Chapter 5
Computation of Convex Hulls

When referring to "computation of convex hulls" we understand this as the task of computing the $\mathcal{H}$-representation of the convex hull of a given finite point set $V \subseteq \mathbb{R}^n$. Depending on the desired application, one might also need to compute all faces, a description of the face lattice or other geometric information.

5.1 Preliminary Considerations

We begin with two simple results. First, Algorithm 5.1 immediately gives a trivial convex hull algorithm, which is, unfortunately, inefficient.

Theorem 3.9, together with the fact that the computed half-spaces define facets, shows that the algorithm is correct. The assumption that the affine hull is full-dimensional is not necessary. Without it, the algorithm can simply be applied to the affine hull of the input.

Algorithm 5.1: A trivial convex hull algorithm

Input: Finite point set $V \subseteq \mathbb{R}^n$ with $\dim \operatorname{aff} V = n$.
Output: Finite set of half-spaces $\{H_1^+, \ldots, H_m^+\}$ such that $\bigcap_{i=1}^m H_i^+ = \operatorname{conv} V$.

```
1 H ← ∅
2 foreach n-element subset W ⊆ V with dim aff W = n − 1 do
3     H ← aff W
4     if V ⊆ H⁺ then
5         H ← H ∪ {H⁺}
6     else
7         if V ⊆ H⁻ then
8             H ← H ∪ {H⁻}
9 return H
```

M. Joswig, T. Theobald, *Polyhedral and Algebraic Methods in Computational Geometry*, Universitext, DOI 10.1007/978-1-4471-4817-3_5,

Secondly, the dual problem, i.e., computing a $\mathcal{V}$-representation of a polytope from its $\mathcal{H}$-representation is, by polarity, algorithmically equivalent to the convex hull problem:

Theorem 5.1 *The problem of computing the $\mathcal{V}$-representation of a polytope from its $\mathcal{H}$-representation can be reduced to the convex hull problem and vice versa.*

Proof Let $P = \bigcap_{i=1}^{m} H_i^+$ be given in the $\mathcal{H}$-representation. Via the linear programs from Example 4.3 and Exercise 4.4 we can compute the affine hull $A = \operatorname{aff} P$ and a point from the relative interior x in $P = P \cap A$. We can thus assume that P is full-dimensional. We can also assume that the origin is an interior point, since we can otherwise apply our computations to A and translate by $-x$.

Since $0 \in \operatorname{int} P$, there exist $h_k^{(i)}$ such that $H_i^+ = [1 : h_1^{(i)} : \cdots : h_n^{(i)}]^+$. We now examine the polar polytope which has, according to Theorem 3.28, the $\mathcal{V}$-representation $P^\circ = \operatorname{conv}\{h^{(1)}, \ldots, h^{(m)}\}$. Using a convex hull algorithm we can obtain an $\mathcal{H}$-representation $P^\circ = \bigcap_{j=1}^{k} [1 : v_1^{(j)} : \cdots : v_n^{(j)}]^+$. Looking at the polar polytope of P° and using Theorem 3.29 we get

$$P = P^{\circ\circ} = \operatorname{conv}\left\{v^{(1)}, \ldots, v^{(k)}\right\}.$$

The reverse direction is similar. In fact, it is easier since it is not necessary to use the linear programming techniques used above. □

In the dual representation of the convex hull problem it becomes clear that the problem can be viewed as a far-reaching generalization of the linear optimization problem: While linear optimization aims at computing *one* specific vertex of an $\mathcal{H}$-polytope (defined by a linear objective function), the dual convex hull algorithm computes *all* vertices of P.

Note that the existence of cyclic polytopes of dimension n with m vertices and $\Theta(m^{\lfloor n/2 \rfloor})$ facets implies that there cannot exist a convex hull algorithm which is polynomial in m and n, since every such algorithm has to write the (in this case) exponentially many facets as output. Theorem 5.1 and the existence of the dual polytopes to cyclic polytopes imply that the dual convex hull problem has exponential run-time in the worst case. Now the natural question is if the naive algorithm from the beginning of this chapter can be optimized at all. There are two answers to this: First, by carefully analyzing the geometry we can exclude many hyperplanes which Algorithm 5.1 considers to be candidates for facets. We demonstrate how to do this in the next section. Secondly, the problem has a different quality when we assume the dimension n to be fixed: In Section 5.3 we will study the case $n = 2$. We provide further remarks and suggested literature at the end of this chapter.

5.2 The Double Description Method

To emphasize the relationship between the linear programming methods from the previous chapter and the convex hull problem, we study the convex hull problem in

its dual form. A basic approach is to order the affine hyperplanes which were given as input. Our goal is to take $\mathcal{V}$-representations of polytopes which are intersections of k hyperplanes to obtain $\mathcal{V}$-representations of polytopes which are intersections of $k+1$ hyperplanes. Such methods are called *iterative*. While reading this section, it is useful to think about how the specific steps can be translated into primal form.

Let P be an $\mathcal{H}$-polytope whose $\mathcal{V}$-representation $P = \operatorname{conv} V$ is already known. We now study how the $\mathcal{V}$-representation must be altered when another half-space H^+ is added. Define $P' = P \cap H^+$. The hyperplane H partitions the point set V into three parts: Points on the hyperplane and points on either of its two sides.

Lemma 5.2 *Let V_0, V_+, V_- be the partition of the point set V defined by*

$$V_0 = V \cap H, \qquad V_+ = V \cap H^+ \setminus H, \qquad V_- = V \cap H^- \setminus H.$$

Then we have

$$P' = \operatorname{conv}\big((V_0 \cup V_+) \cup \big\{[v,w] \cap H : v \in V_+,\ w \in V_-\big\}\big).$$

Proof It is obvious that the points in $V_0 \cup V_+$ are contained in P'. Furthermore, if $v \in V_+$ and $w \in V_-$, then the segment $[v,w]$ intersects the hyperplane H in one point which proves that $P' \supseteq \operatorname{conv} V'$.

For the reverse inclusion it is sufficient to examine the case where V is the vertex set of P. To find the vertices of P' we have to determine which cases have a supporting hyperplane of P' that intersects the polytope in exactly one point v. This happens when either v is a vertex of P (and contained in H^+) or v is the intersection of an edge of P with H. The edges of P are segments between vertices of P. The segment $[v,w]$ intersects the hyperplane H only in the two cases we mentioned, which proves the statement. □

Using Lemma 5.2 we can immediately provide a method to iteratively transform an outer description of a polytope $P \subseteq \mathbb{R}^n$ into an inner description. Without loss of generality we again assume $\dim P = n$.

The name of the method comes from the following concept.

Definition 5.3 Let $V = \{v^{(1)}, \ldots, v^{(m)}\}$ be a point set in $\mathbb{R}^n$ and $\mathcal{H} = \{H_1^+, \ldots, H_k^+\}$ a set of affine half spaces in $\mathbb{R}^n$. The pair $(V, \mathcal{H})$ is called a *double description* of a polytope P if we have

$$P = \operatorname{conv} V = H_1^+ \cap \cdots \cap H_k^+.$$

Exercise 5.4 How should the term 'double description' be extended to arbitrary polyhedra?

Let $P = H_1^+ \cap \cdots \cap H_m^+$, and write $P_k := H_1^+ \cap \cdots \cap H_k^+$. Up to a projective transformation (and renumeration) we can assume that P_{n+1} is an n-simplex (see Exercise 3.57). The $n+1$ vertices of P_{n+1} are precisely the intersections of each set of

Algorithm 5.2: A basic algorithm to compute the double description

Input: A set of affine half-spaces $\mathcal{H} = \{H_1^+, \ldots, H_m^+\}$ in $\mathbb{R}^n$, such that $P = H_1^+ \cap \cdots \cap H_m^+$ is bounded and full-dimensional and $P_{n+1} = H_1^+ \cap \cdots \cap H_{n+1}^+$ is an n-simplex.

Output: Point set V with $\operatorname{conv} V = P$

```
V_{n+1} ← set of vertices of P_{n+1}
for k ← n+2, ..., m do
    Construct V_k with conv V_k = P_k = P_{k-1} ∩ H_k^+ as in Lemma 5.2.
return V_m
```

n hyperplanes from $H_1, \ldots, H_{n+1}$. We can now inductively assume that we have already computed a $\mathcal{V}$-representation of $P_k = \operatorname{conv}\{v^{(1)}, \ldots, v^{(k)}\}$. Using Lemma 5.2 we obtain Algorithm 5.2.

This basic version of the algorithm is already more efficient than the trivial method described at the beginning of this chapter. However, we can still improve it with some simple steps. Note that we have $|V_k| \leq |V_{k-1}|^2$, i.e., the number of points might be squared in each step. The improvement that we introduce below does not completely avoid this "explosion" but it does have a positive effect by avoiding redundant computations, particularly when dealing with actual applications.

The point sets V_k which are iteratively generated in Algorithm 5.2 are in general too large, since they can contain points which are not vertices. Only the vertices are necessary for a $\mathcal{V}$-representation of a polytope. A possible improvement on this method would be to set up a linear program which at each step reduces the point set V_k to the set of vertices of P_k. This technique was previously used in Exercise 4.28.

However, we would like to avoid solving additional linear programs. The above mentioned refinement relies on the observation that vertices of P_k which are not vertices of P_{k-1} are generated by intersections of edges of P_{k-1} with the new hyperplane H_k. This fact was used in the proof of Lemma 5.2. Once we know which pairs of vertices in V_{k-1} generate edges of P_{k-1}, we will only have to test those particular vertices.

For $W \subseteq V$ let

$$\mathcal{H}(W) = \left\{H : H = \partial H^+ \text{ for an } H^+ \in \mathcal{H} \text{ and } W \subseteq H\right\}$$

be the set of supporting hyperplanes from $\mathcal{H}$ that contain all points of W. We abbreviate this as $\mathcal{H}(v, w) := \mathcal{H}(\{v, w\})$.

Lemma 5.5 *Let $(V, \mathcal{H})$ be a double description of an n-polytope $P \subseteq \mathbb{R}^n$. Given two distinct points $v, w \in V$, the set $\operatorname{aff}\{v, w\} \cap P$ is an edge of P if and only if the affine subspace $G := \bigcap \mathcal{H}(v, w)$ is one-dimensional. In this case $\operatorname{aff}\{v, w\} = G$ holds. Furthermore, if v and w are vertices then $\operatorname{conv}\{v, w\} = P \cap G$.*

Proof Observe that $\operatorname{aff}\{v, w\} \subseteq G = \bigcap \mathcal{H}(v, w)$. This is obvious for non-empty $\mathcal{H}(v, w)$. Otherwise we fix here the convention $\bigcap \emptyset = \mathbb{R}^n$.

First, let $e = \operatorname{aff}\{v, w\} \cap P$ be an edge of P. The affine hull of each face F of P is the intersection of those hyperplanes which define the facets of P that contain F. Since $(V, \mathcal{H})$ is a double description of P, the set $\mathcal{H}$ contains all affine hyperplanes that define facets of P. In addition, every affine hyperplane that contains v and w also contains the edge e. This implies that $\operatorname{aff}\{v, w\}$ is the intersection G of all supporting hyperplanes (from $\mathcal{H}$) that contain v and w.

For the reverse direction, let $\dim G = 1$, i.e., $\operatorname{aff}\{v, w\} = G$. In Theorem 3.6 we showed that the faces of faces of P are faces of P themselves. This implies that every intersection of supporting hyperplanes with P defines a face of P. In particular this holds for $G \cap P$ and the assumption about the dimension implies $\dim(G \cap P) \leq 1$. Since the points v and w of G were chosen to be distinct points of P we have that $G \cap P = e$ is an edge. □

To fully realize the advantages resulting from this lemma, we have to study how to make the double description $(V, \mathcal{H})$ accessible as a data structure. We also want to extend the convex hull problem in such a way that we can handle $\mathcal{H}$-descriptions of unbounded (fully-dimensional) pointed polyhedra. Handling non-pointed polyhedra is the task of Exercise 5.13. We showed in Chapter 3 that a polyhedron is pointed if and only if it is projectively equivalent to a polytope. As usual we use homogeneous coordinates. Geometrically, the transformation to homogeneous coordinates can be interpreted as follows. Instead of working with pointed polyhedra $P \subseteq \mathbb{R}^n$, we work with the polyhedral cones which are generated by P:

$$Q = \big\{(\lambda, \lambda x) : x \in P,\ \lambda \geq 0\big\} \subseteq \mathbb{R}^{n+1}.$$

The vertices and rays of P, which we originally wanted to compute, correspond to the uniquely defined minimal generating system of Q as a positive hull: Let $V, R \subseteq \mathbb{R}^n$ be given with

$$P = \operatorname{conv} V + \operatorname{pos} R$$

as in Exercise 3.41. Then we have

$$Q = \operatorname{pos}\big(\big\{(1, v) : v \in V\big\} \cup \big\{(0, r) : r \in R\big\}\big).$$

In the following let

$$W = \big\{w^{(1)}, \ldots, w^{(m)}\big\} := \big\{(1, v) : v \in V\big\} \cup \big\{(0, r) : r \in R\big\} \subseteq \mathbb{R}^{n+1}$$

be a positive generating system of the cone Q. To be able to distinguish P from its *homogenization* Q, we will refer to the elements of W as *vectors*. Through the homogenization, affine half-spaces in $\mathbb{R}^n$ become *linear half-spaces* in $\mathbb{R}^{n+1}$, i.e., affine half-spaces which contain the origin in $\mathbb{R}^{n+1}$. E.g., a simplex in $\mathbb{R}^n$ generates a *simplicial cone* in $\mathbb{R}^{n+1}$. The polytope edges, which played the key role in Lemma 5.5, correspond precisely to the two-dimensional faces of the homogenization.

The following is a useful way to represent the data: The coordinates of vectors from $W = \{w^{(1)}, \ldots, w^{(m)}\}$ are saved as columns of an $(n+1) \times m$-matrix which we will also call W. The linear half-spaces $\mathcal{H} = \{H_1^+, \ldots, H_k^+\}$ are represented by their coordinate vectors $h^{(1)}, \ldots, h^{(k)} \in (\mathbb{R}^{n+1})^*$ where we assume $H_i^+ = \{x : h^{(i)} x \geq 0\}$. By analogy to the vectors, we use $\mathcal{H}$ as the symbol for the $k \times (n+1)$-matrix consisting of the row vectors $h^{(1)}, \ldots, h^{(k)}$. We use the following homogeneous version of the incidence matrix of Section 3.6 and Exercise 3.55.

Definition 5.6 Let $(W, \mathcal{H})$ be the double description of a pointed cone $Q \subseteq \mathbb{R}^{n+1}$ with $W \in \mathbb{R}^{(n+1)\times m}$ and $\mathcal{H} \in \mathbb{R}^{k\times(n+1)}$. The matrix $I(W, \mathcal{H}) \in \{0,1\}^{k\times m}$ with $I(W, \mathcal{H}) = (I_{ij})$ defined by

$$I_{ij} = \begin{cases} 1 & \text{if } w^{(j)} \in H_i = \partial H_i^+, \text{ i.e., } h^{(i)} w^{(j)} = 0, \\ 0 & \text{otherwise} \end{cases}$$

is called the *incidence matrix* of $(W, \mathcal{H})$.

The rows of the incidence matrix $I := I(W, \mathcal{H})$ of the cone Q can be interpreted as the characteristic functions of the set of vectors from W which lie on the corresponding hyperplane. Analogously, the columns of I correspond to sets of supporting hyperplanes which contain a fixed vector from W. In this way we can determine the set $\mathcal{H}(w^{(r)}, w^{(s)})$ from Lemma 5.5 as the intersection of two sets which are given by characteristic functions; many programming languages allow for the efficient implementation of this as a bit-wise "and". This allows us to identify the set $\mathcal{H}(w^{(r)}, w^{(s)})$ with the submatrix consisting of those rows of the matrix $\mathcal{H}$ which have a 1 in their r-th and s-th column. The dimension of the intersection of all supporting hyperplanes which contain $w^{(r)}$ and $w^{(s)}$ is therefore $n+1$ minus the rank of the submatrix $\mathcal{H}(w^{(r)}, w^{(s)})$.

The natural formulation of the crucial Lemma 5.5 shows that it is most convenient to study the double description method in the homogeneous setting. Putting the pieces together, as shown in Algorithm 5.3, we can compute a minimal positive generating system of a polyhedral cone in $\mathbb{R}^{n+1}$ defined by linear inequalities. This is slightly more general than computing convex hulls.

We conclude this section with a detailed description of an example of the functionality of the loop in Steps 8 to 11 of Algorithm 5.3.

Example 5.7 Let $n = 3$ and

$$\mathcal{H} = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 1 & -1 & 2 & 0 \\ 1 & 2 & -1 & 0 \\ 1 & 0 & 0 & 1 \\ 2 & -1 & -1 & -1 \end{pmatrix} \in \mathbb{R}^{5\times 4}.$$

One can easily verify that the cone $Q = \{x \in \mathbb{R}^4 : \mathcal{H}x \geq 0\}$ is full-dimensional, since the ray $\mathbb{R}_{\geq 0}(1,0,0,0)^T$ passes through the interior. Furthermore, we have that $Q_4 =$

Algorithm 5.3: An algorithm for the double description in homogeneous form

Input: Matrix $\mathcal{H} \in \mathbb{R}^{k\times(n+1)}$ with row vectors $h^{(1)}, \ldots, h^{(k)}$ such that $Q = \{x \in \mathbb{R}^{n+1} : \mathcal{H}x \geq 0\}$ is a full-dimensional pointed cone and $Q_{n+1} := \{x \in \mathbb{R}^{n+1} : h^{(1)}x \geq 0, \ldots, h^{(n+1)}x \geq 0\}$ is a simplicial cone.

Output: Set W of vectors with $\operatorname{pos} W = Q$

```
Let W_{n+1} ∈ R^{(n+1)×(n+1)} be a matrix whose columns positively generate Q_{n+1}.
for i ← n+2, ..., k do
    Create W_{i-1}^+ from those columns of W_{i-1} that lie on the positive side of h^(i) and create W_{i-1}^- from the columns on the negative side.
    if W_{i-1}^- = ∅ then
        W_i ← W_{i-1}
    else
        X ← ∅
        foreach Pair (w, w') of columns of W_{i-1}^+ and W_{i-1}^- do
            if rank H_{i-1}(w, w') = n − 1 then
                Choose x as generator of the kernel of the matrix H'_{i-1}(w, w') that consists of the rows of H_{i-1}(w, w') and h^(i).
                X ← X ∪ {x}
        Let W_i be the matrix consisting of the columns of W_{i-1} without the columns of W_{i-1}^- and enhanced by the column vectors from X.
return W_k
```

$\{x \in \mathbb{R}^4 : h^{(1)}x \geq 0, \ldots, h^{(4)}x \geq 0\}$ is a simplicial cone whose rays correspond to the columns of the following matrix

$$W_4 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & -1 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}.$$

Now the fifth and last row of the matrix $\mathcal{H}$ defines the subsets W_4^+ (consisting of the first three columns of W_4) and W_4^- (last column of W_4). The incidence matrix of the double description is then the following:

$$I(W_4, \mathcal{H}_4) = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

As an example we study the pair of rays $(w^{(1)}, w^{(4)}) \in W_4^+ \times W_4^-$. By the definition of the incidence matrix, the first two vectors $h^{(1)}$, $h^{(2)}$ satisfy $h^{(j)}w^{(1)} = 0$ and $h^{(j)}w^{(4)} = 0$ (for $j = 1, 2$). This gives

$$\mathcal{H}_4\big(w^{(1)}, w^{(4)}\big) = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 1 & -1 & 2 & 0 \end{pmatrix}$$

and

$$\mathcal{H}'_4\big(w^{(1)}, w^{(4)}\big) = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 1 & -1 & 2 & 0 \\ 2 & -1 & -1 & -1 \end{pmatrix}.$$

The matrix $\mathcal{H}_4(w^{(1)}, w^{(4)})$ clearly has rank 2, which implies that $\operatorname{pos}\{w^{(1)}, w^{(4)}\}$ is a face of the cone Q_4 of dimension $4-2 = 2 = n-1$. The vector $(1, 1, 0, 1)^T$ spans the kernel of $\mathcal{H}'_4(w^{(1)}, w^{(4)})$. Analogous computations for the pairs $(w^{(2)}, w^{(4)})$ and $(w^{(3)}, w^{(4)})$ yield two more columns. Putting this together we arrive at

$$W = W_5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 4 \end{pmatrix}.$$

If we now *dehomogenize*, i.e., we intersect $Q = \operatorname{pos} W = \{(x_0, x_1, x_2, x_3)^T \in \mathbb{R}^4 : \mathcal{H}x \geq 0\}$ with the affine hyperplane in $\mathbb{R}^4$ defined by $x_0 = 1$, we obtain a simple 3-polytope with five facets which is combinatorially equivalent to a prism over a triangle. The rows of $\mathcal{H}$ and the columns of W describe homogeneous coordinates of facets and vertices of P respectively. The incidence matrix defined by the vertices and facets of P coincides with the incidence matrix of the double description $(W, \mathcal{H})$ of the cone Q:

$$I(W, \mathcal{H}) = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

5.3 Convex Hulls in the Plane

Two dimensional polytopes coincide with convex polygons. The edges of a convex polygon form the facets and the vertices can be ordered cyclically (clockwise or counter-clockwise). Let a finite set of points in the plane be given as columns of a matrix $M \in \mathbb{R}^{2\times m}$. Then the planar convex hull problem is the computation of a list of column indices that defines such a cyclic ordering of the vertices. Depending on the context, it may be necessary to choose one of the two orientations or to fix a specific point as the starting vertex.

Algorithm 5.4: The `Divide-and-Conquer` method for computing convex hulls in the plane

Input: Finite point set $V = \{v^{(1)}, \ldots, v^{(m)}\} \subseteq \mathbb{R}^2$
Output: Vertices of conv V in cyclic order

```
1 if m ≤ 2 then
2     return V
3 else
4     Sort V by the first coordinate.
5     Divide V in two disjoint sets L and R, where L contains the left ⌊m/2⌋
      and R the right ⌈m/2⌉ points of V.
6     Recursively compute conv L and conv R.
7     Compute conv(L ∪ R) from conv L and conv R.
```

Note that degenerate cases of lower dimensional polytopes in $\mathbb{R}^2$ can be coded by such a list as well, which then contains only one index (if the dimension is 0), or two indices (if the dimension is 1). To keep the language simple, we shall call these degenerate polytopes *polygons*.

We now introduce an algorithm of Preparata and Hong [84], which relies on the commonly used "divide-and-conquer" principle of computer science. The basic idea is to divide the original problem into many sub-problems, solve these smaller problems recursively and combine the sub-solutions, thus forming a solution to the original problem. A classic example of this principle is the `MergeSort` sorting algorithm described in Appendix C.1.

To simplify the presentation of Preparata and Hong's algorithm we make an extra assumption. In the exercises at the end of this chapter we will see how to extend the algorithm to the general case. In contrast to the convention used elsewhere in this book, we say that a point set $V \subseteq \mathbb{R}^2$ is in *general position* if no three points are colinear and every *vertical* $[a : -1 : 0]$, for $a \in \mathbb{R}$, contains at most one point in V.

In Algorithm 5.4, the actual computational problem is of course hidden in the last step, where we have to compute the common convex hull of two polygons which are given as a cyclic list of vertices. Our assumption that no two points lie on the same vertical simplifies the situation since this implies that conv L and conv R are disjoint and that there exists a dividing vertical line. The central observation here is that in this situation those vertices of conv$(L \cup R)$ which are vertices of L (or R) are ordered successively by the cyclic ordering.

One consequence of L and R being vertically separated is that there exist four common supporting lines to L and R; see Fig. 5.1. As with smooth convex sets, we call these common supporting lines *double tangents*. Exactly two of these four double tangents define facets of the common convex hull of L and R. Since L and R are vertically separated we can talk about the *upper* and *lower* double tangent. Computing the common convex hull of L and R is therefore equivalent to computing the upper and lower double tangent of two vertically separated polygons. In addition,

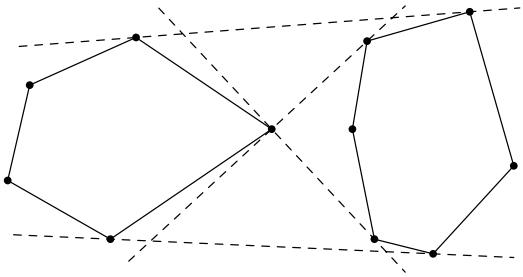

Fig. 5.1 The four double tangents to two disjoint polygons

the problem of computing the upper double tangent and the problem of computing the lower double tangent are equivalent since we can obtain the upper double tangent of L and R by computing the lower double tangent of $(-R, -L)$. Now we obtain an algorithm to compute the convex hull in the plane by combining the following algorithm with the divide-and-conquer method.

It remains to be checked if the Lower-Double-Tangent algorithm is correct. This is not obvious since it has to be shown that the outer loop terminates. To do this we need a further definition and a preliminary lemma.

Each pair of vertices v and w of a polygon defines two polygonal arcs, one in which v appears before w and one where v appears after w with respect to the counter-clockwise cyclic order of the polygon's vertices. For a polygon in general position, the left-most vertex and the right-most vertex define the *upper* and the *lower half*.

Lemma 5.8 *The lower double tangent to two vertically separated polygons L and R intersects both L and R in the lower half.*

Proof The lower half comprises precisely those facets whose outer normal points down. The outer normal of a supporting line to L (or R) which points down lies in the cone of normals of the facets of the lower half. □

Since the algorithm progresses cyclically in a fixed direction on both polygons, its termination is a consequence of the following statement. In some sense the interiors of L and R "block" the algorithm after finitely many steps.

Lemma 5.9 *There is no step of the algorithm where the segment $[v^{(i)}, w^{(k)}]$ could intersect the interior of L or of R.*

Proof In the beginning this condition is satisfied by construction. To show that the condition is satisfied in subsequent steps we use induction. Assume that $[v^{(i)}, w^{(k)}]$ does not intersect the interior of L and R. Using symmetry arguments we can also assume that i will be decreased in the next step. That is, we assume that $[v^{(i)}, w^{(k)}]$ is not a lower supporting line to L. Then $v^{(i-1)}$ lies below the line $\mathrm{aff}\{v^{(i)}, w^{(k)}\}$ and $[v^{(i-1)}, w^{(k)}]$ does not intersect the interior of L. □

Algorithm 5.5: `Lower-Double-Tangent`(L,R)

Input: Two finite polygons $L = (v^{(0)}, \ldots, v^{(l-1)})$ and $R = (w^{(0)}, \ldots, w^{(r-1)})$, given as a list of their vertices in cyclic order counterclockwise, such that there exists a separating vertical line where L lies on the left and R on the right side

Output: Lower double tangent T

```
1 v^(i) ← right-most vertex of L
2 w^(k) ← left-most vertex of R
3 while T ← aff{v^(i), w^(k)} is not lower double tangent do
4     while T is not a lower supporting line to L do
5         i ← i − 1 mod l
6     while T is not a lower supporting line to R do
7         k ← k + 1 mod r
8 return T
```

We would like to determine the complexity of the divide-and-conquer algorithm in its worst case. This will be done in a way that is typical for algorithms of this type. When regarding the input size, we neglect the point coordinates for which, for all geometric primitives, the same unit costs occur. The complexity of the algorithm `Lower-Double-Tangent` (Algorithm 5.5) is clearly $O(l+r)$. If we denote the complexity of `Divide-and-Conquer` by $C(m)$ we have the recursion $C(2m) = 2C(m) + O(m)$. First, we assume that the number of input points $m = 2^b$ is a power of 2. Every division step will then divide the point set into two sets of exactly the same size. Then we obtain

$$\begin{aligned} C(m) &= C\left(2^b\right) \\ &= 2C\left(2^{b-1}\right) + O\left(2^b\right) \\ &= 2\left(C\left(2^{b-2}\right) + O\left(2^{b-1}\right)\right) + O\left(2^b\right) = 2C\left(2^{b-2}\right) + 2O\left(2^b\right) \\ &= 2C\left(2^{b-3}\right) + 3O\left(2^b\right) = \cdots = bO\left(2^b\right) = O(m \log m). \end{aligned}$$

If m is not a power of 2, then the smallest power of 2 that is larger than m is at most twice as large as m. The complexity analysis above changes only by a multiplicative constant which is suppressed in the O-notation. We summarize this with the following theorem.

Theorem 5.10 *The algorithm* `Divide-and-Conquer` *computes the convex hull of m points in $\mathbb{R}^2$ with complexity $O(m \log m)$.*

5.4 Inspection Using `polymake`

`polymake` offers several convex hull algorithms, some of them via interfaces to other software, others as part of the `polymake` system. The double description algorithm is the standard algorithm. Internally, `polymake` calls `cddlib` [43].

We will start with the $\mathcal{V}$-description of a polytope. In contrast to the previous chapter where we entered the coordinates manually, we now use `polymake`'s standard constructions. The function `cube` with the single argument "`3`" generates the standard cube $[-1, 1]^3$.

```
polytope > $C3=cube(3);
```

The following function `edge_middle` takes the cube `$C3` as input, computes its edge mid-points and defines a new polytope as the convex hull of these. The task of Exercise 5.16 is to show that the edge mid-points are always the vertices of the new polytope.

```
$P=edge_middle($C3);
```

The new object `$P` comes with a range of properties which are already known.

```
polytope > print join " ", $P->list_properties();
VERTICES BOUNDED FEASIBLE
```

Each of them can be printed or used for further computations.

```
polytope > print $P->VERTICES;
1 0 -1 -1
1 -1 0 -1
1 1 0 -1
1 0 1 -1
1 -1 -1 0
1 1 -1 0
1 0 -1 1
1 -1 1 0
1 -1 0 1
1 1 1 0
1 1 0 1
1 0 1 1

polytope > print $P->BOUNDED, " ", $P->FEASIBLE;
1 1
```

The property `VERTICES` lists the vertices of the polytope in homogeneous coordinates. The boolean properties `BOUNDED` and `FEASIBLE` indicate that `$P` is a bounded polyhedron, i.e., a polytope, which is not empty. Performing a convex hull computation is now as easy as printing the `FACETS`.

```
polytope > print $P->FACETS;
1 0 0 -1
2 -1 1 -1
1 0 1 0
```

```
2 1 -1 1
1 1 0 0
2 1 1 1
2 1 1 -1
2 1 -1 -1
2 -1 1 1
1 0 0 1
2 -1 -1 1
2 -1 -1 -1
1 0 -1 0
1 -1 0 0
```

The polytope in `$P` is actually a *cuboctahedron* which is one of the *Archimedean solids*; see Fig. 5.2.

5.5 Exercises

Exercise 5.11 Let

$$P = \operatorname{conv}\left\{v^{(1)}, \ldots, v^{(m)}\right\} = H_1^+ \cap \cdots \cap H_l^+ \subseteq \mathbb{R}^n$$

be an n-polytope in double description with pairwise distinct half-spaces H_1^+, $\ldots$, H_l^+, and let $V_i := \{v^{(j)} \in H_i : 1 \leq j \leq m\}$ be the set of given points which lie on the hyperplane H_i. Show that H_i^+ is redundant if and only if there exists an index $k \in \{1, \ldots, l\}$ such that $V_i \subsetneq V_k$.

Exercise 5.12 Let $(V, \mathcal{H})$ be a double description of an $(n+1)$-polytope P and let $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be the linear projection to the first n coordinates. Exercise 3.58 shows that the image $\pi(P)$ is also a polytope. Compute a double description of $\pi(P)$.

Throughout the double description algorithm, the step-wise intersections with hyperplanes become iterated projections to coordinate subspaces in the polar form. In its dual form, this method corresponds to *Fourier–Motzkin-Elimination*. Exercise 5.12 illustrates one elimination step.

Exercise 5.13 How can we alter Algorithm 5.2 so that it also works for non-pointed polyhedra?

Exercise 5.14 How can we alter the divide-and-conquer algorithm from Section 5.3 to compute the area of a polygon that is defined by its vertices?

Exercise 5.15 How can we alter the divide-and-conquer algorithm so that it computes the convex hull of a point set that is not in general position?

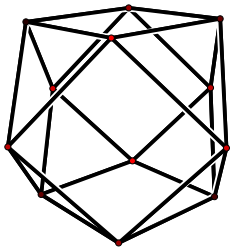

Fig. 5.2 The cuboctahedron

Exercise 5.16 Let P be an arbitrary polytope with vertex set $\{v^{(1)}, \ldots, v^{(m)}\} \subseteq \mathbb{R}^n$ and edge set

$$\left\{\left[v^{(i)}, v^{(j)}\right] : (i, j) \in I\right\}$$

for an appropriate set $I \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$. Show that the set of *edge midpoints*

$$W := \left\{\frac{1}{2}\left(v^{(i)} + v^{(j)}\right) : (i, j) \in I\right\}$$

is the vertex set of the polytope $\operatorname{conv} W$.

Figure 5.2 shows an example of the construction in Exercise 5.16 where P is the standard 3-cube.

5.6 Remarks

The double description algorithm which has briefly been introduced here is used in practical applications and is particularly useful for relatively high-dimensional non-simple polytopes. A detailed description can be found in Fukuda and Prodon [44].

The m vertices of an n-polytope defined by ℓ (facet defining) affine half-spaces can be computed in $O(\ell mn)$ time using the "reverse search" method of Avis and Fukuda [8]; reverse search works for non-simple polytopes as well, but in that setting is often inferior to the double description method; see Avis, Bremner and Seidel [7].

A further class of convex hull algorithms computes from the given point set, in addition to the facets of the convex hull, a triangulation. An example of this class is "beneath-and-beyond"; see Edelsbrunner [38, §8.4] and Joswig [67].

The divide-and-conquer principle can be extended and sometimes yields asymptotically optimal algorithms for lower dimensions. In dimension 2 and 3 one can obtain $O(m \log \ell)$-algorithms; see Clarkson and Shor [23] and Chan [19]. Chan, Snoeyink and Yap [20] describe an $O((m + \ell) \log^2 \ell)$-algorithm to compute the ℓ facets of a 4-polytope defined by m points.

The Upper-bound Theorem limits the number of facets of an n-polytope with m vertices to $\binom{m}{\lfloor n/2 \rfloor}$. When we fix the dimension n as a constant, then $\binom{m}{\lfloor n/2 \rfloor} \in$

$O(m^{\lfloor n/2 \rfloor})$ has a polynomial bound. Chazelle [21] was able to provide an algorithm which, for constant dimension, is in the worst case asymptotically optimal and has a run time of order $O(m \log m + m^{\lfloor n/2 \rfloor})$.

An interesting quality measurement for convex hull algorithms of arbitrary dimension can be obtained when we measure the run time with respect to the combination of input and output size. This is known as the *combined run-time* of a convex hull algorithm. It is unknown if there exists an algorithm that has a polynomially bounded combined run-time which computes the convex hull. Khachiyan et al. [69] recently showed that it is #P-hard (in combined run-time) to enumerate all vertices of an unbounded polyhedron which is given by inequalities. But this result does not imply that it is #P-hard (in combined run-time) to enumerate all vertices and additionally all rays. Therefore the complexity of enumerating all vertices of a polytope is still unknown.