Extended Convex Hull

Komei Fukuda* Thomas M. Liebling† Christine Lütolf†

July 4, 2000

Abstract

In this paper we address the problem of computing a minimal H-representation of the convex hull of the union of k H-polytopes in R^d. Our method applies the reverse search algorithm to a shelling ordering of the facets of the convex hull. Efficient wrapping is done by projecting the polytopes onto the two-dimensional space and solving a linear program. The resulting algorithm is polynomial in the sizes of input and output under the general position assumption.

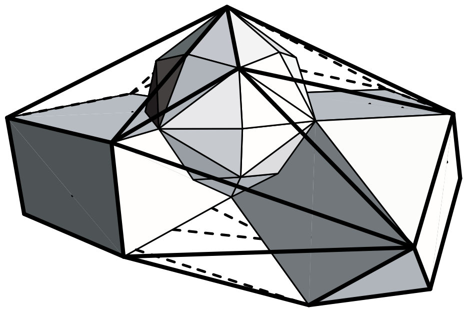

Figure 1: *Extended convex hull of three polytopes in* R^3.

1 Introduction

A *convex polytope* in R^d is the convex hull of a finite set of points. Equivalently, a convex polytope can also be defined as the intersection of a finite set of closed halfspaces of R^d. The former representation is called a *V-representation* and the latter an *H-representation.*

Computing all facets of a set of points and enumerating all vertices of a d-dimensional convex polytope are fundamental problems in mathematical programming and computational geometry.

In this paper we address the problem of computing a minimal H-representation of the convex hull of the union of k H-polytopes in R^d. We call it the *extended convex hull* (ECH) problem, see Figure 1.

A partial answer to this problem is given by Balas [5] for a special class of polyhedra arising in mixed-integer 0-1 programming and differing only in their right-hand sides. In the case of matroid polytopes, the convex hull of the union can be given explicitly [10]. Convexity recognition of the union of polyhedra is treated in [6] and could be used as preprocessing for the ECH problem.

Remark that the ECH problem can be solved by first enumerating the vertices of each polytope P_i, $i = 1, \ldots, k$ and then computing the convex hull of the set of points. But this does not lead to an efficient algorithm as the number of vertices can be exponential ($O(m^{\lfloor \frac{d}{2} \rfloor})$ in the worst case) in the number of facets m. If each polytope P_i, $i = 1, \ldots, k$, is reduced to a single point, representable as an H-polytope with $d+1$ inequalities, the ECH problem comes down to the classical convex hull computation for a point set. Polynomial algorithms for the latter problem are only known under assumptions of nondegeneracy, namely that the resulting convex hull be simplicial.

Our method applies the reverse search technique to a shelling ordering of the facets of the extended convex hull. The resulting algorithm is polynomial in the sizes of input and output under the general position assumption.

We will first present a recursive version of a reverse search algorithm for the ECH problem. Then we will show that under the general position assumption, recursion can be avoided. In the next section we will outline the general ideas of our algorithm before explaining the shelling of a polytope by a line used to order the facets. Then we will describe the reverse search technique applied to the enumeration of all facets of the extended convex hull. Section 3 describes an efficient implementation for generating facets of a facet and our wrapping method for moving from one facet to a neighboring one.

*Department of Mathematics, Swiss Federal Institute of Technology, Lausanne and Zurich, Switzerland

†Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, Switzerland

2 General Recursive Algorithm

In a d-dimensional polytope, a $(d-2)$-face is called *ridge*. Two facets are *adjacent* if they share the same ridge. Given a facet and a ridge, getting the adjacent facet can be formulated as a parametric problem of one degree of freedom, i.e., the normal vector of the new facet is obtained by tilting the given facet in the free direction determined by the ridge and the normal vector of the given facet. We call this the *wrapping procedure*, a term introduced by Chand and Kapur [9]. The solution to the parametric problem is found by solving a linear program over the union of polytopes projected onto the two-dimensional space, see Section 3.

The adjacency relation can be used to design a simple algorithm moving from one facet to all adjacent ones remembering the visited facets. However, for an enumeration of all facets of the extended convex hull to be both time and space efficient, a careful organization of the computation of the facets is needed.

Two important notions in the design of our algorithm is that of a *shelling* of a polytope by a line [8] and reverse search [4]. A shelling of a polytope is a certain linear order of its m facets. It is easy to compute and will allow us to define an optimization criterion for local search in the reverse search algorithm.

Reverse search is a general algorithmic scheme for the enumeration of objects with a specific property. Local search, which is a procedure for moving from one object to a neighboring better one with respect to some objective function, requires the notion of adjacency oracle which finds all neighbors of a given object. In the application of reverse search to the enumeration of the facets of the extended convex hull, the adjacency oracle returns a neighbor of a given facet.

In a simplicial d-polytope every set of $d-1$ vertices of a facet defines a ridge. In this case it is easy to identify all ridges and to treat them in lexicographic order of the d vertices of the facet. In the degenerate case finding all ridges amounts to an extended convex hull problem in one lower dimension. The enumeration of all facets of a facet is done in a recursive way until we can give the faces explicitly.

Further we need an initial facet. It is generated using the same approach as for finding an adjacent facet. Starting with a hyperplane supporting in one vertex, for example the one with largest x_1-component, we successively tilt it in one direction at the time. In this way we construct a sequence of d supporting hyperplanes, each touching one more vertex than the preceding one, the last one containing a facet of the extended convex hull.

2.1 Shelling Ordering

A shelling [8] of a polytope is a certain linear order induced on the facets of the polytope by a line. This property of numbering facets and vertices has been used in algorithms for vertex enumeration [3, 11, 13] and facet enumeration [14, 15]. To understand the shelling principle, we first need a few definitions. Let P be a polytope.

A *shelling* of P is a sequence $(F_1, \ldots, F_m)$ of all facets of P such that, for all i, $1 < i < m$, $(\cup_{j=1}^{i-1} F_j) \cap F_i$ is a topological $(d-2)$-ball. This implies that for each facet F_i, $1 < i \leq m$, there exists an earlier facet F_j, $1 \leq j < m$, which shares a ridge with F_i.

A line through the interior of a polytope P is said to be *admissible* for P if its intersection with each hyperplane supporting P is a distinct point. Any oriented admissible line defines a linear order on the facets of P which is a shelling.

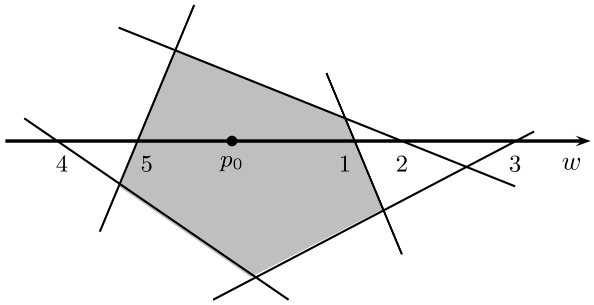

Figure 2: *Shelling of a polytope by a line.*

Let P be a polytope, L an oriented line, admissible for P, and p_0 a point on L and in the interior of P. Starting at p_0, imagine that we move along the line in the direction given by its orientation. We then cross hyperplanes at points $h(t_1), h(t_2), \ldots, h(t_r)$ until we do not cross any others and move towards infinity. We then reappear from the opposite infinity and move back towards p_0 crossing all remaining hyperplanes at $h(t_{r+1}), h(t_{r+2}), \ldots, h(t_s)$. The order of the intersections of the hyperplanes supporting P along its facets, uniquely given by the oriented line L, yields a shelling of P induced by L, see Figure 2. If w is the direction vector of L, then L can be parameterized, as to express the described motion, by

$$L = \left\{ h(t) = \frac{-w}{t} : t \in (-\infty, \infty) \setminus \{0\} \right\}$$

The facets of P are hence ordered in increasing order of their shelling parameters t_i.

Note that, any H-polytope P containing the origin in its interior can be written as $P = \{x \in R^d \mid a_i^T x \leq 1, a_i \in R^d, i = 1, \ldots, m\}$. A shelling of the facets of P then induces an order on the extreme points of the dual polytope $P^* = Conv(a_i : i = 1, \ldots, m)$ given by the shelling parameters $t_i = -a_i^T w$ where w is precisely the direction vector of the line described before.

2.2 Reverse Search for Facet Enumeration

The idea of reverse search by Avis and Fukuda originated in the problem of vertex enumeration for polyhedra or arrangements of hyperplanes [2, 3]. Reverse search algorithms, if successfully designed, have a time complexity proportional to the size of output times a polynomial in the size of input and a space complexity polynomial in the size of input.

Below we describe a reverse search algorithm solving the ECH problem. Its time complexity may not be polynomial in the sizes of input and output. However, as we will see in the next section, under the general position assumption it can be implemented efficiently. For a detailed description of reverse search and its applications we refer to [4].

The *graph of a polytope* is a graph which has a node for each facet and an arc for each pair of adjacent facets. Consider the graph of the extended convex hull. Given a facet F we need an oracle able to generate all facets adjacent to F. Recall that two facets are adjacent if they share the same ridge. The wrapping technique to generate a unique neighbor given a facet F and a ridge R is described in the next section. For the time being let us assume that procedure $Wrapping(F, R)$ does the job.

Except for special cases like for simplicial polytopes, the number of faces of a polytope P is not known a priori. We suppose that an efficient procedure for finding an upper bound on the number of neighboring facets of any facet F of P is available. This means that a call to $deg(F)$ gives us the maximum degree δ of node F of the enumeration tree.

The complete adjacency oracle is then given by $AdjacentFacet(F, j)$ generating, for $j \leq \deg(F)$, the jth facet R of F and returning facet F' adjacent to F on R. For invalid j, it returns null.

Further we have to define a local search algorithm allowing to move from any node towards the best one with respect to some objective function, in our case represented by the line shelling of a polytope. Indeed, if we construct an admissible line oriented from any interior point to an interior point of the initial facet of the extended convex hull, the shelling ordering hence defined leads us to the initial facet which has lowest shelling parameter and becomes the root of the enumeration tree. The shelling parameters play the role of the objective function values in the reverse search and as line shelling orders the facets in increasing order of their parameters, we define *LocalSearch*(F) as the procedure which returns the neighbor of F with smallest shelling parameter.

This way we can construct the tree of facets of the extended convex hull rooted in the initial facet and with the reverse search property that any node F' below node F such that there is a directed path from F' to F has no lower objective function value. Hence every facet is visited exactly once.

The following pseudo-code *ConvexHull*(P) outlines the reverse search procedure for enumerating the facets of the extended convex hull with the traversal done by depth first search. Recursive calls to this function allow us to get the convex hull of any given face F. The recursive calls are part of *AdjacentFacet*(F, j). An initial facet is supposed to be given.

```
procedure ConvexHull(P_1, ..., P_k)
  F_0 := initial facet of P = Conv(P_1 ∪ ... ∪ P_k);
  F := F_0; j := 0; (* j: neighbor counter *)
  repeat
    δ := deg(F);
    while j < δ do
      j := j + 1;
      next := AdjacentFacet(F, j);
      (* reverse traverse *)
      if LocalSearch(next) = F then
        F := next, j := 0
      endif
    endwhile;
    (* forward traverse *)
    if F ≠ F_0 then
      F' := F; F := LocalSearch(F);
      j := 0;
      repeat
        j := j + 1
      until AdjacentFacet(F, j) = F'
    endif
  until F = F_0 and j = δ
endprocedure.
```

Remark that calling *ConvexHull* recursively requires space to store the adjacent facets at each level in the tree and in each dimension. The storage can be avoided if recursion is implemented on the determination of the adjacent facet instead of the whole convex hull as it is done by Rote [14].

3 Efficient Implementation

Let us consider an ECH problem with k H-polytopes $P_1, \ldots, P_k$ in general position. Below, we show that under the general position assumption, the ridges of the extended convex hull can be generated efficiently, allowing us to remove recursion in the above *AdjacentFacet*(F,j) procedure. We then present an efficient wrapping method based on projection of the polytopes $P_1, \ldots, P_k$ onto the two-dimensional space. The complexity of the resulting reverse search algorithm for the extended convex hull computation is polynomial in the sizes of input and output, as will be discussed in Section 4.

3.1 Ridge Generation

For each facet F of the extended convex hull define F_i as $F \cap P_i$ and $d_i = d_i(F)$ as $\dim(F_i)$, $i = 1, \ldots, k$. We say that k polytopes P_i, $i = 1, \ldots, k$, in R^d are in *general position* if $\sum_{i=1}^{k}(d_i + 1) = d$ for each facet F of $Conv(P_1 \cup \ldots \cup P_k)$.

Let $J \subseteq \{1, \ldots, k\}$ be the index set defined as $J = \{j : F_j \neq \emptyset\}$. A facet F of the extended convex hull intersects s $(\leq d)$ polytopes P_j, $j \in J$. A facet R of F is defined by $Conv\left((\cup_{j \in J \setminus \{t\}} F_j) \cup G_t\right)$ where G_t is a facet of F_t, $t \in J$. G_t is obtained by redundancy calculation of the inequalities bound to $F_t \subseteq P_t$. This can be done efficiently by solving as many linear programs as the number of inequalities describing F_t.

To express R by a set of generators, we suppose that there is a method producing generators for every face F_j. Indeed, $d_j + 1$ affinely independent points of F_j can be obtained by $d_j + 1$ applications of the raindrop method [7]. Hence a facet R of F is given by the generators of all F_j, $j \in J \setminus \{t\}$ and G_t.

Note that if we are given a facet F and a ridge R, each with a corresponding set of generators, wrapping of F on R provides a point on the new facet F' which can be used as a generator for F'. Together with the generators of R, F' is then well defined. However, the generators of a face F_j do not necessarily define its facets G_j and the raindrop method may need to be applied to find a set of generators for G_j.

Note also that wrapping of a facet F on R to yield F' either increases by one the dimension d_j of one of the $F_j \subseteq F$ or creates an intersection F'_t of dimension $d_t = 0$ with a new polytope P_t, $t \notin J$. In both cases, the intersections $F'_j = F' \cap P_j$ are easily obtained from the intersections of F with the polytopes thanks to the definition of the general position where $\sum_{j \in J}(d_j + 1) = d$ and the fact that F and F' share the ridge R.

3.2 Wrapping

Here we describe how to implement *Wrapping*(F, R), the procedure for finding the unique adjacent facet F', given facet F and ridge R of the extended convex hull.

Let $P_i = \{x^i \in R^d \mid A^i x^i \leq b^i\}$, $i = 1, \ldots, k$, be k H-polytopes for which the extended convex hull is to be computed and let $P = Conv(P_1 \cup \ldots \cup P_k)$. Let F be the facet defined by $v_1, \ldots, v_{d-1}, v_d \in Conv(P_1 \cup \ldots \cup P_k)$ and R the ridge defined by the first $d - 1$ vertices.

The parametric problem of finding facet F' adjacent to F on R amounts to the maximization of the angle between the outer normal vectors of F and F' such that the hyperplane containing F' is supporting P. To solve this problem, the idea is to project P onto the two-dimensional space (y_1, y_2) where the angle maximization comes down to the maximization of the slope $\frac{y_2}{y_1}$ of the projected supporting hyperplane. The coordinate axes are chosen such that the ridge R is projected onto the origin, vertex $v_d \in F \setminus R$ onto the negative y_2-axis, and the projection $\hat{P}$ of P lies on the positive side of the y_1-axis.

Let us first set up the new coordinate system and the associated projection. Let n_1 be the normalized outer normal of F and $r_1, \ldots, r_{d-2}$ a basis of the linear space spanned by $v_{d-1} - v_1, \ldots, v_2 - v_1$. We have $n_1 \perp r_j$, $j = 1, \ldots, d - 2$.

Let n_2 be the unit vector such that $n_2 \perp n_1$, $n_2 \perp r_j$, $j = 1, \ldots, d - 2$, and $n_2^T(v_d - v_1) < 0$. Clearly $r_1, \ldots, r_{d-2}, n_1, n_2$ form a basis of R^d, $B := [r_1 \ldots r_{d-2} n_1 n_2] : d \times d$.

The projection is given by

$$\begin{array}{rccc} proj: & R^d & \rightarrow & R^2 \\ & x & \mapsto & \hat{x} = Tx \end{array}$$

where the matrix T is

$$T := \left[\begin{array}{ccccc} 0 & \ldots & 0 & -1 & 0 \\ 0 & \ldots & 0 & 0 & 1 \end{array}\right] B^{-1}$$

In addition to the projection where the ridge R is projected to the origin, we shift the polytopes P_i, $i = 1, \ldots, k$ by $v_1 \in R$.

$$A^i(x^i - v_1) \leq b^i \quad \text{ i.e. } \quad A^i x^i \leq \bar{b}^i$$

where $\bar{b}^i = b^i + A^i v_1$.

Any point x in P can be expressed as a convex combination of points of the individual polytopes $P_1, \ldots, P_k$, i.e.,

$$x = \sum_{i=1}^{k} \lambda_i x^i \quad \text{with} \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0$$

where $x^i \in P_i$, $i = 1, \ldots, k$. Hence we have x^i and λ_i as variables. To get rid of the nonlinearity caused by the multiplication of the two variables, we perform the following substitution:

$$\bar{x}^i := \lambda_i x^i$$

and each individual polytope P_i gets scaled by λ_i, i.e.,

$$\bar{P}_i = \{\bar{x}^i \mid A^i \bar{x}^i - \bar{b}^i \lambda_i \leq 0, \lambda_i \geq 0\}$$

The shifted variable of P is then $x = \sum_{i=1}^{k} \bar{x}^i$.

Once P is projected and shifted, the objective function to maximize is the slope $\frac{y_2}{y_1}$ of the projected supporting hyperplane containing $\hat{R}$. To avoid the fraction, we homogenize $\hat{P}$ such as to limit the feasible region to a subset of hyperplane $y_1 = 1$. In fact, the sum of the previously introduced variables λ_i plays the role of the homogenization variable of $\hat{P}$, see Figure 3.

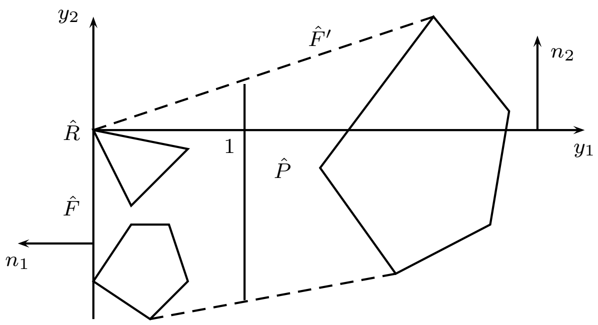

Figure 3: *The projection of P onto the (y_1, y_2)-space.*

Finally, the problem can be formulated as a linear program.

$$LP: \quad \begin{array}{llll} \max y_2 & & & \\ \text{s.t.} \; A^i \bar{x}^i & \leq & \bar{b}^i \lambda_i & i = 1, \ldots, k \\ \lambda_i & \geq & 0 & i = 1, \ldots, k \\ x & = & \sum_{i=1}^{k} \bar{x}^i & \\ y & = & Tx & \\ y_1 & = & 1 & \end{array}$$

If this LP has an optimal solution $(1, y_2^*, x^*, \lambda^*)$, it determines a point on the new facet. Indeed, $p^* = \frac{x^*}{\sum_{i=1}^{k} \lambda_i^*}$ and there exists a unique hyperplane containing $\hat{R}$ and p^*. Note that the solution to LP is not necessarily a vertex of the new facet but any point lying on it.

Let us show that $x = 0$ can never be solution to the LP and that consequently $\sum_{i=1}^{k} \lambda_i^* > 0$. Suppose, by contradiction, that $\lambda_i = 0$, $\forall i$. This implies that $x = 0$ and therefore $y = Tx = 0$ which contradicts the fact that $y_1 = 1$.

Lemma 3.1 *The time complexity of our wrapping algorithm is $O(LP(m + k, kd))$ where $LP(\tilde{m}, \tilde{n})$ denotes the time to solve a linear program with $O(\tilde{m})$ constraints and $O(\tilde{n})$ variables and m is the sum of all constraints describing P_i, $i = 1, \ldots, k$.*

Proof. In our wrapping procedure, besides the resolution of a linear program, the projection involves an inversion of a $d \times d$ matrix which can be done in $O(d^3)$. All other operations are $O(d)$ scalar products of vectors in R^d and a matrix multiplication in $O(d^2)$. In the first group of constraints in the LP there are $m = m_1 + \ldots + m_k$ constraints where m_i is the number of constraints of P_i, $i = 1, \ldots, k$. The total number of constraints of the LP is $m+k+d+2+1$, i.e., $O(m+k)$. The total number of variables is $kd + k + d + 2$, i.e., $O(kd)$. The main effort is therefore spent in the resolution of the LP dominating the complexity of solving the linear system to inverse the $d \times d$ matrix. The overall complexity is therefore $O(LP(m + k, kd))$. ∎

Note that this way of wrapping is efficient because no explicit inequality description of the projected polytopes is needed. Amenta and Ziegler [1] showed that a two-dimensional projection of a d-polytope with f_{d-1} facets may have as many as $\Theta\left(f_{d-1}^{\lfloor d/2 \rfloor}\right)$ vertices for fixed d.

The extended convex hull problem can be solved even if the k polytopes are not given explicitly but by an oracle deciding for a polyhedron P, a certain point p, and a hyperplane H whether H is separating p from P. In this case, efficient wrapping can be done using binary search.

4 Complexity

Theorem 4.1 *Under the general position assumption, there is an implementation of the extended convex hull* *algorithm applied to k H-polytopes in R^d which runs in time*

$$O\left(f_{d-1}d\bar{m}\mathrm{LP}(\bar{m},d)+ f_{d-2}\left(\bar{m}d^3+\mathrm{LP}(m+k,kd)+\log(f_{d-1})\right)\right)$$

where f_{d-1} is the number of facets and f_{d-2} the number of ridges of the extended convex hull, $m = m_1+\ldots+m_k$, and $\bar{m} = max\{m_1,\ldots,m_k\}$ with m_i being the number of constraints describing P_i, $i = 1,\ldots,k$.

Proof. Let $P_i = \{x \in R^d \mid A^i x \leq b^i\}$ with A^i an $m_i \times d$ matrix and $b^i \in R^{m_i}$, $i = 1,\ldots,k$ be the k polytopes for which the extended convex hull is to be computed.

A node of the enumeration tree of the reverse search algorithm represents a facet F of the extended convex hull. Let us denote the s nonempty intersections of F with the polytopes P_j by $F_j = F \cap P_j \neq \emptyset$, $j \in J \subset \{1,\ldots,k\}$ with $|J| = s$. A facet of F is given by $(\cup_{j\in J\setminus\{t\}} F_j) \cup G_t$ where G_t is a facet of F_t, $t \in J$. The facets G_j of each fixed F_j are obtained by redundancy calculation which means that m_j linear programs with m_j constraints and d variables need to be solved. As there are at most d nonempty intersections of F with the polytopes and m_j can be bounded by $\bar{m}$, $\forall j$, the amount of work to enumerate the facets of all faces F_j is $\sum_{j=1}^{s} m_j LP(m_j,d) \leq d\bar{m}LP(\bar{m},d)$.

In the worst case, we then need to compute generators for each facet G_j of F_j. Producing a set of generators, i.e., d_j+1 affinely independent points by the raindrop method takes $O(m_j d^3)$ time. Generator sets are needed for as many faces G_j, $j \in J$, as there are facets of F. Replacing again m_j with $\bar{m}$, $\forall j$, the work done for producing all generators is $O(f_{d-2}^{(F)}\bar{m}d^3)$ where $f_{d-2}^{(F)}$ denotes the number of ridges of the extended convex hull corresponding to F. For all other faces composing the ridge, generators are known and the ridges corresponding to F are now well defined.

To get the neighboring facets of F, wrapping is done on each of the $f_{d-2}^{(F)}$ ridges. Hence $f_{d-2}^{(F)}$ applications of our wrapping method which has a time complexity of $O(LP(m+k,dk))$, see Lemma 3.1, yield all facets adjacent to F in $O(f_{d-2}^{(F)}LP(m+k,dk))$. Finally the shelling parameter of a given facet is determined in $O(d)$.

So far we have evaluated all work to be done at each node of the enumeration tree. In order to avoid recalculation of adjacent facets at different moments in the reverse search algorithm, we propose to put all information concerning F in a list whenever a new F is discovered. The facets of the extended convex hull are uniquely defined by their shelling parameter and this parameter can be used to sort the list. This list should be maintained in a balanced (e.g. AVL) tree [12] so that the look-up and the insertion can be done in $O(\log(f_{d-1}))$ time. The list is accessed twice for each edge of the enumeration tree, resulting a complexity of $O\left(f_{d-2}\log(f_{d-1})\right)$.

The forward and reverse traversals then come down to a constant time operation of comparing shelling parameters of facets from the list. Finally, the total time complexity for the extended convex hull algorithm is

$$O\left(f_{d-1}d\bar{m}LP(\bar{m},d)+ f_{d-2}\left(\bar{m}d^3+LP(m+k,kd)+\log(f_{d-1})\right)\right)$$

■

Note that the look-up time for the list containing the information about each facet of the output is probably dominated by the complexity of the wrapping procedure. However, we could not find a bound on the number of output facets proving this statement. The algorithm as presented in the proof of the theorem also needs some space to store information about the output facets.

Apart from the complexity of the wrapping procedure, a lot of time is spent in calculating the ridges of the extended convex hull. In the special case where the d-polytopes P_i, $i = 1,\ldots,k$ are simplicial, ridge generation is easy. Under the general position assumption, all facets are then $(d-1)$-simplices and every subset of $d-1$ vertices out of the d vertices defining the facet yields a ridge. Any facet of the extended convex hull has therefore exactly d facets and hence d adjacent facets. The jth facet of F is then given by $R = \{v_1,\ldots,v_d\} \setminus v_j$ if F is defined by $v_1,\ldots,v_d$. The time complexity of our algorithm in the case of simplicial d-polytopes, implemented such that no additional storage is needed, is given in the following corollary.

Corollary 4.2 *There is an implementation of the extended convex hull algorithm applied to k simplicial d-dimensional H-polytopes in general position which runs in time*

$$O\left(f_{d-1}d^2LP(m+k,kd)\right)$$

and with space complexity $O(md)$.

Proof. Let us analyze the two steps making up the time complexity of the reverse search algorithm. They are the forward and the reverse traversals executed exactly once for each facet of the output.

In the forward step, the ridge generation is now done in constant time as all facets are simplices. It remains, for each facet F, d calls to the wrapping procedure which has a complexity of $O(LP(m+k, kd))$, see Lemma 3.1, and the determination of the shelling parameter in $O(d)$ to find the best neighbor. Forward traversal is therefore done in $O(dLP(m+k, kd))$ for each facet F. The reversability check amounts to performing d wrapping steps per neighbor of F and takes therefore $O(dLP(m+k, kd))$ time for each facet of F.

The total work can hence be bounded by $O(f_{d-2}dLP(m+k, kd) + f_{d-1}dLP(m+k, kd))$. As in this particular case, $f_{d-2} = df_{d-1}$ we get the result.

The space complexity is independent of the cardinality of the output. The only space needed is $O(md)$, i.e., the space to store the input.

■

Acknowledgment: We would like to thank Günter Rote for his suggestions leading to the wrapping method.

References

[1] N. Amenta and G. M. Ziegler. Deformed products and maximal shadows of polytopes. *Advances in Discrete and Computational Geometry*, 223:57–90, 1999.

[2] D. Avis and K. Fukuda. A basis enumeration algorithm for linear systems with geometric applications. *Appl. Math. Lett.*, 5:39–42, 1991.

[3] 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.

[4] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65:21–46, 1996.

[5] E. Balas. On the convex hull of the union of certain polyhedra. *Oper. Res. Lett.*, 7(6):279–283, 1988.

[6] A. Bemporad, K. Fukuda, and F. D. Torrisi. On convexity recognition of the union of polyhedra. Preprint, 2000.

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

[8] H. Bruggesser and P. Mani. Shellable decompositions of cells and spheres. *Math. Scand.*, 29:197–205, 1971.

[9] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17(1):78–86, 1970.

[10] M. Conforti and M. Laurent. On the facial structure of independence system polyhedra. *Math. Oper. Res.*, 13(4):543–555, 1988.

[11] K. Fukuda. An efficient pivot algorithm for finding all edges and all vertices of convex polytopes: 3-dimensional case. Unpublished note, 1984.

[12] D. E. Knuth. *The Art of Computer Programming Vol. 3: Sorting and Searching.* Addison Wesley, 1973.

[13] K. G. Murty. An algorithm for ranking all the assignments in order of increasing costs. *Oper. Res.*, 16:682–687, 1968.

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

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