26 CONVEX HULL COMPUTATIONS

Raimund Seidel

INTRODUCTION

The "convex hull problem" is a catch-all phrase for computing various descriptions of a polytope that is either specified as the convex hull of a finite point set in $\mathbb{R}^d$ or as the intersection of a finite number of halfspaces. We first define the various problems and discuss their mutual relationships (Section 26.1). We discuss the very special case of the irredundancy problem in Section 26.2. We consider general dimension d in Section 26.3 and describe the most common general algorithmic approaches along with the best run-time bounds achieved so far. In Section 26.4 we consider separately the case of small dimensions $d = 2, 3, 4, 5$. Finally, Section 26.5 addresses various issues related to the convex hull problem.

26.1 DESCRIBING CONVEX POLYTOPES AND POLYHEDRA

"Computing the convex hull" is a phrase whose meaning varies with the context. Consequently there has been confusion regarding the applicability and efficiency of various "convex hull algorithms." We therefore first discuss the different versions of the "convex hull problem" along with versions of the "halfspace intersection problem" and how they are related via polarity.

CONVEX HULLS

The generic convex hull problem can be stated as follows: Given a finite set $S \subset \mathbb{R}^d$, compute a description of $P = \text{conv}S$, the ***polytope*** formed by the convex hull of S.

A convex polytope P can be described in many ways. In our context the most important descriptions are those listed below.

GLOSSARY

(See Chapter 15 for basic concepts and results of polytope theory.)

Vertex description: The set of all vertices of P (specified by their coordinates).

Facet description: The set of all facets of P (specified by their defining linear inequalities).

Double description: The set of vertices of P, the set of facets of P, and the incidence relation between the vertices and the facets (specified by an incidence matrix).

Lattice description: The face lattice of P (specified by its *Hasse diagram* (cf.

below), with vertex and facet nodes augmented by coordinates and defining linear inequalities, respectively).

Boundary description: A triangulation of the boundary of P (specified by a simplicial complex, with vertices and maximal simplices augmented by coordinates and defining normalized linear inequalities, respectively).

Hasse diagram: A directed graph of an order relation that joins nodes a to b iff $a \leq b$ and there are no elements between a and b in the sense that if $a \leq c \leq b$ then either $c = a$ or $c = b$. For the face lattice, the order relation is containment.

The five descriptions above assume that P is full-dimensional. If it is not, then a specification of the smallest affine subspace containing P has to be added to all but the vertex description.

These five descriptions make explicit to varying degrees the geometric information carried by polytope P and the combinatorial information of its facial structure. The vertex description and the facet description each carry only rudimentary geometric information about P. We therefore call them ***purely geometric descriptions***. The other three descriptions we call ***combinatorial*** since they also carry more or less complete combinatorial information about the face structure of P. As a matter of fact, these three descriptions are equivalent in the sense that one can be computed from the other by purely combinatorial means, i.e., without the use of arithmetic operations on real numbers.

Which description is to be computed depends on the application at hand. It is important to keep in mind, however, that these descriptions can differ drastically in terms of their sizes (see Section 26.3).

INTERSECTION OF HALFSPACES

Closely related to the convex hull problem is the ***halfspace intersection*** problem: Given a finite set H of halfspaces in $\mathbb{R}^d$, compute a description of the *polyhedron* $Q = \bigcap H$.

Convex polyhedra are more general objects than convex polytopes in that they need not be bounded. Consequently their descriptions are slightly more complicated. Every polyhedron Q admits a "factorization" $Q = L + C + R$, where L is a linear subspace orthogonal to C and R, the set C is a convex cone, and R is a convex polytope. The "vertex description" of Q then consists of a minimal set of vectors spanning L, the set of extreme rays of C, and the set of vertices of R. Our other four description methods for convex polytopes have to be adjusted accordingly in order to apply to polyhedra. Also, the triangulations appearing in the boundary description need to allow for unbounded simplices (this concept makes sense if one views a k-simplex as an intersection of $k + 1$ halfspaces).

Because polyhedra are more general than polytopes, all statements about the size differences among the various descriptions of the latter apply also to the former.

POLARITY

The relationship between computing convex hulls and computing the intersection of halfspaces arises because of ***polarity*** (Section 15.1.2). Let S be a finite set in $\mathbb{R}^d$ and let H_S be the set of halfspaces $\{h_p \mid p \in S\}$, with $h_p = \{x \mid \langle x, p \rangle \leq 1\}$. Let

$P = \mathrm{conv}S$ and let $Q = \bigcap H_S$. Polarity yields a 1-1 correspondence between the k-faces of Q and the $(d-k)$-faces of P that admit supporting hyperplanes having P and the origin strictly on the same side. In particular, if the origin is contained in the relative interior of P, then the face lattices of P and Q are anti-isomorphic.

It is thus easy to reduce a convex hull problem to a halfspace intersection problem: First translate S by $-\sum_{p\in S} p/|S|$ to insure that the origin is contained in the relative interior of P, and compute $Q = \bigcap H_S$ for the resulting H_S. The polytope Q is then the polar P^Δ of P, and, assuming that P is full-dimensional, we have straightforward correspondences between the vertex description of Q and the facet description of P, between the facet description of Q and the vertex description of P, between the double descriptions of Q and of P (reverse the roles of vertices and facets), and between the lattice descriptions of Q and P (reverse the order of the lattice). Note that there is *no* correspondence between the boundary descriptions. If P has dimension $l < d$ then $Q = Q' \times L$, where polytope Q' has dimension l and L is a linear subspace of dimension $d-l$. The indicated correspondences then hold between P and Q'.

Reducing a halfspace intersection problem to a convex hull problem is more difficult. Polarity assumes all halfspaces to be describable as $\{x \mid \langle a, x\rangle \leq 1\}$, which means they must strictly contain the origin. In general not all halfspaces in a set H will be of such a form. In order to achieve this form the origin must be translated to a point r that is contained in the interior of $Q = \bigcap H$. Determining such a point r requires solving a linear program. Moreover, such an r does not exist if Q is empty, in which case the halfspace intersection problem has a trivial solution, or if Q is not full-dimensional, in which case one has to perform some sort of dimension reduction.

In general, halfspace intersection appears to be a slightly more general and versatile problem, especially in a homogenized formulation, which very elegantly avoids various special cases (see, e.g., [MRTT53]). Nevertheless, we will concentrate exclusively on the convex hull problem. The stated results can be translated *mutatis mutandis* to the halfspace intersection problem. In many cases the algorithms can be "dualized" to apply directly to the halfspace intersection problem, or the algorithms were originally stated for the halfspace intersection problem and were "dualized" to the convex hull problem.

26.2 THE IRREDUNDANCY PROBLEM

GLOSSARY

Irredundancy problem: Given a set S of n points in $\mathbb{R}^d$, compute the vertex description of $P = \mathrm{conv}S$.

$\lambda(n,d)$: The time to solve a linear programming problem in d variables with n constraints. $O(n)$ for fixed d (see Chapter 49).

This problem seeks to compute all points in S that are irredundant, in the sense that they cannot be represented as a convex combination of the remaining points in S. The equivalent polar formulation requires computation of the facet description of $Q = \bigcap H$, given a set H of n halfspaces in $\mathbb{R}^d$. We will follow the primal formulation.

The flavor of this version of the convex hull problem is very different from the other versions. Testing whether a point $p \in S$ is irredundant amounts to solving a linear programming problem in d variables with $n-1$ constraints. The straightforward method of successively testing points for irredundancy results in an algorithm with running time $O(n\lambda(n-1,d))$, which for fixed dimension d is $O(n^2)$.

Clarkson [Cla94] and independently Ottmann et al. [OSS95] have ingeniously improved this method so that every linear program involves only at most V constraints, where V is the number of vertices of P, i.e., the output size. The resulting running time is $O(n\lambda(V,d))$, which for fixed d is $O(nV)$.

In each of these two methods the n linear programs that occur are closely related to each other. This can be exploited, at least theoretically, by using data structures for so-called linear programming queries [Mat93, Cha96a, Ram00]. This was first done by Matoušek for the naive method [Mat93], and then by Chan for the improved method [Cha96], resulting for fixed $d > 3$ in an asymptotic time bound of

$$O(n \log^{d+2} V + (nV)^{1-1/(\lfloor d/2 \rfloor+1)} \log^{O(1)} n)\,.$$

Finally, note that for the small-dimensional case $d = 2, 3$ there are even algorithms with running time $O(n \log V)$ (see Chapter 42), which can be shown to be asymptotically worst-case optimal [KS86].

26.3 COMPUTING COMBINATORIAL DESCRIPTIONS

GLOSSARY

Facet enumeration problem: Compute the facet description of $P = \operatorname{conv} S$, given S.

Vertex enumeration problem: Compute the vertex description of $Q = \bigcap H$, given H.

The facet and vertex enumeration problems are classical and were already considered as early as 1824 by Fourier (see [Sch86, pp. 209–225] for a survey). Interestingly, no *efficient* algorithm is known that solves these enumeration problems without also computing, besides the desired purely geometric description, some combinatorial description of the polyhedron involved. Consequently we now concentrate on computing combinatorial descriptions.

THE SIZES OF COMBINATORIAL DESCRIPTIONS

It is important to understand how the three combinatorial descriptions differ in terms of their sizes. Let S be a set of n points in $\mathbb{R}^d$ and let $P = \operatorname{conv} S$. Assume that P is a d-polytope and that it has m facets. As a consequence of McMullen's Upper Bound Theorem (Chapter 15) and of polarity, the following inequalities hold between n and m and are tight:

$$n \le \mu(d,m) \qquad \text{and} \qquad m \le \mu(d,n)\,,$$

where

$$\mu(d,x) = f_{d-1}(C_d(x)) = \binom{x - \lceil d/2 \rceil}{\lfloor d/2 \rfloor} + \binom{x - 1 - \lceil (d-1)/2 \rceil}{\lfloor (d-1)/2 \rfloor},$$

which is $\Theta(x^{\lfloor d/2 \rfloor})$ for fixed d.

For the sake of definiteness let us define the sizes of the various descriptions as follows. For the double description of P it is the number of vertex-facet incidences, for the lattice description it is the total number of faces (of all dimensions) of P, and for the boundary description it is the number of $(d-1)$-simplices in the boundary triangulation.

Note that for the double and the lattice descriptions the sizes are completely determined by P, whereas the size of a boundary description depends on the boundary triangulation that is actually used. The sizes of those triangulations for a given P can vary quite drastically, even if, as we assume from now on, all vertices of the triangulation must be from S.

These size measures are only crude approximations of the space required to store such descriptions in memory (in particular, in case of the lattice description the edges of the Hasse diagram are completely ignored). However, these approximations suffice to convey the possible similarities and differences between the sizes of the different descriptions.

For such a comparison between the description sizes of $P = \text{conv}S$ consider Table 26.3.1, whose columns deal with three cases. The first column lists worst-case upper bounds in terms of n and d. The second column lists upper bounds in terms of m and d under the assumption that S is in nondegenerate position, i.e., no $d+1$ points in S lie in a common hyperplane, which means that P must be simplicial. Note that in this case there is a unique boundary description. Finally, the third column lists asymptotic bounds (d fixed) for *products of cyclic polytopes* $CC_d(n)$, a certain class of highly degenerate polytopes described in [ABS97]. (See Section 15.1.4 for a discussion of cyclic polytopes.) In this third table column, $\delta = \lfloor \sqrt{d/2} \rfloor$.

TABLE 26.3.1 Polytope description sizes.

DESCRIPTION	WORST CASE	NONDEGENERATE	DEGENERATE CLASS $CC_d(n)$
Double	$d \cdot \mu(d,n)$	$d \cdot m$	$\Theta(n \cdot m^{1-1/\delta})$
Lattice	$2^d \cdot \mu(d,n)$	$2^d \cdot m$	$\Theta((n+m)^\delta)$
Boundary	$\mu(d,n)$	m	$\Omega((n+m)^\delta)$

The bounds in the table are based on the fact that all description sizes are maximized when P is a cyclic polytope, that each facet of a simplicial d-polytope contains 2^d faces, and that the Upper Bound Theorem also applies to simplicial spheres. The lower bound on the size of the boundary description of $CC_d(n)$ applies no matter which triangulation of the boundary is actually used.

The implication of this table is that in the worst case and also in the nondegenerate case all three combinatorial descriptions of P have approximately the same size. If d is considered constant, then the sizes are $\Theta(n^{\lfloor d/2 \rfloor})$ in the worst case, where n is the number of points in S (i.e., n is the input size), and the description sizes are $\Theta(m)$ in the nondegenerate case, where m is the number of facets of P (in a way the output size). The third column of the table, however, shows that in the general case the double description of a polytope P may be substantially more compact than the lattice description or the boundary description.

MAIN RESULTS AND OPEN PROBLEMS

The main positive results are that in the sense of asymptotic worst case complexity the convex hull problem has been solved completely, and that in the case of nondegenerate input, each of the three combinatorial descriptions can be found in time polynomial in the size of the input and the size of the output. In the case of general input this has only been shown for the lattice and for a boundary description, whereas it is unknown whether this is also possible for the double description.

In the following let $P = \text{conv}S$ be a d-polytope, and $|S| = n$.

THEOREM 26.3.1 *Chazelle* [Cha93]

If the dimension d is considered constant, then given S, each of the three combinatorial descriptions of $P = \text{conv}S$ can be computed in time $O(n \log n + n^{\lfloor d/2 \rfloor})$ using space $O(n^{\lfloor d/2 \rfloor})$. This is asymptotically worst-case optimal.

THEOREM 26.3.2 *Avis-Fukuda* [AF92]

Given S, a boundary description of $P = \text{conv}S$ can be computed in time $O(dnM)$ using space $O(dn)$, where M is the size of the boundary description produced.

If S is nondegenerate, then each of the three combinatorial descriptions of P can be computed in time $O(d^{O(1)}nM)$, where M is the size of the respective description.

THEOREM 26.3.3 *Swart* [Swa85] *and Chand-Kapur* [CK70]

Given S, the lattice description of $P = \text{conv}S$ can be computed in time and space polynomial in d, n, and the size of the output.

OPEN PROBLEM 26.3.4

Is there an algorithm that, given S, computes the double description of $P = \text{conv}S$ in time polynomial in d, n, and the size of the double description?

The algorithm in Chazelle's theorem appears to be of theoretical interest only. The algorithm of Avis-Fukuda is quite practical, the algorithms of Swart and of Chand and Kapur are less so because of the potentially large space requirements. (See Chapters 67 and 68 for descriptions of available code.) The running times of the last two algorithms admit some theoretical improvements, as will be discussed in the following sections.

Almost all algorithms that have been published for solving the different versions of the convex hull problem and the halfspace intersection problem appear to be variations of three general methods: incremental, graph traversal, and divide-and-conquer. We discuss the incremental and the graph traversal methods in the next two subsections. Divide-and-conquer has proven useful only for very small dimension, and we will discuss it in that context in Section 26.4. Methods that fall outside this threefold classification are discussed in Subsection 26.3.3.

26.3.1 THE INCREMENTAL METHOD

The incremental method puts the points in S in some order $p_1, \ldots, p_n$ and then successively computes a description of $P_i = \text{conv}S_i$ from the description of P_{i-1} and p_i, where $S_i = \{p_1, \ldots, p_i\}$.

Before discussing details it should be noted that no matter how the incremental method is implemented, it has a serious shortcoming in that the intermediate polytopes P_i may have many more facets than the final $P_n = P$ (see, e.g., [ABS97]). Thus the description sizes of the intermediate polytopes may be much larger than the size of the description of the final result, and hence this method cannot have running time that depends reasonably on the output size.

This is not necessarily just the result of an unfortunate choice of the insertion order, since Bremner [Bre99] has shown that if S is the vertex set of the aforementioned product of cyclic polytopes $CC_d(n)$, then P_{n-1} has $\Omega(m^{\lfloor\sqrt{d/2}\rfloor-1})$ facets no matter which insertion order is used, where m is the number of facets of $P_n = P$.

We first present a selection of algorithms implementing the incremental method and list their asymptotic worst-case or expected running times for fixed d (Table 26.3.2). All these algorithms compute boundary descriptions, except for [Sei81] (see also [Ede87, Section 8.4]), which can also be made to compute a lattice description, and [MRTT53], which computes a double description.

TABLE 26.3.2 Sample of incremental algorithms.

ALGORITHM	TIME	BOUND TYPE
Kallay [PS85, Section 3.4.2]	$n^{\lfloor d/2\rfloor+1}$	worst-case
Seidel [Sei81]	$n\log n + n^{\lceil d/2\rceil}$	worst-case
Chazelle [Cha93]	$n\log n + n^{\lfloor d/2\rfloor}$	worst-case
Clarkson-Shor [CS89]	$n\log n + n^{\lfloor d/2\rfloor}$	expected
Clarkson et al. [CMS93]	$n\log n + n^{\lfloor d/2\rfloor}$	expected
Motzkin et al. [MRTT53]	$n^{3\lfloor d/2\rfloor+1}$	worst-case

We now concentrate on how P_{i-1} and P_i differ. For the sake of simplicity we will first assume that S is nondegenerate and hence all involved polytopes are simplicial. Moreover we will ignore how the insertion method starts and assume that P_{i-1} and P_i are full-dimensional. We say that a facet of P_{i-1} is ***visible*** (from p_i) if its supporting hyperplane separates P_{i-1} and p_i. Otherwise the facet is ***obscured***.

The facet set of P_i consists of "old facets," namely all obscured facets of P_{i-1}, and "new facets," namely facets of the form $\text{conv}(R \cup \{p_i\})$, where R is a "horizon" ridge of P_{i-1}, i.e., R is contained in a visible and in an obscured facet of P_{i-1}.

Updating P_{i-1} to P_i thus requires solving three subproblems: finding (and deleting) all visible facets of P_{i-1}; finding all horizon ridges; forming all new facets. The various incremental algorithms only differ in how they solve those subproblems, and they differ in the type of insertion order used.

Visible facets. The simplest way of finding the visible facets is simply to check each facet of P_{i-1}. This is done in Kallay's "beneath-beyond" method [PS85, Section 3.4.2] and in the "double description method" of Motzkin et al. [MRTT53].

Since P_i may have $\Theta(i^{\lfloor d/2 \rfloor})$ facets, such an approach automatically leads to a suboptimal overall running time of $\Omega(n^{\lfloor d/2 \rfloor + 1})$ in the worst case.

Another way is to maintain "conflict lists" between facets and not yet inserted points. In the worst case this is no better than the previous method. However, if the insertion order is a random permutation of the points in S, then in expectation this method works in $O(n^{\lfloor d/2 \rfloor})$ time [CS89].

The last method requires the maintenance of a ***facet graph***, whose nodes are the facets and whose arcs connect facets if they share a common ridge. The visible facets form a connected subgraph of this facet graph. Thus they can be determined by graph search, such as depth-first search. This takes time proportional to the number of visible facets, which means that in the amortized sense this takes no time since all those visible facets will be deleted. This graph search requires that one starting visible facet be known. Such an initial visible facet can be determined relatively efficiently by a special choice of the insertion order, as in [Sei81], by maintaining "canonical visible facets," as in [CS89] and [CMS93], or by linear programming, as in [Sei91].

Horizon ridges. Determining the horizon ridges is trivial if the facet graph is used, since those ridges correspond to arcs connecting visible and obscured facets. Otherwise one has to use data structuring techniques to determine which of the ridges incident to the visible facets are incident to exactly one visible facet.

New facets. After the horizon ridges are determined, the new facets are easily constructed in time proportional to their number. Keeping this number small is one of the main difficulties of making the insertion method efficient. In the worst case there may be as many as $\mu(d-1, i-1) = \Theta(i^{\lfloor (d-1)/2 \rfloor})$ such new facets. For even d this is $\Theta(i^{\lfloor d/2 \rfloor - 1})$, which is the main reason why it was relatively easy to obtain an asymptotically worst-case optimal running time of $O(n^{\lfloor d/2 \rfloor})$ for even d [Sei81]. For general d, using a random insertion order [CS89, CMS93, Sei91] appears to be the only known way to keep this number low, at least in terms of expectation. Chazelle's celebrated deterministic algorithm [Cha93] applies derandomization and thus in effect "simulates" random insertion order so that the number of new facets is not only small in the expected sense but also in the worst case.

Finally, if a facet graph is used, then the arcs corresponding to the ridges between the new facets need to be generated, which can be done via data structuring techniques, as in [Sei91], or by graph traversal techniques, as in [CS89, CMS93]. We should mention that if we remove the nondegeneracy assumption this problem of determining the new ridges seems to become very difficult.

Degenerate input. So far we have assumed that the input set S be nondegenerate. If this is not the case, then this can be simulated using perturbation techniques [Sei96]. This way the algorithms produce a boundary description from which a lattice description or a double description could be computed in $O(n^{\lfloor d/2 \rfloor})$ worst-case time.

The algorithm of Seidel [Sei81] (see also [Ede87, Section 8.4]) also works with degenerate input and then produces a lattice description. Most interesting, though, in the case of degeneracy is the so-called double description algorithm of Motzkin et al. [MRTT53].

THE DOUBLE DESCRIPTION METHOD

Although it is one of the oldest published incremental algorithms, this method has received little attention in the computational geometry community. This method maintains only the double descriptions of the polytopes P_i. It makes no assumptions about nondegeneracy. In fact, despite its poor worst-case complexity, empirically this method works well for degenerate inputs, where all other methods seem to fail, running out of time or space.

The algorithm determines the visible facets by simply checking all facets of P_{i-1}. The interesting point is how it determines the horizon ridges, from which the new facets are then constructed. In contrast to the other methods it does not maintain ridges, since, as we already mentioned, determining the new ridges created during an insertion is difficult. The double description method simply considers each pair of visible and obscured facets of P_{i-1} and checks whether their intersection A forms a horizon ridge. This is achieved by testing whether the vertex set in A is contained in some other facet of P_{i-1}. If it is, then A is not a ridge and hence not a horizon ridge.

A straightforward implementation of this idea will require $\Theta(i^{3\lfloor d/2 \rfloor})$ time in the worst case to discover all horizon ridges of P_{i-1}, resulting in a high worst-case overall running time. Although a number of heuristics have been proposed to speed up this process (see [Zie94, p. 48]), experiments show that this method is unbearably slow in the nondegenerate case when compared to other algorithms. However, in the case of degenerate input it still appears to be the method of choice with the new primal-dual approach (Section 26.3.3) as a possible contender.

Finally, we should mention that convex hull algorithms based on so-called Fourier-Motzkin elimination are nothing but incremental algorithms dressed up in an algebraic formulation.

26.3.2 THE GRAPH TRAVERSAL METHOD

This method attempts to traverse the facet graph of polytope $P = \text{conv}S$ in an organized fashion. The basic step is: given a facet F of P and a ridge R contained in F, find the other facet F' of P that also contains R. Geometrically this amounts to determining the point $p \in S$ such that the hyperplane spanned by R and p maximizes the angle to F. In analogy to a 3D physical realization this operation is therefore known as a "gift-wrapping step," and these algorithms are known as ***gift-wrapping algorithms***. In the polar context of intersecting halfspaces, this step corresponds to moving along an edge from one vertex to another and is equivalent to a pivoting step of the simplex algorithm for linear programming. Thus these algorithms are also known as ***pivoting algorithms***.

The basic outline of the graph traversal method is as follows: Find some initial facet of $P = \text{conv}S$ and the ridges that it contains. As long as there is an ***open ridge*** R, i.e., one for which only one containing facet F is known, perform a gift-wrapping step to discover the other facet F' containing R and determine the ridges that F' contains.

This general method faces three problems:

(a) How does one maintain the set of open ridges?

(b) How can the ridges of the new facet F' be quickly discovered?

(c) How can an individual gift-wrapping step be performed quickly?

THE NONDEGENERATE CASE

Let us again first assume that the input set S is in nondegenerate position. This trivializes problem (b) since every facet is a $(d-1)$-simplex and each of the d subsets with $d-1$ of its d vertices will span a ridge.

The most straightforward way to deal with problem (a) is to use some sort of dictionary data structure to store the set of open ridges. The most straightforward way to deal with (c) is to scan through all the points in S to find the best candidate, leading to work proportional to n per discovered facet. This straightforward method has been proposed many times (see [Sch86, p. 224] and [Chv83, p. 282] for references) and has running time $O(d^2nM)$ using $O(d(M+n))$ space, where M is the number of facets of P.

The gift-wrapping steps can be performed faster if a special data structure (for the dual of ray-shooting queries) is used. This was developed by Chan [Cha96], who achieved for fixed $d > 3$ an asymptotic time bound of

$$O(n \log M + (nM)^{1-1/(\lfloor d/2 \rfloor+1)} \log^{O(1)} n)\,.$$

Avis and Fukuda [AF92] proposed an ingenious way to deal with problem (a) so that no storage space is needed. They pointed out that there is a way of defining a canonical spanning tree T of the facet graph of polytope P so that the arcs of T can be recognized locally. Gift-wrapping steps are then performed only over ridges corresponding to arcs of T. Doing this in the form of a depth-first search traversal of T avoids the use of any extra storage space. Facets can be output as soon as they are discovered. Their algorithm is eminently practical and has a running time of $O(dnM)$ using only $O(dn)$ space.

In theory the gift-wrapping step improvement of Chan also could be applied to the algorithm of Avis and Fukuda. However, this appears to be of little practical relevance.

A completely different way of simultaneously addressing problems (a) and (c) was suggested by Seidel [Sei86a]. He proposed to try to discover the facets in an order corresponding to a straight-line shelling of P. In many cases gift-wrapping steps over several currently open ridges would yield the same new facet F'. However, in that case the entire vertex set of F' is known already and the expensive scan to solve problem (c) is not necessary. The facets of P for which this trick is not applicable can be discovered in advance by linear programming. This "shelling algorithm" has running time $O(n\lambda(n-1,d-1) + d^3M \log n)$, where $\lambda(n-1,d-1)$ is again the time necessary to solve a linear program with $n-1$ constraints in $d-1$ variables. From the way a shelling proceeds, one can prove that the space requirement for storing the open ridges is somewhat lower than in an ordinary gift-wrapping algorithm.

The linear programs that need to be solved are similar to the ones in the irredundancy problem of Section 26.2. Again, improvements can be achieved by applying linear programming queries ([Mat93]), and the $n\lambda(n-1,d-1)$ factor can be improved to $n^{2-2/(\lfloor d/2 \rfloor+1)} \log^{O(1)} n$).

THE GENERAL CASE

There are two ways to approach the general case where P is not simplicial. The first is again to apply perturbations in order to simulate nondegeneracy of S. This way all previously mentioned algorithms still apply, however they now compute a boundary description of P. The parameter M is now the size of the triangulation that happens to be constructed. Moreover, the perturbed computations slow down the running times by a polynomial factor in d.

The second way to deal with the general case is to generalize the algorithms so that they compute the lattice description of P. The main obstacle that must be overcome in the degenerate case is problem (b), the discovery of the ridges of a new facet F'. The obvious way to address this problem is to view the construction of F' as a recursive subproblem one dimension down. Some care must be taken however that in the many recursions small-dimensional faces are not reconstructed too often. This method was proposed by Chand and Kapur [CK70] and their algorithm was later improved and analyzed by Swart [Swa85] who showed a running time of $O(d^2nK_1 + d^3K_2 \log K_0)$, where K_i is the number of directed $(i{+}1)$-vertex paths in the Hasse diagram of the face lattice of P.

Rote [Rot92] generalized the algorithm of Avis and Fukuda to produce the lattice description using little storage space. Its running time is $O(dK_{d+1}n)$ and it appears to be not as relevant in practice as the original algorithm.

Finally, Seidel [Sei86b] generalized his shelling algorithm to produce the lattice description in time $O(n\lambda(n-1, d-1) + K_2(d^2 + \log K_0))$. Because of the recursive nature of straight-line shellings, this generalization avoids reconstruction of small-dimensional faces. Again the improvement via linear programming queries applies.

26.3.3 OTHER METHODS

THE BRUTE-FORCE APPROACH

Let S be a set of n points in $\mathbb{R}^d$ and let $P = \text{conv}S$. Assume w.l.o.g. that the origin is contained in the interior of P (otherwise apply a translation) and assume that S is irredundant in the sense that every point in S is a vertex of P (otherwise apply the results of Section 26.2).

A set $T \subset S$ spans a face of P iff there is a halfspace that has T on its boundary and $S \setminus T$ in its interior. Algebraically this can be tested by determining

$$y_T = \max\{y \in \mathbb{R} | \exists x \in \mathbb{R}^d : \forall p \in T : \langle x, p\rangle = 1 \text{ and } \forall p \in S \setminus T : \langle x, p\rangle + y \leq 1\},$$

which can be computed via linear programming, and checking that $y_T > 0$.

This characterization immediately yields a straightforward algorithm with running time $O(2^n\lambda(n, d))$ for generating all faces and also the lattice description of P: Simply test each subset of S whether it spans a face of P. This brute-force approach can be substantially improved by applying backtrack-search techniques ([Bal61],[FLM97]). Fukuda et al. [FLM97] even achieve a running time of $O(nK_0\lambda(n, d))$ this way, using just $O(dn)$ space. Unfortunately this backtrack-search approach does not seem to yield an efficient method to compute the double description of P.

THE PRIMAL-DUAL METHOD

Let S be a set of n points in $\mathbb{R}^d$, let $P = \text{conv}S$, and let $\mathcal{F}$ be the set of facets of P. Determining $\mathcal{F}$ from S is difficult if P is degenerate in the sense that it is not simplicial, i.e., its facets are not all simplices. However, in this case determining S from $\mathcal{F}$ may not be so difficult. The primal-dual method [BFM98] of Bremner, Fukuda, and Marzetta tries to exploit this possibility, despite the fact that $\mathcal{F}$ is unknown and S is the input.

The basic idea of their algorithm is as follows: For a facet $F \in \mathcal{F}$, let H_F be the halfspace that has F on its boundary and contains P, and for $\mathcal{G} \subset \mathcal{F}$ let $H_\mathcal{G} = \{H_G | G \in \mathcal{G}\}$. Assume some $\mathcal{G} \subset \mathcal{F}$ is known already. Enumerate the vertices of the polyhedron $P_\mathcal{G} = \bigcap H_\mathcal{G} \supset P$. If all the vertices found are points in S and if $P_\mathcal{G}$ is bounded, then it must be the case that $P_\mathcal{G} = P$ and $\mathcal{G} = \mathcal{F}$ and all facets of P have been found, and we are done. If this is not the case (and this can be determined after at most $n+1$ vertices of $P_\mathcal{G}$ have been enumerated), then it is easy to find a point $v \in P_\mathcal{G} \setminus P$ (either a vertex not in S or a point on an extreme ray of $P_\mathcal{G}$). But now clearly $\mathcal{G} \neq \mathcal{F}$. Moreover it is easy to find a facet $G \in \mathcal{F} \setminus \mathcal{G}$ (or rather the halfspace H_G) that separates v from P. This amounts to performing the initial facet finding step of the gift-wrapping algorithm and can be done (without linear programming!) in $O(d^2 n)$ time. Now add G to $\mathcal{G}$ and repeat.

The method suggests that the complexity of computing the facet description of a polytope P from its vertex description is related to the complexity of computing the vertex description from the facet description. It is difficult to make this theoretical statement precise without introducing assumptions about the intermediate polyhedra $P_\mathcal{G}$. However, on the practical side, the authors of [BFM98] present experimental evidence showing that the primal-dual method outperforms other algorithms in certain “degenerate” cases.

26.4 THE CASE OF SMALL DIMENSION

Convex hull computations in very small dimension are special. We have strong geometric intuitions about 2D and 3D space (and via Schlegel diagrams even about 4-polytopes). Moreover the situation is simpler in the case $d = 2, 3$ since our five polytope descriptions cannot differ much in terms of their sizes (they are all within a constant factor of each other), which means there is little need for keeping an exact distinction. Algorithmically, small dimensions are special in that besides the incremental and the graph traversal method, divide-and-conquer methods have also been brought to fruition.

THE 2-DIMENSIONAL CASE

The planar convex hull problem has drawn considerable attention and many different algorithmic paradigms have been tried (see textbooks such as [PS85] or [O’R98]). The graph traversal method was rediscovered and is known in the planar case as the ***Jarvis march*** with running time $O(nM)$, and the incremental method was rediscovered and is known in a rather different guise as the ***Graham scan*** with running time $O(n \log n)$ (as usual n and M are the sizes of the input and output, respectively). It was easy and natural to apply the divide-and-conquer paradigm to obtain further $O(n \log n)$ time algorithms. By giving this paradigm the extra twist of “marriage-before-conquest” it was possible even to obtain an $O(n \log M)$ algorithm, which was also shown to be worst-case optimal in the algebraic computation tree model of computation [KS86]. This algorithm required the use of 2D linear programming. Much later Chan, Snoeyink, and Yap [CSY97] showed how to avoid this and substantially simplified the algorithm in way that allowed its generalization to higher dimensions. Later, Chan [Cha96] showed quite surprisingly that by using simple data structures and the method of guessing the output size by repeated squaring, the Jarvis march algorithm can be sped up to also run in time $O(n \log M)$.

THE 3-DIMENSIONAL CASE

In 3 dimensions the output size M is $O(n)$ in the worst case. However, the straightforward implementations of the standard incremental and the graph traversal methods only yield algorithms with worst-case running time $O(n^2)$. In this context the use of the divide-and-conquer paradigm was decisive in obtaining $O(n \log n)$ running time, which was achieved by Preparata and Hong (see [PS85, Section 3.4.4]; for a more detailed account, [Ede87, Section 8.5]). This running time was later matched in the expected sense by the randomized incremental algorithm of Clarkson and Shor [CS89], who also gave another randomized algorithm with expected performance $O(n \log M)$.

The question whether this optimal output-size sensitive bound could also be achieved deterministically was open for a long time. Edelsbrunner and Shi [ES91] first generalized the "marriage-before-conquest" method of [KS86] but achieved only a running time of $O(n \log^2 M)$. Eventually Chazelle and Matoušek [CM95] succeeded in derandomizing the randomized algorithm of Clarkson and Shor and obtained, at least theoretically, this optimal $O(n \log M)$ time bound. Later, Chan [Cha96] showed that there is a relatively simple algorithm for achieving this bound, again by the method of speeding up the gift-wrapping method using data structures and guessing the output size by repeated squaring.

THE CASE $d = 4{,}5$

In this case the sizes of the combinatorial descriptions may be as large as $\Theta(n^2)$. All the methods and bounds mentioned in Section 26.3 apply. In addition there are methods for computing a boundary description based on sophisticated divide-and-conquer and some additional pruning mechanisms. Worst-case time bounds of $O((n + M) \log^{d-2} M)$ were achieved by Chan, Snoeyink, and Yap [CSY97] for $d = 4$, and by Amato and Ramos [AR96] for $d = 4, 5$. The latter paper also states that their bound applies to computing the lattice description in the case $d = 4$.

26.5 RELATED TOPICS

There has been some work on determining the intrinsic computational complexity of versions of the convex hull problem. The strongest results at this point are:

1. For fixed $d \geq 2$ the time necessary to determine whether exactly V of n points in $\mathbb{R}^d$ are extreme is $\Omega(n \log V)$ in the algebraic computation tree model [KS86]. This is asymptotically best possible for $d = 2$.

2. For fixed $d \geq 2$ the time necessary to determine whether the convex hull of n points in $\mathbb{R}^d$ has exactly M facets is $\Omega(n^{\lceil d/2 \rceil - 1} + n \log n)$ in a specialized but realistic model of computation [Eri99]. This is asymptotically best possible for odd $d > 1$.

The expected sizes of convex hulls of point sets drawn according to some statistical distribution are typically much smaller than the worst-case sizes. Constructing such convex hulls has been explicitly studied by several authors (see, e.g.,[DT81, Dwy91, BGJR91]). One should also mention in this context the randomized incremental algorithm [CS89]. With input set $S \subset \mathbb{R}^d$ its expected running time for constructing a boundary description is

$$O\left(\sum_{d+1<r\leq n} df_r(S)/r + \sum_{d+1\leq r<n} d^2 n f_r(S)/r^2 \right),$$

where $f_r(S)$ is the expected size of the boundary description of the convex hull of a random subset of S of size r. For many distributions f_r is sufficiently sublinear so that this randomized incremental algorithm has $O(n)$ expected running time.

The problem of maintaining convex hulls under insertions and deletions of points has been addressed also. In higher dimensions randomized incremental algorithms have been adapted by several authors to process updates [Mul94, Sch91, CMS93]. However, the analyses are all based on some probabilistic model of which updates actually occur. More satisfactory solutions have only been obtained in the planar case. Solutions with $O(\log n)$ update time were obtained for the insertions-only case (see [PS85, Section 3.3.6]) and also for the deletions-only case [HS92]. For the general dynamic case $O(\log^2 n)$ update times were achieved early on [OL81, Gow80], and only very recently they were improved to $O(\log n)$ in [Cha01, BJ02].

For some time there was hope that additional input information might help compute convex hulls. Although this is true in the planar case, where having points presorted or having them given along a nonintersecting polygonal line [Mel87] leads to linear-time algorithms, it has been shown [Sei85] that for dimension $d \geq 3$ such additional information does not help. Having a 3D set S presorted or even knowing a nonself-intersecting polyhedral surface whose vertex set is S does not in general make it easier to find the convex hull of S.

There have been some attempts to generalize the convex hull construction problem so that the input S does not consist of points but of more general objects such as algebraically described regions in the plane [BK91, NY98], balls in $\mathbb{R}^d$ [BCD$^+$96], ellipsoids in $\mathbb{R}^3$ [Wol02], or sets of polyhedra [FLL01].

Finally, parallel algorithms for the convex hull problem have been developed; see Chapter 46.

26.6 SOURCES AND RELATED MATERIALS

FURTHER READING

[Zie94]: A modern account of polytope theory.

[MR80]: A survey of vertex enumeration methods from the dual standpoint.

RELATED CHAPTERS

Chapter 15: Basic properties of convex polytopes
Chapter 17: Face numbers of polytopes and complexes
Chapter 27: Voronoi diagrams and Delaunay triangulations
Chapter 49: Linear programming

REFERENCES

[ABS97] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Comput. Geom.*, 7:265–301, 1997.

[AF92] 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.

[AR96] N.M. Amato and E.A. Ramos. On computing Voronoi diagrams by divide-prune-and-conquer. In *Proc. 12th Sympos. Comput. Geom.*, pages 166–175, ACM Press, 1996.

[BK91] C.L. Bajaj and M.-S. Kim. Convex hulls of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 1991.

[Bal61] M.L. Balinski. An algorithm for finding all vertices of convex polyhedral sets. *SIAM J. Appl. Math.*, 9:72–81, 1961.

[BCD$^+$96] J.-D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. *Comput. Geom.*, 6:123–130, 1996.

[BGJR91] K.H. Borgwardt, N. Gaffke, M. Jünger, and G. Reinelt. Computing the convex hull in the Euclidean plane in linear expected time. In P. Gritzmann and B. Sturmfels, editors, *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, vol. 4 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 91–107, AMS, Providence, 1991.

[Bre99] D. Bremner. Incremental convex hull algorithms are not output sensitive. *Discrete Comput. Geom.*, 21:57–68, 1999.

[BFM98] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. *Discrete Comput. Geom.*, 20:333–357, 1998.

[BJ02] G.S. Brodal and R. Jacob. Dynamic planar convex hull. *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, pages 617–626, 2002.

[Cha93] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[Cha96] T.M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[Cha96a] T.M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th Sympos. Comput. Geom.*, pages 284–290, ACM Press, 1996.

[Cha01] T.M. Chan. Dynamic planar convex hull operations in near-logarithmic time. *J. ACM*, 48:1–12, 2001.

[Chv83] V. Chvátal. *Linear Programming.* W.H. Freeman, New York, 1983.

[CK70] D.R. Chand and S.S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17:78–86, 1970.

[Cla94] K.L. Clarkson. More output-sensitive geometric algorithms. In *Proc. 35th IEEE Sympos. Found. Comput. Sci.*, pages 695–702, 1994.

[CM95] B. Chazelle and J. Matoušek. Derandomizing an output-sensitive convex hull algorithm in three dimensions. *Comput. Geom.*, 5:27–32, 1995.

[CMS93] K.L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom.*, 3:185–212, 1993.

[CS89] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[CSY97] T.M. Chan, J. Snoeyink, and C.K. Yap. Primal dividing and dual pruning: Output-sensitive construction of four-dimensional polytopes and three-dimensional Voronoi diagrams. *Discrete Comput. Geom.*, 18:433–454, 1997.

[DT81] L. Devroye and G.T. Toussaint. A note on linear expected time algorithms for finding convex hulls. *Computing*, 26:361–366, 1981.

[Dwy91] R.A. Dwyer. Rex A. Dwyer Convex hulls of samples from spherically symmetric distributions. *Discrete Appl. Math.*, 31:113–132, 1991.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, vol. 10 of *EATCS Monogr. Theoret. Comput. Sci.* Springer-Verlag, Heidelberg, 1987.

[ES91] H. Edelsbrunner and W. Shi. An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem. *SIAM J. Comput.*, 20:259–277, 1991.

[Eri99] J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28:1198–1214, 1999.

[FLL01] K. Fukuda, T.M. Liebling, and C. Lütolf. Extended convex hull. *Comput. Geom.*, 20:13–23, 2001.

[FLM97] K. Fukuda, T.M. Liebling, and F. Margot. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Comput. Geom.*, 8:1–12, 1997.

[Gow80] I.G. Gowda. *Dynamic problems in computational geometry.* M.Sc. thesis, Dept. Comput. Sci., Univ. British Columbia, Vancouver, 1980.

[HS92] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32:249–267, 1992.

[KS86] D.G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287–299, 1986.

[Mat93] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993.

[Mel87] A. Melkman. On-line construction of the convex hull of a simple polyline. *Inform. Process. Lett.*, 25:11–12, 1987.

[MR80] T.H. Mattheiss and D. Rubin. A survey and comparison of methods for finding all vertices of convex polyhedral sets. *Math. Oper. Res.*, 5:167–185, 1980.

[MRTT53] T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games II*, vol. 8 of *Ann. Math. Stud.*, pages 51–73. Princeton University Press, 1953.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, 1994.

[NY98] F. Nielsen and M. Yvinec. Output-sensitive convex hull algorithms of planar convex objects. *Comput. Geom.* 8:39-66, 1998.

[OL81] M.H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[O'R98] J. O'Rourke. *Computational Geometry in C*, second edition. Cambridge University Press, 1998.

[OSS95] T.A. Ottmann, S. Schuierer, and S. Soundaralakshmi. Enumerating extreme points in higher dimensions. In *Proc. 12th Sympos. Theoret. Aspects Comp. Sci.*, vol. 900 of LNCS, pages 562–570, Springer, Berlin, 1995.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[Ram00] E.A. Ramos. Linear optimization queries revisited. In *Proc. 16th Sympos. Comput. Geom.*, pages 176–181, ACM Press, 2000.

[Rot92] G. Rote. Degenerate convex hulls in high dimensions without extra storage. In *Proc. 8th Sympos. Comput. Geom.*, pages 26–32, ACM Press, 1992.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience, New York, 1986.

[Sch91] O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *Proc. 32nd IEEE Sympos. Found. Comput. Sci.*, pages 197–206, 1991.

[Sei81] R. Seidel. *A convex hull algorithm optimal for point sets in even dimensions.* M.Sc. thesis, Dept. Comput. Sci., Univ. British Columbia, Vancouver, 1981.

[Sei85] R. Seidel. A method for proving lower bounds for certain geometric problems. In G.T. Toussaint, editor, *Computational Geometry*, pages 319–334, North-Holland, Amsterdam, 1985.

[Sei86a] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Sympos. Theory Comput.*, pages 404–413, ACM press, 1986.

[Sei86b] R. Seidel. *Output-size sensitive algorithms for constructive problems in computational geometry.* Ph.D. thesis, Dept. Comput. Sci., Cornell Univ., Ithaca, 1986.

[Sei91] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[Sei96] R. Seidel. The meaning and nature of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1996.

[Swa85] G.F. Swart. Finding the convex hull facet by facet. *J. Algorithms*, 6:17–48, 1985.

[Wol02] N. Wolpert. *An exact and efficient approach for computing a cell in an arrangement of quadrics.* Ph.D. thesis, FR Informatik, Univ. des Saarlandes, Saarbrücken, 2002.

[Zie94] G.M. Ziegler. *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Math.* Springer-Verlag, New York, 1994.