Splitting a Complex of Convex Polytopes In Any Dimension*

Chandrajit L. Bajaj Valerio Pascucci

Computer Sciences Department
Purdue University
West Lafayette, IN 47907

1 Introduction

We present a locality-based algorithm to solve the problem of splitting a complex of convex polytopes with a hyperplane or a convex subset of it. The solution to this problem has several applications. One goal is to perform boolean set operations. The solution can also be used to decompose a polyhedron into convex polytopes [3] and to generate good meshes [4]. In higher dimensional spaces it can be used to efficiently compute isocontours of linear approximations of scalar fields (a basic technique of Scientific Visualization) [17, 19]. The approach taken here can also be included in a set of robust algorithms [11, 13, 15, 20, 27, 28] based on finite precision arithmetic. It is also defined in a dimension independent framework [5, 16, 24, 25].

The main contributions of this approach are: (i) it can be applied to polyhedral complexes of any dimension d; (ii) the algorithm is robust (it always produces valid output) and consistent (the topological structure of the result always has a geometric counterpart); (iii) it can be used to split a polyhedral complex with any convex subset of a hyperplane; (iv) degradation of the decomposition quality in the result is avoided since the computations are performed only locally, in the zones where the splitting really takes place; (v) the worst case time complexity for the intersection between two d-dimensional polyhedra with m vertices and n facets is $O(m^{1+\epsilon} + j\ n\ m^{1-1/\lfloor d/2 \rfloor} \log m + p\ q\ n)$, where j,p and q are (small) constants that depend on shape and relative position of the two polyhedra.

The main idea for robustness (similar to [13]) is to compute the result using mostly symbolic manipulations and further reduce any numerical computations as much as possible. The algorithm for splitting a polyhedral complex with a hyperplane h is divided into three phases: (i) in the first phase, primary numerical computations are performed to classify vertex positions with respect to h; (ii) in the second phase, symbolic manipulations return the topological structure of the result; (iii) in the final phase, secondary numerical computations are used to detail the geometric structure of the result. It is also possible to collapse small undesired polytopes using symbolic postprocessing.

A leading idea is the maintenance of a search structure. The difference of our approach is that we do not use partitioning trees [23, 26, 29, 31] as search trees structure since its optimization is lost as cascading computations are performed. This happens both on trees optimized with respect to the number of cells induced by the tree decomposition [26] and for trees optimized with respect to the expected traversal time [22]. Moreover, performing boolean set operations with a BSP [23, 29] involves many extraneous computations since each traversal of a face in a tree requires splitting the face at each visited node. If the leaves of the tree reached by a face at the end of the traversal all have equal classification then the computed subdivisions are discarded. Note that if we intersect two complexes of convex polytopes these extraneous subdivisions are computed for all the internal faces of one complex that intersect the internal (external) cells of the other complex.

The alternative scheme we use is to define the traversal of the search structure associated with the complex of polytopes as a range search query [1, 18]. In particular, when we intersect two polyhedra, A and B, we wish to limit the computation of split cells in polyhedron A to the zones intersecting the boundary facets of B. This is achieved by performing, for each face of B, some halfspace range reporting queries and an incremental update of the search structure (as described in [2]) when new vertices are introduced in A. Since we intersect the boundary facets of B with the polytopes of A we avoid having to perform the detection [8] and computation of intersections between pairs of convex polytopes [6, 7]. The basic step of our algorithm is the splitting of a convex polytope with a hyperplane. This has the ad-

*Supported in part by AFOSR grant F49620-93-10138, NSF grant CCR 92-22467 and ONR grant N00014-94-1-0370

vantage of simplicity and known topological structure in any dimension [14].

Overview In Section 2 we outline the approach for a simplified context where only a single polytope is taken into account and numerical computations are considered exact. In Section 3 the method is extended for a complex of convex polytopes. We show how to apply the algorithm locally to only the zones where the splitting affects the complex. In Section 4 the consistency of the method is enforced and a postprocessing is provided to remove small or degenerate polytopes from the result. In Section 5 we briefly analyze the asymptotic worst case complexity of the algorithm.

2 Polytope Splitting

Consider the (relative-closed) faces of a cell c and the relation of inclusion between pairs of such faces. We can construct for the cell c a graph G with the same structure of the Hasse diagram [30] but with oriented arcs corresponding to the direction of the inclusion relation. Considering the set of all the cells in a complex $\mathcal{C}$ we obtain an extended graph G^*. Figure 1 shows the graph G^* for the complex $\mathcal{C}= \{c, d\}$. The set $N^*(N)$ of the nodes of G^* (G) is partitioned into $d+1$ sets $N_i^*(N_i)$, $i = 0, \ldots, d$, where $N_i^*(N_i)$ contains all the nodes representing the i-dimensional faces of $\mathcal{C}(c)$.

With reference to the graph G^*, we call the face f_1 *directly incident* to the face f_2 if there is a directed arc from f_1 to f_2. We call the two faces f_1 and f_2 simply *incident* if there is some path from f_1 to f_2 or from f_2 to f_1. Moreover if there is a face f_3 direct incident to both f_2 and f_1 we say that f_1 is *adjacent* to f_2.

Definition 1 *We say that the complex $\mathcal{C}^*$ is one* split *of the complex $\mathcal{C}$ with a convex subset g of a hyperplane h if every k-dimensional cell c of $\mathcal{C}^*$ intersects one and only one k-dimensional cell of $\mathcal{C}$ and at most one k-dimensional face of g.*

For the remainder of this section we focus on the simple case $\mathcal{C} = \{c\}$ and $g \equiv h$, that is on the problem of splitting a single cell c with an entire hyperplane h. In the following sections we will show how this approach can be generalized.

2.1 Basic Structural Properties

In this subsection we show how to use the graph G to symbolically compute the splitting of a polytope c with a hyperplane h. We first state some properties of G relating to h.

Assume in the following that h does not contain any vertex of c. The configuration with h intersecting a vertex of c is a degenerate case. Following the SoS paradigm [10], we ignore it. In practice this is a *general position* assumption (see [21]) by which we do not consider the cases that occur with zero probability. This does not mean that they are impossible but that they can be removed with a perturbation.

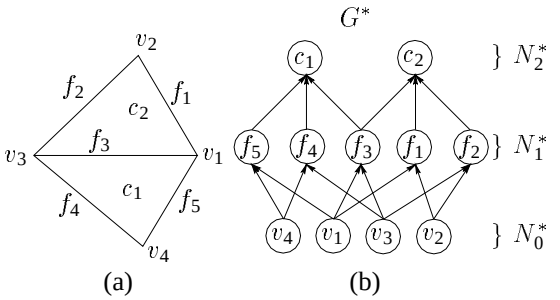

FIGURE 1: A complex of convex polytopes (a) composed of two triangles c, d (2-polytopes), five edges f, g, h, i, j (1-polytopes), and four vertices u, v, w, x (0-polytopes), and (b) the associated graph G^*.

Hence if λ is the linear form associated with the hyperplane h, we classify a vertex v as $\boxed{+}$ (positive) if $\lambda(v) \geq 0$ and as $\boxed{-}$ (negative) if $\lambda(v) < 0$. The implicit assumption is that the value $\lambda(v)$ can be computed with infinite precision. We postpone the problem of making the vertices' classification consistent even if finite precision arithmetic is used until Section 4. In the present section we continue to assume that the computations are exact and hence the classification of the vertices is consistent.

Property 1 *The intersection of the hyperplane h and the polytope c of intrinsic dimension d is either empty or equal to a polytope c' of dimension $d-1$.*

Let h^+ and h^- be the two halfspaces separated by the hyperplane h.

Property 2 *The intersection of h^+ (h^-) with the polytope c of intrinsic dimension d is either empty or equal to a polytope c^+ (c^-) of dimension d.*

Note that if c^+ is empty then $c^- = c$, and if c^- is empty then $c^+ = c$. If the hyperplane h effectively intersects the cell c we have also (see Figure 2 for c equal to a triangle).

Property 3 *Let $f_1, \ldots, f_n$ be the facets of the polytope c and compute $f = c \cap h$, $f_i^+ = f_i \cap h^+$, $f_i^- = f_i \cap h^-$, with $i = 0, \ldots, n$. If c^+ (c^-) exists then the set of its facets is composed of f and all the facets f_i^+ (f_i^-).*

Property 4 *Let $f_1, \ldots, f_n$ be the facets of the polytope c and compute $f = c \cap h$. The set of the facets of f is the set of the non-empty intersections $f_i \cap h$.*

As a consequence of Properties 3 and 4 the structure of the intersection $c \cap h$ can be computed by simply grouping the nonempty intersections $f_i \cap h$, with f_i ranging over the set of c facets. The facets f_i are, in turn, convex polytopes of dimension $d-1$. Hence their intersection with h can be computed by grouping the intersections of their boundary facets with h. This process can be applied recursively until

we reach the 0-dimensional faces that we assumed cannot intersect h.

What is the minimum amount of numerical computations required to determine the structure of the result? As a consequence of the earlier discussion we need only (and cannot avoid) to determine the position of the vertices of the polytope with respect to the hyperplane h. These are the *primary* numerical computations needed to compute the topological structure of the result.

Note that at first we are not required to compute the coordinates of the intersections of the edges with h, since we only need to know if such intersections exist. This information is inferred from the vertex classification. When we know which edges intersect h we can proceed with the symbolic computations creating new nodes in N_0. Each new node in N_0 (a new vertex of G) is associated to one hyperplane–edge intersection. The numerical evaluation of such intersections can be postponed as *secondary* computation.

2.2 The Hyperplane Splitting Algorithm

From the discussion above it is clear that we can define an algorithm that symbolically computes the splitting of a d-dimensional polytope c with a hyperplane h. The algorithm iteratively performs the splitting of the k-faces of c, with k increasing from 1 to d. The classification of the vertices of c with respect to h is performed as a preliminary computation.

– `Split` –

Begin

Step 1 *(primary numerical)* Classify all the vertices of c either [+] or [-]. Then set $k = 1$.

Step 2 *(symbolic computation)* For each $c \in N_k$ do:

- If none of its facets is [+] ([-]) then classify c [-] ([+]) and goto Step 3;
- Create a new $(k-1)$-polytope f (classified as [=]) and connect it to each $(k-2)$-polytope in c classified as [=]. Create two polytopes c^+ and c^- connected both (down) to f and (up) to all the $k+1$ polytopes connected with c. Connect each $(k-1)$-polytope in c classified as [+] to c^+, and each one classified as [-] to c^-. Remove c from N_k.

Step 3 If $k < d$ then $k = k + 1$ and goto Step 2. Else continue to Step 4.

Step 4 *(secondary numerical)* For each vertex v classified [=], compute its coordinates by geometrically intersecting the hyperplane h with the edge divided in two parts by v.

End

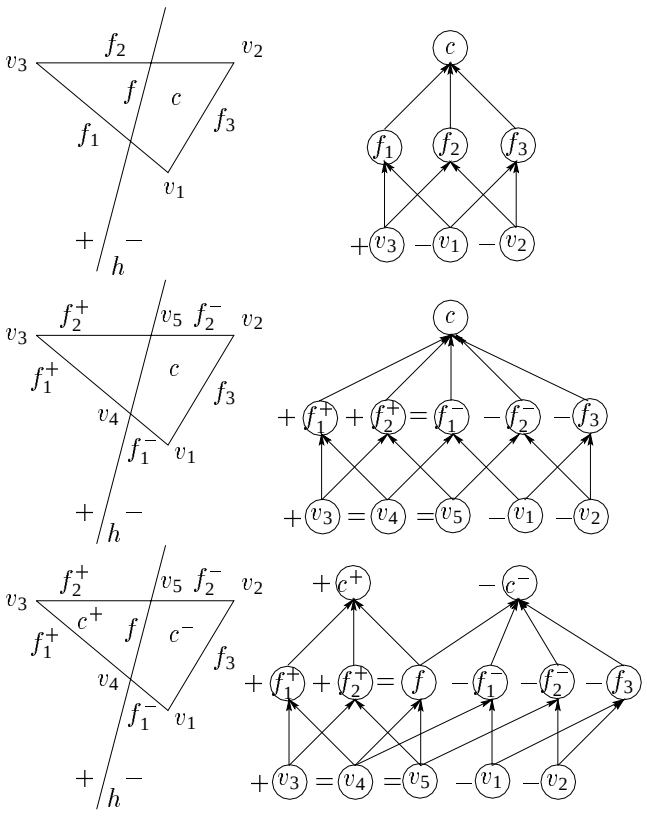

FIGURE 2: The structure of a triangle in three intermediate phases of its split.

In Step 4 we need to compute the coordinates of the newly created vertices. To make the algorithm robust we need to compute each vertex such that: (i) the computation never fails, (ii) the resulting vertex always falls between the two extreme vertices of the divided edge. These two conditions must apply even if the two extremes of the divided edge belong to the same side of h. For this minor detail see Appendix A.

The algorithm `Split` can be used to split all the polytopes that form a complex $\mathcal{C}$. To avoid extraneous computations we can discard, during Step 1, the polytopes that have all the vertices in the same side of h. In fact, the polytopes that need to be processed in Step 2 are the polytopes that have at least one vertex above and one vertex below h. In Section 3.3 a method to select such polytopes is outlined. Here, we want to note that the symbolic and secondary numerical computations directly lead to the construction of the result; that is, no intermediate data structure needs to be constructed and then discarded. The only overhead might arise in the first step in which we need to classify all the vertices of the complex $\mathcal{C}$ to discard the polytopes that are not split with h.

Another important issue is that the algorithm is suitable not only for convex polytopes represented by the set of their geometric faces, but for any convex polytope whose boundary is represented as a decomposition in convex polytopes of lower dimension. In fact the only properties that are required

are that: (i) the polytope is split in no more than two parts; (ii) each "face" is, in turn, a convex polytope. In Section 3 we show how to take advantage of this fact.

As final remark we should notice the similarity of this approach with the beneath–beyond [9] algorithm for the computation of the convex hull of a set of points. In fact both algorithms are based on the incremental update of the incidence graph of a polytope. The present approach can be viewed as an application of the convex hull algorithm in the dual space where faces become vertices and vice versa. The extension of a polytope (convex hull) by addition of an external vertex is equivalent in the dual space to the reduction of its dual polytope by intersection with a halfspace. Hence if the origin of the space is inside the positive half c^+ of the cell c by application of the beneath–beyond algorithm in the dual space we would have obtained c^+. The only difference with the present approach is that we consider both halves, c^+ and c^-, generated by the splitting of c with h. This, in the dual space, implies the computation of an "unbounded convex hulls of points", that requires a generalization of the concept of a convex hull.

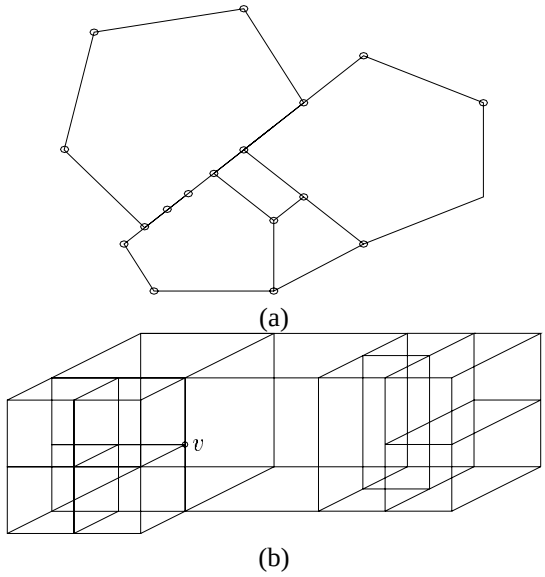

FIGURE 3: Two weak complexes that are not complexes. In (a) the weak complex is composed of a set of five convex polygons. In (b) the weak complex is composed of a set of six cubes and four parallelepipeds.

3 Weak Complex

In this section we introduce the definition of the data structure required to apply `Split` to a set of polytopes instead of a single polytope. Remember that a set of convex polytopes forms a *complex* $\mathcal{C}$ if each pair of polytopes $c, g \in \mathcal{C}$ has intersection $c \cap g$ either empty or equal to a face of both c and g (and such a face belongs to $\mathcal{C}$).

We define a structure similar to the complex but less restrictive and equally suitable for the algorithm of Section 2.

Definition 2 *A set of convex polytopes forms a* weak complex $\mathcal{WC}$ *if, for each pair of polytopes* c *and* g*, the intersection* $f = c \cap g$ *satisfies one of the following two conditions:*

1. $c \cap g = \emptyset$;
2. $c \cap g = \partial c \cap \partial d = f_1 \cup \cdots \cup f_i$, *with* $f_1, \ldots, f_i \in \mathcal{WC}$.

With some abuse of terminology we call a k-face of $c \in \mathcal{WC}$, any k-polytope $f \in \mathcal{WC}$ such that $f \subset \partial c$.

In general, a weak complex need not be complex. Conversely every complex is also a weak complex since it always satisfies condition *2* of Definition 2, with $i = 1$ and f_1 being a k-face of both c and g. In Figure 3 two weak complexes of intrinsic dimension two (a) and three (b) are depicted.

3.1 Splitting a Weak Complex

Proposition 1 *If* l *polytopes* $\{c_1, c_2, \ldots, c_l\}$*, of dimension* $k \leq d$*, of a weak complex* $\mathcal{WC}$ *are split along a hyperplane* h *then the resulting set of polytopes form a weak complex.*

Proof: The proof is inductive in the growing dimension k of the split polytopes of $\mathcal{WC}$ and incremental in the number of split polytopes. In particular, assume that we are splitting one k-polytope c_i a time, and that the $(k-1)$-skeleton of $\mathcal{WC}$ has been decomposed in a $(k-1)$-dimensional weak complex $\mathcal{WC}^*$ by splitting the boundary faces of all the polytopes $\{c_1, c_2, \ldots, c_l\}$. Then we can split c_i in two halves c_i^+, c_i^- by adding a $(k-1)$-polytope to $\mathcal{WC}^*$ to obtain a weak complex $\mathcal{WC}' = (\mathcal{WC} + \{c_i^+, c_i^-\}) - \{c_i\}$.

This process holds for $k = 0, 1$ since the 0-polytopes cannot be split. Assume now (by induction) that we have the $(k-1)$-dimensional weak complex $\mathcal{WC}^*$. The additional $(k-1)$-face f that splits c_i in two halves is bounded by (all and only) the $(k-2)$-faces created by splitting ∂c along h. Then $\mathcal{WC}^* \cup \{f\}$ is a weak complex. $\mathcal{WC}'$ is a weak complex because, by construction, the intersection between any pair of polytopes in $\mathcal{WC}'$ is a weak sub-complex of $\mathcal{WC}^* \cup \{f\}$. Each time we split a polytope c_i we get a new weak complex. Hence, after we split, in sequence, the polytopes c_i, for $i = 1, \ldots, l$, we obtain a weak complex. $\diamond$

In summary, if we have a weak complex we can split some of its polytopes along a hyperplane h to obtain, again, a weak complex. This is an important property that does not hold for the nonweak complexes and that allows us to apply the splitting process locally. In fact, if we want to maintain a complex after each splitting we must either split all the polytopes that intersect h, or decompose the polytopes incident to $f_1, f_2, \ldots, f_l$.

3.2 Reconstructing a Complex

We now reduce a weak complex to a complex as is required in many applications. Note, however, that a weak complex is a decomposition into convex polytopes, and sometimes this is sufficient.

The problem we want to solve is: *Given a weak complex $\mathcal{WC}$, construct a complex $\mathcal{C}$ whose polytopes are subsets of the polytopes in $\mathcal{WC}$.*

Proposition 2 *A k-dimensional weak complex is always a complex for $k \leq 1$.*

Proof: This holds because a 0-polytope cannot be split. Therefore the boundary of a 1-polytope is always composed of two 0-polytopes (the two extreme vertices). Hence, an 1-dimensional weak complex is also a complex. $\diamond$

As a consequence, we can construct the complex $\mathcal{C}$ inductively starting from the 1-skeleton of $\mathcal{WC}$ and decomposing the polytopes of increasing dimension. At each step, we assume that the boundary of each polytope forms a complex and not simply a weak complex. Such inductive procedure can be applied if we can prove, in general, that given a weak complex $\mathcal{WC}$ which $(k-1)$-skeleton si a complex $\mathcal{C}$ we can decompose the cells of $\mathcal{WC}$ such that its k-skeleton becomes also a complex $\mathcal{C}^*$.

Proposition 3 *If the $(k-1)$-skeleton of a weak complex $\mathcal{WC}$ of dimension k is a complex $\mathcal{C}^*$, then we can construct a k-dimensional complex $\mathcal{C}$ whose $(k-1)$-skeleton contains $\mathcal{C}^*$.*

Proof: Take a generic convex polytope c of $\mathcal{WC}$. Its boundary is decomposed into a complex of convex polytopes. If such boundary polytopes are all geometric faces of c then nothing needs to be done. Otherwise, since the boundary polytopes of c are smaller than the geometric faces, we must divide c into a set of smaller polytopes.

Take a new point p inside c (e.g. the barycenter). Then assume that the $(k-1)$-skeleton of c is composed of m polytopes and the $(k-2)$-skeleton of n polytopes. Construct n new $(k-1)$-polytopes connecting v with each polytope of the $(k-2)$-skeleton of c. These n polytopes can be added to $\mathcal{C}^*$ to form a new complex. The polytope c can now be replaced in $\mathcal{WC}$ by m new polytopes. The $(k-1)$-skeleton of each new polytope is composed of one facet f of c and the $(k-1)$-polytopes that are constructed by connecting v to the facets of f. The new polytopes (with their lower dimensional faces) now form a complex. After all the polytopes of $\mathcal{WC}$ have been divided in this way, a complex $\mathcal{C} \supset \mathcal{C}^*$ is obtained. $\diamond$

The validity of this proposition is important because it implies that the polytopes that have not been directly affected by the splitting process do not need to be split. Moreover, since the input data was a complex, we can know directly which are the faces of a polytope that do not correspond to a geometric face. In fact, in the beginning no face was split. So we can give the same tag to the polytopes that come from the same original face. Hence, the selection of the polytopes that need to be further decomposed can be performed symbolically by checking, for each polytope, if there is a pair of its faces that have the same tag. If we had to perform this check numerically it would have required testing whether any pair of k-faces were embedded in the same k-dimensional affine subspace.

In conclusion, the price we have to pay to obtain the complex $\mathcal{C}$ from $\mathcal{WC}$ is the introduction of some new points in the decomposition. It is evident that this is not necessary in dimension two.

Proposition 4 *A 2-dimensional weak complex can be decomposed into a complex without adding any new vertex.*

In Figure 4, a possible decomposition of the weak complex of Figure 3 is shown. This case is particularly important

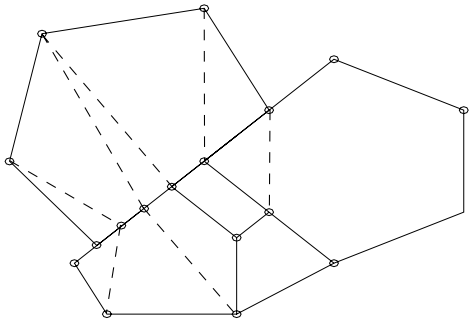

FIGURE 4: A weak complex is shown in filled lines. A complex is obtained by adding the dashed lines.

because we can use the algorithm of the previous section to perform boolean operations on boundary representations of polyhedra embedded in E^3 (in this case the boundary of a polyhedron is a 2-complex). We can split the boundaries of the two operand polyhedra and then select some of their polytopes to obtain the union, the intersection or the difference. The polytopes that are involved in this operation all have dimension $d \leq 2$. This means that we can perform the complete operation without adding any auxiliary point. If the available boundary representation is not composed of convex polytopes, then we can use the same method to reduce it to a set of convex polytopes. In fact, for each nonconvex cell we can split an enveloping convex polytope (e.g. the bounding box) with its boundary (that must be a complex). From the result it is easy to select the polytopes that form a decomposition in convex polytopes of the initial nonconvex polygon.

The question of how to achieve a decomposition without auxiliary vertices in any dimension seems to be an open problem. A possible strategy could be to maintain the non-weak complex after the split of each single polytope c. In this way, we can assume that the boundary of any adjacent

polytope g has only a few decomposed faces. This implies that at each step we divide not only c but also the polytopes that share with c any face that has been split. This approach could imply an increased number of split polytopes.

3.3 Polytope Classification

The first step of the algorithm `Split` on page 3 requires the classification of the positions of all the vertices with respect to the hyperplane h. If our aim is to split the polytopes of $\mathcal{WC}$ along a convex polytope g embedded in h, and not along the entire hyperplane h, we can take advantage of the vertices' classification to reduce the number of split polytopes. This is desirable to enhance the algorithm performance and possible because we deal with a weak complex instead of a complex. In fact, if we split all the polytopes that intersect the hyperplane h, we produce more splitting in $\mathcal{WC}$ than strictly required by the insertion of the polytope g. This induces two main negative consequences: (a) the number of polytopes of the complex increases much more than required; (b) the time spent to compute extraneous subdivisions slows down the method. This is a consequence of the fact that a basically *local* problem is approached as if it were *global*. Here we exploit the locality of the problem to improve the expected performance of the algorithm.

Assume that the polytope g that split $\mathcal{WC}$ has m facets $\{f_1, \ldots, f_m\}$. The problem is to select a subset $\mathcal{WC}'$ of $\mathcal{WC}$ of polytopes that can be split by g. Assume that $\mathcal{WC}^{\pm}|_h$ is the set of polytopes intersected by h and $\mathcal{WC}^{-}|_h$ is the set of polytopes contained in the halfspace h^- for any hyperplane h. Then we compute the set $\mathcal{WC}'$ as follows:

$$\mathcal{WC}' = \left(\mathcal{WC}^{\pm}|_h - \bigcup_{f_i} \mathcal{WC}^{-}|_{f_i} \right)$$

where f_i ranges over the boundary facets f_i of g (or, more precisely, over the set of the hyperplanes in which the boundary facets of g are embedded). It is easy to see that for any polytope $c \in \mathcal{WC}$ the two following properties are satisfied:

- if $c \cap g \neq \emptyset$ then $c \in \mathcal{WC}'$;
- if $c \notin \mathcal{WC}'$ then $c \cap g = \emptyset$.

That is, $\mathcal{WC}'$ is a (hopefully small) superset of the set of polytopes that intersect g. Then, instead of splitting all the polytopes in $\mathcal{WC}^{\pm}|_h$, we can split only the polytopes in $\mathcal{WC}'$.

Assume that, for a given hyperplane h, we can detect the set V^- of the vertices that belong to the halfspace h^-. We wish to determine the polytopes across h and the polytopes inside h^-. For each element $v \in V^-$, we collect the incident polytopes that contain another vertex $w \notin V^-$. Such polytopes are the polytopes in $\mathcal{WC}^{\pm}|_h$. To obtain $\mathcal{WC}^{-}|_h$ we collect the polytopes that do not have any vertex w outside V^-. It is evident that the two sets can be constructed at the same time since every polytope c incident to $v \in V^-$ must be contained either in $\mathcal{WC}^{\pm}|_h$ or in $\mathcal{WC}^{-}|_h$. Thus, the preliminary step can be reduced to a *half-space range-reporting problem*. We use the search structure described in [2]. For a set of m points, the construction of this search structure requires $O(m^{1+\epsilon})$ time. This method performs each search in $O(m^{1-1/\lfloor d/2 \rfloor} \log m)$ time and the incremental insertion of new vertices in $O(m^{\epsilon})$ time.

4 Enforcing Consistency

Strictly speaking, the algorithm described above cannot fail, since situations like a division by zero cannot occur. In fact, the sequence of operations is designed so that a result is always achieved. However a problem arises in guaranteeing the consistency of the result if the numerical computations are not exact. In this section we present a technique that can be used to force the primary numerical computations to be consistent with the symbolic structure of the input data. The technique adopted here is, basically, an extension of the approach introduced by Hopcroft and Kahn [13]. Our approach will also prevent the algorithm from generating (almost) degenerate polytopes in the result.

4.1 Consistent Classification

In Section 2, we have assumed that the numerical computations were correct and we conceptually perturbed the splitting hyperplane h so that no vertex would belong to it. The consequence was that the only polytopes that could belong to h were the polytopes generated by splitting a polytope of higher dimension. Now we allow the input polytopes to lie on h; thereby avoiding the creation of very small features. To enforce the consistency of the classification performed in Step 1 of `Split` we apply the following property (in [13] the property was considered only for $k = 2$):

Property 5 *If $k+1$ affinely independent vertices of a k-polytope c lie on h then c lies on h.*

The property we apply requires testing (i) whether some vertices of c are on h and (ii) if such vertices are affinely independent. The first test must be performed with tolerance δ. The second test must be based only on the topological structure of c.

When we compute the position of a vertex v with respect to a hyperplane h we have to take into account the accuracy that the numerical computations can achieve. In particular, if the value of $\lambda(v)$ of the linear form associated with h is included in the range $[-\delta, \delta]$ we do not know if v is above or below h. In this case v is classified as $\boxed{=}$.

Next, we start classifying the edges. If an edge has at least one $\boxed{=}$ extreme it cannot be split. This is the basic step that prevents the algorithm from generating small edges. If both vertices of one edge are classified $\boxed{=}$ then the edge is

classified $\boxed{=}$. More generally, we test a k-polytope c after we have forced all of its faces to have a consistent classification. Property 5 tells us that if c does not lie on h then all its vertices classified $\boxed{=}$ must lie on one facet of c. Hence the rule we apply is as follows:

> If there are a vertex v and a facet f of c that are both classified $\boxed{=}$, but v is not contained in f, then all the vertices of c must be classified $\boxed{=}$.

The rule must be propagated to the neighboring polytopes. This propagation is performed consistently, because any vertex that has been classified $\boxed{+}$ can become $\boxed{=}$ but not $\boxed{-}$ (and vice versa). Furthermore, once a polytope has become $\boxed{=}$, it cannot be reclassified to either $\boxed{+}$ or $\boxed{-}$.

– `Reclassify` –

Begin

Step 1 Set k=1. Mark all the polytopes that have at least one vertex classified $\boxed{=}$ as *unchecked*.

Step 2 For each *unchecked* polytope c of intrinsic dimension k do:

- Mark c *checked*.
- If there are one vertex v and one facet f of c both classified $\boxed{=}$ such that v is not a vertex of f (or of any other facet obtained with f by splitting the same original facet f') then:
  - Classify all the vertices of c as $\boxed{=}$.
  - Mark all the faces of c as *unchecked*.
  - Set k=0 and continue to Step 3

Step 3 If $k < d$ then set $k = k+1$ and goto Step 2, otherwise Stop.

End

It is important to note that when c has both $\boxed{+}$ and $\boxed{=}$ facets it must not be split. The test in Step 2 of `Split` accounts for this situation since, in this case, there is no $\boxed{-}$ facet and so c is classified $\boxed{+}$. In fact, there can never be three facets classified as $\boxed{+}$, $\boxed{-}$ and $\boxed{=}$ in c.

In this way, we obtain a classification that is consistent for each polytope of the complex and, hence, for the whole complex. The consistency of the result can be stated inductively. We assume that the edges are consistently split. This simply means that each edge has two distinct extremes.

Proposition 5 *Let c be a k-polytope, split along h, using* `Split` *enforced by* `Reclassify`*. The interiors of the two halves c^+ and c^-, generated by h, are connected.*

Proof: We prove the proposition for c^+. Assume, on the contrary, that the interior of c^+ is not connected and that c_1^+, c_2^+ are two of its connected components. By construction, the facets of c_1^+ and c_2^+ that are not classified $\boxed{=}$ must be (part of) the original facets of c. Since the facets of c^+ have been consistently classified (`Reclassify`) then there exist two vertices $v_1 \in c_1^+$ and $v_2 \in c_2^+$ such that $\lambda(v_1) > \delta$ and $\lambda(v_2) > \delta$. That is, two vertices that have been correctly classified to belong to the interior of h^+. Then we can select two points v_1' and v_2' that belong to the interior of c_1^+ and c_2^+, respectively, and belong to the interior of h^+. A straight line connecting v_1' with v_2' must intersect one facet of c_1^+ that has been classified $\boxed{+}$. But, since the interior of c is convex, there exists a straight line in h^+ that connects v_1' with v_2' without intersecting any facet of c. This contradicts the initial assumption. Therefore, the interior of c^+ is connected. The proof is similar for c^-. ◇

Proposition 5 assures us that `Reclassify` will remove all the inconsistencies that might be caused by wrong numerical computations. The resulting representation is consistent in the sense that, for each polytope representation, there exists a geometric model that has the same structure. To verify that this is true, we have to establish what requirement the polytope must satisfy. It is well known that a necessary and sufficient condition for a graph to represent the edges of a linear convex polytope in E^3 is that it must be planar and triply connected [12]. For k-dimensional polytopes it is also known that the edges form a k-connected graph (this condition is only necessary). In the case of the polytopes of a weak complex, these properties are not always satisfied because their faces can be partitioned. Consider, for example, the vertex v in Figure 3(b). In one of the large cubes, v is incident only to two edges (actually it splits an edge of the cube). This means that the edges of that cube do not form a triply connected graph. For this reason we use a different kind of argument to state that each polytope is consistently represented. In particular, we show that the boundary of each k-polytope is homeomorphic to a $(k-1)$-sphere.

Proposition 6 *Let c be a k-polytope split along h using* `Split` *enforced by* `Reclassify`*. Then c^+ and c^- are both homeomorphic to a $(k-1)$-sphere.*

Proof: By Proposition 5, c^+ is a single connected polytope and, by construction, all but one of its facets are (part of) facets of c. In fact, the only facet of c^+ that does not belong to ∂c is the facet f that separates c^+ from c^-. Select a point v inside c^+ and a $(k-1)$-sphere S that contains c. From v we can project the facets of c^+ that are (part of) facets of c. This projection is a one-to-one mapping that covers a certain region S^+ of S. We now have to map f to the remaining part S^-. This can be done because S^- is a connected region. Furthermore, we can build a hypersurface f' with the same topology of f by taking a point w on h inside c and connecting it to all the facets of f. We can map f' to cover all S^- without any overlapping. Then ∂c^+ is homeomorphic to a $(k-1)$-sphere. For c^- a similar argument holds. ◇

4.2 Collapsing Polytopes

Even if we assume, as in [13], that the input data is an α-representation so that we obtain in output a β-representation, with β suitably related to α, we may need to remove some small polytopes to obtain a good decomposition under certain constraints.

The basic operation required is the deletion of small edges whose length is less than ϵ. We assume that if the length of one edge is less than ϵ it can be collapsed to a point without affecting the convexity of the incident polytopes. This assumption, though reasonable, is important in order to allow us to annihilate the edge without requiring a redecompositon of the incident polytopes. We can so perform this operation simply as a symbolic operation over the incidence graph structure of the polytopes complex. In particular, we only need to update the polytopes that have been modified by the edge deletion. The procedure is as follows:

– Collapse –

Begin

Step 1 Make the two extremes of one degenerate edge coincident. That is, merge its two nodes in N_0^* with the relative set of directed arcs that connect them to the incident edges. Mark the incident edges and set $k = 1$ as ?.

Step 2 For each k-polytope c marked ? do:

- if c has no facet, remove c and mark all the $(k+1)$-polytopes to which c is directly incident as ?.
- if c has only one facet f then remove c and mark all the $(k+1)$-polytopes to which c is directly incident as ?. If f was directly incident only to c then remove f and check its facets recursively.

Step 3 If $k < d$ then increment k by one and go to Step 2. Otherwise Stop.

End

Figure 5 shows how the complex of Figure 2 is modified by moving the vertex v_4 to be coincident with the vertex v_5 and hence deleting f, f_1^+, f_2^+ and v_3.

If we have a k-polytope c, with $k > 1$, whose measure is small, but has no small edges, then it must have a pair of adjacent facets f_1 and f_2 that form an angle close to π. Split the polytopes with a hyperplane h that forms the same angle with f_1 and f_2 and passes through the $(k-2)$-polytopes that they have in common. Then apply this procedure again; that is, either remove small edges of the resulting halves or split them recursively. At the end of this procedure all the degenerate edges will have been removed.

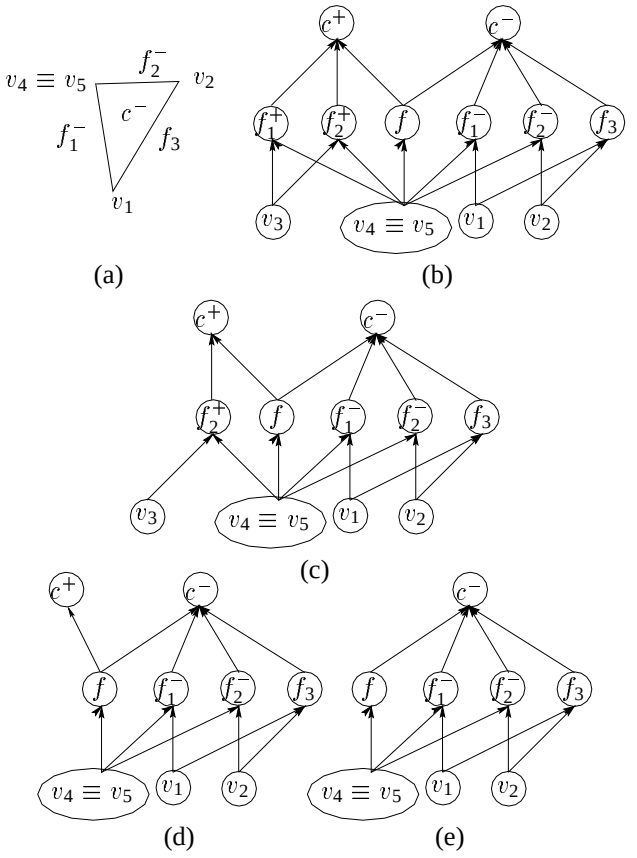

FIGURE 5: Modification of the complex in Figure 2 induced by making the vertex x coincident to y.

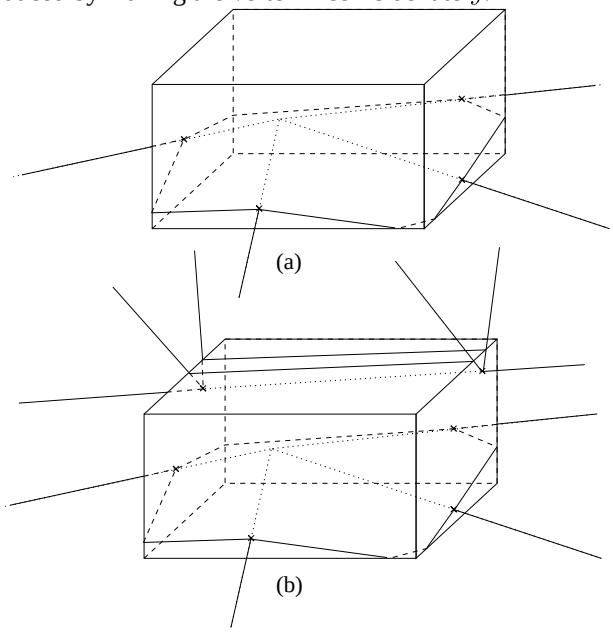

FIGURE 6: The polytope in (a) is split by four facets that have a common face (a vertex). The polytope in (b) is split by six facets that do not have a common face.

5 Complexity

Here we state some bounds on the efficiency of the method and on the amount of splitting it induces. We concentrate our attention on the intersection (difference) between two polyhedra P and Q represented as complexes of convex polytopes. The boolean operation is performed by splitting P with the boundary of Q.

Assume that Q has n boundary facets and that each of the facets intersects p convex polytopes of P on average. Under a certain condition, the number of new polytopes induced by the splitting is pn. The condition is that all the facets of Q that intersect the same polytope c of P have a common face. In fact, if c is intersected by l facets that have a common face then c is split in precisely $l+1$ parts. Figure 6 shows two cases in which such a condition is satisfied (a) and not satisfied (b). Generally, the condition holds when Q does not have features that are small with respect to the dimension of the polytopes of P. In practice, the case of Figure 6(b) is not common and even in such a configuration, the number of splits induced on the polytope is close to $l+1$. We conjecture that, in a boolean operation performed as above, the number of newly generated polytopes is $O(pn)$.

The algorithm iterates over two basic steps: (i) classification of the vertices of P to select the polytopes to be split, (ii) the splitting of a polytope. Assume that the number of vertices in P after the splitting has been performed is $\hat{m}$, the mean number of faces of a convex polytope in P is $O(q)$, and the mean number of facets of each boundary facet of Q is $O(j)$. The preliminary step is the construction of the search structure for the vertices' classification, which requires $O(m^{1+\epsilon})$ time. This structure must be updated $(\hat{m}-m)$ times. The update time for m points is $O(m^{\epsilon})$, so the total update time can be bounded by $O((\hat{m}-m)\ \hat{m}^{\epsilon})$. The search takes place $O(j)$ times for each of the n facets of ∂Q and requires $O(m^{1-1/\lfloor d/2 \rfloor}\log m)$ when the structure contains m points. The total search time will be $O(j\ n\ \hat{m}^{1-1/\lfloor d/2 \rfloor}\log \hat{m})$. During the split of a polytope, each of its $O(q)$ faces is considered once for the consistent classification and once for the actual split, so that it will require $O(q)$ time. The split is performed in $O(p\ q\ n)$ time. The overall time complexity will so be $O(m^{1+\epsilon}+(\hat{m}-m)\ \hat{m}^{\epsilon}+j\ n\ \hat{m}^{1-1/\lfloor d/2 \rfloor}\log \hat{m}+p\ q\ n)$. Assuming that $O(\hat{m})=O(m)$ it reduces to $O(m^{1+\epsilon}+j\ n\ m^{1-1/\lfloor d/2 \rfloor}\log m+p\ q\ n)$.

References

[1] AGARWAL, P. K., AND MATOUŠEK, J. On range searching with semialgebraic sets. *Discrete Comput. Geom. 11* (1994), 393–418.

[2] AGARWAL, P. K., AND MATOUŠEK, J. Dynamic halfspace range reporting and its applications. *Algorithmica 13* (1995), 325–345.

[3] BAJAJ, C., AND DEY, T. K. Convex decomposition of polyhedra and robustness. *SIAM J. Comput. 21* (1992), 339–364.

[4] BERN, M., AND EPPSTEIN, D. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, D.-Z. Du and F. K. Hwang, Eds., vol. 1 of *Lecture Notes Series on Computing*. World Scientific, Singapore, 1992, pp. 23–90.

[5] BRISSON, E. Representing geometric structures in d dimensions: Topology and order. *Discrete Comput. Geom. 9* (1993), 387–426.

[6] CHAZELLE, B. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput. 21*, 4 (1992), 671–696.

[7] CHAZELLE, B., AND DOBKIN, D. P. Intersection of convex objects in two and three dimensions. *J. ACM 34* (1987), 1–27.

[8] DOBKIN, D. P., AND KIRKPATRICK, D. G. Fast detection of polyhedral intersection. *Theoret. Comput. Sci. 27* (1983), 241–253.

[9] EDELSBRUNNER, H. *Algorithms in Combinatorial Geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[10] EDELSBRUNNER, H., AND MÜCKE, E. P. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph. 9* (1990), 66–104.

[11] FORTUNE, S., AND MILENKOVIC, V. Numerical stability of algorithms for line arrangements. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.* (1991), pp. 334–341.

[12] GRÜNBAUM, B. *Convex Polytopes*. Wiley, New York, NY, 1967.

[13] HOPCROFT, J. E., AND KAHN, P. J. A paradigm for robust geometric algorithms. *Algorithmica 7* (1992), 339–380.

[14] KINCSES, J. On polytopes cut by flats. *Discrete Comput. Geom. 14* (1995), 287–294.

[15] LI, Z., AND MILENKOVIC, V. Constructing strongly convex hulls using exact or rounded arithmetic. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.* (1990), pp. 235–243.

[16] LIENHARDT, P. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry & Applications 4*, 3 (1994), 275–324.

[17] LORENSEN, W., AND H.CLINE. Marching cubes: a high resolution 3d surface construction algorithm. *A.C.M. Computer Graphics 21*, 4 (1987), 163–170.

[18] MATOUŠEK, J. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom. 10*, 2 (1993), 157–182.

[19] MAX, N., HANRAHAN, P., AND CRAWFIS, R. Area and volume coherence for efficient visualization of 3d scalar functions. *A.C.M. Computer Graphics 24*, 5 (1990), 27–33.

[20] MILENKOVIC, V. Robust polygon modeling. *Computer-Aided Design 25*, 9 (1993). (special issue on Uncertainties in Geometric Design).

[21] MULMULEY, K. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[22] NAYLOR, B. Constructing good partitioning trees. In *Proc. Graphics Interface '93* (Toronto, ON, 1993), pp. 181–191.

[23] NAYLOR, B., AMANTIDES, J., AND THIBAULT, W. Merging bsp trees yields polyhedral set operations. *Comput. Graph. 24*, 4 (1990), 115–126. Proc. SIGGRAPH '90.

[24] PAOLUZZI, A., BERNARDINI, F., CATTANI, C., AND FERRUCCI, V. Dimension–independent modeling with simplicial complexes. *ACM Transactions on Graphics 12*, 1 (Jan. 1993), 56–102.

[25] PASCUCCI, V., FERRUCCI, V., AND PAOLUZZI, A. Dimension-independent convex-cell based HPC: Representation scheme and implementation issues. In *Solid Modeling '95, Third ACM/IEEE Symposium on Solid Modeling and Applications* (Salt Lake City, Utah, 1995), C. Hoffmann and J. Rossignac, Eds., ACM Press, pp. 163–174.

[26] PATERSON, M. S., AND YAO, F. F. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete Comput. Geom. 5* (1990), 485–503.

[27] STEWART, A. J. Local robustness and its application to polyhedral intersection. *International Journal of Computational Geometry & Applications 4*, 1 (1994), 87–118.

[28] SUGIHARA, K. A robust and consistent algorithm for intersecting convex polyhedra. *Comput. Graph. Forum 13*, 3 (1994), 45–54. Proc. EUROGRAPHICS '94.

[29] THIBAULT, W. C., AND NAYLOR, B. F. Set operations on polyhedra using binary space partitioning trees. In *Proc. SIGGRAPH'87* (1987), pp. 153–162.

[30] TROTTER, W. T. *Combinatorics and Partially Ordered Sets: Dimension Theory*. Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press, 1992.

[31] VANĚČEK JR., G. Brep-index: a multidimensional space partitioning tree. *Internat. J. Comput. Geom. Appl. 1*, 3 (1991), 243–261.

APPENDIX A: Fault immune hyperplane-edge intersection

The method used to intersect an edge l, of extreme vertices v_1,v_2, with a hyperplane h defined by the linear form $\lambda(x) = 0$ is reported here. This method is applied only when the result has already been symbolically computed. For this reason, we require the method to give a result even if the input data is incorrect. The algorithm must return, in any case, a point v^* such that:

- v^* belongs to the edge l;
- if the input is correct (that is, v_1 and v_2 are located on the opposite sides of h and both with a value of the linear form λ greater than δ) v^* is the actual intersection between l and h, within the accuracy of the available numerical computations;

The intersection point v^* is computed with the formula:

$$v^* = \alpha v_1 + \beta v_2, \tag{1}$$

with $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$. This expression assures that the point v^* belongs to the edge l. Now, we must compute α and β without failure and such that v^* is $l \cap h$.

Assume, without loss of generality, that $|\lambda(v_1)| \leq |\lambda(v_2)|$ (if this is not the case exchange v_1 with v_2), and let $\gamma = |\lambda(v_1)|$ and $\eta = |\lambda(v_2)| - |\lambda(v_1), |$.

If $\eta \geq \delta$ or $\gamma \geq \delta$ we can compute the two parameters with the exact formula:

$$\begin{aligned} \alpha &= \frac{\gamma}{2\gamma + \eta}; \\ \beta &= \frac{\gamma + \eta}{2\gamma + \eta}. \end{aligned}$$

Since $2\gamma + \eta \geq \delta$ the two divisions can be performed.
If $\eta < \delta$ and $\gamma < \delta$ we will directly set:

$$\alpha = \beta = \frac{1}{2}. \tag{2}$$

and obtain a result that satisfies the above properties.