subset C$ is clear.

Let $x \in C$. We shall show that $x \in C'$ and hence $C \subseteq C'$. Since $x \in C$, x is a nonnegative combination of r_j's over $j \in J$, i.e., there exist $\lambda_j \ge 0$ for $j \in J$ such that

$$x = \sum_{j \in J} \lambda_j r_j. \tag{9.3}$$

If there is no positive λ_j with $j \in J^+$ in the expression above then $x \in C'$. Suppose there is some $k \in J^+$ with $\lambda_k > 0$. Since $x \in C$, we have $A_i\, x \le 0$. This together with (9.3) implies that there is at least one $h \in J^-$ with $\lambda_h > 0$. Now by construction, $hk \in J'$ and

$$r_{hk} = (A_i\, r_h) r_k - (A_i\, r_k) r_h. \tag{9.4}$$

By subtracting an appropriate positive multiple of (9.4) from (9.3), we obtain an expression of x as a positive combination of some vectors r_j $(j \in J')$ with new coefficients λ_j where the number of positive λ_j's with $j \in J^+ \cup J^-$ is strictly smaller than in the first expression. As long as there is $j \in J^+$ with positive λ_j, we can apply the same transformation. Thus we must find in a finite number of steps an expression of x without using r_j with $j \in J^+$. This proves $x \in C'$, and hence $C \subseteq C'$. ∎

This algorithm can be used to prove Minkowski's Theorem constructively.

Proof. (**of Theorem 3.10**) By Lemma 9.1, it is sufficient to show that one can find an initial DD pair (A_K, R) for some K. The trivial case is when $K = \emptyset$ and $C(A_K) = \mathbb{R}^d$. In this case, the set of $2d$ vectors $R = \{e_1, -e_1, e_2, -e_2, \ldots, e_d, -e_d\}$ generates the space $\mathbb{R}^d$ by their nonnegative combinations. (Actually, one can find $d+1$ vectors which positively span $\mathbb{R}^d$. How?) This completes the proof. ∎

Here we write the DD method in procedural form.

```
procedure DoubleDescriptionMethod($A$);
begin
    Obtain any initial DD pair $(A_K, R)$
    while $K \neq \{1, 2, \ldots, m\}$ do
    begin
        Select any index $i$ from $\{1, 2, \ldots, m\} \setminus K$
        Construct a DD pair $(A_{K+i}, R')$ from $(A_K, R)$
            /* by using Lemma 9.1 */
        $R := R'$    $K := K + i$;
    end
    Output $R$
end.
```

The DD method given here is very primitive, and the straightforward implementation is not quite useful, because the size of J increases very fast and goes beyond any tractable limit. One reason for this is that many (perhaps, most) vectors $r_{jj'}$ the algorithm generates (defined in Lemma 9.1), are unnecessary. To avoid generating redundant generators, we store and update the adjacency of generators. Such a refinement can reduce the size of the output drastically.

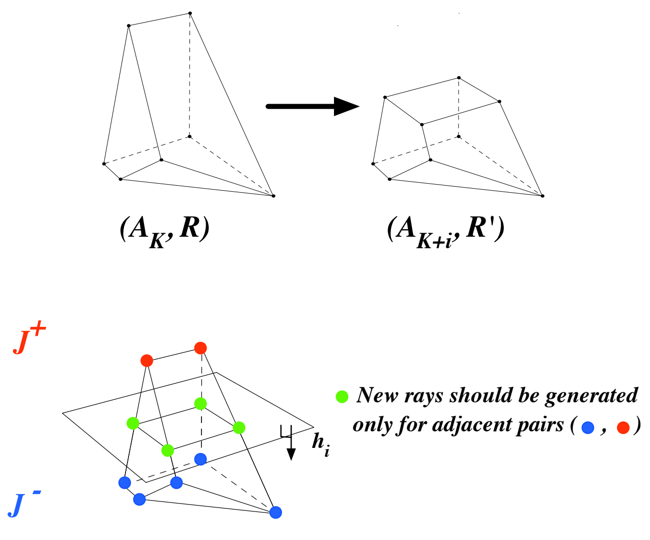

(A_K, R)
(A_{K+i}, R')
J^+
J^-
h_i
● New rays should be generated only for adjacent pairs (● , ●)

Figure 9.1: The Double Description Algorithm

Figure 9.1 depicts a general step of adding the ith inequality with a refined double description algorithm. The polytopes should be considered as a cut section of 4-dimensional pointed cones with some hyperplane so that each vertex represents one dimensional extreme ray starting from the origin.

Two generators are said to *adjacent* if the common set of active constraints is maximal among all pairs of generators. This means that the line segment connecting a adjacent pair meets the new hyperplane $h_i = \{x : A_i x = 0\}$ at a point lying on a minimal face of the cone $C(A_{K+i})$. It is easy to see that such a point must be in any V-representation.

The double description algorithm at the ideal form not generating any redundant generators is still not easy to analyze. The main problem is that the size of a V-representation of intermediate cone is not easy to estimate. The size also depends heavily on the insertion order of constraints.

Here are somewhat surprising behaviors of the refined double desciption method with respect to different insertion orders. In the figure below, the input is (the homogenized cone of) a 15-dimensional polytope with 32 facets. The output is a list of 368 vertices. It is important to note that the conversion is **highly degenerate**, meaning that the number of active inequalities at each output vertex is much higher than the dimension.

We consider the five different orderings of the inequalities. The ordering **lexmin** is simply sort the rows of A by lexicographic ordering, comparing the first component first, then the second in case of tie, and the third, etc. The ordering **maxcutoff** (**mincutoff**)is a dynamic ordering in which at each iteration the next inequality is selected to maximize (minimize) the size $|J^+|$. The lexmin is a sort of shelling ordering which appears to perfom the best among all orderings tested.

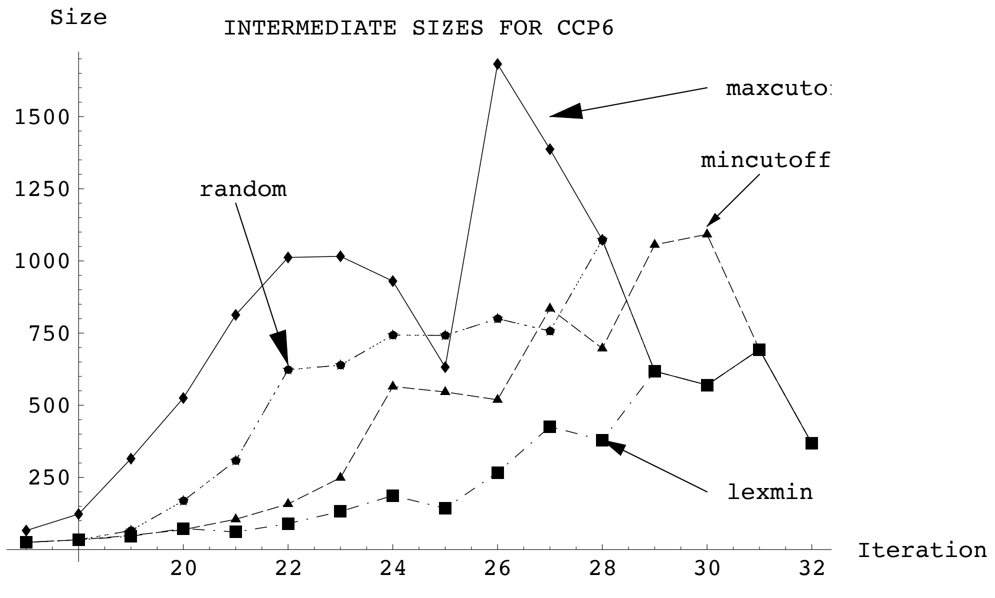


Figure 9.2: Comparison of Intermediate Sizes for a Degenerate Input

A more striking comparison is given below where the input is a 10-dimensional cross polytope with 2^{10} facets. The output is a list of 20 vertices. The highest peak is attained by maxcutoff ordering, following by random and mincutoff. The ordering lexmin is the best among all and the peak intermediate size is less than 30. Note that the graph of lexmin is too low to see it in the figure below.

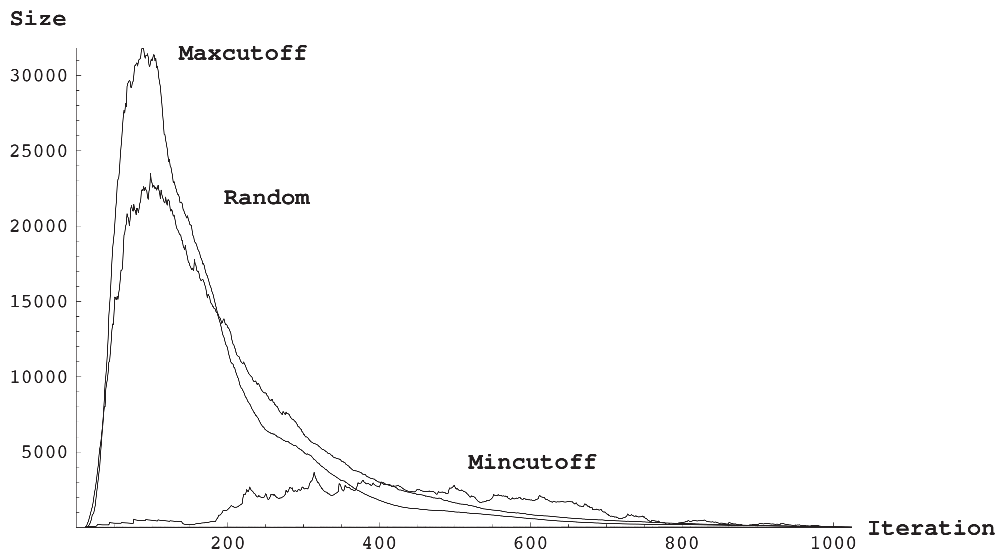
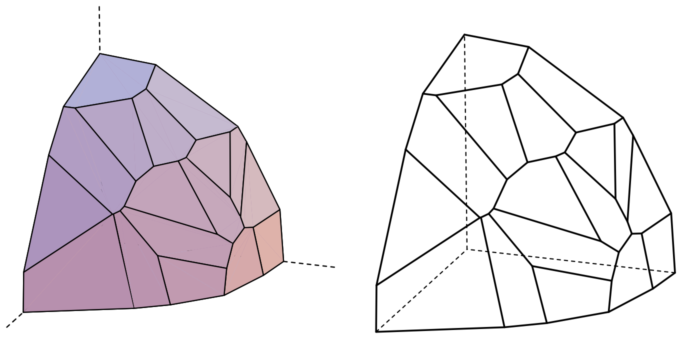


Figure 9.3: Comparison of Intermediate Sizes for a Highly Degenrate Input

9.2 Pivoting Algorithms

One can design pivoting algorithms to visit all vertices of a convex polytope systematically. The idea is quite simple. The graph of a convex polytope is connected, and in fact d-connected if the polytope is d-dimensional, due to Balinski [7]. Thus, one can trace the graph systematically until no new vertex can be found.

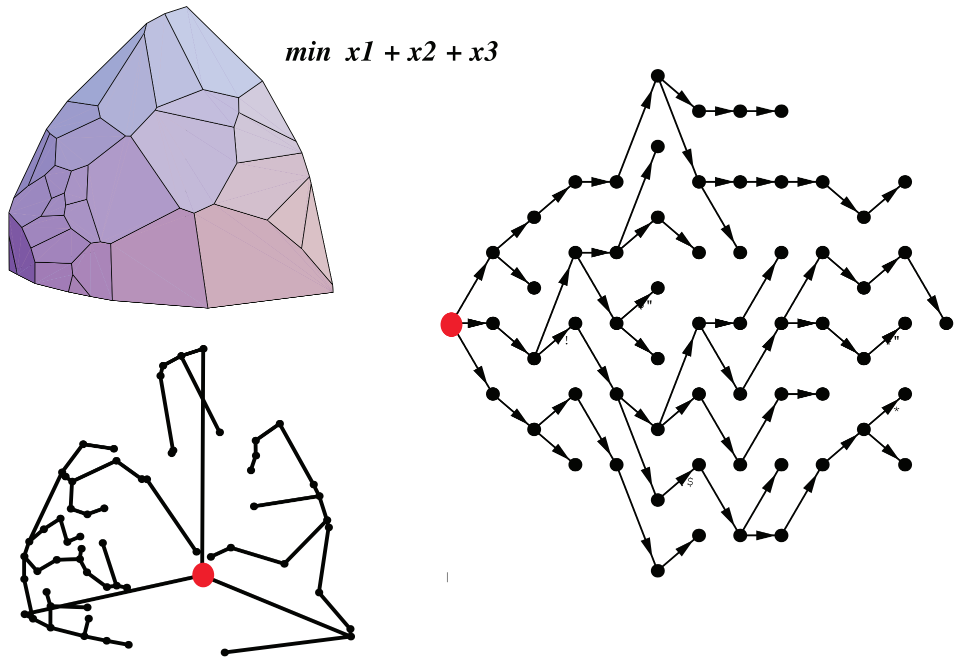


A polytope P and its graph (1-skeleton)

Historically, there are many pivoting algorithms proposed by Balinski [6], Murty [42], Dyer and Proll [19], etc. The weakness of pivoting algorithms is that when the polytope is degenerate, i.e., non-simple, pivoting may not be able to trace the graph of a polytope efficiently. Typical way to resolve degeneracy is a symbolic perturbation of constraints which may create an exponentially large number of new extreme points. Under the nondegeneracy assumption that the number of active constraints at each vertex is exactly d, the algorithm due to Dyer and Proll [19] is an output-polynomial algorithm. Yet, it must store all visited vertices in memory and thus is not a compact algorithm.

In this section, we present a compact output-polynomial algorithm for the nondegenerate case, based on the reverse search technique due to Avis and Fukuda.

The main idea is to reverse the simplex method from the optimal vertex in all possible ways. Here the objective function is set to any generic one so that the optimal vertex is unique and no edge of the polytope is parallel to an objective contour.



Also, another important thing is to make sure that the simplex algorithm is finite and selects a next pivot uniquely at each vertex. This can be achieved, for example, by employing the minimum index rule (Bland's rule). Under these, the edges used by the refined simplex method form a directed spanning tree of the graph $G(P)$ of a polytope P rooted at the optimal vertex. We will see that the resulting algorithm enumerates all f_0 vertices in time $O(mdf_0)$ and $O(md)$-space under nondegeneracy when the input H-polytope is given by m inequalities in d variables. Thus, it is compact and output-polynomial.

For a formal description, let us define two functions. A finite local search f for a graph $G = (V, E)$ with a special node $s \in V$ is a function: $V \setminus \{s\} \to V$ satisfying

(L1) $\{v, f(v)\} \in E$ for each $v \in V \setminus \{s\}$, and

(L2) for each $v \in V \setminus \{s\}$, $\exists k > 0$ such that $f^k(v) = s$.

For example, let $P = \{x \in \mathbb{R}^d : A\,x \leq b\}$ be a simple polytope, and $c^T x$ be any generic linear objective function. Let V be the set of all vertices of P, s the unique optimal, and $f(v)$ be the vertex adjacent to v selected by the simplex method which selects a pivot uniquely if v is not the optimal vertex.

A *adjacency oracle* or simply *A-oracle* Adj for a graph $G = (V, E)$ is a function (where δ a upper bound for the maximum degree of G) satisfying:

(i) for each vertex v and each number k with $1 \leq k \leq \delta$ the oracle returns $\mathrm{Adj}(v, k)$, a vertex adjacent to v or extraneous *null* (null),

(ii) if $\mathrm{Adj}(v, k) = \mathrm{Adj}(v, k') \neq 0$ for some $v \in V$, k and k', then $k = k'$,

(iii) for each vertex v, $\{\mathrm{Adj}(v, k) : \mathrm{Adj}(v, k) \neq 0, 1 \leq k \leq \delta\}$ is exactly the set of vertices adjacent to v.

For example, when $P = \{x \in \mathbb{R}^d : A\,x \leq b\}$ is a simple polytope, let V be the set of all vertices of P, δ be the number of nonbasic variables and $\mathrm{Adj}(v, k)$ be the vertex adjacent to v obtained by pivoting on the kth nonbasic variable at v.

Now we are ready to describe a general reverse search algorithm to generate all vertices of the underlying graph G assuming that the two functions f and Adj are given.

```
       procedure ReverseSearch(Adj,$\delta$,$s$,$f$);
           $v := s$; $j := 0$; (* $j$: neighbor counter *)
           repeat
               while $j < \delta$ do
                   $j := j + 1$;
(r1)               $next := \mathrm{Adj}(v, j)$;
                   if $next \neq null$ then
(r2)                   if $f(next) = v$ then  (* reverse traverse *)
                       $v := next$; $j := 0$
                       endif
                   endif
               endwhile;
               if $v \neq s$ then (* forward traverse *)
(f1)               $u := v$;    $v := f(v)$;
(f2)               $j := 0$;    repeat $j := j + 1$ until $\mathrm{Adj}(v, j) = u$ (* restore $j$ *)
               endif
           until $v = s$ and $j = \delta$
```

We can evaluate the complexity of reverse search above as follows. Below we denote by $t(f)$ and $t(\mathrm{Adj})$ the time to evaluate the functions f and Adj, respectively.

Theorem 9.2 *Suppose that a local search (G, s, f) is given by an A-oracle. Then the time complexity of ReverseSearch is $O(\delta\, t(\mathrm{Adj})|V| + t(f)|E|)$.*

Proof. It is easy to see that the time complexity is determined by the total time spent to execute the four lines (r1), (r2), (f1) and (f2). The first line (r1) is executed at most δ times for each vertex, and the total time spent for (r1) is $O(\delta\ t(\text{Adj})|V|)$. The line (r2) is executed as many times as the degree $deg(v)$ for each vertex v, and thus the total time for (r2) is $O(t(f)|E|)$. The third line (f1) is executed for each vertex v in $V \setminus \{s\}$, and hence the total time for (f1) is $O(t(f)(|V|-|S|))$. Similarly, the total time for (f2) is $O(\delta\ t(\text{Adj})(|V|))$. Since $|V| \leq |E|$, by adding up the four time complexities above, we have the claimed result. ∎

Corollary 9.3 *Suppose that a local search (G, s, f) is given by an A-oracle. Then the time complexity of ReverseSearch is $O(\delta\ (t(\text{Adj}) + t(f))|V|)$. In particular, if δ, $t(f)$ and $t(\text{Adj})$ are independent of the number $|V|$ of vertices in G, then the time complexity is linear in the output size $|V|$.*

Proof. The claim follows immediately from Theorem 9.2 and the fact that $2|E| \leq \delta|V|$. ∎

One can improve the complexity of reverse search algorithms by exploiting special structures. We give the best known complexity of reverse search for the representation conversion for convex polytopes without proof, see [4] for details.

Theorem 9.4 *There is an implementation of reverse search algorithm to enumerate all vertices of a nondegenerate H-polytope $P = \{x : Ax \leq b\}$ in time $O(mdf_0)$ and space $O(md)$, where $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$ and f_0 is the number of vertices of P. In particular, it is a compact output-polynomial algorithm for nondegenerate inputs.*

There are many applications of reverse search in geometry and combinatorics, see [5].

Finally, what is the difference between reverse search and depth-first search? The quick answer is: reverse search is a memory-free search while depth-first search must store all nodes visited so far to distinguish those vertices from the rest. In other words, reserve search is depth-first search applied to a unique spanning tree of the graph defined by local search function f.

9.3 Pivoting Algorithm vs Incremental Algorithm

- Pivoting algorithms, in particular the reverse search algorithm (lrs, lrslib [2]), work well for high dimensional cases.
- Incremental algorithms work well for low (say, up to 12) dimensional cases and highly degenerate cases. For example, the codes cdd/cddlib [22] and porta [13] are implemented for highly degenerate cases and the code qhull [8] for low (up to 10) dimensional cases.
- The reverse search algorithm seems to be the only method that scales very efficiently in massively parallel environment.
- Various comparisons of representation conversion algorithms and implementations can be found in the excellent article [3] by Avis, Bremner and Seidel.

10 Hyperplane Arrangements and Point Configurations

In Sections 5, 6 and 7, we studied the combinatorial structure of convex polytopes. In this section, we look at not only polytopes but also the dissection of the whole space by a set of hyperplanes which induces a polyhedral complex. Formally, it is known as an arragement of hyperplanes and its dual structure is known as a configuration of points or vectors.

10.1 Cake Cutting

An intuitively appealing way to study the dissection of the plane by a set of lines is through cake cutting. Just consider a round cake from above (i.e, a 2-dimensional disk), and try to cut it by a knife a few times. With m straight cuts, how many pieces can one produce? Of course, it depends on cut intersection patterns, as one can see in Figure 10.1.

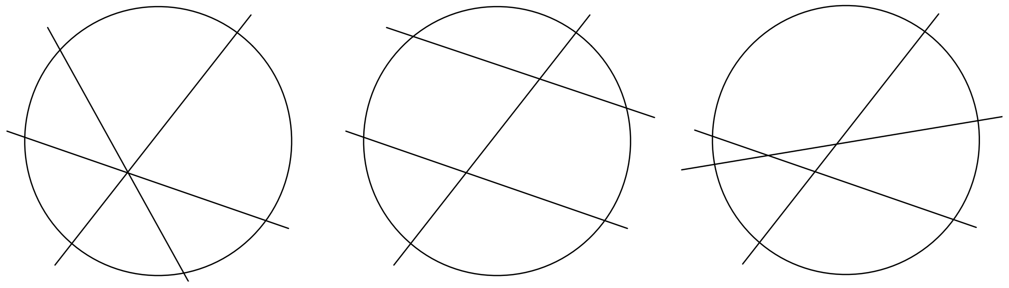
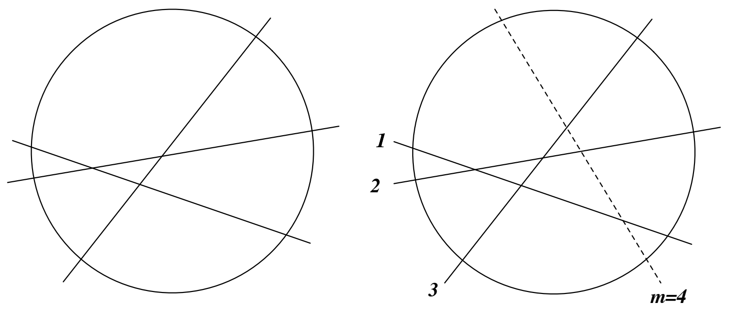


Figure 10.1: Cake Cutting Problem

Let us denote by $p_2(m)$ the maximum number of pieces one can produce by m cuts in 2D. Clearly, $p_2(0) = 1$ and $p_2(1) = 2$. It is not hard to give an explicit formula for this by looking at a simple recursive formula. We can easily see that if the mth cut intersects with the previous $m-1$ cuts at distinct points (in the interior of the cake), then it generates **additional** m **pieces**. It is obvious that this is an upper bound of the number of pieces one can generate.



Is this upper bound attainable? We argue that this is always attainable by placing cuts properly. A 2D cake cutting with m cuts is defined to be *nondegenerate* if any two distinct cuts intersect in the interior of the cake and no three distinct cuts have a common intersection. For any m, nondegenerate cuts exist. Just place m cuts so that no two cuts are parallel and no three cuts intersect. If some two cuts do not intersect in the interior of the cake, just dilate the cake (centered at the origin) without changing the cut placements. If the dilation is large enough, all intersections of the lines will be placed inside the cake. Then, shrink the whole space so that the cake becomes the original size.

This observation leads to a simple recursion:

$$p_2(m) = p_2(m-1) + m \tag{10.1}$$

which implies

$$p_2(m) = p_2(0) + 1 + 2 + \cdots + m = 1 + \frac{(m+1)m}{2}. \tag{10.2}$$

Now we go up to one higher dimension. The cake looks like a 3-dimensional ball, and we try to cut out the largest number of pieces with m cuts. We now imagine how a "watermelon" can be dissected into pieces by a knife into pieces, see Figure 10.2. (A cut does not go through the center although the figure shows such a case.)

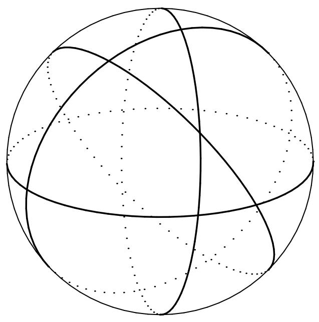


Figure 10.2: 3-Dimensional Cake Cutting Problem

Let us denote by $p_3(m)$ the maximum number of pieces one can produce by m cuts in 3 dimension. Can one write a simple recursive formula for $p_3(m)$? Yes, it is possible, once one notices that the cut section at mth cut in the 3D cake could look like a 2D cake cutting at $(m-1)$st step, as long as the mth cut intersects with the previous $(m-1)$ cuts at distinct lines. A key observation is that the number of 2D pieces at the cut section is exactly the increment of the number of pieces by mth cut. Thus, when the mth cut section is nondegenerate, the increment is largest and thus the observation leads to the recursion

$$
\begin{aligned}
p_3(m) &= p_3(m-1) + p_2(m-1) \qquad (10.3)\\
&= p_3(m-1) + 1 + \frac{m(m-1)}{2}\\
&= p_3(0) + m + \sum_{i=1}^{m} \frac{i(i-1)}{2}\\
&= 1 + m + \frac{1}{2}\left(\sum_{i=1}^{m} i^2 - \sum_{i=1}^{m} i\right)\\
&= 1 + m + \frac{1}{6}(m+1)\left(m+\frac{1}{2}\right)m - \frac{1}{4}m(m+1). \qquad (10.4)
\end{aligned}
$$

$$\text{(Recall the identity: } \sum_{i=1}^{m} i^2 = \frac{1}{6}(m+1)(2m+1)m \text{)}$$

We have two explicit formulas, one for 2D (10.2) and the other for 3D (10.4). Can we guess a general formula for $p_d(m)$? Well, not quite easy to guess from what we have. But, it is much easier once we rewrite the two equations in the following form:

$$
\begin{aligned}
p_2(m) &= \binom{m}{0} + \binom{m}{1} + \binom{m}{2}\\
p_3(m) &= \binom{m}{0} + \binom{m}{1} + \binom{m}{2} + \binom{m}{3}.
\end{aligned} \qquad (10.5)
$$

Exercise 10.1 Verify the correctness of the equations (10.5).

Now, we are ready to prove the general cake cutting theorem.

Theorem 10.1 *The number $p_d(m)$ of the maximum number of pieces dissected from the d-dimensional ball by m (hyperplane) cuts is given by*

$$p_d(m) = \sum_{i=0}^{d} \binom{m}{i}. \qquad (10.6)$$

Proof. We prove the correctness of the formula and the fact that the the value is attained by any nondegenerate cake cut, by induction on d. Here, we say a d-dimensional cake cutting of a d-ball (cake) with m cuts is defined to be *nondegenerate* if any d distinct cuts intersect in the interior of the cake and no $(d+1)$ distinct cuts have a common intersection. The formula is correct for $d = 2$ and attained by any nondegenerate cutting. Consider any unknown case d assuming that the formula is correct for any smaller dimension. First of all, $m = 0$, the formula $p_d(m)$ is correct that is 1. Here we use second induction on m. Consider any unknown case m assuming that the formula is correct for any smaller values of m. By extending the recursion (10.3), we have

$$p_d(m) = p_d(m-1) + p_{d-1}(m-1). \qquad (10.7)$$

By induction hypothesis, one can apply the formula to the RHS and we have:

$$\begin{aligned} p_d(m) = &\binom{m-1}{0} + \binom{m-1}{1} + \cdots + \binom{m-1}{d} \\ &+ \binom{m-1}{0} + \cdots + \binom{m-1}{d-1}. \end{aligned}$$

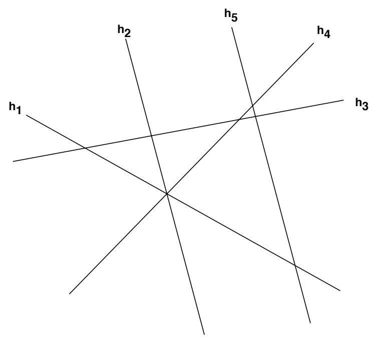
Finally, since $\binom{m-1}{-1} = 0$, the last equation above leads to

$$\begin{aligned} p_d(m) = &\sum_{k=0}^{d} \left(\binom{m-1}{k} + \binom{m-1}{k-1} \right) \\ = &\sum_{k=0}^{d} \binom{m}{k}. \end{aligned} \tag{10.8}$$

This completes the proof. ∎

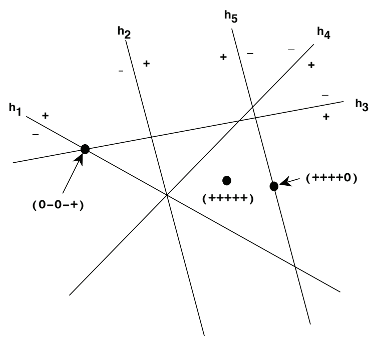
10.2 Arrangements of Hyperplanes and Zonotopes

Cake cutting is a less formal way of presenting the mathematical notion of arrangements of hyperplanes in $\mathbb{R}^d$. A finite family $\mathcal{A} = \{h_i : i = 1, 2, \ldots, m\}$ of hyperplanes in $\mathbb{R}^d$ is called an *arrangement of hyperplanes*.



We are mostly interested in combinatorial structures underlying hyperplane arrangements. For this, it is convenient to define the partition of the space $\mathbb{R}^d$ into three sets:

$$h_i^+ = \{x : A_i\, x < b_i\}, \tag{10.9}$$
$$h_i^0 = \{x : A_i\, x = b_i\}, \tag{10.10}$$
$$h_i^- = \{x : A_i\, x > b_i\}. \tag{10.11}$$

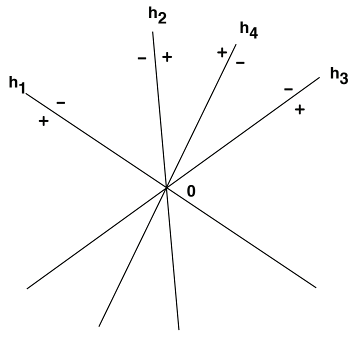


There is a natural way to associate each point x in the space $\mathbb{R}^d$ with the sign vector $\sigma(x) \in \{-, 0, +\}^m$ defined by:

$$\sigma(x)_i = \begin{cases} + & \text{if } x \in h_i^+ \\ 0 & \text{if } x \in h_i^0 \\ - & \text{if } x \in h_i^- \end{cases} \qquad i \in E.$$

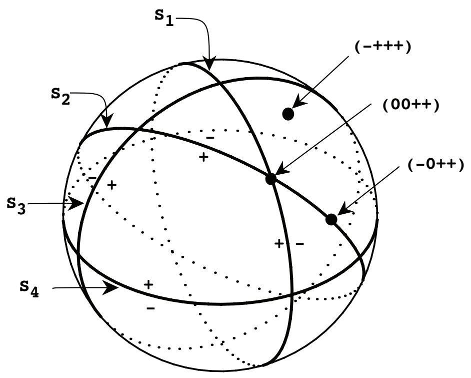
The set of points with a given sign vector is a relatively open polyhedron, is called a *open face* of the arrangement, and its topological closure is called a *face* of the arrangement. The full dimensional faces are called the *cells* or *regions* of the arrangement. The set of all faces forms a polyhedral complex, called the *complex of the arrangement*. One can represent the facial incidence in the complex by a binary relation among sign vectors. For two sign vectors $X, Y \in \{-, 0, +\}^m$, we say X *conforms to* Y (denoted as $X \preccurlyeq Y$) if $i \in [m]$ and $X_i \neq 0$ implies $X_i = Y_i$. The poset $\sigma(\mathbb{R}^d) := \{\sigma(x) : x \in \mathbb{R}^d\}$ ordered by conformal relation is a combinatorial representation of the complex. This poset is the *face poset* $\mathcal{F}(\mathcal{A})$ *of the arrangement* $\mathcal{A}$.

The poset $\mathcal{F}(\mathcal{A})$ behaves nicely if all the hyperplanes contains the origin. An arrangement of hyperplanes in which all its hyperplanes contain the origin 0 is called a *central arrangement of hyperplanes*.



For example, $\mathcal{F}(\mathcal{A})$ contains the zero vector $\mathbf{0}$ which is the unique smallest element. Also, it is symmetric with respect to the origin: if a sign vector X is in $\mathcal{F}(\mathcal{A})$, its negative $-X$ is in $\mathcal{F}(\mathcal{A})$. By adding the artificial greatest element $\mathbf{1}$ of all 1's to $\mathcal{F}(\mathcal{A})$, we obtain what we call the face lattice $\hat{\mathcal{F}}(\mathcal{A})$ of the central arrangement. We will see this lattice is isomorphic to the lattice of a very special polytope.

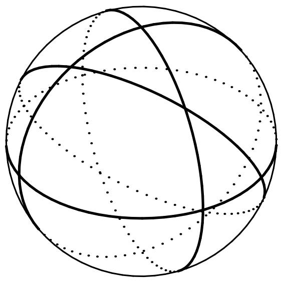
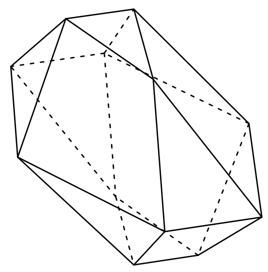
Let $\mathcal{A}$ be a central arrangement of hyperplanes represented by a matrix A, i.e, $h_i = \{x : A_i\, x = 0\}, \forall i = 1, \ldots, m$. Geometrically, it is convenient to look at the cut section of the arrangement with the unit $(d-1)$-sphere $S^{d-1} := \{x \in \mathbb{R}^d : ||x|| = 1\}$, where each hyperplane becomes a $(d-2)$-sphere $s_i := h_i \cap S^{d-1}$. Thus, the cut section is an arrangement of $(d-2)$-spheres in the unit sphere S^{d-1}. The complex of the arrangement is essentially represented in the sphere arrangement, namely, $\sigma(\mathbb{R}^d) = \sigma(S^d) \cup \{\mathbf{0}\}$.



Consider the following H-polyhedron given by 2^m inequalities:

$$P_A = \{x : y^T\, A\, x \leq 1, \forall\, y \in \{-1, +1\}^m\}.$$

Theorem 10.2 *Let A be a column full rank matrix representing a central arrangement $\mathcal{A}$. Then P_A is a polytope, and the face lattice $\hat{\mathcal{F}}(\mathcal{A})$ of $\mathcal{A}$ is isomorphic to the face lattice of the polytope P_A.*

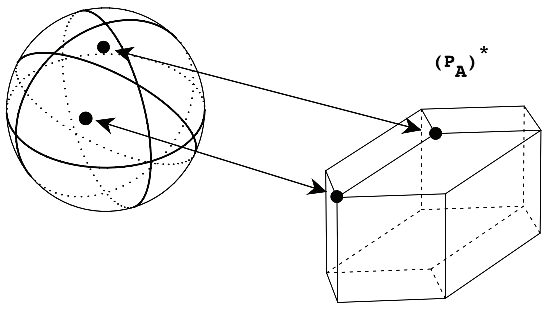



The central arrangement $\mathcal{A}$ and the polytope P_A

The polar of the polytope P_A is a very special polytope. In fact, it is a zonotope.

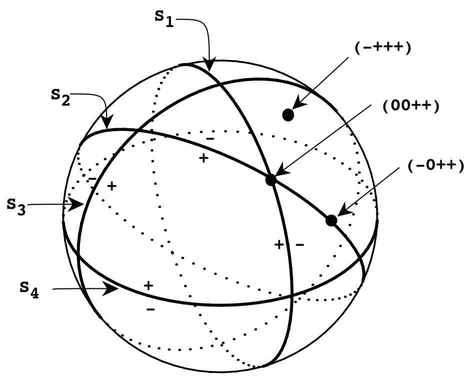
$$\begin{aligned}
(P_A)^* &= \operatorname{conv}\{y^T A \in \mathbb{R}^d : y \in \{-1, +1\}^m\} \\
&= \{y^T A \in \mathbb{R}^d : y \in [-1, +1]^m\} \\
&= L_1 + L_2 + \cdots + L_m,
\end{aligned}$$

where each *generator* L_i is the line segment $[-A_i, A_i]$.



10.3 Face Counting Formulas for Arrangements and Zonotopes

We denote by $f_k(\mathcal{A})$ the number of k-dimensional faces of an arrangement $\mathcal{A}$ of hyperplanes in $\mathbb{R}^d$. We assume all arrangements are central and thus can be seen as a sphere arrangement in S^{d-1}.



With the sphere arrangement setting, it is not hard to relate any central arrangement of m hyperpanes in $\mathbb{R}^d$ to a cake cutting. Let s^0_m be the last sphere in the arrangement. It is the boundary of two hemispheres $s^+_m := h^+_m \cap S^{d-1}$ and $s^-_m := h^-_m \cap S^{d-1}$. The arrangement of spheres restricted to one of the hemispheres is combinatorially equivalent to the cake cutting of a $d-1$-dimensional ball by $m-1$ cuts. This observation together with Theorem 10.1 implies the following theorem.

Theorem 10.3 (Upper Bound Theorem for Arrangements) *For any central arrangement $\mathcal{A}$ of m hyperplanes in $\mathbb{R}^d$,*

$$f_d(\mathcal{A}) \leq 2 \sum_{i=0}^{d-1} \binom{m-1}{i} \quad \textit{and} \quad f_1(\mathcal{A}) \leq 2 \binom{m}{d-1}.$$

Note that if one restrict the arrangement to the unit sphere, the LHS expressions represent $f_{d-1}(\mathcal{A} \cap S^{d-1})$ and $f_0(\mathcal{A} \cap S^{d-1})$.

Using the duality of arrangements and zonotopes, Theorem 10.3 implies the upper bound theorem for zonotopes.

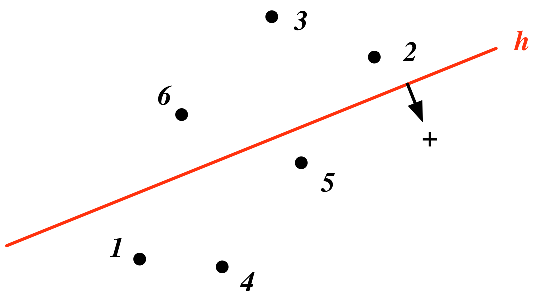
Theorem 10.4 (Upper Bound Theorem for Zonotopes) *Let P be a d-dimensional zonotope given by m generators ($m \geq d$). Then,*

$$f_0(P) \leq 2 \sum_{i=0}^{d-1} \binom{m-1}{i} \quad \textit{and} \quad f_{d-1}(P) \leq 2 \binom{m}{d-1}.$$

For fixed d, both $f_{d-1}(P)$ and $f_0(P)$ are $O(m^{d-1})$.

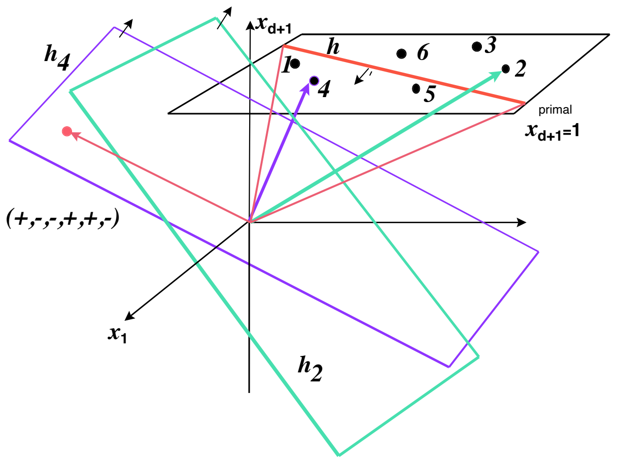
10.4 A Point Configuration and the Associated Arrangement

A *point configuration* is a set $P = \{p_1, p_2, \ldots, p_n\}$ of points in $\mathbb{R}^d$. The relative locations of the points with respect to an arbitrary hyperplane represent the underlying combinatorial structure.



Let $\hat{p}_i = \begin{bmatrix} p_i \\ 1 \end{bmatrix}$ be the lifted points in $\mathbb{R}^{d+1}$, and the hyperplanes $h_i = \{x : \hat{p}_i^T\, x = 0\}$. The resulting arrangement $\mathcal{A} = \{h_1, \ldots, h_n\}$ in $\mathbb{R}^{d+1}$ encodes the combinatorial structure of P nicely.

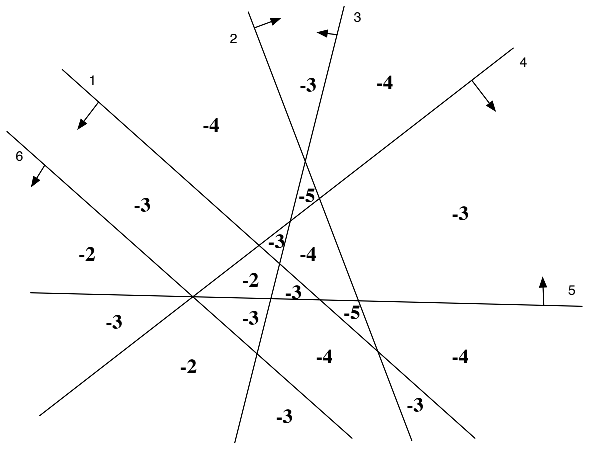
A open halfspace h^+ is represented by the sign vector $X \in \{+, -, 0\}^n$ of a region in the dual hyperplane arrangement with $j \in X^+$ iff $p_j \in h^+$.



The partition $(\{1, 4, 5\}, \{2, 3, 6\})$ by the hyperplane h corresponds to the region $(+, -, -, +, +, -)$.

10.4.1 Application: Largest Feasible Subsystem

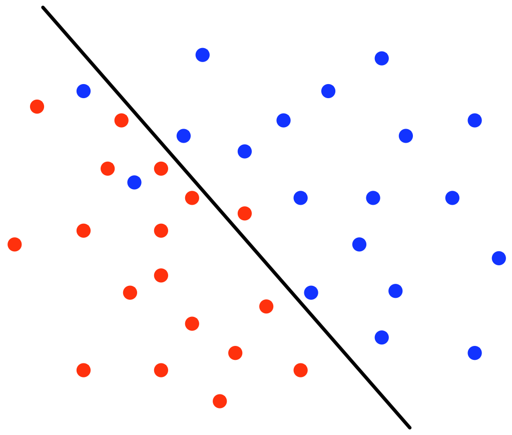
Given an inconsistent linear inequality system $Ax < b$, find a subsystem that is consistent and largest possible. In other words, try to remove as few inequalities as possible to make it feasible.



This problem is known to be NP-hard. One must rely on some kind of enumeration or approximation algorithms to solve this.

10.4.2 Applications: Best Separation of Points by a Hyperplane

Given two blue and red sets of points in $\mathbb{R}^d$, find a (separation) hyperplane which is best possible, i.e. the number of misclassified points is minimized.



This problem is NP-hard, and in fact, one can reduce this to the largest feasible subsystem problem. The number of separations represents the underlying complexity of enumeration.

11 Computing with Arrangements and Zonotopes

As we learned in the previous section that central arrangements of hyperplanes and zonotopes are essentially the same object mathematically. More specifically, if $\mathcal{A}(A)$ is a central arrangement with an $m \times d$ representation matrix A, then its face lattice $\hat{\mathcal{A}}$ is anti-isomorphic to the zonotope $Z(A)$ generated by the line segments $L_i = [-A_i, A_i]$, $j \in [m]$.

This duality implies that one can translate an algorithm for arrangements to an algorithm for zonotopes. In particular, the following pairs of problems with input matrix A given are equivalent.

Problem 11.1 Cell Enumeration for Arrangements/Vertex Enumeration for Zonotopes

(a) Generating all cells of $\mathcal{A}(A)$.

(b) Generating all vertices of $Z(A)$.

Problem 11.2 Vertex Enumeration for Arrangements/Facet Enumeration for Zonotopes

(a) Generating all 1-faces (rays) of $\mathcal{A}(A)$.

(b) Generating all facets of $Z(A)$.

Problem 11.3 Face Enumeration for Arrangements/Face Enumeration for Zonotopes

(a) Generating all faces of $\mathcal{A}(A)$.

(b) Generating all faces of $Z(A)$.

There is a compact output-polynomial algorithm [5] due to Avis and Fukuda for Problem 11.1. Also, there is a worst-case optimal algorithm [20] due to Edelsbrunner, O'Rourke and Seidel for Problem 11.1.

There is a output-polynomial algorithm [49] due to Seymour for Problem 11.2. No compact output-polynomial algorithm is known for Problem 11.2. When input is nondegenerate, Problem 11.2 has a trivial algorithm which is compact and output-polynomial, just go though all $\binom{m}{d-1}$ combinations. This suggests that when input is only "slightly" degenerate, the naive algorithm might be practical.

The paper [27] shows that there is an output-polynomial algorithm to generate all faces of $\mathcal{A}(A)$ from the list of cells. This means that together with the compact output-polynomial algorithm [5] for Problem 11.1, Problem 11.3 can be solved by an output-polynomial algorithm.

11.1 Cell Generation for Arrangements

Here we present the reverse search algorithm [5] which is the only compact output-polynomial algorithm for generating all cells of an arrangement. By duality, this is a compact output-polynomial algorithm for enumerating all vertices of a zonotope.

Let $\mathcal{A}$ be an arrangement of distinct hyperplanes $\{h_i : i \in [m]\}$ in $\mathbb{R}^d$, where each hyperplane is given by a linear equality $h_i = \{x : A_i x = b_i\}$. The two sides of h_i are

$h_i^+ = \{x : A_i x \geq b_i\}$ and $h_i^- = \{x : A_i x \leq b_i\}$. For each $x \in \mathbb{R}^d$, the sign vector $\sigma(x)$ of x is the vector in $\{-, 0, +\}^m$ defined by

$$\sigma(x)_i = \begin{cases} - & \text{if } x \in h_i^- \\ 0 & \text{if } x \in h_i \\ + & \text{if } x \in h_i^+ \end{cases} \qquad (i \in [m]).$$

Let V_{CELL} be the set of sign vectors of points in $\mathbb{R}^d$ whose nonzero support is $[m]$. We can identify each vector c in V_{CELL} with the open cell (open d-face) of the arrangement defined by $\{x : \sigma(x) = c\}$. For two cells c and c', let $sep(c, c')$ be the set of separators of c and c', that is, the set of elements i in $[m]$ such that c_i and c'_i have opposite signs. We say that two cells c and c' are *adjacent* in G_{CELL} if they differ in only one component, or equivalently, $|sep(c, c')| = 1$. The following lemma is important.

Lemma 11.4 *For any two distinct cells c and c' in V_{CELL}, there exists a cell c'' which is adjacent to c and $sep(c, c'') \subset sep(c, c')$.*

Proof. Let c and c' be two distinct cells, and let x (x') be a point in c (in c', respectively) in general position. Moving from x toward x' on the line segment $[x, x']$, we encounter the sequence of cells: $c_o = c, c_1, c_2, \ldots, c_k = c'$, and we can easily verify that c_1 is adjacent to c and $sep(c, c_1) \subset sep(c, c')$. ∎

Let us assume that V contains the cell c^* of all $+$'s. Lemma 11.4 implies that for each cell c different from c^*, there is a cell c'' which is adjacent to c and $sep(c^*, c'') \subset sep(c^*, c)$. Let us define $f_{CELL}(c)$ as such c'' that is lexico-largest (i.e., the unique element in $sep(c, c'')$ is smallest possible). Then, $(G_{CELL}, S_{CELL}, f_{CELL})$ is a finite local search with $S_{CELL} = \{c^*\}$.

Figure 11.1 describes the trace of the local search on a small example with $d = 2$ and $m = 4$.

By reversing this local search, we obtain an algorithm to list all cells in an arrangement. There are a few things to be explained for an implementation. First, we assumed that the cell c^* of all $+$'s is given, but we can pick up any cell c in the arrangement, and consider it as the cell of all $+$'s since replacing some equality $A_i x = b_i$ by $-A_i x = -b_i$ does not essentially change the arrangement. Note that one can obtain an initial cell by picking up any random point in $\mathbb{R}^d$ and perturbing it if it lies on some hyperplanes.

Now, how can we realize ReverseSearch(Adj_{CELL},δ_{CELL},S_{CELL},f_{CELL}) in an efficient way? First we can set $\delta_{CELL} = m$ and $S_{CELL} = \{c^*\}$. For any cell $c \in V_{CELL}$ and $k \in M$, the function $\text{Adj}_{CELL}(c, k)$ can be realized via solving an LP of the form

$$\begin{array}{lll} \text{minimize (maximize)} & y_k & \\ \text{subject to} & y = Ax - b, & \\ & y_i \geq 0 \text{ for all } i \neq k \text{ with } c_i = +, & \\ & y_i \leq 0 \text{ for all } i \neq k \text{ with } c_i = -, & \end{array} \qquad (11.1)$$

where minimization (maximization) is chosen when $c_k = +$ ($c_k = -$, respectively). The function returns the adjacent cell c' with $sep(c, c') = \{k\}$ if and only if LP (11.1) has a feasible solution with negative (positive) objective value. The time $t(\text{Adj}_{CELL})$ depends on how an LP with d variables and $m - 1$ inequalities is solved. We denote this as a function $\text{LP}(d, m)$ of m and d, as we used this notation in Section 8.

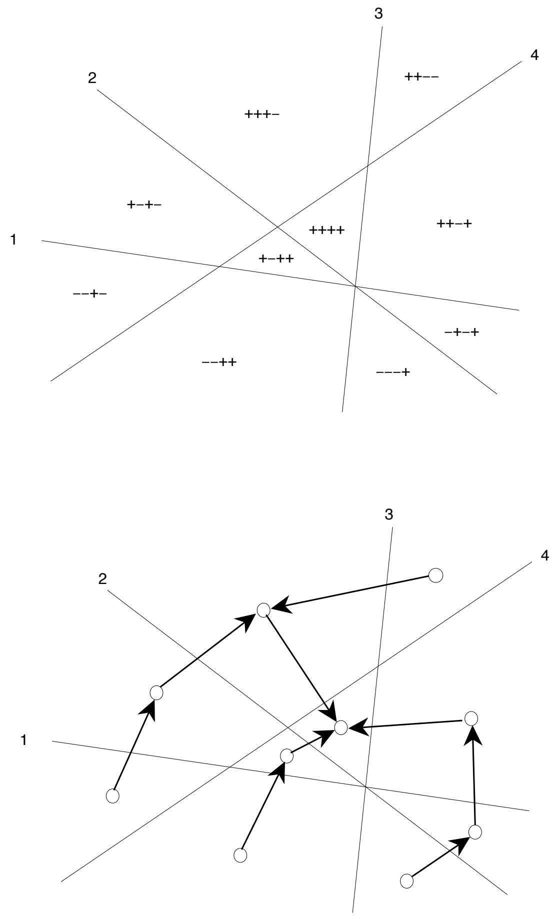


Figure 11.1: An arrangement of hyperplanes and the trace of f_{CELL}

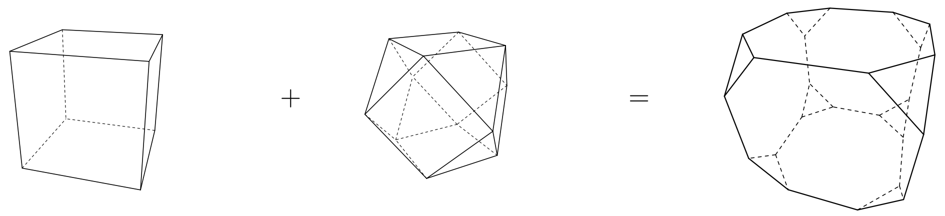
There is a straightforward implementation of f_{CELL}, which solves a sequence of LP's similar to (11.1) with objective functions $y_1, y_2, y_3, \ldots$. This means we may have to solve $O(m)$ LP's in the worst case. Presently we don't know how to implement it in a more efficient manner.

Theorem 11.5 *There is an implementation of ReverseSearch(*Adj_{CELL}*,* δ_{CELL}*,* S_{CELL}*,* f_{CELL}*) for the cell enumeration problem with time complexity* $O(m\ d\ \mathrm{LP}(d,m)|V_{CELL}|)$ *and space complexity* $O(m\ d)$*.*

Proof. To prove this, first we recall that Theorem 9.2 says, the time complexity of ReverseSearch is $O(\delta\ t(\mathrm{Adj})|V|+t(f)|E|)$. As we remarked earlier, $\delta_{CELL} = m$, $t(\mathrm{Adj}_{CELL}) = O(\mathrm{LP}(d,m))$, and $t(f_{CELL}) = O(m\ \ \mathrm{LP}(d,m))$. Since $|E_{CELL}| \le d\ |V_{CELL}|$ holds for any arrangement (see [27]), the claimed time complexity follows. The space complexity is clearly same as the input size $O(m\ d)$. ∎

12 Minkowski Additions of Polytopes

A zonotope is a very special Minkowski sum of polytopes, namely, a Minkowski of line segments. In this section, we study the complexity of Minkowski sums of polytopes P_1, …, P_k in $\mathbb{R}^d$ and some algorithms for computing Minkowski sums of polytopes.



There are three basic variations of the problem. When input is H-polytopes and output is also H-polytope, Tiwary [50] has recently proved that the associated decision problem is NP-hard for $k = 2$. Here the *associated decision problem* is to test whether a given H-polytope P is the Minkowski sum of given H-polytopes P_1, …, P_k. When input is V-polytopes and output is H-polytope, the problem contains the representation conversion for polytopes as a special case ($k = 1$) whose complexity is still unknown. The last case when both input and output are V-polytopes is the only case for which an output-polynomial algorithm is known.

In this section, we present a compact output-polynomial algorithm for the last case. The algorithm is a natural extension of (the dual form of) the reverse search algorithm given in Section 11.1.

Faces, Minkowski Decomposition and Adjacency

For a polytope P and for any vector $c \in \mathbb{R}^d$, the set of maximizers x of the inner product $c^T x$ over P is denoted by $S(P; c)$. Thus each nonempty face of P is $S(P; c)$ for some c. We denote by $F(P)$ the set of faces of P, by $F_i(P)$ the set of i-dimensional faces, and by $f_i(P)$ the number of i-dimensional faces, for $i = -1, 0, \ldots, d$, For each nonempty face F, the relatively open polyhedral cone of outer normals of P at F is denoted by $N(F; P)$. Thus, $c \in N(F; P)$ if and only if $F = S(P; c)$. The *normal fan* $N(P)$ of P is the cell complex $\{N(F; P) | F \in F(P)\}$ whose body is $\mathbb{R}^d$. If F is i-dimensional ($i = 0, 1, \ldots, d$), the normal cone $N(F; P)$ is $(d-i)$-dimensional. Thus the extreme points of P are in one-to-one correspondence with the full dimensional faces (which we call the *regions* or *cells*) of the complex.

Proposition 12.1 *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let $P = P_1 + P_2 + \cdots + P_k$. Then a nonempty subset F of P is a face of P if and only if $F = F_1 + F_2 + \cdots + F_k$ for some face F_i of P_i such that there exists $c \in \mathbb{R}^d$ (not depending on i) with $F_i = S(P_i; c)$ for all i. Furthermore, the decomposition $F = F_1 + F_2 + \cdots + F_k$ of any nonempty face F is unique.*

Proof. The equivalence follows directly from the obvious relation [30, Lemma 2.1.4]

$$S(P_1 + P_2 + \cdots + P_k; c) = S(P_1; c) + S(P_2; c) + \cdots + S(P_k; c) \text{ for any } c \in \mathbb{R}^d.$$

For the uniqueness, let F be a nonempty face with $F = S(P; c)$ for some c and let $F = F_1 + F_2 + \cdots + F_k$ be any decomposition. First, note that $F_i \subseteq S(P;c)$ for all i, because the value $c^T x$ for any $x \in F$ is the sum of the maximum values $c^T x_i$ subject to $x_i \in P_i$ for $i = 1, \ldots, k$, and thus if $x \in F$ and $x = x_1 + x_2 + \cdots + x_k$ for $x_i \in F_i$, then $x_i \in S(P_i, c)$. Now suppose there exists F_i properly contained in $S(P_i; c)$. Let v be an extreme point of $S(P_i; c)$ not in F_i. Then there is a linear function $w^T x$ such that $w^T v$ is strictly greater than any value attained by $x \in F_i$. Now let x^* be any point attaining the maximum of $w^T x$ over the polytope $F_1 + F_2 + \cdots F_{i-1} + F_{i+1} + \cdots + F_k$. Clearly $x^* + v \in F$ but this point cannot be in $F_1 + F_2 + \cdots + F_k$, a contradiction. This proves the uniqueness. ∎

We refer the unique decomposition $F = F_1 + F_2 + \cdots + F_k$ of a nonempty face F as the *Minkowski decomposition*. Here, the dimension of F is at least as large as the dimension of each F_i. Thus we have the following.

Corollary 12.2 *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let $P = P_1 + P_2 + \cdots + P_k$. A vector $v \in P$ is an extreme point of P if and only if $v = v_1 + v_2 + \cdots + v_k$ for some extreme point v_i of P_i and there exists $c \in \mathbb{R}^d$ with $\{v_i\} = S(P_i; c)$ for all i.*

For our algorithm to be presented in the next section, it is important to characterize the adjacency of extreme points in P.

Corollary 12.3 *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let $P = P_1 + P_2 + \cdots + P_k$. A subset E of P is an edge of P if and only if $E = E_1 + E_2 + \cdots + E_k$ for some face E_i of P_i such that $\dim(E_i) = 0$ or 1 for each i and all faces E_i of dimension 1 are parallel, and there exists $c \in \mathbb{R}^d$ with $E_i = S(P_i; c)$ for all i.*

The following variation of the above is useful for the algorithm to be presented. The essential meaning is that the adjacency of extreme points is inherited from those of Minkowski summands.

Proposition 12.4 *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let $P = P_1 + P_2 + \cdots + P_k$. Let u and v be adjacent extreme points of P with the Minkowski decompositions: $u = u_1 + u_2 + \cdots + u_k$ and $v = v_1 + v_2 + \cdots + v_k$. Then u_i and v_i are either equal or adjacent in P_i for each i.*

Proof. Let u and v be adjacent extreme points. It is sufficient to show that $[u, v] = [u_1, v_1] + [u_2, v_2] + \cdots + [u_k, v_k]$ and each $[u_i, v_i]$ is a face of P_i. Let $c \in \mathbb{R}^d$ be such that $[u, v] = S(P; c)$. Because $[u, v] = S(P_1; c) + S(P_2; c) + \cdots + S(P_k; c)$ and by the uniqueness of decomposition of u and v, both u_j and v_j are in $S(P_j, c)$, for all j. This implies that $[u_j, v_j] \subseteq S(P_j, c)$, for all j. On the other hand, one can easily see that in general $[u, v] \subseteq [u_1, v_1] + [u_2, v_2] + \cdots + [u_k, v_k]$. The last two relations give $[u_j, v_j] = S(P_j, c)$ for all j. This completes the proof. ∎

This proposition immediately provides a polynomial algorithm for listing all neighbors of a given extreme point using linear programming.

12.1 Complexity of Minskowski Sums of V-Polytopes

The nontriviality of computing Minkowski sums of V-polytopes can be understood by how the complexity of Minkowski sums varies from some instances to another. In particular, we are most concerned with the complexity of sums in terms of the size of summands.

The first proposition shows that the vertex complexity of Minknowski sums is linearly bounded by the vertex complexity of summand polytopes.

Proposition 12.5 (Linearly Bounded Minkowski-Addition) . *For each $k \geq 2$ and $d \geq 2$, there is an infinite family of Minkowski additions for which $f_0(P_1 + P_2 + \cdots + P_k) \leq f_0(P_1) + f_0(P_2) + \cdots + f_0(P_k)$.*

Proof. Suppose $k \geq 2$ and $d \geq 2$. First pick up any d-polytope, say Q, with at least k extreme points, and select k extreme points. For each jth selected extreme point v^j, make a new polytope P_j from Q by truncating only v^j with one or more hyperplanes. Now we claim that the number $f_0(P_1 + P_2 + \cdots + P_k) \leq f_0(P_1) + f_0(P_2) + \cdots + f_0(P_k)$. See Figure 12.1 for an example for $k = 2$, $d = 3$ and Q is a 3-cube. To see this, let v be an extreme point of P_j for some fixed j. There are three cases. The first case is when v is an unselected one, i.e. an extreme point of Q not selected. In this case, it can be an Minkowski summand of an extreme point of P in a unique way, since any linear function maximized exactly at v over P_j is maximized exactly at v over other P_i's. The second case is when v is a newly created vertex by the truncation of v^j. Since it is obtained by the truncation of v^j, any linear function maximized exactly at v over P_j is maximized exactly at v^j over other other P_i's. The last case is when $v = v^i$ for some $i \neq j$. This case is essentially the same as the second case where v contributes uniquely to a new extreme point with each truncation vertex of P_i. By Corollary 12.2, every extreme point of P_j contributes at most once to $f_0(P_1 + P_2 + \cdots + P_k)$. This completes the proof. ∎

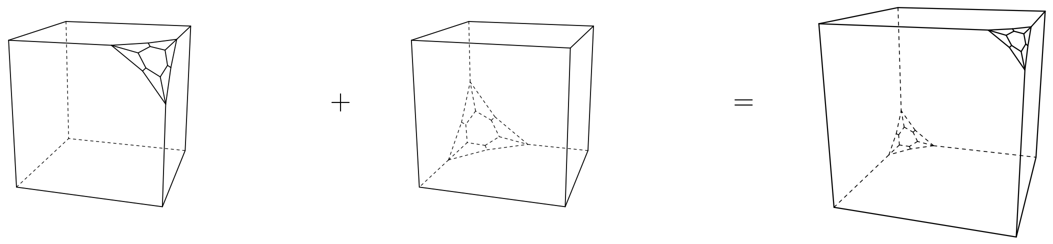


Figure 12.1: Minkowski Sum of Truncated Cubes

The following theorem gives the other extreme to the previous proposition. Namely, the obvious upper bound of the vertex complexity can be achieved for a large class of Minkowski sums of polytopes.

Theorem 12.6 (Tight Upper Bound [28]) .
In dimension $d \geq 3$, it is possible to choose k $(\leq d-1)$ polytopes $P_1, \ldots, P_k$ so that the trivial upper bound for the number of vertices is attained by their Minkowski sum.

$$f_0(P_1 + P_2 + \cdots + P_k) = f_0(P_1) \times f_0(P_2) \times \cdots \times f_0(P_k).$$

Proof. Here we give outline only, see [28] for a precise construction. On k $(\leq d-1)$ orthogonal planes in $\mathbb{R}^d$, place v_i points in convex position. Perturb the points slightly to make each P_i full dimensional. Figure 12.2 shows the case when $f_0(P_1) = f_0(P_2) = 4$ and $f_0(P) = f_0(P_1) \times f_0(P_2) = 16$ ∎

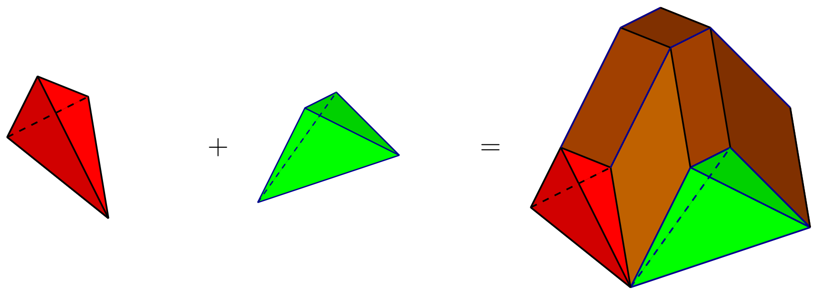


Figure 12.2: Minkowski Sum of $(d-1)$ Thin Polytopes in Orthogonal Spaces

12.2 Extension of a Zonotope Construction Algorithm

We assume in this section that P_1, P_2, ..., P_k are polytopes in $\mathbb{R}^d$ given by the sets V_1, V_2, ..., V_k of extreme points. We also assume that the graph $G(P_j)$ of P_j is given by the adjacency list $(\text{Adj}_j(v,i) : i = 1, \ldots, \delta_j)$ of vertices adjacent to vertex $v \in V_j$ in graph $G(P_j)$, where δ_j is the maximum degree of $G(P_j)$ for each $j = 1, \ldots, k$. If the degree $\deg_j(v)$ of v is less than δ_j in $G(P_j)$, we assume that $\text{Adj}_j(v,i) = \textit{null}$ for all $i > \deg_j(v)$. Finally we define $\delta = \delta_1 + \delta_2 + \cdots + \delta_k$, an upper bound of the maximum degree of $G(P)$, due to Proposition 12.4. For example, when the input polytopes are simple and full dimensional then $\delta_j = d$ for all j and $\delta = k\,d$. Note that for a given set V_j, one can compute the adjacency list in polynomial time using linear programming.

Recall that the Minkowski addition problem is to compute the set V of extreme points of $P = P_1 + P_2 + \cdots + P_k$. We shall present a compact polynomial algorithm for the Minkowski addition problem.

The key idea in our algorithm design

The main algorithmic idea is quite simple. Just like for the vertex enumeration for convex polyhedra using reverse search given in Section 9.2, it traces a directed spanning tree T of the graph $G(P)$ of P rooted at an initial extreme point v^*. The difference from the vertex enumeration algorithm is that the polytope P is not given by a system of inequalities (i.e. not an H-polytope) in the present setting but as a Minkowski-addition of V-polytopes. Thus we need to introduce a new way of defining a directed spanning tree that is easy to trace. We shall use the following simple geometric property of normal fans.

Proposition 12.7 *Let v and v' be two distinct extreme points of P, and let $c \in N(v;P)$ and $c' \in N(v';P)$. Then there exists an extreme point v'' adjacent to v such that $N(v'';P)$ contains a point of form $(1-\theta)c + \theta c'$ for some $0 \le \theta \le 1$.*

Proof. Since $v \neq v'$, their outer normal cones are two distinct full dimensional cones in the normal fan $N(P)$. This means that the parameterized point $t(\theta) := c + \theta(c' - c)$ ($0 \le \theta \le 1$) in the line segment $[c, c']$ must leave at least one of the bounding halfspaces of the first cone $N(v;P)$ as θ increases from 0 to 1. Since the bounding halfspaces of $N(v;P)$ are in one-to-one correspondence with the edges of G incident to v, any one of the halfspaces violated first corresponds to a vertex v'' adjacent to v claimed by the proposition. ∎

Let us fix v^* as an initial extreme point of P. Finding one extreme point of P is easy. Just select any generic $c \in \mathbb{R}^d$, and find the unique maximizer extreme point v^i of $c^T x$ over P_i, for each i. The point $v = v^1 + v^2 + \cdots + v^k$ is an extreme point of P.

Now we construct a directed spanning tree of $G(P)$ rooted at v^* as follows. Let $v \in V$ be any vertex different from v^*. We assume for the moment that there is some canonical way to select an interior point of the normal cone of P at any given vertex, as we shall give one method to determine such a point later. Let c and c^* be the canonical vector of $N(v;P)$ and $N(v^*;P)$, respectively. By Proposition 12.7, by setting $v' = v^*$, we know that there is a vertex v'' adjacent to v such that $N(v'';P)$ meets the segment $[c, c^*]$. In general there might be several such vertices v'' (degeneracy). We break ties by the standard symbolic perturbation of c as $c + (\epsilon^1, \epsilon^2, \ldots, \epsilon^d)^T$ for sufficiently small $\epsilon > 0$. Define the mapping $f : V \setminus \{v^*\} \to V$ as $f(v) = v''$. This mapping, called a *local search function* in reverse search, determines the directed spanning tree $T(f) = (V, E(f))$ rooted at v^*, where $E(f)$ is the set of directed edges $\{(v, f(v)) | v \in V \setminus \{v^*\}\}$.

Proposition 12.8 *The digraph $T(f)$ is a spanning tree of $G(P)$ (as undirected graph) and v^* is a unique sink node of $T(f)$.*

Proof. By the construction, v^* is a unique sink node of $T(f)$. It is sufficient to show that $T(f)$ has no directed cycle. For this, take any edge $(v, v'' = f(v)) \in E(f)$. Let c, c^* be the canonical vector for v, v^*, respectively. Without loss of generality, we assume nondegeneracy, since one can replace c with the perturbed vector $c + (\epsilon^1, \epsilon^2, \ldots, \epsilon^d)^T$. Since c is an interior point of $N(v;P)$,

$$c^T(v - v'') > 0. \tag{12.1}$$

Again, by the construction and because the canonical points are selected as interior points of the associated normal cones, there exists $0 < \theta < 1$ such that $\hat{c} := (1-\theta)c + \theta c^* \in N(v'';P)$. This implies $\hat{c}^T(v'' - v) > 0$, that is,

$$\begin{aligned} 0 &< ((1-\theta)c + \theta c^*)^T(v'' - v) \\ &= (1-\theta)c^T(v'' - v) + \theta(c^*)^T(v'' - v) \\ &< \theta(c^*)^T(v'' - v) \qquad (\text{ by } (12.1)\ ) . \end{aligned}$$

This implies that the vertex v'' attains a strictly higher inner product with c^* than v. Therefore, there is no directed cycle in $T(f)$. ∎

Figure 12.3 shows an example of the directed spanning tree $T(f)$ in green.

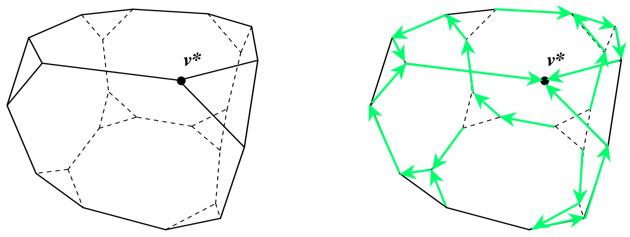


Figure 12.3: The Graph $G(P)$ and A Rooted Spanning Tree $T(f)$

A reverse search algorithm, to be presented below, traces reversely the tree from the root v^* in depth-first manner, using an adjacency oracle.

The critical computation in our algorithm is solving a linear programming problem. We denote by $\mathrm{LP}(d, m)$ the time, as we used in Section 8. necessary to solve a linear programming in d variables and m inequalities.

Now we can state the complexity of our algorithm.

Theorem 12.9 *There is a compact polynomial algorithm for the Minkowski addition of k polytopes that runs in time $O(\delta\, \mathrm{LP}(d, \delta) f_0(P))$ and space linear in the input size.*

The algorithm

The sequel of the section is devoted to present the technical details of a reverse search algorithm that traces $T(f)$ starting from its root vertex v^* against the orientation of edges. We shall prove Theorem 12.9 at the end.

As usual, our reverse search algorithm requires, in addition to the local search function f, an adjacency oracle function that implicitly determines the graph $G(P)$.

Let v be any vertex of P with the Minkowski decomposition $v = v_1 + v_2 + \cdots + v_k$ (see, Corollary **??**). Let

$$\Delta = \{(j, i) : j = 1, \ldots, k \text{ and } i = 1, \ldots, \delta_j\}. \tag{12.2}$$

Recall that for any $(j, i) \in \Delta$, $\mathrm{Adj}_j(v_j, i)$ is the ith vertex adjacent to v_j whenever it is not *null*. We shall call a pair (j, i) *valid* for v if $\mathrm{Adj}_j(v_j, i) \neq$ *null*, and *invalid* otherwise. Let us define the associated edge vectors $e_j(v_j, i)$ by

$$e_j(v_j, i) = \begin{cases} \mathrm{Adj}_j(v_j, i) - v_j & (j, i) \text{ is valid for } v \\ \textit{null} & \text{otherwise.} \end{cases} \tag{12.3}$$

Proposition 12.4 shows that all edges of P incident to v are coming from the edges incident to v_j's, or more precisely, each edge of P incident to v is parallel to some $e_j(v_j, i)$. This immediately implies that δ is an obvious upper bound of the degree of v. For each $(s, r) \in \Delta$, let us group the same (parallel) directions together as

$$\Delta(v, s, r) = \{(j, i) \in \Delta : e_j(v_j, i) \parallel e_s(v_s, r)\}. \tag{12.4}$$

Consider it as the empty set if (s, r) is invalid. Now, for any given pair $(s, r) \in \Delta$, checking whether $e_s(v_s, r)$ determines an edge direction of P is easily reducible to an LP (or more precisely, a linear feasibility problem):

$$\begin{array}{lll} e_s(v_s, r)^T\lambda & < 0, & \\ e_j(v_j, i)^T\lambda & \geq 0 & \text{for all valid } (j, i) \notin \Delta(v, s, r). \end{array} \tag{12.5}$$

More precisely, the system (12.5) has a solution λ if and only if the direction $e_s(v_s, r)$ determines an edge of P incident to v. If it has a feasible solution, then by Proposition 12.4, the vertex $\hat{v}$ adjacent to v along this direction is given by

$$\hat{v} = \hat{v}_1 + \hat{v}_2 + \cdots + \hat{v}_k$$
$$\hat{v}_j = \begin{cases} \mathrm{Adj}_j(v_j, i) & \text{if there exists } i \text{ such that } (j, i) \in \Delta(v, s, r) \\ v_j & \text{otherwise.} \end{cases}$$

Let us denote by $\Delta(v)$ as the set of all pairs $(s, r) \in \Delta$ such that $e_s(v_s, r)$ determines an edge of P and (s, r) is a member of $\Delta(v, s, r)$ with the smallest first index. This set represents a duplication-free index set of all edge directions at v.

Now we are ready to define our adjacency oracle as a function $\mathrm{Adj} : V \times \Delta \to V \cup \{\mathit{null}\}$ such that

$$\mathrm{Adj}(v, (s, r)) \;\; = \begin{cases} \hat{v} & \text{if } (s, r) \in \Delta(v) \\ \mathit{null} & \text{otherwise.} \end{cases} \tag{12.6}$$

Lemma 12.10 *One can evaluate the adjacency oracle* $\mathrm{Adj}(v, (s, r))$ *in time* $\mathrm{LP}(d, \delta)$.

Proof. The essential part of the evaluation is solving the system (12.5). Since $\delta = |\Delta|$, the system has d variables and at most δ inequalities and the claim follows. ∎

Lemma 12.11 *There is an implementation of the local search function* $f(v)$ *with evaluation time* $O(\mathrm{LP}(d, \delta))$*, for each* $v \in V \setminus \{v^*\}$ *with the Minkowski decomposition* $v = v_1 + v_2 + \cdots + v_k$.

Proof. The implementation of f essentially depends on how we define the canonical vector of the normal cone $N(v; P)$. Like in the adjacency oracle implementation, we use an LP formulation. Since the set of directions $e_j(v, i)$ for valid $(j, i) \in \Delta$ include all edge directions at v, the normal cone $N(v; P)$ is the set of solutions λ to the system

$$e_j(v_j, i)^T\lambda \;\; \leq 0 \quad \text{for all valid } (j, i) \in \Delta.$$

Since we need an interior point of the cone, we formulate the following LP:

$$\begin{array}{lll} \max & \lambda_0 & \\ \text{subject to} & & \\ e_j(v_j, i)^T\lambda + & \lambda_0 \leq 0 & \text{for all valid } (j, i) \in \Delta \\ & \lambda_0 \leq K. & \end{array} \tag{12.7}$$

Here K is any positive constant. Since v is a vertex of P, this LP has an optimal solution. We still need to define a unique optimal solution. For this, we use a very pragmatic definition: fix one deterministic algorithm and define the canonical vector as the unique solution returned by the algorithm. Since the number of variables is $d+1$ and the number of inequalities is at most $\delta+1$, the assumptions on LP implies the time complexity $O(\mathrm{LP}(d,\delta))$ to compute the canonical vector. Note that for practical purposes, we should probably add bounding inequalities for λ to the LP (12.7) such as $-1 \le \lambda_i \le 1$ for all i to make sure that the optimal solution stays in a reasonable range. This does not change the complexity.

An execution of f requires to compute the canonical vectors c and c^*. Once they are computed, the remaining part is to determine the first bounding hyperplane of the normal cone $N(v;P)$ hit by the oriented line $t(\theta) := c + \theta(c^* - c)$ (as θ increases from 0 to 1). This amounts to solving at most δ one-variable equations, and is dominated by the canonical vector computation. ∎

In Figure 12.4, we present the resulting reverse search algorithm, where we assume that the δ index pairs (j,i) in Δ are ordered as $(1,1) < (1,2) < \cdots < (1,\delta_1) < (2,1) < \cdots < (k,\delta_k)$.

```
     procedure MinkowskiAddition($Adj$,($\delta_1,\ldots,\delta_k$), $v^*$,$f$);
          $v := v^*$; $(j,i) := (1,0)$; (* $(j,i)$: neighbor counter *)
          output $v$;
          **repeat**
               **while** $(j,i) < (k,\delta_k)$ **do**
                    increment $(j,i)$ by one;
(r1)                $next := Adj(v,(j,i))$;
                    **if** $next \ne null$ **then**
(r2)                     **if** $f(next) = v$ **then**   (* reverse traverse *)
                              $v := next$;$(j,i) := (1,0)$;
                              output $v$
                         **endif**
                    **endif**
               **endwhile**;
               **if** $v \ne v^*$ **then** (* forward traverse *)
(f1)                $u := v$;    $v := f(v)$;
(f2)                restore $(j,i)$ such that $Adj(v,(j,i)) = u$
               **endif**
          **until** $v = v^*$ and $(j,i) = (k,\delta_k)$.
```

Figure 12.4: Reverse Search Algorithm for Minkowski Sum

Finally, we are ready to prove the main theorem, Theorem 12.9.

Proof. We use the general complexity result, Corollary 9.3, saying the time complexity of the reverse search in Figure 12.4 is $O(\delta(t(\mathrm{Adj}) + t(f))|V|)$. By Lemma 12.10 and Lemma 12.11, both $t(\mathrm{Adj})$ and $t(f)$ can be replaced by $\mathrm{LP}(d,\delta)$. Since $f_0(P) = |V|$, the claimed time complexity follows. The space complexity is dominated by those of the functions f and Adj which are clearly linear in the input size. ∎

13 Problem Reductions in Polyhedral Computation

In this section, we look at some basic problems in polyhedral computation. Just like in combinatorial optimization, it is quite hard to distinguish hard problems (typically NP-hard problems) from easy problems. Here there are two sorts of easy problems. The first group consists of decision problems that are polynomially solvable. The second group consists of enumeration problems that may require output whose size is exponential in the input size, but are output-polynomially solvable.

We shall present some hard decision problems in Section 13.1, and discuss some hard enumeration problems in Section 13.2.

13.1 Hard Decision Problems in Polyhedral Computation

We start with two decision problems in polyhedral computation that are related to linear programming but are known to be hard.

For $A \in \mathbb{Q}^{m\times d}$ and $b \in \mathbb{Q}^m$, let $P_H(A, b)$ be the H-polyhedron

$$P_H(A, b) := \{x \in \mathbb{R}^d : Ax \leq b\}, \tag{13.1}$$

and let $\mathcal{A}(A, b)$ be the associated arrangement of hyperplanes:

$$\mathcal{A}(A, b) := \{h_1, \ldots, h_m\}, \tag{13.2}$$

$$h_i := \{x \in \mathbb{R}^d : A_i x \leq b_i\}. \tag{13.3}$$

Problem 13.1 Optimal Vertex of a Polyhedron (OVP)

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^d$ and $K \in \mathbb{Q}$.
- Question: Does there exists a vertex v of $P_H(A, b)$ with $c^T v \geq K$?

Problem 13.2 K-Vertex of a Polyhedron (KVP

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^d$ and $K \in \mathbb{Q}$.
- Question: Does there exists a vertex v of $P_H(A, b)$ with $c^T v = K$?

Theorem 13.3 ([26]) *The decision problems OVP and KVP are both NP-complete.*

Proof. It is clear that both problems are in the class NP. The proofs of the NP-completeness will be obtained by a polynomial time transformation from the following problem, known to be NP-complete in the strong sense [29]:

Problem 13.4 Directed Hamiltonian Path (DHP)

- Input: A directed graph $G = (V, A)$ and two distinct vertices $s, t \in V$.
- Question: Does G contain a directed Hamiltonian path from s to t ?

Let $G = (V, A)$ be a directed graph and $s \neq t \in V$. Associate a variable x_{ij} with each arc $(i, j) \in A$. Let $P(G)$ be the polytope given by:

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0, \qquad \text{for each } i \in V - \{s,t\}, \tag{13.4}$$

$$\sum_{j|(s,j)\in A} x_{sj} - \sum_{j|(j,s)\in A} x_{js} = 1, \tag{13.5}$$

$$\sum_{j|(t,j)\in A} x_{tj} - \sum_{j|(j,t)\in A} x_{jt} = -1, \tag{13.6}$$

$$x_{ij} \geq 0, \qquad \text{for each } (i,j) \in A. \tag{13.7}$$

The matrix of the coefficients of these inequalities is totally unimodular ([43], Proposition 2.6, p. 542) implying that $P(G)$ is integral. It follows that an extreme point x of $P(G)$ is the characteristic vector of a directed path joining s to t in G and, possibly, a set of circuits. If a circuit C exists, then x is a convex combination of the two points obtained by adding or subtracting small $\epsilon > 0$ on all the arcs of the circuit, a contradiction. Hence x is the characteristic vector of a simple directed path joining s to t. One verify easily that all such paths are extreme points of $P(G)$, proving that the extreme points of $P(G)$ are exactly the characteristic vectors of the simple directed paths joining s to t in G. These two facts thus imply that, for $K = |V| - 1$ and $c = \mathbf{1}$ (the vector of all 1's), both the OVP and the KVP problems for $P(G)$ are NP-complete in the strong sense. This completes the proof. ∎

There are similar complexity results for arrangements of hyperplanes.

Problem 13.5 Optimal Vertex of an Arrangement (OVA)

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^d$ and $K \in \mathbb{Q}$.
- Question: Does there exists a vertex v of $\mathcal{A}(A, b)$ with $c^T v \geq K$?

Problem 13.6 Optimal Vertex of an Arrangement (KVA)

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^d$ and $K \in \mathbb{Q}$.
- Question: Does there exists a vertex v of $\mathcal{A}(A, b)$ with $c^T v = K$?

Theorem 13.7 ([26]) *The decision problems OVA and KVA are both NP-complete.*

Proof. Consider an instance of DHP and build a corresponding instance for OVA and KVA as follows: To each arc $(i, j) \in A$, we associate a variable x_{ij}. Let $d := |A|$, $K := |V|-1$, $c = \mathbf{1}$ and define the arrangement generated by the following set of hyperplanes:

$$H_i \quad := \quad \{\mathbf{x} \in \mathbb{R}^d | \sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = 0\}, \qquad \text{for each } i \in V - \{s,t\}, \tag{13.8}$$

$$H_s \quad := \quad \{\mathbf{x} \in \mathbb{R}^d | \sum_{j|(s,j)\in A} x_{sj} - \sum_{j|(j,s)\in A} x_{js} = 1\}, \tag{13.9}$$

$$H_t \quad := \quad \{\mathbf{x} \in \mathbb{R}^d | \sum_{j|(t,j)\in A} x_{tj} - \sum_{j|(j,t)\in A} x_{jt} = -1\}, \tag{13.10}$$

$$H_{ij} \quad := \quad \{\mathbf{x} \in \mathbb{R}^d | \, x_{ij} = 0\}, \qquad \text{for each } (i,j) \in A. \tag{13.11}$$

First we observe that if DHP has a "yes" answer, so does the corresponding instance of OVA and KVA, as the characteristic vector of any directed Hamiltonian path lies on the $|V|$ hyperplanes H_i for $i \in V$ as well as on $(|A|-(|V|-1)) = |A|-|V|+1$ of the hyperplanes H_{ij} for $i \neq j \in V$. Note that the $|V|$ hyperplanes H_i for $i \in V$ are not linearly independent, but any subset of $(|V|-1)$ of them are. Hence there are $(|A|-|V|+1)+(|V|-1) = |A|$ linearly independent hyperplanes containing the characteristic vector of any directed Hamiltonian path joining s to t in G, implying that the latter is a vertex of the given arrangement.

Now suppose that KVA or OVA has a "yes" answer produced by a vertex v of the constructed instance. One can write the $|A|$ equations defining the hyperplanes of the instance as a system of the form $Ax = b$. It is well known that the matrix $[A, b]$ is totally unimodular (see [43] for example). Thus any vertex of the arrangement has only $+1, -1$, or 0 coordinates, as shown by Cramer's rule for solving a linear system.

Let S be a set of n linearly independent hyperplanes of the given family whose intersection is v. As the $|V|$ hyperplanes in $\{H_i | i \in V\}$ are not linearly independent, the number of these hyperplanes which are in S is at most $(|V|-1)$. Hence the number of non zero coordinates of v is at most $(|V|-1)$. As $c = \mathbf{1}$ and $c^T v \geq K = (|V|-1)$, we have that exactly $(|V|-1)$ coordinates of v are $(+1)$, all the others being (0)'s. Thus v is the characteristic vector of a set P of $(|V|-1)$ arcs of A. This also implies that KVA has a "yes" answer if and only if OVA has a "yes" answer

If P is a directed Hamiltonian path in G joining s to t, then we are done. Otherwise, P contains a directed path joining s to t in G and at least one directed cycle C. But consider $v' \in R^n$ defined by

$$v'_{ij} = \begin{cases} 0 & \text{if } (i,j) \in C, \\ v_{ij} & \text{otherwise,} \end{cases} \qquad \text{for each } (i,j) \in A. \tag{13.12}$$

This complete the proof. ∎

13.2 Hard Enumeration Problems in Polyhedral Computation

For matrices $V \in \mathbb{Q}^{s\times d}$ and $R \in \mathbb{Q}^{t\times d}$, the V-polyhedron with representation pair (V, R) is denoted by $P_V(V, R)$, i.e.,

$$P_V(V,R) := \{x : x = V\lambda + R\mu, \mathbf{1}^T\lambda = 1, \lambda \geq \mathbf{0}, \mu \geq \mathbf{0}\}. \tag{13.13}$$

The following decision problem is arguably the most important problem in polyhedral computation.

Problem 13.8 Polyhedral Verification Problem (PVP)

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, $V \in \mathbb{Q}^{s\times d}$ and $R \in \mathbb{Q}^{t\times d}$.
- Question: Is $P_H(A,b) \neq P_V(V,R)$?

It is not difficult to prove that if PVP is in P, then there is an output-polynomial algorithm for the polyhedral representation conversion problem discussed in Section 9, see Polyhedral Computation FAQ [23].

PVP is easily seen to be in NP, because if the polyhedra are not equal, there is a succinct certificate for it, a point x in one of the polyhedra which is not in the other. Unfortunately, the complexity of PVP is still open. The decision problem PVP was first posed by Lovasz, see [49], and has been extensively studied by many researchers.

One of the most exciting progresses is the NP-completeness of a closely related problem, due to Khachiyan et al. [35].

Problem 13.9 Vertex Enumeration for an Unbounded Polyhedron (VEU)

- Input: $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$ and $V \in \mathbb{Q}^{s\times d}$.
- Question: Does the H-polyhedron $P_H(A,b)$ contain a vertex not in V?

Theorem 13.10 ([26]) *The decision problems VEU is NP-complete.*

Proof. (Outline) It is easily seen to be in NP, because if the answer is yes, then there is at least one vertex not in V. The proof uses a reduction from the NP-complete problem:

Problem 13.11 Negative Circuit Enumeration (NCE)

- Input: A digraph $G = (V, E)$ with edge weight $w : E \to \mathbb{Q}$, and a family S of negative circuits of G.
- Question: Does G contain a negative circuit not in the family S?

Here, a *negative circuit* is a directed circuit $C \subset E$ whose total weight $\sum_{e\in C}$ is negative. It is shown by Khachiyan et al. [35] that NCE is NP-complete from a reduction from SAT. ∎

(This section is to be extended.)

14 Evolutions and Applications of Polyhedral Computation

Polyhedral Computation has been shaped and polished through actual demands from numerous mathematicians, scientists, engineers and even social scientists. In this section, we present the author's personal involvements in various external or internal projects which have driven the advancement of polyhedral computation and software developments.

1987 – 1992: The First Stage. The first generation of codes for polyhedral representation conversion were written first for mathematicians to understand certain combinatorial polyhedra, such as **cut polytopes**, cut cones, and **traveling salesman polytopes**. It is extremely difficult to determine the facet inequalities of these polyhedra because typical associated combinatorial optimization problems are NP-hard. However, by computing the H-representation from a V-representation for small instances, many new facet inequalities were discovered and used for finding a stronger LP relaxation of NP-hard optimization problems. The first version of my implementation of the double description algorithm described in Section 9.1 was released in January 1988 is called PDD where p stands for the programming language Pascal. It helped the early stage of research on cut polytopes by Michel Deza and Monique Laurent, see [17, 18].

1993 – 1996: The Critical Second Stage. Then, a more computationally demanding task was needed for research in **material science**. Two physicists G. Ceder and G.D. Garbulsky at MIT contacted both David Avis and myself in 1993, and asked for our computational help in enumerating all extreme points of a highly degenerate polytope in dimension 8 given by 729 inequalities. The vertices represent physically stable states of a ternary (3 elements) alloy. David had a C-implementation named RS (which was replaced by LRS later) of the reverse search algorithm given in Section 9.2 then, and I had a C-version named CDD of the earlier code PDD. Both David and myself devoted our effort to compute the vertices, and finally it took us about a month to compute the results. Our greatest excitement came when we verified that the final results computed by our implementations of two totally different algorithms returned exactly the same results. This successful computation lead to a paper by the four of us [12].

About the same time, then a doctoral student Francisco Valero of **neuromuscular systems** laboratory at Stanford University contacted me. The application Valero discovered then surprised me considerably. I did not imagine that one can find an application of polyhedral computation in human bodies and muscles. He email in May 1994 describing his application reads

> My application deals with **finger muscles having force limits** (i.e., form zero force to their maximum physiological force for each finger muscle) which defines a hypercube in a dimension equal to the number of muscles under consideration. Other mechanical, functional or anatomical characteristics produce further constrain equations (i.e., need the force of the finger to be zero in certain directions, finger size/configuration, etc.). The vertex enumeration technique helps me identify the limits of muscle force production

> under these constraints, which in turn maps into functional limits such as maximum finger forces, accelerations, etc, which are independent of muscle force coordination. Coordination can be studied with standard linear programming techniques. The limits of function, however, require the explicit enumeration of the vertices of convex polyhedra in n-dimensional space.

Valero has been a strong advocate of computational geometry techniques applied to **biomedical** and **biomechanics** fields since then. A recent paper [36] shows the **analysis of muscle redundancies** using the vertex enumeration in polyhedra.

From the software development front, a new C++ version of CDD, called CDD+, was released in April 1995 which has the capability of using both floating-point and rational exact arithmetic using GMP [1].

1997 – 2007: Developments of Polyhedral Computation Libraries. Further advancements were made during this period for the development of software C-libraries CDDLIB and LRSLIB, based on CDD and LRS, respectively by Fukuda [22] and Avis [2]. Naturally, these libraries have been integrated into other programs.

A versatile **R-interface** of CDDLIB was written by the statistician Charles Geyer of University of Minnesota. It is available from

http://www.stat.umn.edu/~charlie/

A webpage of computing **all Nash equilibria** of bimatrix games using LRSLIB written by Rahul Savani became available at

http://banach.lse.ac.uk/form.html.

A **Matlab toolbox** for the study of **control theory** with an emphasis on parametric optimization was written by a group of researchers at the system dynamics and control group at ETH Zurich. It has an interface called CDDMEX to cddlib and is available at

http://control.ee.ethz.ch/research/software.en.html

A **Python** interface PyPolyhedron to cddlib was written by Pearu Peterson. He wrote in his email in 2007 "I am using it for analyzing multi-soliton interactions. In terms of computational geometry, I just construct special polyhedron in $(N + 1)$-D space, project it to N-D space, and then find its intersection with 2-D hyperplane, which after projecting to 2-D space gives an interaction pattern of the N-soliton solution." It is available at http://cens.ioc.ee/projects/polyhedron/

Polymake is a platform to do polyhedral and algebraic computation mainly for mathematicians whose two core engines are CDDLIB and LRSLIB. It is available at http://www.polymake.org/doku.php

TOPCOM [45] is a package for computing Triangulations Of Point Configurations and Oriented Matroids. It uses the LP code of CDDLIB for the recognition of regular triangulations.

Minksum http://www.cs.dartmouth.edu/~weibel/minksum.php is a program to compute the V-representation (i.e. the set of vertices) of the Minkowski addition of several convex polytopes given by their V-representation. It is an implementation in C++

language of the reverse search algorithm given in Section 12.2 whose time complexity is polynomially bounded by the sizes of input and output. It relies on the exact LP solver of CDDLIB.

Gfan [33] is a program to list all reduced **Gröbner bases** of a general polynomial ideal given by a set of generating polynomials in -variables. It is an implementation in C++ language of the reverse search algorithm [25]. It relies on the exact LP solver of CDDLIB.

2004 – 2011: Expanding Application Fields. An application of **Minkowski sum of polytopes** presented in Section 12 is given in a doctoral thesis of J.P. Petit [44] in 2004, which is **computer aided tolerancing in design and manufacturing** using a mathematical model with convex polytopes in dimension 6. The dimension is simply $3+3$ where the first 3 is the dimension of the space and the latter is the freedom of movement in 3-space.

A polyhedral model was introduced in a doctoral research at Queen's University Belfast guided by Cecil Armstrong on **aircraft stress load evaluation and optimization**. The essential problem is to detect the most critical parts of aircrafts against a set of many likely stresses, which is reduced to the redundancy removal in linear inequality systems, the theme of Section 8.2.

To analyze the **effects of United Nations peacekeeping operations**, the danger of using the high dimensional analysis is pointed out in a paper by political scientists in [46], after a few researchers in computational geometry including myself presented counter-intuitive facts in higher dimensional spaces. In particular, one serious problem of estimating the effect of a future operation, a relatively small set of past instances represented by high dimensional points cannot be a reliable guidance, due to the fact that a new point will most likely be (far) outside of the **convex hull of the past data points**, and thus a wild extrapolation occurs at a high probability.

Future. From my personal involvements in polyhedral computation during the past 24 years, once reliable and efficient codes of polyhedral computation become available, new users might show up from any field of science, engineering, humanities and even arts. Thus, the main purpose of writing this book is to present the fundamental theory of polyhedra and the basic computational problems associated polyhedra with most efficient algorithmic techniques. My belief is that interesting applications should follow once researchers have easy access to the theory and computational codes. After all, convex polyhedra appear almost everywhere, implicitly or explicitly.

15 Literatures and Software

Course materials will be distributed as pdf files from our course webpage.

For the theory of linear programming, recommended sources are [47, 14, 24]. Some of the excellent books on combinatorial optimization are [16, 31, 48]. There are excellent books on convex polyhedra, see [39, 32, 51]. For good discussions on algorithmic and combinatorial aspects of polyhedra, see [41, 37, 34]. The standard text on oriented matroids is [9], and an excellent introduction is included in the dissertation [21] which is available online.

We use some of the freely available software packages such as LRSLIB [2] and CDDLIB [22] for polyhedral representation conversion.

References

[1] GMP, GNU's library for arbitrary precision arithmetic. http://gmplib.org/.

[2] D. Avis. *lrs homepage*. McGill University. http://cgm.cs.mcgill.ca/~avis/C/lrs.html.

[3] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms. *Computational Geometry: Theory and Applications*, 7:265–302, 1997.

[4] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8:295–313, 1992.

[5] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[6] M. Balinski. An algorithm for finding all vertices of convex polyhedral sets. *Journal of the Society for Industrial and Applied Mathematics*, pages 72–88, 1961.

[7] M. Balinski. On the graph structure of convex polyhedra in n-space. *Pacific J. Math.*, 11:431–434, 1961.

[8] C. Barber, D. Dobkin, and H. Huhdanpaa. *qhull, Version 2003.1*, 2003. program and report available from http://www.qhull.org/.

[9] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler. *Oriented matroids*. Cambridge University Press, Cambridge, second edition, 1999.

[10] D. Bremner. Incremental convex hull algorithms are not output sensitive. preprint, School of Computer Science, McGill University, Montreal, Canada, 1997. to appear in Discrete Comput. Geom.

[11] H. Bruggesser and P. Mani. Shellable decomposition of cells and spheres. *Mathematica Scandinavia*, 29:197–205, 1971.

[12] G. Ceder, G. Garbulsky, D. Avis, and K. Fukuda. Ground states of a ternary fcc lattice model with nearest and next-nearest neighbor interactions. *Physical Review B*, 49(1):1–7, 1994. pdf file available from http://prola.aps.org/abstract/PRB/v49/i1/p1_1.

[13] T. Christof and A. Löbel. PORTA: Polyhedron representation transformation algorithm (ver. 1.3.1), 1997. http://www.zib.de/Optimization/Software/Porta/.

[14] V. Chvatal. *Linear Programming.* W.H.Freeman and Company, 1983.

[15] K. L. Clarkson. More output-sensitive geometric algorithms. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 695–702, 1994. http://cm.bell-labs.com/who/clarkson/pubs.html.

[16] W. Cook, W. Cunningham, W. Pullyblank, and A. Schrijver. *Combinatorial optimization.* Series in Disctrete Mathematics and Optimization. John Wiley & Sons, 1998.

[17] M. Deza, K. Fukuda, and M. Laurent. The inequicut cone. *Discrete Mathematics*, 119:21–48, 1993.

[18] M. Deza and M. Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and Combinatorics.* Springer-Verlag, Berlin, 1997.

[19] M. Dyer and L. Proll. An algorithm for determining all extreme points of a convex polytope. *Mathematical Programmming*, 12:81–96, 1977.

[20] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[21] L. Finschi. *A graph theoretical approach for reconstruction and generation of oriented matroids.* Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 2001. http://www.ifor.math.ethz.ch/staff/finschi/.

[22] K. Fukuda. *cdd, cddplus and cddlib homepage.* Swiss Federal Institute of Technology, Zurich. http://www.inf.ethz.ch/personal/fukudak/cdd_home/index.html.

[23] K. Fukuda. Polyhedral computation FAQ, 2004. Both html and ps versions available from http://www.inf.ethz.ch/personal/fukudak/fukuda.html.

[24] K. Fukuda. Lecture notes: Optimization techniques, linear and combinatorial optimization. Technical report, Department of Mathematics, Swiss Federal Institute of Technology, Zurich, 2009. ps file available from ftp://ftp.ifor.math.ethz.ch/pub/fukuda/lecture/a09otnotes.pdf.

[25] K. Fukuda, A. Jensen, and R. Thomas. Computing Gröbner fans. preprint, 2005. http://www.arxiv.org/abs/math.AC/0509544, submitted to Mathematics of Computation.

[26] K. Fukuda, T. M. Liebling, and F. Margot. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry*, 8:1–12, 1997. http://www.sciencedirect.com/science/journal/09257721.

[27] K. Fukuda, S. Saito, and A. Tamura. Combinatorial face enumeration in arrangements and oriented matroids. *Discrete Applied Mathematics*, 31:141–149, 1991.

[28] K. Fukuda and C. Weibel. f-vectors of Minkowski additions of convex polytopes. *Discrete Comput. Geom.*, 37:503–516, 2007. http://www.springerlink.com/content/r552751m4081506l/.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

[30] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM J. Dics. Math.*, 6:246–269, 1993.

[31] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer-Verlag, Berlin, 1988.

[32] B. Grünbaum. *Convex polytopes*, volume 221 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, second edition, 2003. Prepared and with a preface by Volker Kaibel, Victor Klee and Günter M. Ziegler.

[33] A. Jensen. Gfan version 0.3: A user's manual, 2007. available from http://www.math.tu-berlin.de/~jensen/software/gfan/gfan.html.

[34] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Math. Programming*, 79(1-3, Ser. B):217–233, 1997. Lectures on mathematical programming (ismp97) (Lausanne, 1997), ps file available from http://www.ma.huji.ac.il/~kalai/papers.html.

[35] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. *Discrete Comput. Geom.*, 39(1-3):174–190, 2008.

[36] J. J. Kutcha and F. J. Valero-Cuevas. Muscle redundancy does not imply robustness to muscle dysfunction. *Mathematika*, 44:1264–1270, 2011.

[37] J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 2002.

[38] P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

[39] P. McMullen and G. Shephard. *Convex polytopes and the upper bound conjecture.* Cambridge University Press, 1971.

[40] T. Motzkin, H. Raiffa, G. Thompson, and R. Thrall. The double description method. In H. Kuhn and A.W.Tucker, editors, *Contributions to Theory of Games, Vol. 2.* Princeton University Press, Princeton, RI, 1953.

[41] K. Mulmuley. *Computational Geometry, An Introduction Through Randamized Algorithms.* Prentice-Hall, 1994.

[42] K. Murty. An algorithm for ranking all the assignments in order of increasing costs. *Operations Research*, 16:682–687, 1968.

[43] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[44] J. Petit. *Specification Geometrique des Produits: Methode d'Analyse de Tolerances.* Ph.D. Thesis, Université de Savoie, France, 2004.

[45] J. Rambau. *TOPCOM, a package for computing Triangulations of point configurations and oriented matroids.* http://www.rambau.wm.uni-bayreuth.de/.

[46] N. Sambanis and M. Doyle. No easy choices: Estimating the effects of Unoted Nat ions peacekeeping (response to King and Zeng. *International Studies Quarterly*, 51:217–226, 2007.

[47] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, New York, 1986.

[48] A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. A, B, C*, volume 24 of *Algorithms and Combinatorics.* Springer-Verlag, Berlin, 2003.

[49] P. Seymour. A note on hyperplane generation. *J. Combin. Theory, Series B*, 61:88–91, 1994.

[50] H. R. Tiwary. On the hardness of computing intersection, union and Minkowski sum of polytopes. *Discrete Comput. Geom.*, 40(3):469–479, 2008.

[51] G. Ziegler. *Lectures on polytopes.* Graduate Texts in Mathematics 152. Springer-Verlag, 1994.