Combinatorial face enumeration in convex polytopes

Komei Fukuda\textsuperscript{a,*},1, Vera Rosta\textsuperscript{b}

\textsuperscript{a}Graduate School of Systems Management, University of Tsukuba, 3-29-1 Otsuka, Bunkyo-ku, Tokyo 112, Japan
\textsuperscript{b}Faculty of Mathematics, Temple University Japan, 2-2 Minami-Osawa, Hachioji-shi, Tokyo, Japan 192-03

Communicated by Joseph Zaks; submitted 7 April 1993; accepted 19 February 1994

Abstract

Let \( P \) be a \( d \)-dimensional convex polytope with \( n \) facets \( F_1, F_2, \ldots, F_n \). The (combinatorial) representation of a face \( F \) of \( P \) is the set of facet indices \( j \) such that \( F \subseteq F_j \). Given the representations of all vertices of \( P \), the combinatorial face enumeration problem is to enumerate all faces in terms of their representations.

In this paper we propose two algorithms for the combinatorial face enumeration problem. The first algorithm enumerates all faces in time \( O(f^2d \min\{m,n\}) \), where \( f \) and \( m \) denotes the number of faces and vertices, respectively. For the case of simple polytopes, the second algorithm solves the problem in \( O(fd) \) time, provided that a good orientation of the graph of the polytope is also given as input.

Key words: Convex polytope; Enumeration; Face lattice

1. Introduction

Let \( P \) be a \( d \)-dimensional convex polytope with \( n \) facets \( F_1, F_2, \ldots, F_n \). We call an \( i \)-dimensional face an \( i \)-face, and the set of \( i \)-faces of \( P \) is denoted by \( \mathcal{F}_i \). The combinatorial representation of a face \( F \) of \( P \) is the set of facet indices \( j \) such that \( F \) is a subface of \( F_j \).

In this paper we represent a face with its combinatorial representation. A small caution must be taken, since this means that for any two faces \( F, F' \) of \( P \), a face \( F \) is...
a subface of $F'$ if and only if $F' \subseteq F$, reversing the geometrical containment relation between the faces. Given all vertices $P_0$ of $P$, the **combinatorial face enumeration problem** is to enumerate all faces of $P$ in terms of their representations without duplications.

There is a natural dual problem: given the set of facets of a $d$-dimensional convex polytope with vertices $v_1, v_2, \ldots, v_m$, each facet represented by the set of indices of its vertices, list all faces in terms of vertex representation. Needless to say, this is the same problem by the duality of convex polytopes, see [6]. The dual problem might look more natural since the vertex representation of faces preserves the geometrical containment relation among faces. However, in order to make use of the graph structure of a convex polytope in Section 3, it appears to be more convenient in this paper to deal with the primal problem. This is the reason why we use the facet representation in the sequel.

It is easy to see that the combinatorial face enumeration problem is a special case of constructing the *intersection family* of any given family of subsets of a finite set $E = \{1, 2, \ldots, n\}$. Thus, a trivial approach is to consider all possible combinations of vertices and take their intersections. Unfortunately, this method is not practical because in the worst case it takes a time exponential in terms of $d$, $n$, and $f$, where $f$ is the number of faces of $P$. For instance, in the $d$-hypercube $P$, the number $|\mathcal{P}_0|$ of vertices is $2^d$ and the number of subsets of $\mathcal{P}_0$ is $2^{2d}$ which is double exponential in $d$, while the number of faces is only $3^d$.

In this paper, by using some basic properties of convex polytopes, we develop an algorithm whose time complexity is polynomially bounded by $d$, $n$, and $f$. The main idea of the enumeration is to construct pairwise intersection of vertices as candidates for edges of the polytope and similarly in the general step we take pairwise intersection of $k$-faces as candidates for $(k + 1)$-faces. Subsequently, we update the temporary dimension of each candidate until we obtain the real dimension of a face. Using binary tree implementation our algorithm determines all the faces together with their dimension in time $O(f^2dn)$. Furthermore, by selecting the primal or the dual problem, whichever is smaller, one can reduce the complexity of the face enumeration to $O(f^2d \min \{m, n\})$.

It is often expected that the nondegeneracy assumption that the polytope $P$ is simple makes the problem easier to solve. In fact one can easily see that the enumeration of faces is quite simple because a subset of $\{1, 2, \ldots, n\}$ is a $k$-face of $P$ if and only if it is a $d - k$-subset of some vertex. Yet, listing all faces without duplication stays nontrivial if a more efficient algorithm is required than the first algorithm. We found that an additional information about the graph $G(P)$ of the polytope can in fact simplify the problem considerably. Following Kalai [5], an orientation of $G(P)$ is **good** if for every non-empty face $F$ of $P$, $G(F)$ has exactly one sink (vertex with out-degree zero). Good orientations can be obtained, for example, by orienting the edges according to the value of a linear functional on $\mathbb{R}^d$ that is $1-1$ on the vertices of $P$. Given the list of vertices together with a good orientation, we show that the **combinatorial enumeration problem** can be solved in time $O(fd)$ when $P$ is simple. It
is noteworthy that this algorithm is extremely simple and has a small space complexity of \(O(d)\) if it reads the input data from an external storage device.

2. Face enumeration algorithm

In this section we propose an algorithm enumerating all faces of a polytope from the set of vertices. The following basic properties of convex polytopes \([4, 6]\) play an important role. Recall that each face is identified with its combinatorial representation and thus the geometrical containment relation among faces are reversed in our representation.

Lemma 2.1. For every \(d\)-polytope the following statements hold:

(a) The intersection of two distinct faces of dimension at least \(k\) is a face of dimension at least \(k + 1\).

(b) Each \((k + 1)\)-face is the intersection of two distinct \(k\)-faces.

From this lemma we immediately obtain a naive algorithm to enumerate all the faces. First take all pairwise intersections of vertices (in "level zero"), and select only maximal ones to create the level 1. In the general step, take all pairwise intersections of faces in the \(k\)-th level, and select only the maximal subsets from them to get \(\mathcal{P}_{k+1}\). Selecting the maximal subsets at each level would require \(O(f^4 n)\) time since there are \(O(f^2)\) candidates with possible duplication of the same face. Doing this for \(d\) steps will result in the time complexity of \(O(f^4 dn)\).

Below we propose a more efficient algorithm where the maximality of a candidate for \(k\)-face is checked while we are creating the candidates for \((k + 1)\)-faces. Employing a binary-tree data structure, we can further reduce the time complexity. We shall discuss the implementation of the binary tree later.

**General Face Enumeration Algorithm**

Input: the set \(\mathcal{P}_0\) of vertices of a polytope \(P\).

Output: the set \(\mathcal{P}\) of faces of \(P\) together with their dimensions.

**procedure** FaceEnumeration \((\mathcal{P}_0; \) vertices)\;

begin
 create a binary tree \(T\) with set of leaves \(\mathcal{P}_0\);
 \(k := 0; f_0 := |\mathcal{P}_0|; \mathcal{P}_{(0)} := \mathcal{P}_0;\)
 while \(f_k \geq 2\) do
 \(f_{k+1} := 0; \mathcal{P}_{(k+1)} := \emptyset;\)
 for each pair of elements \(F\) and \(F'\) in \(\mathcal{P}_{(k)}\) do
 \(F'' := F \cap F'\);
 if \(F''\) is not in \(T\) then
 if \(F'' = F\) or \(F'' = F'\) then
 delete \(F''\) from \(\mathcal{P}_{(k)}\);
 \(f_k := f_k - 1\)
end
Theorem 2.2. At the end of k-th iteration, the face enumeration algorithm Face Enumeration1 outputs exactly the set of k-faces of the polytope P, for k = 0, 1, . . . , d − 1.

Proof. We prove the slightly stronger statement than the theorem by induction:

(*) at the end of the k-th iteration \( \mathcal{P}_k = \mathcal{P}_k^{(k)} \), and \( \mathcal{P}_{k+1} \) contains all \((k + 1)\)-faces and contains no faces of dimension \( k \) or lower, for each \( k = 0, 1, . . . , d - 1 \).

For \( k = 0 \) we initially set in the algorithm \( \mathcal{P}_{(0)} = \mathcal{P}_0 \). The equality will hold at the end of 0-th iteration as well since there are no comparable sets in \( \mathcal{P}_0 \), and no member of \( \mathcal{P}_{(0)} \) will be eliminated during the iteration. On the other hand, since at the end of the 0-th iteration \( \mathcal{P}_{(1)} \) is exactly the set of intersections of distinct pairs of vertices, by Lemma 2.1, the statement (*) is valid for \( k = 0 \).

Assume (*) is valid at the end of the \( j \)-th iteration, for a fixed \( j \geq 0 \). Let \( k = j + 1 \). During the \( k \)-th iteration, the intersections of distinct two faces in \( \mathcal{P}_k \) are generated sequentially (in some arbitrary order). By the induction hypothesis, \( \mathcal{P}_k \) contains all \( k \)-faces and contains no faces of dimension \( k - 1 \) or lower. These imply that a face in \( \mathcal{P}_k \) will be removed from the set during the \( k \)-th iteration if and only if it is a proper subset of some \( k \)-face (i.e., it is a face of dimension \( (k + 1) \) or higher). Therefore, at the end of the iteration \( \mathcal{P}_k = \mathcal{P}_k^{(k)} \). Furthermore, since all the intersections of two distinct \( k \)-faces will be added to \( \mathcal{P}_{k+1} \) by the end of the iteration, by Lemma 2.1, it must eventually contain all \((k + 1)\)-faces. It is clear that no faces of dimension \( k \) or lower will be appended to \( \mathcal{P}_{k+1} \) since we always take an intersection of two distinct faces of dimension \( k \) or higher by the description of the algorithm and the induction hypothesis. This completes the proof of (*) and hence the theorem. \( \square \)

To evaluate the complexity of this algorithm, we must describe how to implement the set \( \mathcal{P}_k \) and the tree \( T \).

Firstly, we simply use the linked list structure for storing \( \mathcal{P}_k \). This enables us to append an element or to delete an element located by a pointer variable in constant time.

Now we consider \( T \). The algorithm uses the membership operation for \( T \) whose time complexity is critical for the time complexity of the algorithm. We employ a binary tree of depth \( n \) where each node \( N \) of depth \( k \) is represented by 0-1 vector \((s_1, s_2, \ldots, s_k)\). The root is the empty vector (\( ) \). If a node \( N \) is the left (right) side of
a node $N'$ with address $(s_1, s_2, \ldots, s_k)$, then the address of $N$ is $(s_1, s_2, \ldots, s_k, 0)$ $((s_1, s_2, \ldots, s_k, 1)$, respectively). A face $F$ is then stored in the node of address $(s_1, s_2, \ldots, s_k)$ where $s_k = 0$ if and only if $k \in F$ for $k = 1, 2, \ldots, n$. It is clear that the membership query $F \in T$? can be answered in $O(n)$ time. Also adding a new face $F$ in $T$ can be done in $O(n)$ time as well. Note that here the 0–1 addresses of nodes are used merely for the explanation of the tree structure. For the actual implementation, the tree structure may be constructed and manipulated by pointer variables.

Now we can evaluate the time complexity of the algorithm as $O(f^2dn)$. This follows immediately from the following remarks:

(a) the number of intersections to take at the k-th iteration is at most $O(f^2)$;

(b) finding the intersection of two faces in $P_{(k)}$ is $O(n)$; then to find $F$ on the binary tree $T$ takes $O(n)$ time;

(c) there are exactly $d$ iterations.

The space complexity is $O(fn)$, since both the binary tree and the linked list requires at most $O(fn)$ space.

When the number $m$ of vertices is smaller than the number $n$ of facets, one can simply apply the same algorithm to the dual polytope having $m$ vertices and $n$ facets for the same purpose. This means that we have a better time complexity $O(f^2d \min \{m, n\})$ of the face enumeration.

The Hasse diagram of a polytope $P$ can be also computed with the same complexity. $F \in P_{k+1}$ and $F' \in P_k$ are linked in the Hasse diagram if and only if the representation of $F$ is subset of the representation of $F'$. This means that $n_{f_{k+1}} f_k$ comparisons are enough to create these links for $0 \leq k < d$. Therefore the total time complexity is no more than $O(f^2dn)$.

3. Face enumeration in simple polytopes

For simple polytopes the following lemma holds.

**Lemma 3.1.** Let $N = \{1, 2, \ldots, n\}$ be the set of facet indices. A subset of $N$ is a $k$-face if and only if it is a $(d - k)$ subset of some vertex.

As a special case, $d - 1$ subsets of vertices are 1-faces of the polytope, and thus edges of the graph of the polytope.

There is a trivial algorithm enumerating all faces, namely generating all possible subsets of vertices. But this procedure would enumerate each face as many times as the number of vertices the face contains. Using the binary tree to avoid duplication would lead to an algorithm with $O(f_0d^2n)$ time complexity.

In this section, we show that the computation becomes much simpler and faster if we have some additional information on the graph structure of the polytope, namely a good orientation.

Following Kalai [5], an orientation of the graph of the polytope is said to be good if its subgraph induced by each face of $P$ has exactly one sink. Every polytope has good
orientation because we can always direct the edges according to any linear functional which is $1 - 1_{\mathcal{S}_0}$. With a given good orientation there is a canonical way to represent each face as a subset of vertices.

Let $O$ be a good orientation on the graph of the polytope $P$. For each vertex $v$ in $\mathcal{S}_0$ we denote by

\[ v^+ = \{ j \in v \mid \text{the edge } v - j \text{ is directed toward } v \text{ in } O \}, \]
\[ v^- = \{ j \in v \mid \text{the edge } v - j \text{ is directed away from } v \text{ in } O \}, \]

The following lemma contains the key ideas to design an efficient algorithm.

**Lemma 3.2.** Let $P$ be a simple polytope with a good orientation $O$.

(a) If $F$ is a face of $P$ then $F = v^- \cup A(F)$, where $v$ is the unique sink of $F$ with respect to $O$ and $A(F) = \{ j \in v^+ \mid v - j \text{ is not an edge of } F \}$.

(b) If $A$ is a subset of $v^+$ then $F = v^- \cup A$ is a face of $P$ whose unique sink with respect to $O$ is $v$, and $A = A(F)$.

**Proof.** (a) Let $v$ be the unique sink of $F$. Then $F$ is uniquely generated by a set of edges directed toward $v$. Moreover for each $j \in v$, $j$ is a member of $F$ if and only if $v - j$ is not an edge of $F$. That is $F = \{ j \in v \mid v - j \text{ is not an edge of } F \}$. Since $v$ is the sink of $F$ we have $F = v^- \cup \{ j \in v^+ \mid v - j \text{ is not an edge of } F \} = v^- \cup A(F)$.

(b) Let $A \subseteq v^+$ and $F = v^- \cup A$. By Lemma 3.1, $F$ is a face of $P$. For $j \in v^-$, $v - j$ cannot be an edge of $F$, so $v$ is the sink of $F$. By a) $F = v^- \cup A(F)$ if $v$ is the sink of $F$. Therefore $A = A(F)$. \qed

According to this lemma if we have a vertex enumeration together with a good orientation $O$ on the edges of the graph of the polytope, then we can represent each vertex $v \in \mathcal{S}_0$ as a set of signed set $(v^+, v^-)$ and by enumerating all possible sets $v^- \cup A$ for each $v \in \mathcal{S}_0$ and each $A \subseteq v^+$ we enumerate each face of $P$ exactly once.

**Face Enumeration Algorithm for Simple Polytopes**

**Input:** the set $\mathcal{S}_0$ of vertices of a polytope $P$ with a good orientation represented by the set of signed sets $(v^+, v^-)$ for $v \in \mathcal{S}_0$.

**Output:** the faces of $P$.

**procedure** FaceEnumeration2;

\begin{verbatim}
begin
  for each $v \in \mathcal{S}_0$
    for each $A \subseteq v^+$
      $F := v^- \cup A$;
      output $F$
  endfor
endfor
\end{verbatim}

end.
The time complexity $O(fd)$ of the face enumeration algorithm for simple polytopes is optimal since it is equal to the time necessary to output all faces of $P$. This algorithm actually suggests that storing the vertices and a good orientation by the set of signed sets is not far from listing all faces, and can be more useful than storing all faces explicitly.

By replacing "for each $A \subset v^+$" with "for each $A \subset v^+$ with $|v^+ \setminus A| = k$" in FaceEnumeration2, we obtain an algorithm to enumerate all $k$-faces, for any fixed $k$. The time complexity is $O(f_0d + f_4d)$.

4. Final remarks

We have presented purely combinatorial algorithms for enumerating all faces of a $d$-polytope from the combinatorial description of vertices (and some information on edges). Dually our algorithm can be used to construct all faces from facets each represented by the set of its vertices. These algorithms can be used to analyze the combinatorial structure of a $d$-polytope given as the solution set to a system of linear inequalities, if a vertex enumeration algorithm such as the Avis-Fukuda algorithm [1] is incorporated. It is noteworthy that the Avis-Fukuda algorithm can output a good orientation together with the list of vertices with no extra cost when the system is nondegenerate.

It is interesting to note that there is a finite (but exponential-time) procedure to find a good orientation of the graph of a simple convex polytope without using any geometrical information of the polytope, see Kalai [5]. Unfortunately, no polynomial algorithm is known for this construction, and it appears to be important to have a good orientation together with the graph structure for face enumeration in simple polytopes.

Our algorithm for general polytopes has been implemented in the language Mathematica and it is publicly available as a supplementary package of the Vertex Enumeration package of Avis–Fukuda algorithm [2].

Parallel to the problem of the present paper is the combinatorial face enumeration problem in arrangements of hyperplanes which is to enumerate the incidence $\{0, 1, -1\}$-vectors of all faces of an arrangement from the set of regions (i.e., full-dimensional faces) of the arrangement. An efficient algorithm for this problem was given in [3].

Acknowledgments

We would like to thank Professor Pierre Hansen of École Polytechnique de Montréal and an anonymous referee for their helpful comments with which we could improve the time complexity of the general face enumeration algorithm.
References


