Discrete Comput Geom 16:69–112 (1996)

Discrete & Computational
Geometry

Mixed-Volume Computation by Dynamic Lifting Applied to Polynomial System Solving

J. Verschelde,[1] K. Gatermann,[2] and R. Cools[1]

[1] Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200 A, B-3001 Heverlee, Belgium
Jan.Verschelde@cs.kuleuven.ac.be
Ronald.Cools@cs.kuleuven.ac.be

[2] Konrad-Zuse-Zentrum für Informationstechnik Berlin,
Heilbronner Str. 10, D-10711 Berlin-Wilmersdorf, Germany
gatermann@sc.ZIB-Berlin.de

Abstract. The aim of this paper is to present a flexible approach for the efficient computation of the mixed volume of a tuple of polytopes. In order to compute the mixed volume, a mixed subdivision of the tuple of polytopes is needed, which can be obtained by embedding the polytopes in a higher-dimensional space, i.e., by lifting them. Dynamic lifting is opposed to the static approach. This means that one considers one point at a time and only fixes the value of the lifting function when the point really influences the mixed volume. Conservative lifting functions have been developed for this purpose. This provides us with a deterministic manipulation of the lifting for computing mixed volumes, which rules out randomness conditions. Cost estimates for the algorithm are given. The implications of dynamic lifting on polyhedral homotopy methods for the solution of polynomial systems are investigated and applications are presented.

1. Introduction

The aim of this paper is to present an algorithm for computing the mixed volume of a tuple of polytopes. Although the motivation for this paper stems from the polyhedral homotopy methods for sparse polynomial systems, our approach is of independent interest, see Section 9 of [25] for other applications of volume and mixed-volume computation. The algorithm is developed from a geometric viewpoint. Recently, much research has been devoted to the computation of the mixed volume.

Definition 1.1. The *mixed volume* $V_n(\mathcal{P})$ of an n-tuple of polytopes

$$\mathcal{P} = (P_1, P_2, \ldots, P_n)$$

is

$$V_n(\mathcal{P}) := \sum_{I \subset \{1,2,\ldots,n\}} (-1)^{n-\#I} \operatorname{vol}_n \left(\sum_{i \in I} P_i \right), \tag{1}$$

where $\operatorname{vol}_n(P)$ equals the volume of P and $P_1 + P_2 = \{\mathbf{x} + \mathbf{y} | \mathbf{x} \in P_1, \ \mathbf{y} \in P_2\}$.

If all polytopes in $\mathcal{P}$ are identical, then $V_n(P, P, \ldots, P) = n! \operatorname{vol}_n(P)$. The mixed volume is multilinear and invariant under a shift of the polytopes. See, e.g., Chapter 4 of [9] for more on mixed volumes.

Note that formula (1) is in general not a good way for computing mixed volumes. In [52] Verschelde *et al.* showed how the recursion formula, used in [3] for computing the mixed volume, is already useful for solving practical problems, despite its combinatorial implementation. Based on an idea of Betke [4], a more flexible approach for the computation of the mixed volume has been presented by Huber and Sturmfels in [30]. We henceforth call their approach the lifting method. In [10] and [18] Canny and Emiris applied it to the efficient computation of sparse mixed resultants. The exploitation of symmetry relations has been examined in [50], which led to the development of the symmetric lifting method.

The idea of this paper is to apply the concepts of incremental convex hull constructing algorithms, see [12], [15], [25], [28], and [43]. In a lifting method all points are lifted, i.e., embedded into an $(n + 1)$-dimensional space. See [4] for the application on two polytopes and [45] for the generalization to tuples of polytopes. Afterward, the faces of the lower hull of the lifted points need to be computed. This is a static approach. By dynamic lifting, a point is only lifted when it is sure to belong to the subdivision, which is achieved by placing (or pushing) the point with respect to a regular subdivision, see [35]. This offers a flexible computational tool to investigate which points influence the mixed volume.

For polynomial systems the mixed volume of the Newton polytopes of the polynomials gives an upper bound for the number of isolated solutions [3], see Section 4. In [30] a so-called polyhedral homotopy method based on lifting has been presented, which enables all isolated solutions of a polynomial system to be computed. The approach presented in this paper offers a flexible computational tool to investigate which coefficients can have an influence on the number of solutions of the system. Hereby an algorithm is presented for incrementally solving polynomial systems, which tends to be more stable than the static polyhedral homotopy continuation.

This paper consists of four parts. The first part is devoted to the case of computing a regular triangulation for one polytope. In the second part the dynamic lifting algorithm is generalized to the computation of the mixed volume. The impact on polynomial system solving is discussed in the third part. This paper concludes with a section summarizing the main properties of the algorithms investigated.

2. Dynamic Construction of Regular Triangulations

This part describes the dynamic lifting algorithm applied to one polytope. Its structure is as follows. After some preliminary definitions, the basic version of the algorithm is sketched. The following subsections describe the key steps in this algorithm. Some cost estimates are given in the final subsection.

2.1. *Regular Triangulations*

For completeness we start with some well-known definitions [34]. We assume the points to be vectors in Euclidean space $\mathbb{E}^n$, equipped with the standard inner product $\langle \cdot, \cdot \rangle$.

Definition 2.1. Given a set $A \subseteq \mathbb{E}^n$. The *dimension* of A equals d, denoted by $\dim(A) = d$, if A contains at most $d+1$ points $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_d$ such that $\mathbf{c}_0 - \mathbf{c}_j$, $j = 1, 2, \ldots, d$, are linearly independent. For a finite set $A \subset \mathbb{E}^n$, the *convex hull* of A, denoted by $\mathrm{conv}(A)$, is the smallest convex set that contains A. The *polytope* P of A is defined by $P = \mathrm{conv}(A)$. A *face* of a polytope P is the intersection of hyperplanes which define half-spaces that contain the polytope entirely. The polytope itself is considered as a *trivial* face. All other faces, the empty set included, are *proper* faces. A *vertex* of a polytope is a face of dimension zero. A face of dimension k is called a *k-face*.

Definition 2.2. Given a polytope P in n-dimensional space, with $\dim(P) = d$. A *facet* ∂P of P is a face of P, with $\dim(\partial P) = d-1$. ∂P is defined as the intersection of P with one hyperplane that defines a half-space which contains P entirely and is characterized by its *inner normal* γ:

1. $\forall \mathbf{x}, \mathbf{y} \in \partial P, \langle \mathbf{x}, \gamma \rangle = \langle \mathbf{y}, \gamma \rangle$.
2. $\langle \cdot, \gamma \rangle$ attains its minimum at ∂P, i.e., $\langle \mathbf{y}, \gamma \rangle > \langle \mathbf{x}, \gamma \rangle, \forall \mathbf{x} \in \partial P, \forall \mathbf{y} \notin \partial P$.

Since the functional $\langle \cdot, \gamma \rangle$ is constant on ∂P we denote $\langle \partial P, \gamma \rangle := \langle \mathbf{x}, \gamma \rangle$, for one $\mathbf{x} \in \partial P$. The facet itself is denoted by $\partial_\gamma P = \mathrm{conv}(\partial_\gamma A)$, with $\partial_\gamma A = \{\mathbf{x} \in A \mid \langle \mathbf{x}, \gamma \rangle = \min_{\mathbf{y} \in A} \langle \mathbf{y}, \gamma \rangle\}$.

Definition 2.3. The *lower hull* of a polytope P consists of all facets $\partial_\gamma P$ with $\gamma_n > 0$.

The following definitions are based on the definitions in [30] and in [35]. See Lecture 5 of [56] for a more detailed mathematical background.

Definition 2.4. Given a set of points $A \subset \mathbb{E}^n$, a *subdivision S of A* consists of a collection of cells $S = \{C_1, C_2, \ldots, C_m\}$, with $C_k \subset A, \forall k$, which satisfies:

1. $\dim(C_k) = n$.
2. $\mathrm{conv}(C_l) \cap \mathrm{conv}(C_k)$ is a proper face of both $\mathrm{conv}(C_l)$ and $\mathrm{conv}(C_k)$, $l \neq k$.
3. $\bigcup_{k=1}^{m} \mathrm{conv}(C_k) = \mathrm{conv}(A)$.

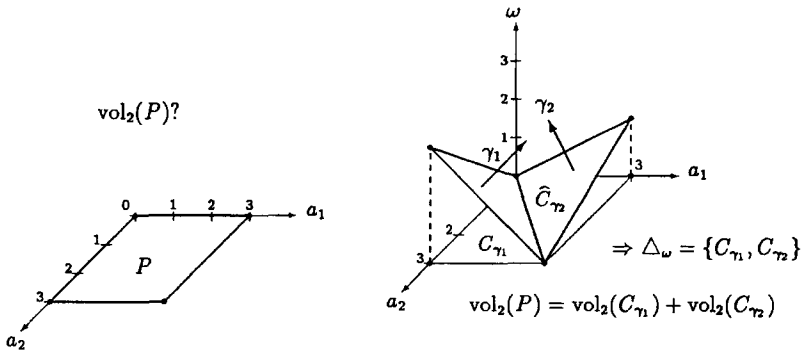


Fig. 1. The construction of a regular triangulation by lifting.

See [25] and [15] for references on how subdivisions can be computed by means of a lifting function.

Definition 2.5. A *lifting function* ω lifts every point of a set A, ω: $A \to \mathbb{E}$: $\mathbf{a} \mapsto \omega(\mathbf{a})$. This yields the *lifted point*

$$\widehat{\mathbf{a}} = \begin{pmatrix} \mathbf{a} \\ \omega(\mathbf{a}) \end{pmatrix}.$$

The set of lifted points is denoted by $\widehat{A}$. The lifted polytope is denoted by $\widehat{P} = \text{conv}(\widehat{A})$. The collection of lifted cells $\widehat{C}$ is denoted by $\widehat{S}$. Note that lifted points are also sometimes referred to as *weighted* points.

For every lifting function a subdivision is induced by associating the cells with the projected facets $\partial_\gamma \widehat{P}$ of the lower hull of $\widehat{P}$, $P = \text{conv}(A)$. As each cell C is characterized by the inner normal γ of $\widehat{C} = \partial_\gamma \widehat{P}$, the cell C can be denoted by C_γ, see Fig. 1.

Definition 2.6. A subdivision $S = \{C_1, C_2, \ldots, C_m\}$ of A is called *a regular subdivision* when there is a lifting function ω which induces S. It is denoted by S_ω.

Note that in [30] a regular subdivision is called a *coherent* subdivision.

As mentioned in the introduction the usefulness of subdivisions lies in the volume computation. Given a subdivision S of a polytope P, its volume can be computed by

$$\text{vol}_n(P) = \sum_{C \in S} \text{vol}_n(C). \tag{2}$$

The computation of the volumes of the cells is straightforward for special subdivisions:

Definition 2.7. A subdivision S is called *a triangulation*, and therefore denoted by $\Delta = \{C_1, C_2, \ldots, C_m\}$ when $\#C_j = n + 1$, $\forall j$, i.e., $\text{conv}(C_j)$ is an n-dimensional

simplex. A triangulation Δ is said to be *a regular triangulation* if it can be induced by a lifting function ω.

For cells $C = \{c_0, \ldots, c_n\}$ of a triangulation we have

$$\mathrm{vol}_n(C) = \frac{1}{n!} |\det(\mathbf{c}_1 - \mathbf{c}_0, \ldots, \mathbf{c}_n - \mathbf{c}_0)|. \tag{3}$$

Other applications of triangulations, e.g., Voronoi diagrams, can be found in [15] and [43].

A regular triangulation is denoted as Δ_ω. It can be computed by random lifting functions. See [16] for an algorithm to construct a regular triangulation, with the assumption that the lifted points are in general position. An implementation of this algorithm in three dimensions has been presented in [19].

Proposition 2.8 (see [30]). *If the lifting function ω is chosen sufficiently at random, then the induced subdivision is a triangulation. This holds even for integer lifting functions.*

In practice it is often desired to exploit the structure of the polynomial system (see Section 4 for the relationship between polynomials and subdivisions) and to construct a special subdivision, e.g., a symmetric one, like in [50]. In this case, the assumption of randomness, which does not take the additional constraints of the system into account, cannot be relied upon. The aim of dynamic lifting is to provide a deterministic lifting algorithm which enables the construction of subdivisions with a special geometry.

2.2. *The Dynamic Lifting Algorithm*

The basic version of an incremental construction of a regular triangulation is described in Algorithm 2.9. There we write $\widehat{\Delta}_\omega^{\mathbf{x}}$ for the set of lifted cells which contain $\mathbf{x}$ in the triangulation for the points that are already processed. The general notation is the following. Let S_ω be a regular subdivision of A and consider $\mathbf{x} \in A$. Denote $S_\omega^{\mathbf{x}} = \{C \in S_\omega \mid \mathbf{x} \in C\}$. Analogously, $\widehat{S}_\omega^{\widehat{\mathbf{x}}} = \{\widehat{C} \in \widehat{S}_\omega \mid \widehat{\mathbf{x}} \in \widehat{C}\}$.

Algorithm 2.9. The dynamic lifting algorithm:

Input: A.	*a set of points*
Output: Δ_ω, ω.	*a regular triangulation of A*
$\Delta_\omega := \{C_0\}$;	*compute an initial cell*
$\omega := 0$;	*with lifting value = 0*
$E := C_0$;	*the points already processed*
$B := \emptyset$;	*the set of interior points*

```
while E ≠ A do                          invariant: Δ_ω is a triangulation of E
  x ∈ A\E;                                                  choose next point
  E := E ∪ {x};                                             to be processed
  if x ∈ Δ_ω                                        ∃? C ∈ Δ_ω: x ∈ conv(C)
  then B := B ∪ {x};                                update the set of interior points
  else ω := ω(Δ̂_ω, x);                                      determine next value
       x̂_{n+1} := ω;                                        of the lifting function
       Δ_ω^x := New_Facets(Δ_ω, x);                         compute new facets
       Δ_ω := Δ_ω ∪ Δ_ω^x;                                  update the lower hull
  end if;
end while;
∀x ∈ B: x̂_{n+1} := ω + 1.                                   lift out the interior points
```

There are five subalgorithms in Algorithm 2.9:

1. Compute an initial cell C_0.
2. Choose the next point $\mathbf{x}$ to be processed $\mathbf{x} \in A \backslash E$.
3. Check whether a point $\mathbf{x}$ already belongs to the triangulation $\mathbf{x} \in \Delta_\omega$, i.e., whether there is a $C \in \Delta_\omega$ such that $\mathbf{x} \in \text{conv}(C)$.
4. Give a point the lifting value $\omega(\widehat{\Delta}_\omega, \mathbf{x})$.
5. Compute the set of new cells in the triangulation $\Delta_\omega^{\mathbf{x}}$.

The initialization steps for Algorithm 2.9 are described in the next section. In order to achieve a simple update of the triangulation, i.e., like $\Delta_\omega := \Delta_\omega \cup \Delta_\omega^{\mathbf{x}}$, it is necessary to apply special lifting functions, defined in Section 2.4. In Section 2.5 a pivoting mechanism and an efficient data structure are presented to compute the new cells. To control the condition of the lifting, the regular triangulation will be made flat, as described in Section 2.6. In the last section it is proven that this algorithm runs in polynomial time in $\#\Delta_\omega$.

2.3. *Computation of an Initial Cell and Vertices*

At this early stage of computation, degeneracy, i.e., $\dim(A) < n$, should be detected. By characterizing the degenerate case, an initial cell can be computed by the Greedy Algorithm, see p. 212 of [27]. Algorithm 2.10 formulates this algorithm.

Algorithm 2.10. Computation of an initial cell:

Input: $A \subset \mathbb{E}^n$.
Output: $C = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_i\}$, a collection of linearly independent points.
If $\dim(A) = n$, then $i = n$, otherwise $i < n$.

1. Let $\gamma \neq \mathbf{0}$. If $\partial_\gamma A = \partial_{-\gamma} A$, then $\dim(A) < n$.
   Otherwise, take $\mathbf{c}_0 \in \partial_\gamma A$ and $\mathbf{c}_1 \in \partial_{-\gamma} A$, $\mathbf{c}_1 \neq \mathbf{c}_0$. This yields $C := \{\mathbf{c}_0, \mathbf{c}_1\}$.
2. For i from 2 to n do the following. Let $\gamma \neq \mathbf{0}$: $\langle \mathbf{c}_j - \mathbf{c}_0, \gamma \rangle = 0$, for $j = 1, 2, \ldots, i-1$. If $\partial_\gamma A = \partial_{-\gamma} A$, then $\dim(A) < n$. Otherwise, if $\langle \partial_\gamma A, \gamma \rangle \neq \langle \mathbf{c}_0, \gamma \rangle$, then take $\mathbf{c}_i \in \partial_\gamma A$, else take $\mathbf{c}_i \in \partial_{-\gamma} A$. This yields $C := C \cup \{\mathbf{c}_i\}$.

Except for nongeneric choices of γ the points in C are vertices.

An implementation of Algorithm 2.10 is now described. If $\mathbf{0} \in A$, we can take $\mathbf{c}_0 := \mathbf{0}$. Then the procedure for computing an initial cell C consists of finding n linearly independent vertices of the point set A. The determination of the directions γ's is as follows. It is natural to take the first direction along the first coordinate axis. The ith direction γ is not explicitly chosen as in Algorithm 2.10, but equivalently a change of coordinates is performed such that the ith unit vector can be chosen as γ. The change of coordinates is done by a unimodular transformation U that preserves the first $i-1$ unit vectors and maps $\mathbf{c}_{i-1}$ to the $(i-1)$th unit vector (modulo a scalar multiple). By these orthogonality assumptions, the next point $\mathbf{c}_i$ can be chosen by taking the point with the largest ith component.

Example 2.11. Consider the matrix M whose columns contain the points of A. Because $\mathbf{0} \notin A$, let $\mathbf{c}_0 = (1, 0)^t$ and perform an affine shift with $\mathbf{c}_0$, so that $\mathbf{0} \in A'$. Let $\gamma = (1, 0)^t$, then $\mathbf{c}_1$ corresponds to the third column in M', which has the largest (underlined) first component. This yields $\mathbf{c}_1 = (4, 2)^t$.

$$M = \begin{bmatrix} 1 & 3 & 4 & 2 & 0 \\ 0 & 0 & 2 & 4 & 2 \end{bmatrix} \rightarrow M' = \begin{bmatrix} 0 & 2 & \underline{3} & 1 & -1 \\ 0 & 0 & 2 & 4 & 2 \end{bmatrix}.$$

The unimodular transformation

$$U = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} \quad \text{gives} \quad UM' = \begin{bmatrix} 0 & 2 & 1 & -3 & -3 \\ 0 & -4 & 0 & \underline{10} & 8 \end{bmatrix}.$$

In Step 2 the maximum with respect to $\gamma = (0, 1)^t$ equals 10. Since we maintained the positions of the points in the matrix, the point $\mathbf{c}_2$ of the initial cell is the fourth column of the original matrix M. Because the unimodular transformation is volume preserving and at each step the maximum value along the coordinate axis has been taken (see the underlined entries in the matrices), the cell will have a volume as large as possible.

Note that the steps in Algorithm 2.10 can be used for searching an optimal point to be added next. Optimal means that this point has a large contribution to the volume. To achieve this goal approximately, we propose to choose a certain direction randomly and then compute a vertex with respect to that direction. This corresponds to the principle of randomized incremental constructions, see [12] and [28].

Once Algorithm 2.10 terminates with an initial cell C_0, the volume computation problem is nondegenerate. For an efficient computation of the volume, the nonvertex points of the remainder point set $A \backslash C_0$ can be omitted in advance, as those points will have no influence on the volume. A vertex of a polytope can be considered as the solution of a linear optimization problem, see [46] and p. 184 of [27]. In [26] a feasibility problem is proposed for computing an irredundant representation of a polytope. Let

$A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$, to solve the membership question $\mathbf{x} \in \text{conv}(A)$ we use the following:

$$\begin{array}{ll} \min & \mu_0 + \mu_1 + \cdots + \mu_n \\ \text{subject to} & \left\{ \begin{array}{rl} \displaystyle\sum_{i=1}^{m} \lambda_i a_{ij} + \mu_j x_j = x_j, & j = 1, 2, \ldots, n, \\ \displaystyle\sum_{i=1}^{m} \lambda_i + \mu_0 = 1, & \\ \lambda_i \geq 0, & i = 1, 2, \ldots, m, \\ \mu_j \geq 0, & j = 0, 1, \ldots, n. \end{array} \right. \end{array} \tag{4}$$

It has as a trivial feasible but not optimal solution all $\mu_j = 1$ and all $\lambda_i = 0$. If the problem has an optimal feasible solution, with all $\mu_j = 0$, for $j = 0, 1, \ldots, n$, then $\mathbf{x}$ can be written as a convex combination of the other points in A, the coefficients in this combination are given by the λ_i's and thus $\mathbf{x} \in \text{conv}(A)$.

The vertex set of A can be computed by repetitive application of (4):

Algorithm 2.12. Computation of the vertex set:

```
Input: A.                              a set of points
Output: V                              the vertex set

V := A;                                initialization
while A ≠ ∅ do                         enumerate points in A
  choose x ∈ A;                        next point to be checked
  A := A\{x};                          update A
  if x ∈ conv(V\{x})                   to decide, solve (4)
    then V := V\{x};                   update V
  end if;
end while.
```

So, an option has been added to Algorithm 2.9. One can start with all points or compute the vertices first. In the latter case, the test $\exists?\ C \in \Delta_\omega$: $\mathbf{x} \in \text{conv}(C)$ in Algorithm 2.9 is always false. For the choice of the next point to be processed in the latter case, we observe the following. For $n = 2$ it can be guaranteed that the number of cells in the subdivision will be minimal. For $n > 2$ it is still necessary to choose the point along a random direction to achieve this goal approximately.

2.4. *Conservative Lifting Functions*

The purpose of this section is to introduce special lifting functions which allow the dynamic construction of a regular triangulation.

Lemma 2.13. *Given* $\widehat{S}_\omega = \{\widehat{C}_\gamma\}$, *consider* $\mathbf{x} \notin \text{conv}(C_\gamma)$. *Choose a lifting value* $\omega(\mathbf{x})$ *so that* $\langle \widehat{\mathbf{x}}, \gamma \rangle > \langle \widehat{C}_\gamma, \gamma \rangle$. *Let* S'_ω *be the induced subdivision of* $C_\gamma \cup \{\mathbf{x}\}$, *then* $\widehat{S}_\omega \subset \widehat{S}'_\omega$. *For each new cell* $C \in S'_\omega \backslash S_\omega$: $\mathbf{x} \in C$.

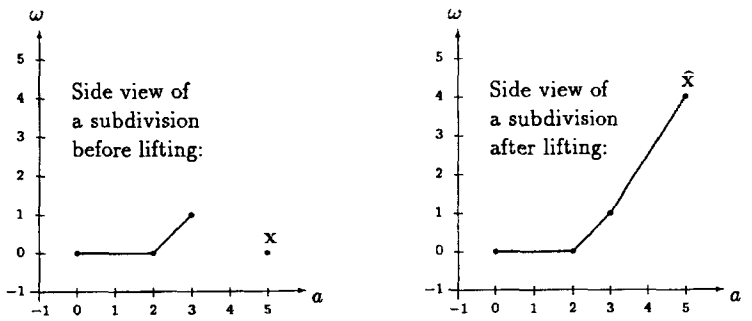


Fig. 2. A conservative lifting of $\mathbf{x}$: the cells in the subdivision are preserved.

Proof. As $\langle \widehat{\mathbf{x}}, \gamma \rangle > \langle \widehat{\mathbf{c}}, \gamma \rangle$, $\forall \widehat{\mathbf{c}} \in \widehat{C}_\gamma$, γ still attains its minimum at $\widehat{C}_\gamma$, so $\widehat{C}_\gamma \in \widehat{S}'_\omega$. Hence $\widehat{S}_\omega \subset \widehat{S}'_\omega$. The second statement is trivial to prove. □

Figure 2 illustrates the application of Lemma 2.13 and introduces the concept of *conservative lifting*.

Definition 2.14. Let $\widehat{C}_\gamma$ be a cell. Consider a point $\mathbf{x}$. Let $\omega(\mathbf{x}) = \widehat{x}_{n+1}$ so that $\langle \widehat{\mathbf{x}}, \gamma \rangle > \langle \widehat{\mathbf{c}}, \gamma \rangle$, $\forall \widehat{\mathbf{c}} \in \widehat{C}_\gamma$, then ω is called a *conservative lifting with respect to* $\widehat{C}_\gamma$. Let $\widehat{S}_\omega$ be a regular subdivision. If ω is a conservative lifting with respect to each cell $\widehat{C}_\gamma \in \widehat{S}_\omega$, then ω is called a *conservative lifting with respect to* $\widehat{S}_\omega$.

This kind of lifting preserves the cells in the subdivision, so it is *conservative*. The advantage of using conservative lifting functions lies in the simplicity of placing [12], [15], [35] the points in the triangulation: no deletion or modification of existing cells is required. By using different orders of placing the points, any *placeable* triangulation can be obtained. Note however that not every regular triangulation is placeable, see [35]. Theorem 2.15 implies that a regular triangulation can always be maintained by successive applications of a conservative lifting function with respect to a regular triangulation, on a point $\mathbf{x}$.

Theorem 2.15. *Let S_ω be a regular subdivision of A induced by ω. Consider a point $\mathbf{x}$, lifted conservatively with respect to $\widehat{S}_\omega$. Let S'_ω be the regular subdivision of $A \cup \{\mathbf{x}\}$, then $\widehat{S}_\omega \subseteq \widehat{S}'_\omega$. For each new cell $C \in S'_\omega \backslash S_\omega$: $\mathbf{x} \in C$. If $\exists C_\gamma \in S_\omega$: $\mathbf{x} \in \text{conv}(C_\gamma)$, then $\widehat{S}_\omega = \widehat{S}'_\omega$.*

Proof. By applying Lemma 2.13 successively on all individual cells in the subdivision, the theorem is proven. It is sufficient to see that, for any $\widehat{C}_\gamma$, $\langle \widehat{\mathbf{x}}, \gamma \rangle > \langle \widehat{C}_\gamma, \gamma \rangle$. □

The efficient computation of the new cells $\widehat{S}'_\omega \backslash \widehat{S}_\omega$ is the topic of Section 2.5. The last statement of Theorem 2.15 shows that a nonvertex point inside the convex hull of one of the cells can be *lifted out*.

An explicit formula for constructing an optimal conservative lifting function is given by the following:

Proposition 2.16. *Let $\widehat{C}_\gamma$ be a cell. Then the lowest lifting value obtained by a discrete conservative lifting function is given by*

$$\omega(\widehat{C}_\gamma, \mathbf{x}) := \max\left(1, \left(\frac{\langle \widehat{C}_\gamma, \gamma\rangle - \sum_{k=1}^{n} x_k\gamma_k}{\gamma_{n+1}}\right) + 1\right). \tag{5}$$

Given a regular subdivision $\widehat{S}_\omega$. The lowest lifting value obtained by a conservative lifting function is given by

$$\omega(\widehat{S}_\omega, \mathbf{x}) := \max_{\widehat{C}_\gamma \in \widehat{S}_\omega} \omega(\widehat{C}_\gamma, \mathbf{x}). \tag{6}$$

The lifting described in Proposition 2.16 is the one applied in Fig. 2.

2.5. *Computation of New Cells by Pivoting*

The aim of this section is the efficient computation of new cells and to show how a triangulation can always be obtained. It leads to the implementation of the subalgorithm *New_Facets* of Algorithm 2.9. We need the following fundamental lemma (Carathéodory, see, e.g., [34] and [56]).

Lemma 2.17. *Let $C = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n\}$ be a cell, spanning the n-dimensional simplex* $\mathrm{conv}(C)$. *Consider a point* $\mathbf{x}$. *Then*

$$\mathbf{x} \in \mathrm{conv}(C) \quad \Leftrightarrow \quad \exists \lambda_k\colon \mathbf{x} = \sum_{k=0}^{n} \lambda_k \mathbf{c}_k, \qquad \sum_{k=0}^{n} \lambda_k = 1, \quad \lambda_k \geq 0, \quad k = 0, 1, \ldots, n, \tag{7}$$

i.e., $\mathbf{x}$ can be written as a convex combination of the points in C. Moreover, this representation is independent of the order of the points inside C.

Computation of the λ_k's can be done by solving a linear system. Consider the set of shifted points $\{\mathbf{c}_1 - \mathbf{c}_0, \ldots, \mathbf{c}_n - \mathbf{c}_0, \mathbf{x} - \mathbf{c}_0\}$ to be the columns of the matrix M. Solve the homogeneous linear system defined by $M\Lambda = \mathbf{0}$. The existence of a solution is guaranteed by $\dim(C) = n$ and the uniqueness is guaranteed by $\#C = n + 1$.

To decide whether $\mathbf{x} \in P$, apply the following:

Theorem 2.18 (Carathéodory). *Let Δ be a triangulation of the polytope P, then* $\mathbf{x} \in P \Leftrightarrow \exists C \in \Delta\colon \mathbf{x} \in \mathrm{conv}(C)$.

Lemma 2.17 gives a simple computable criterion to implement the membership test

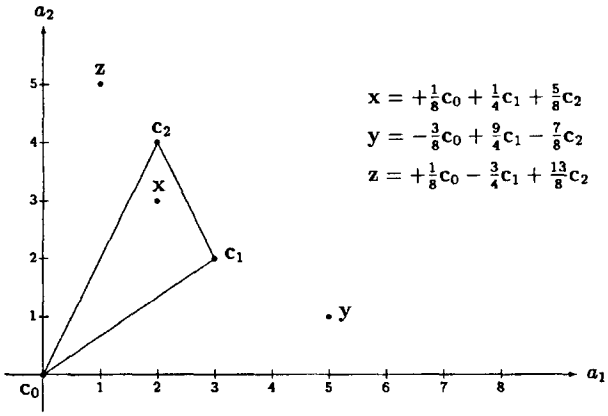


Fig. 3. Given a simplex defined by $C = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2\}$ and three possible point configurations: $\mathbf{x} \in \text{conv}(C)$, adding $\mathbf{y}$ to $\text{conv}(C)$ leads to two new simplices, while for $\mathbf{z}$ there is only one new simplex.

$\mathbf{x} \in \text{conv}(C)$. As illustrated in Fig. 3, computing the decomposition of $\mathbf{x}$ with respect to the points in C provides additional information indicating which new cells arise. So, it also illustrates conditions (8) in Lemma 2.19.

Lemma 2.19. *Let* $S_\omega = \{C_\gamma\}$ *and* $\widehat{C}_\gamma = \{\widehat{\mathbf{c}}_0, \widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_n\}$. *Consider* $\widehat{\mathbf{x}}, \widehat{\mathbf{x}} = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k$, *with* $\sum_{k=0}^{n} \lambda_k = 1$ *and* $\widehat{x}_{n+1}$ *such that* $\langle \widehat{\mathbf{x}}, \gamma \rangle > \langle \widehat{C}_\gamma, \gamma \rangle$. *Let* S'_ω *be the induced subdivision of* $C_\gamma \cup \{\mathbf{x}\}$. *Then the new cells* $\widehat{C} \in \widehat{S}'_\omega \backslash \widehat{S}_\omega$ *are given by*

$$\{\widehat{\mathbf{c}}_0, \ldots, \widehat{\mathbf{c}}_{k-1}, \widehat{\mathbf{x}}, \widehat{\mathbf{c}}_{k+1}, \ldots, \widehat{\mathbf{c}}_n\} \qquad \textit{where} \quad \lambda_k < 0. \tag{8}$$

Proof. First bring the cell $\widehat{C}_\gamma$ in a diagonal form $\widehat{C}'_\gamma$, by a shift and a transformation: $\widehat{\mathbf{c}}'_k = T(\widehat{\mathbf{c}}_k - \widehat{\mathbf{c}}_0), \widehat{\mathbf{c}}'_k \in \widehat{C}'_\gamma : \widehat{\mathbf{c}}'_0 = \mathbf{0}$ and $\widehat{\mathbf{c}}'_k$ is the kth unit vector, $k = 1, 2, \ldots, n$. Then

$$\widehat{\mathbf{x}} = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k \quad \Leftrightarrow \quad \widehat{\mathbf{x}} - \widehat{\mathbf{c}}_0 = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k - \left(\sum_{k=0}^{n} \lambda_k \right) \widehat{\mathbf{c}}_0, \qquad \text{because} \quad \sum_{k=0}^{n} \lambda_k = 1$$

$$\Leftrightarrow \quad \widehat{\mathbf{x}}' = T(\widehat{\mathbf{x}} - \widehat{\mathbf{c}}_0) = \sum_{k=0}^{n} \lambda_k T(\widehat{\mathbf{c}}_k - \widehat{\mathbf{c}}_0)$$

$$\Leftrightarrow \quad \lambda_k = \widehat{x}'_k, \quad k = 1, 2, \ldots, n, \qquad \lambda_0 = 1 - \sum_{k=1}^{n} \widehat{x}'_k.$$

Due to Lemma 2.13 all new cells contain $\mathbf{x}$. Since a cell has $n + 1$ points we need only consider the case that $\mathbf{x}$ replaces one $\mathbf{c}_k$. There are two cases.

Case 1: $\widehat{\mathbf{x}}'$ *replaces* $\widehat{\mathbf{c}}_0'$. The inner normal δ of this facet C_δ satisfies

$$
\begin{aligned}
k \neq 0, \quad l \neq 0, \qquad \langle \widehat{\mathbf{c}}_k', \delta \rangle = \langle \widehat{\mathbf{c}}_l', \delta \rangle \quad &\Leftrightarrow \quad \delta_l = \delta_k, \\
\langle \widehat{\mathbf{x}}', \delta \rangle = \langle \widehat{\mathbf{c}}_1', \delta \rangle \quad &\Leftrightarrow \quad \delta_1 \left(\sum_{k=1}^{n} \widehat{x}_k' \right) + \widehat{x}_{n+1}' \delta_{n+1} = \delta_1 \\
&\Leftrightarrow \quad \delta_1 \left(1 - \sum_{k=1}^{n} \widehat{x}_k' \right) = \widehat{x}_{n+1}' \delta_{n+1} \\
&\Leftrightarrow \quad \lambda_0 = \frac{\widehat{x}_{n+1}' \delta_{n+1}}{\delta_1}.
\end{aligned}
$$

C_δ is a cell of S_ω' if $\widehat{C}_\delta$ is in the lower hull. The inner normal of a facet in the lower hull satisfies $\delta_{n+1} > 0$ and $\langle \widehat{\mathbf{c}}_k', \delta \rangle < \langle \widehat{\mathbf{c}}_0', \delta \rangle \Leftrightarrow \delta_k < 0$, for all $k \neq 0$. Thus $\{\mathbf{x}, \mathbf{c}_1, \ldots, \mathbf{c}_n\} \in S_\omega' \backslash S_\omega$, if and only if $\lambda_0 < 0$.

Case 2: $\widehat{\mathbf{x}}'$ *replaces* $\widehat{\mathbf{c}}_k'$, $k > 0$. For the inner normal δ we compute

$$
\begin{aligned}
l \neq k, \qquad \langle \widehat{\mathbf{c}}_l', \delta \rangle = \langle \widehat{\mathbf{c}}_0', \delta \rangle \quad &\Leftrightarrow \quad \delta_l = \delta_0, \\
\langle \widehat{\mathbf{x}}', \delta \rangle = \langle \widehat{\mathbf{c}}_0', \delta \rangle \quad &\Leftrightarrow \quad \widehat{x}_k' \delta_k + \widehat{x}_{n+1}' \delta_{n+1} = 0 \\
&\Leftrightarrow \quad \lambda_k = \frac{\widehat{x}_{n+1}' \delta_{n+1}}{\delta_k}.
\end{aligned}
$$

If $\widehat{C}_\delta$ is part of the lower hull, then $\langle \widehat{\mathbf{c}}_k', \delta \rangle > \langle \widehat{\mathbf{c}}_0', \delta \rangle \Leftrightarrow \delta_k > 0$. Thus $\{\mathbf{c}_0, \ldots, \mathbf{c}_{k-1}, \mathbf{x}, \mathbf{c}_{k+1}, \ldots, \mathbf{c}_n\} \in S_\omega' \backslash S_\omega$, if and only if $\lambda_k < 0$. □

Lemma 2.19 enables the computation of new cells if the triangulation consists of one simplex. If the triangulation consists of several simplices a direct and simple approach for computing $\widehat{\Delta_\omega^{\mathbf{x}}}$ is successively applying Lemma 2.19 to all cells in the subdivision. Afterward, it must be checked whether each new cell corresponds to a facet of the lower hull of $\text{conv}(\widehat{A})$. An obvious way to do this is by computing the inner products of their normal with all the other points.

Better ways to avoid the computation of spurious cells use a kind of minimalist data structure, like in [16]. The triangulation is stored as a list of cells. Each cell C_γ is characterized by $n+1$ vertices and an inner normal γ. For each vertex there is a pointer to the neighboring cell, which can be obtained by replacing that vertex by another one, see Fig. 4. Note that in Fig. 4 each vertex appears only once and that no null pointers are drawn.

The application of Lemmas 2.17 and 2.19 requires the solution of a linear system $M\Lambda = \mathbf{0}$, defined after Lemma 2.17. As this has to be done for each new point to be added to Δ, the factorization matrix for the first n columns of M can be stored. This reduces the solution of a linear system to a back substitution, which requires only $O(n^2)$ operations, whereas the factorization of an n-dimensional matrix requires $O(n^3)$ operations.

To decide for a new point $\mathbf{x}$ whether $\mathbf{x} \in \Delta$, Theorem 2.18 can be implemented by enumerating all cells C and computing the decomposition of $\mathbf{x}$ with respect to the vertices of C. A better way is to exploit the neighborship relation. Starting with one cell

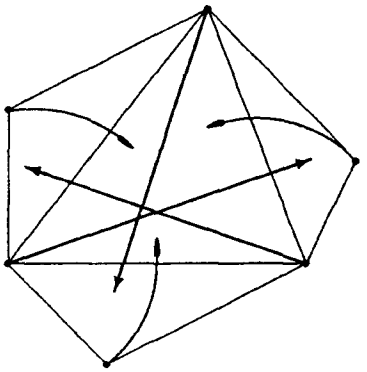


Fig. 4. Four connected cells.

C, e.g., the initial cell, walk toward $\mathbf{x}$ along one chain of cells as pictured in Fig. 5. One ends either with a cell C with $\mathbf{x} \in \text{conv}(C)$ or with a cell close to $\mathbf{x}$. For each cell in the chain the representation $\mathbf{x} = \sum_k \lambda_k \mathbf{c}_k$ is computed. If $\mathbf{x} \notin \text{conv}(C)$, then some $\lambda_k < 0$ is chosen and the walk continues at the cell to which the pointer associated to the vertex $\mathbf{x}_k$, corresponding to λ_k, points to. The walk stops when a null pointer associated to a vertex with $\lambda_k < 0$ is encountered.

The data structure can be used for the determination of the new cells in two ways. The walk described above finishes with one cell close to $\mathbf{x}$. With Lemma 2.19 one new cell is determined. The neighboring new cells can be determined by considering the neighboring points on the edge of the triangulation and by computing their decomposition with respect to this one new cell which contains $\mathbf{x}$. Alternatively, all outer cells, i.e., those with vertices with null pointer, while computing the decomposition $\mathbf{x} = \sum_k \lambda_k \mathbf{c}_k$, can be enumerated. The vertices with $\lambda_k < 0$ and null pointer at vertex c_k yield the new cells due to Lemma 2.19.

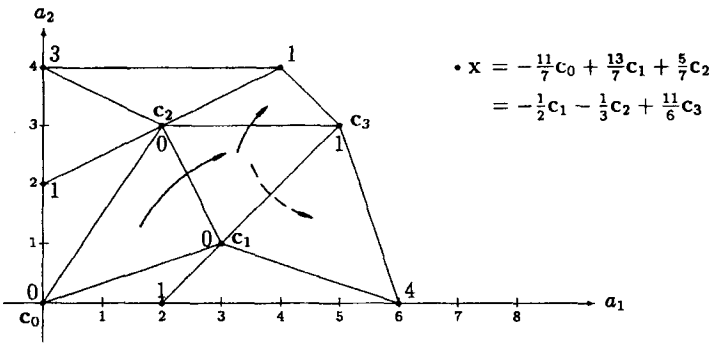


Fig. 5. An efficient walk from the simplex defined by $C = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2\}$ to the point $\mathbf{x} = (7, 4)^t$. The lifting values are indicated by the big numbers. Only three instead of nine simplices need to be visited. The dashed arrow represents an alternative path.

To compare the cost of both alternatives the number of decompositions which have to be determined has to be counted. Such a decomposition is needed for each cell in the walk. As a conclusion, we can state that the walk will be fruitful when the triangulation contains inner points, like the case in Fig. 5. When all nonvertex points have been eliminated in advance, such a walk can be more expensive than the simple enumeration of all outer cells.

2.6. *Flattening a Regular Subdivision*

Section 2.4 provided lower bounds on the lifting for a new point to be added to the triangulation. This section deals with specifying operational upper bounds on the lifting, necessary to counter the following numerical problems. With fixed-point arithmetic, the lifting value might cause an overflow, see Fig. 2 for how the lifting becomes steeper and steeper. With floating-point arithmetic, the rounding errors which occur while solving an ill-conditioned linear system might produce an erroneous result.

To control the condition of the lower hull $\widehat{A}$, we present the *flattening* of a regular subdivision. This involves a modification of the lifting function and provides an extension of the basic version of the dynamic lifting algorithm, as described in Algorithm 2.9.

Definition 2.20. A cell $\widehat{C}$ is called *flat*, if all lifting values are zero. A subdivision is called *flat*, if all its cells are flat.

The subdivision induced by a zero lifting consists of one cell with normal $(0, \ldots, 0, 1)$. A subdivision S induced by a conservative lifting function is associated with zero lifting and this lifting is used for the following steps of the dynamic lifting algorithm. This process is called *flattening* a subdivision.

As flattening is performed several times a list of regular triangulations $S_0, S_1, S_2, \ldots$, where S_i is a regular subdivision of the set $A^{(i)}$ such that $\omega^i_{|A^{(i-1)}} \equiv 0$ and S_{i-1} is a subdivision of one cell $(= A^{(i-1)})$ of the subdivision S_i, is obtained. This can be interpreted as reversing the order of refinement, as defined in [35].

All S_i together form the subdivision which is computed with the pivoting algorithm together with the variant of checking normals. Usually the other variants using the data structure do the same. However, in very special situations these variants compute an even finer subdivision. This detail is discussed in Section 3.7 as well. We assume that this final subdivision is induced by a lifting function, but because of the flattening this lifting is never computed explicitly.

From an implementational point of view the flattening only changes the lifting values, but all old cells remain stored. The pivoting algorithm, presented in the previous section, only requires the lifting to be conservative and has no other demands on the lifting function. So, flattening does not alter the algorithm for computing new cells.

Concerning the flattening mechanism, there is a bound on the maximal lifting value that needs to be set. Each time this bound is exceeded after determining the value of the conservative lifting function for the new point, the whole subdivision will be flattened. For reasons of simplicity, this bound can be set to one. The second alternative is to invoke

the flattening automatically, when numerical problems occur during the update. With an exception handling mechanism like in Ada this can be implemented in a straightforward way. The third alternative is to flatten after each update, so that a lifting value equal to one can be taken for each new point.

2.7. *Complexity and Cost Estimates*

The complexity model used is the binary Turing machine, see [21] for complexity theory. The complexity of volume and mixed-volume computation is known to be #P-hard, see [13], [14], and [25]. In the theory of computational convexity, the following important result has been derived:

Proposition 2.21 [25]. *When the dimension n is fixed, the volume of a polytope can be computed in polynomial time in the input size.*

The idea of this result is based on linear programming, which can be used to derive, in polynomial time, a description of the polytope in terms of an inequality system, representing its facets. Once the facets are given in this representation, by an enumeration of the facets with respect to a common point of the polytope, the volume can be computed, straightforwardly by calculating determinants when the polytope is simplicial. Note that it is crucial here to consider n as a constant number.

In this section the cost for the optimal, average, and worst case is measured by counting the number of arithmetical operations in the important steps of the dynamic lifting algorithm. Fixed-point arithmetic is assumed throughout. Evidence is provided that the algorithm runs in polynomial time. The results are interpreted empirically in Section 4.3.1.

Lemma 2.22. *Given a finite set of points $A \subset \mathbb{E}^n$. The computation of an initial cell requires $O(n^3)$ operations.*

Proof. Computing an initial cell is equivalent to computing n linearly independent vectors, which can be brought back to the triangulation of a matrix with n rows. □

For the following, it is assumed that the problem is nondegenerate. Here we denote $A' = A \backslash C$, where C is an initial cell.

Lemma 2.23. *The cost of checking whether a point belongs to a triangulation Δ requires at least $O(n^2)$, at most $O((\#\Delta)n^2)$, and on average $O((\#\Delta)n)$ arithmetical operations.*

Proof. As the factorization matrices are stored, only the solving of a triangular system is required for each cell. This requires $O(n^2)$ arithmetical operations for one cell. In the optimal case the decomposition (see Lemma 2.17) of one cell can suffice, while in the worst case all cells need to be considered, which takes time $O((\#\Delta)n^2)$. In the average

case assume a uniform distribution of the cells in Δ and start the walk (see Fig. 5) at the center of Δ. Each path only goes through one facet of the starting cell. Hence, the path will on average only cross $(1/n)$th of the cells in Δ, which takes time $O((\#\Delta)n)$. □

Lemma 2.24. *The cost of adding a point* $\mathbf{x}$ *to a triangulation* Δ *requires at least* $O((\#\Delta^{\mathbf{x}})n^3 + n^2)$, *at most* $O((\#\Delta^{\mathbf{x}})n^3 + (\#\Delta)n^2)$, *and on average* $O((\#\Delta^{\mathbf{x}})n^3 + (\#\Delta)n)$ *arithmetical operations, with* $\Delta^{\mathbf{x}}$ *the collection of new cells.*

Proof. Factorization matrices are computed only once, each time a new cell is constructed. So, for all cases, time $O((\#\Delta^{\mathbf{x}})n^3)$ is needed. In the optimal case only one decomposition has to be computed, which takes time $O(n^2)$, while for the worst case all cells in Δ need to be considered, which takes time $O((\#\Delta)n^2)$. The average cost bound is derived by application of Lemma 2.23. □

Theorem 2.25. *The construction of a placeable triangulation* Δ *of a point set* $A \subset \mathbb{E}^n$ *takes at least* $O((\#\Delta)n^3 + (\#A')n^2)$, *at most* $O((\#\Delta)n^3 + (\#A')(\#\Delta)n^2)$, *and on average* $O((\#\Delta)n^3 + (\#A')(\#\Delta)n)$ *arithmetical operations.*

Proof. Factorization matrices are computed only once, each time a new cell is constructed. So, for both cases, time $O((\#\Delta)n^3)$ is needed. In the optimal case, at each step, at least one decomposition needs to be computed. Because there are $\#A'$ steps in the algorithm, this takes time $O((\#A')n^2)$. Hence $O((\#\Delta)n^3 + (\#A')n^2)$ is obtained as the cost in the optimal case. In the worst case, at each step, for $j = 1, 2, \ldots, \#A'$, all cells in the corresponding triangulation $\Delta^{(j)}$ need to be considered. Applying Lemma 2.24 for each step, a total number of $O(\sum_{j=1}^{\#A'}(\#\Delta^{(j)})n^2)$ arithmetical operations is needed. As $\#\Delta^{(j)} \leq \#\Delta$, for $j = 1, 2, \ldots, \#A'$, $O((\#\Delta)n^3 + (\#A')(\#\Delta)n^2)$ is obtained as a bound for the cost in the worst case. In the average case application of Lemma 2.23 yields $O((\#\Delta)n^3 + (\#A')(\#\Delta)n)$. □

Theorem 2.25 indicates the bottleneck of the algorithm: determining the cells in Δ for which pivoting yields new cells. All cost bounds contain three important factors which influence the general cost of the algorithm. On the input side we have the number of points $\#A'$ and the dimension n of the problem. On the output side the complexity of the facet structure of the lifted polytope plays a role, as the number of cells $\#\Delta$ is also taken into account. This number of cells can be influenced by the choice of the lifting function. It is natural to assume that a random lifting will induce a triangulation with an average number of cells. In the dynamic lifting algorithm a random placeable triangulation is obtained by adding the points in a random order.

Note that the number of cells can grow exponentially, and $\#\Delta \gg n^3$, with the bound for $\#\Delta$ given by $O((\#A)^{\lfloor n/2 \rfloor})$, see p. 92 of [43] for more precise bounds on the number of facets. Still, the bound is polynomial in the output size. The cost with respect to the space of the dynamic lifting algorithm is proportional to the number of cells, as the list of cells is maintained during the computations.

3. Dynamic Construction of Regular Simple Mixed Subdivisions

This part deals with the application of the dynamic lifting algorithm to the general case. Therefore we first need to extend the definitions of Section 2.1 to the case of several polytopes. Algorithm 2.9 can be applied in a straightforward manner, by means of the Cayley trick, as explained in the second section. However, the other sections consider a more complicated generalization of Algorithm 2.9. The third section deals with the degeneracy check by the computation of an initial mixed cell. In the fourth section the idea of Betke is elaborated explicitly with fans, which leads to a powerful computational tool for the computation of the mixed volume by means of static lifting. Conservative lifting functions are extended to the general case in the fifth section. In the sixth section the connectivity of the mixed cells is conjectured and the main kernel of the dynamic lifting algorithm is presented. The idea of unfolding cells with the same normal is described in the seventh section. The computational complexity of the problem and cost estimates of the algorithm are treated in the last section of this part.

3.1. *Regular Simple Mixed Subdivisions*

Definition 3.1. Let $\mathcal{A}$ be a tuple of point sets with a respective tuple of polytopes $\mathcal{P}$. Assume that the sets in $\mathcal{A}$ are ordered in the following way:

$$\begin{aligned} \mathcal{A} &= (\underbrace{A_1, \ldots, A_1}_{k_1}, \underbrace{A_2, \ldots, A_2}_{k_2}, \ldots, \underbrace{A_r, \ldots, A_r}_{k_r}) \\ \mathcal{P} &= (\overbrace{P_1, \ldots, P_1}^{k_1}, \overbrace{P_2, \ldots, P_2}^{k_2}, \ldots, \overbrace{P_r, \ldots, P_r}^{k_r}) \end{aligned} \quad \text{with} \quad \sum_{i=1}^{r} k_i = n. \tag{9}$$

The tuples $\mathcal{A}$ and $\mathcal{P}$ are said to be *unmixed* when $r = 1$, *semimixed* when $1 < r < n$, and *fully mixed* when $r = n$.

Let $C = (C_1, C_2, \ldots, C_r)$, $C_i \subset A_i$, be a cell. The volume of C is written as $\mathrm{vol}_n(C) = \mathrm{vol}_n(\mathrm{conv}(C))$ where the following conventions are used:

$$\mathrm{conv}(C) = \mathrm{conv}(C_1 + C_2 + \cdots + C_r) \subset \mathbb{E}^n, \tag{10}$$

$$\mathrm{type}(C) = (\dim(C_1), \dim(C_2), \ldots, \dim(C_r)) \in \mathbb{N}^r. \tag{11}$$

Given a tuple of point sets, based on a subdivision of the Minkowski sum, Definition 2.4 can be extended in the following way, see [30]:

Definition 3.2. Assume the union of the sets A_i in $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ affinely spans $\mathbb{E}^n$. A *subdivision* of $\mathcal{A}$ is a collection $S = \{C_1, \ldots, C_m\}$ of m cells $C_j = (C_{j1}, C_{j2}, \ldots, C_{jr})$, $C_{ji} \subset A_i$, satisfying:

1. $\dim(C_j) = n$ for $j = 1, \ldots, m$.
2. $\mathrm{conv}(C_j) \cap \mathrm{conv}(C_k)$ is a proper common face of $\mathrm{conv}(C_j)$ *and* $\mathrm{conv}(C_k)$, $j \neq k$.
3. $\bigcup_{j=1}^{m} \mathrm{conv}(C_j) = \mathrm{conv}(\mathcal{A})$.

The subdivision is called *mixed* if the following additional property holds:

4. $\sum_{i=1}^{r} \dim(C_{ji}) = n$ for all cells $C_j \in S$.

The subdivision is called *fine mixed* if:

5. $\sum_{i=1}^{r} (\#(C_{ji}) - 1) = n$ for all cells $C_j \in S$.

Note that the fifth item in Definition 3.2 represents an additional property, not implied by the previous ones.

Definition 3.3. A cell C is called *mixed* when it has a contribution to the mixed volume.

If the subdivision is mixed, then all cells C with $\text{type}(C) = (k_1, \ldots, k_r)$ are mixed, see [30]. For computation of the mixed volume, it is sufficient that the collection of the mixed cells is fine mixed, which is weaker than the last part of Definition 3.2.

Definition 3.4. A mixed subdivision is called a *simple mixed subdivision* when, for all mixed cells C_j, $\sum_{i=1}^{r} (\#(C_{ji}) - 1) = n$.

In Example 3.7 a regular simple mixed subdivision is given for which the subdivision is not fine mixed.

Definition 3.5. A cell C which is mixed and satisfies $\#C_i = k_i + 1, i = 1, \ldots, r$, is called *simple mixed.*

Definition 3.6. A subdivision S of a tuple $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ is called *regular* if there is a tuple ω of lifting functions, $\omega = (\omega_1, \omega_2, \ldots, \omega_r)$, so that the cells of S are the facets of the lower hull of $\sum_{i=1}^{r} \text{conv}(\widehat{A}_i)$.

As before, a regular subdivision induced by a lifting ω is denoted by S_ω and the cells by C_γ, as they are facets $\partial_\gamma(\sum_{i=1}^{r} \text{conv}(\widehat{A}_i))$ characterized by their inner normal γ. In [48] a mixed subdivision induced by a lifting function is called a *coherent mixed subdivision* (*CMD*) and a regular fine mixed subdivision is called a *tight coherent mixed subdivision* (*TCMD*). For the relationship with fiber polytopes, see [5].

Given a mixed subdivision, it is sufficient to consider only the mixed cells, as shown in [30], to compute the mixed volume:

$$V_n(\mathcal{P}) = \sum_{\substack{C \in \Delta_\omega \\ \text{type}(C) = (k_1, \ldots, k_r)}} k_1! \cdots k_r! \ \text{vol}_n(C). \tag{12}$$

Given a simple mixed cell $C \in S$, $C = (C_1, C_2, \ldots, C_r)$, with $C_i = \{\mathbf{c}_{0i}, \mathbf{c}_{1i}, \ldots, \mathbf{c}_{k_i i}\}$, $i = 1, 2, \ldots, r$. Its volume can be computed by

$$\text{vol}_n(C) = \frac{1}{k_1! \cdots k_r!} |\det(\mathbf{c}_{11} - \mathbf{c}_{01}, \ldots, \mathbf{c}_{1i} - \mathbf{c}_{0i}, \ldots, \mathbf{c}_{k,i} - \mathbf{c}_{0i}, \ldots, \mathbf{c}_{k,r} - \mathbf{c}_{0r})|. \tag{13}$$

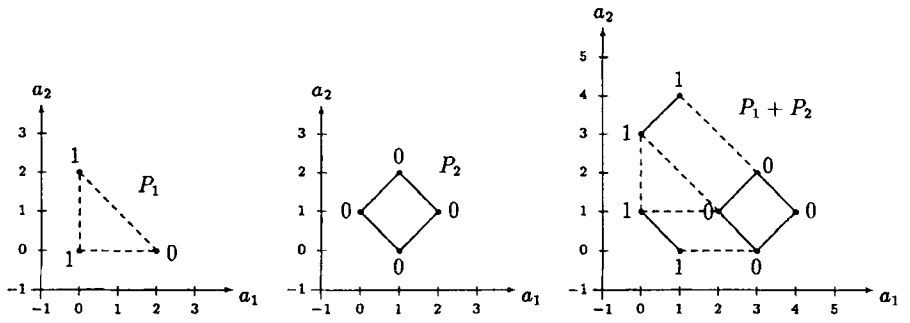


Fig. 6. The polytopes P_1 and P_2 with their sum. The big numbers indicate the values of the lifting function. The mixed cells can be found as parallelograms (with one side dashed and one side solid) in the subdivision of $P_1 + P_2$.

In case the subdivision is not mixed, a recursive scheme as presented in [50] can be applied for computation of the mixed volume.

The following example illustrates the definitions.

Example 3.7. Consider the following tuple of lifted point sets:

$$\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2) = \left(\left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \right\} \right).$$

The polytopes in the tuple $\mathcal{P} = (P_1, P_2)$ and the sum $P_1 + P_2$ are shown in Fig. 6. The two mixed cells in the subdivision of $\widehat{\mathcal{A}}$ are

$$\left(\left\{ \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \right\} \right)$$

and

$$\left(\left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right),$$

with respective inner normals $\gamma_1 = (1, -1, 4)$ and $\gamma_2 = (1, 1, 2)$. The other two unmixed cells are

$$\left(\widehat{A}_1, \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right) \quad \text{and} \quad \left(\left\{ \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \right\}, \widehat{A}_2 \right),$$

with respective inner normals $\gamma_3 = (1, 0, 2)$ and $\gamma_4 = (0, 0, 1)$. This subdivision is not fine mixed as the second component of the last cell contains too many points. Note that by giving the rightmost point of P_2 lifting value > 0, the subdivision becomes fine mixed.

However, we only need to have a simple mixed subdivision and the mixed volume can be computed by

$$V_n(\mathcal{P}) = 1! \cdot 1! \frac{1}{1! \cdot 1!} \operatorname{vol}_2(C_{\gamma_1}) + 1! \cdot 1! \frac{1}{1! \cdot 1!} \operatorname{vol}_2(C_{\gamma_2}) = 4 + 2 = 6.$$

In Fig. 6 in the subdivision of $P_1 + P_2$ the unmixed cells can be found as the cells with respectively either all sides dashed or all sides solid. Their volumes equal respectively $\operatorname{vol}_2(P_1) = 4/2!$ and $\operatorname{vol}_2(P_2) = 4/2!$. Adding the volumes of all cells yields $\operatorname{vol}_2(P_1 + P_2) = 10$. Note how formula (1) can be applied: $V_n(\mathcal{P}) = \operatorname{vol}_2(P_1 + P_2) - \operatorname{vol}_2(P_1) - \operatorname{vol}_2(P_2) = 10 - 2 - 2 = 6$.

3.2. *The Cayley Trick*

In this section Algorithm 3.8 presents a geometric description of the so-called *Cayley trick*. This geometric description is due to Sturmfels, as mentioned in [29]. See [22], [23], and [31] for other references.

Algorithm 3.8. The Cayley trick:

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_r)$, $\sum_{i=1}^{r} k_i = n$. *a tuple of point sets*

Output: Δ_ω. *a regular fine-mixed subdivision of* $\mathcal{A}$

$\widetilde{A}_i := \left\{ \begin{pmatrix} \mathbf{a} \\ \mathbf{e}_{i-1} \end{pmatrix} \;\middle|\; \mathbf{a} \in A_i \right\}$; *construct extended point sets with* $\mathbf{e}_0 = \mathbf{0}$ *and* $\mathbf{e}_i = i$th *unit vector in* $\mathbb{E}^{r-1}$

$\lhd_\omega := \operatorname{Triang}\left(\operatorname{conv}\left(\bigcup_{i=1}^{r} \widetilde{A}_i\right)\right)$; *construct a regular triangulation*

$\Delta_\omega := \{C \mid \exists \widetilde{C} \in \lhd_\omega : \pi(\widetilde{C}) = C\}$. π *projects to original coordinates*

An example for the first step of Algorithm 3.8 is given in Fig. 7.

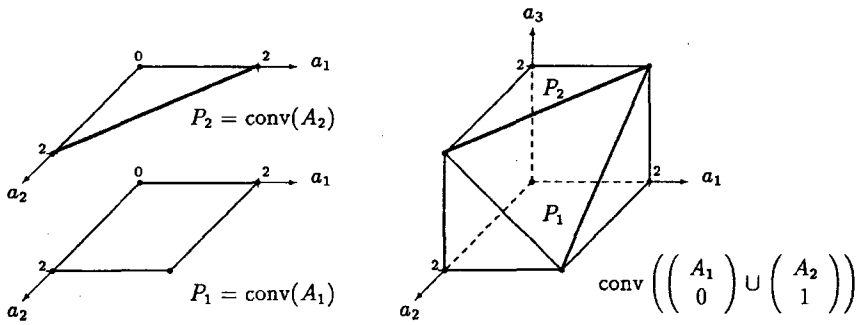


Fig. 7. The polytopes P_1 and P_2 are shown on the left. On the right is the big polytope as the convex hull of the extended point sets.

Denote $\widehat{\pi}$ as the projection map that forgets the additional coordinates introduced by the Cayley trick, but which keeps the lifting coordinate. Following the notations of Algorithm 3.8, we state the following proposition.

Proposition 3.9. *Let $\widehat{\lhd_\omega}$ be a regular triangulation of $\widetilde{\mathcal{A}}$. Then $\widehat{\Delta_\omega} = \widehat{\pi}(\widehat{\lhd_\omega})$ is a regular fine-mixed subdivision of $\mathcal{A}$.*

Proof. The projection $\widehat{\pi}$ copies the lifting ω on $\widetilde{\mathcal{A}}$ onto $\mathcal{A}$. As the extended coordinates are the same for all points which belong to the same component, there is a one-to-one correspondence between the cells $\widehat{C}_\gamma \in \widehat{\lhd_\omega}$ and $\widehat{\pi}(\widehat{C}_\gamma) \in \widehat{\Delta}_\omega$. The property of fine mixed is obtained because $\lhd_\omega$ is a triangulation. □

In the following sections we describe an approach to avoid calculation of the cells that do not contribute to the mixed volume.

3.3. *Computation of an Initial Mixed Cell*

The aim is to detect the degeneracy $V_n(\mathcal{P}) = 0$ by computing a mixed cell

$$C = (C_1, C_2, \ldots, C_r),$$

with $C_i = \{\mathbf{c}_{0i}, \mathbf{c}_{1i}, \ldots, \mathbf{c}_{k_i i}\} \subset A_i$. A straightforward approach could be to apply Algorithm 2.10 to compute, from each set A_i in $\mathcal{A}$, k_i linearly independent points with respect to a common origin. However, this approach is only sure to work when, for each A_i, $\dim(A_i) = n$, as is illustrated in the following example.

Example 3.10. Consider the following tuple $\mathcal{A} = (A_1, A_2)$, $n = 2 = r$:

$$\mathcal{A} = \left(\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \right). \tag{14}$$

The first two points of A_1 should not be taken, because then there is no other linearly independent point left in A_2 to choose. Therefore A_2 must be considered first.

Ordering the sets according to their dimension is only enough whenever the A_i's lie in complementary affine spaces. However, for the applications we have in mind, the sparsity of the vectors also plays an important role, as is illustrated in the following example.

Example 3.11. Consider the following tuple $\mathcal{A} = (A_1, A_2)$, $n = 3$, $r = 2$, with the type of mixture given by (1, 2):

$$\mathcal{A} = \left(\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\} \right). \tag{15}$$

Both sets have the same dimension. The last point of the first set A_1 must be chosen, otherwise no initial mixed cell can be found.

The instrument to deal with sparse vectors is provided in the following definition.

Definition 3.12. Given a set $A \subset \mathbb{E}^n$. Consider $i \in \{1, 2, \ldots, n\}$. The *occurrence of i with respect to A* is $\text{occ}(i, A) = \#\{\mathbf{a} \in A | a_i \neq 0\}$. Consider $\mathbf{x} \in \mathbb{E}^n$. The *occurrence of* $\mathbf{x}$ *with respect to A* is $\text{occ}(\mathbf{x}, A) = \min_{x_i \neq 0} \text{occ}(i, A)$, if $\mathbf{x} \neq \mathbf{0}$ whereas $\text{occ}(\mathbf{0}, A) = \#A+1$. This extends to the tuple of sets $\mathcal{A} = (A_1, A_2, \ldots, A_r)$: the *occurrence of i with respect to* $\mathcal{A}$ is $\text{occ}(i, \mathcal{A}) = \sum_{j=1}^{r} \text{occ}(i, A_j)$ and the *occurrence of* $\mathbf{x}$ *with respect to* $\mathcal{A}$ is $\text{occ}(\mathbf{x}, \mathcal{A}) = \sum_{j=1}^{r} \text{occ}(\mathbf{x}, A_j)$.

Example 3.13. Consider again the tuple $\mathcal{A}$ of Example 3.11. The vector (8 2 1 8 2 1) gives the respective occurrences of the respective vectors with respect to $\mathcal{A}$, as listed in the order in which they appear in (15), i.e., $\text{occ}((0\ 0\ 0)^t, \mathcal{A}) = 8$, $\text{occ}((1\ 0\ 0)^t, \mathcal{A}) = 2$, etc.

The key step for an algorithm for computing an initial mixed cell is presented in Algorithm 3.14.

Algorithm 3.14. Computation of the ith component of an initial mixed cell:

Input: i, k_i, — *current component*
$E \subseteq \bigcup_{j=1}^{i-1} A_j$, — *set of chosen points*
$\mathcal{R} = (A_i, \ldots, A_r)$ with — *remainder of* $\mathcal{A}$
$\begin{cases} \mathbf{0} \in A_j, \ \#A_j = \dim(A_j) + 1, \\ \dim(A_j \cup E) \leq \dim(A_{j+1} \cup E). \end{cases}$ — *invariant conditions*
Output: C_i, — *i*th *component of cell*
$E, \mathcal{R}$. — *updated sets*

$C_i := \{\mathbf{0}\}$; — *initialization*
for j from 1 to k_i do
  choose $\mathbf{x} \in A_i$ such that
    (1) $\dim(E \cup \{\mathbf{x}\}) > \dim(E)$ — *linearly independent*
    (2) $\mathbf{x}$ has minimal occurrence with respect to $\mathcal{R}$ — *take sparsity into account*
      of all possible choices which satisfy (1);
  exit when no such $\mathbf{x}$ can be found;
  $C_i := C_i \cup \{\mathbf{x}\}$; $E := E \cup \{\mathbf{x}\}$; $R_i := R_i \backslash \{\mathbf{x}\}$; — *update* C_i, E *and* $\mathcal{R}$
end for.

Algorithm 3.14 has to be applied r times, each time respecting the invariant conditions given in the input specification. When the algorithm terminates with a component C_i, $\#C_i \leq k_i$, then no linearly independent points could be found and the problem is degenerate. Proposition 3.15 provides a formal guarantee for Algorithm 3.14. The proof interprets Algorithm 3.14 as the computation of one nonzero term in the expansion of the permanent of a matrix in $\mathbb{Z}_2^{n \times n}$. See [53] for an efficient algorithm for computing permanents of degree matrices.

Proposition 3.15. *If Algorithm* 3.14 *yields a* C_i *with* $\#C_i \leq k_i$, *then* $V_n(\mathcal{P}) = 0$.

Proof. Without loss of generality, the sparsest case can be focused, i.e., all vectors in $\mathcal{A}$ are unit vectors, denoted by $\mathbf{e}_i$. For notational convenience, $r = n$ is assumed. With $\mathcal{A}$ the following matrix M can be associated: $M_{ij} = 1$, if $\mathbf{e}_j \in A_i$, otherwise $M_{ij} = 0$. So, the ith row of M represents A_i, for $i = 1, 2, \ldots, n$. Then $V_n(\mathcal{P}) = 0 \Leftrightarrow \text{per}(M) = 0$. When Algorithm 3.14 is iterated on each row, it picks from each row exactly one element on that column for which in the following rows there are a maximum number of zeros. Only when $\text{per}(M) = 0$ will the algorithm not find enough nonzero elements. □

By computing the vertex set of each separate point set in the tuple, those points which will certainly not influence the mixed volume, regardless of the subdivision used to compute it, can also be eliminated.

3.4. *Normal Fans and Mixed Cells*

Powerful tools to investigate the combinatorial structure of polytopes and subdivisions are *fans*. Based on this abstraction, we present a major algorithmic cornerstone of both the static and dynamic lifting algorithm. General definitions can be found in Lecture 7 of [56]. Here we reformulate the definitions using our notations.

Definition 3.16. Let $A \subset \mathbb{E}^n$, then

$$K(A) = \{\gamma \in \mathbb{E}^n \mid \forall \mathbf{x}, \mathbf{y} \in A\colon \langle \mathbf{x}, \gamma \rangle = \langle \mathbf{y}, \gamma \rangle\}$$

is *the normal cone*[1] *on* A. Let $A \subset B$, then

$$K(A, B) = \{\gamma \in K(A) \mid \forall \mathbf{y} \in B \backslash A, \forall \mathbf{x} \in A\colon \langle \mathbf{y}, \gamma \rangle < \langle \mathbf{x}, \gamma \rangle\}$$

is *the normal cone on* A *with respect to* B.

Definition 3.17. Let $A \subset \mathbb{E}^n$, then

$$\mathcal{N}^j(A) = \{K(\partial_\gamma A, A) \mid \dim(\partial_\gamma A) = j\}$$

is *the normal cone complex of the* j*-faces* $\partial_\gamma A$ *of* $\text{conv}(A)$. If A is a lifted polytope, the normal cone complex of the j-faces of the lower hull of $\text{conv}(A)$ will be denoted by $\mathcal{N}^j_\vee(A)$. The *normal fan* $\mathcal{N}(A)$ of A is the set of all normal cone complexes $\mathcal{N}^j(A)$, for $j = 0, 1, \ldots, n$.

As the definition is given for point sets, it extends naturally to polytopes. We refer to p. 193 of [56] for an alternative definition of normal fans.

Definition 3.18. Given two normal cone complexes $\mathcal{F}$ and $\mathcal{G}$. Their *common refinement* is defined as

$$\mathcal{F} \wedge \mathcal{G} := \{K \cap K' \mid K \in \mathcal{F},\ K' \in \mathcal{G}\}.$$

The *root of a normal cone complex* $\mathcal{F}$ is defined by $\bigcap_{K \in \mathcal{F}} K$.

---

[1] Cone = Kegel, both in German and Dutch.

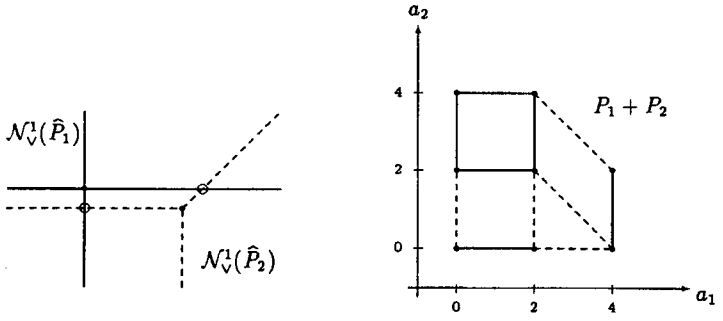


Fig. 8. The normal complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^1(\widehat{P}_2)$, with P_1 and P_2 as in Fig. 7. The circles around the intersection points indicate the mixed cells.

In [4] Betke proposed a new idea for the computation of mixed volumes based on lifting and normal fans. See also Lemma 2.1.5 of [25]. In [45] this idea has been generalized to more than two polytopes. Note that the idea of intersecting normal cones also appeared in [8]. Here we reformulate the idea in the following theorem.

Theorem 3.19. *Given* $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$, *a tuple of lifted point sets, with* $\sum_{i=1}^r k_i = n$, *see* (9). *The normals in the common refinement* $\bigwedge_{i=1}^r \mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ *of the normal cone complexes of the* k_i*-faces of the lower hull of* $\operatorname{conv}(\widehat{A}_i)$ *are the outer normals of the facets of the lower hull of* $\operatorname{conv}(\widehat{A}_1 + \widehat{A}_2 + \cdots + \widehat{A}_r)$. *These facets determine the mixed cells in the subdivision of* $\mathcal{A}$.

Example 3.20. Theorem 3.19 is illustrated in Fig. 8, on the same polytopes that have been used to illustrate the Cayley trick. The lifting is assumed to be linear. The lifted square $\widehat{P}_1$ has four 1-faces. The corresponding normal cones are two-dimensional and share the one-dimensional normal cone $K(\widehat{P}_1)$ which defines the root, drawn as a point. For the triangle there are three normal cones of 1-faces. They are two-dimensional and have one common one-dimensional normal cone, drawn as a point. The intersection points in Fig. 8 represent the normals that span the cones of $\mathcal{N}_\vee^1(\widehat{P}_1) \wedge \mathcal{N}_\vee^1(\widehat{P}_2)$. These normals define the mixed cells in the subdivision.

In [10], [18], and [30] the idea of Betke has been used to compute all mixed cells. Here the algorithm suggested by Theorem 3.19, and in [10] and [18] described as the lift-and-prune algorithm, is elaborated and presented to prepare the dynamic lifting algorithm.

Given a tuple of lifted point sets $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$, any lifted cell $\widehat{C}_\gamma$ of a regular subdivision can be characterized by its inner normal as

$$\widehat{C}_\gamma = (\partial_\gamma \widehat{A}_1, \partial_\gamma \widehat{A}_2, \ldots, \partial_\gamma \widehat{A}_r). \tag{16}$$

Since $\operatorname{conv}(\widehat{C}_\gamma) = \operatorname{conv}(\sum_{i=1}^r \partial_\gamma \widehat{A}_i)$ is a facet of the lower hull, the functional $\langle \cdot, \gamma \rangle$

attains its minimum over $\widehat{A}_i$ at $\partial_\gamma \widehat{A}_i$, i.e.,

$$\langle \widehat{\mathbf{a}}, \gamma \rangle = \langle \widehat{\mathbf{b}}, \gamma \rangle, \qquad \forall \widehat{\mathbf{a}}, \widehat{\mathbf{b}} \in \partial_\gamma \widehat{A}_i, \quad i = 1, 2, \ldots, r, \tag{17}$$

$$\langle \widehat{\mathbf{a}}, \gamma \rangle > \langle \widehat{\mathbf{b}}, \gamma \rangle, \qquad \forall \widehat{\mathbf{a}} \in \widehat{A}_i \backslash \partial_\gamma \widehat{A}_i, \quad \forall \widehat{\mathbf{b}} \in \partial_\gamma \widehat{A}_i, \quad i = 1, 2, \ldots, r. \tag{18}$$

Algorithm 3.21 presents a way to compute all mixed cells by searching for feasible solutions to the constraints (17) and (18). The algorithm enumerates all possible combinations of k_i-faces, with proper feasibility tests to limit the search space. The order of enumeration is organized so that mixed cells which share some faces also share a part of the factorization work to be done to solve the system defined by (17).

Conditions (17) and (18) are used in the following way. After choosing a k_i-face $\widehat{C}_i = \{\widehat{\mathbf{c}}_{0i}, \widehat{\mathbf{c}}_{1i}, \ldots, \widehat{\mathbf{c}}_{k_i i}\}$ of $\widehat{A}_i$, it is checked whether $(\widehat{C}_1, \ldots, \widehat{C}_i)$ can lead to a mixed cell in the induced subdivision. In the ith step we already have the upper triangular matrix M_1, with κ rows, originating from the vectors $(\widehat{\mathbf{c}}_{j\nu} - \widehat{\mathbf{c}}_{0\nu})^t$, $\nu = 1, 2, \ldots, i-1$, $j = 1, 2, \ldots, k_\nu$, with $\kappa = \sum_{\nu=1}^{i-1} k_\nu$. Then this matrix M_1 is extended with new rows defined by the vectors $(\widehat{\mathbf{c}}_{ji} - \widehat{\mathbf{c}}_{0i})^t$, $j = 1, 2, \ldots, k_i$. By making row combinations and pivoting, we obtain $M_1 := L \cdot M_1$ with entries $M_1 = (m_{kl})$, such that $m_{\mu l} = 0$, $\mu = \kappa + 1, \ldots, \kappa + k_i$, $l = 1, 2, \ldots, \kappa$. A unimodular transformation U yields an upper triangular form: $M_1 := U \cdot M_1$. Condition (17), with $\nu = 1, 2, \ldots, i$, is then equivalent to $M_1 \gamma = \mathbf{0}$. If $M_1 \gamma = \mathbf{0}$ implies $\gamma_{n+1} = 0$, then $\mathrm{conv}(\widehat{C}_1) + \cdots + \mathrm{conv}(\widehat{C}_i)$ lies in a hyperplane perpendicular to the lifting axis and hence $(\widehat{C}_1, \ldots, \widehat{C}_i)$ cannot be part of any mixed cell. This concludes the first feasibility test.

For the second feasibility test, the rows of the matrix M_2 are extended with the vectors $(\widehat{\mathbf{a}}_i - \widehat{\mathbf{c}}_{0i})^t$, $\widehat{\mathbf{a}}_i \in \widehat{A}_i \backslash \widehat{C}_i$, since, in (18), it is sufficient to consider only $\widehat{\mathbf{b}} = \widehat{\mathbf{c}}_{0i}$. In the ith step we use $M_1 \gamma = 0$ to eliminate $\kappa + k_i$ unknowns, which reduces the dimension of the space of inequalities to $n - \kappa - k_i$. If the system of inequalities $M_2 \gamma \geq \mathbf{0}$ implies $-\gamma_{n+1} \geq 0$, then $\mathrm{conv}(\widehat{C}_1) + \cdots + \mathrm{conv}(\widehat{C}_i)$ is not a lower facet of $\mathrm{conv}(\widehat{A}_1 + \cdots + \widehat{A}_i)$ and hence $(\widehat{C}_1, \ldots, \widehat{C}_i)$ cannot be part of any mixed cell. The test whether $M_2 \gamma \geq \mathbf{0} \Rightarrow -\gamma_{n+1} \geq 0$ is equivalent to determining whether the vector $(0, \ldots, 0, -1)$ belongs to the cone spanned by the vectors in the columns of M_2. The Farkas lemma (see, e.g., [56]) deals with this problem and can be worked out by linear programming.

Algorithm 3.21. Shared factorizations subject to inequality constraints:

Input: $(k_1, k_2, \ldots, k_r)$, $n = \sum_{i=1}^{r} k_i$, *type of mixture*
$(\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$, *lifted point sets*
$(\widehat{\mathfrak{F}}_1, \widehat{\mathfrak{F}}_2, \ldots, \widehat{\mathfrak{F}}_r)$. *$k_i$-faces of lower hull of* $\mathrm{conv}(\widehat{A}_i)$
Output: $\widehat{\mathfrak{S}}_\omega = \{\widehat{C} \in \widehat{S}_\omega \mid V_n(C) > 0\}$. *collection of lifted mixed cells*

At level i, $1 \leq i < r$:
DATA and INVARIANT CONDITIONS:

(M_1, κ): $M_1 \gamma = \mathbf{0} \not\Rightarrow \gamma_{n+1} = 0$, *equalities* (17)
$\kappa = \sum_{j=1}^{i-1} k_j$ *upper triangular up to row* κ
(M_2, κ): $M_2 \gamma \geq \mathbf{0} \not\Rightarrow -\gamma_{n+1} \geq 0$ *inequalities* (18)
$\dim(M_2) = n - \kappa$ *still feasible and reduced*

ALGORITHM:
  for each $\widehat{C}_i \in \widehat{\mathfrak{F}}_i$ do — *enumerate over all k_i-faces*
    Triangulate$(M_1, \kappa, \widehat{C}_i)$; — *ensure invariant conditions*
    if $M_1\gamma = 0 \not\Rightarrow \gamma_{n+1} = 0$ — *test for feasibility with respect to* (17)
    then *Eliminate*$(M_1, M_2, \kappa, \widehat{C}_i, \widehat{A}_i)$; — *eliminate unknowns*
      if $M_2\gamma \geq \mathbf{0} \not\Rightarrow -\gamma_{n+1} \geq 0$ — *test for feasibility with respect to* (18)
        then proceed to next level $i + 1$;
      end if;
    end if;
  end for.
At level $i = r$:
  Compute γ: $M_1\gamma = \mathbf{0}$;
  Merge the new cells with the list $\widehat{\mathfrak{S}}_\omega$.

Note that (18) had to be weakened to $\geq$ type inequalities in order to be able to compute also subdivisions that are not mixed. This also explains the merge operation at the end. The feasibility tests in the algorithm allow an efficient computation of the mixed cells. As an alternative to these feasibility tests we refer to similar criteria which can be verified by means of linear programming, as presented by Emiris and Canny in [18].

In Algorithm 3.21 a lot of face–face combinations that do not lead to mixed cells need to be tested. For semimixed tuples of polytopes, it might often be beaten by the Cayley trick and it is certainly not the appropriate tool for the unmixed case. The dynamic lifting algorithm applies Algorithm 3.21 with a relatively small input set, which assures its efficiency.

3.5. *Conservative Lifting Functions Applied to Mixed Cells*

A linear lifting function ω is face structure preserving, i.e., there is a one-to-one correspondence between the faces $\partial_\gamma A_i$ of A_i and the faces of the lower hull of $\widehat{A}_i$. For a generic choice of linear ω a mixed subdivision $\mathcal{A}$ is induced. However, in the context of dynamic lifting a nonlinear lifting is more appropriate.

Analogously to Definition 2.14, we define conservative lifting functions. Denote $\widehat{\mathfrak{S}}_\omega$ as the collection of mixed cells in the regular subdivision of $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ and $\widehat{S}_\omega^{(i)}$ as the corresponding regular subdivision of A_i, $i = 1, 2, \ldots, r$.

Definition 3.22. Let $\widehat{C}_\gamma = (\widehat{C}_1, \widehat{C}_2, \ldots, \widehat{C}_r)$ be a cell. Consider a point $\mathbf{x}$ with respect to $\widehat{C}_i$. Let $\omega(\mathbf{x}) = \widehat{x}_{n+1}$ so that $\langle(\widehat{\mathbf{x}}, \gamma\rangle > \langle\widehat{C}_i, \gamma\rangle$, then ω is called a *conservative lifting with respect to the* ith *component of* $\widehat{C}_\gamma$. If ω is a conservative lifting with respect to all cells in $\widehat{S}_\omega^{(i)}$ and with respect to the ith component of all cells in $\widehat{\mathfrak{S}}_\omega$, then ω is called a *conservative lifting with respect to* $\widehat{S}_\omega^{(i)}$ *and with respect to the* ith *component of* $\widehat{\mathfrak{S}}_\omega$.

Analogously to Proposition 2.16, optimal discrete lifting functions can be defined.

Proposition 3.23. *Consider a point* $\mathbf{x}$ *with respect to* A_i. *Then the lowest possible value of a discrete conservative lifting function is given by*

$$\omega_i(\widehat{\mathfrak{S}}_\omega, \widehat{S}_\omega^{(i)}, \mathbf{x}) := \max\left(\omega_i(\widehat{S}_\omega^{(i)}, \mathbf{x}), \max_{\widehat{C}_\gamma \in \widehat{\mathfrak{S}}_\omega} \omega_i(\widehat{C}_{\gamma i}, \mathbf{x})\right), \qquad i = 1, 2, \ldots, r, \quad (19)$$

where, on the right-hand side, formulas (5) *and* (6) *are applied respectively.*

Analogously to the unmixed case, the mixed cells are preserved by conservative lifting. Denote the collection of mixed cells in the regular subdivision of $(A_1, \ldots, A_i \cup \{\mathbf{x}\}, \ldots, A_r)$ by $\widehat{\mathfrak{S}'_\omega}$ and the corresponding subdivision of $A_i \cup \{\mathbf{x}\}$ by $\widehat{S'}_\omega^{(i)}$.

Lemma 3.24. *Consider* $\mathbf{x}$ *with respect to* A_i. *Let* $\dim(A_i \cup \{\mathbf{x}\}) < n$. *Let* $\mathbf{x}$ *be lifted conservatively with respect to the* i*th component of* $\widehat{\mathfrak{S}}_\omega$. *Then* $\widehat{\mathfrak{S}}_\omega \subseteq \widehat{\mathfrak{S}'_\omega}$. *If* $\mathbf{x} \in \operatorname{conv}(A_i)$, *then* $\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$, *otherwise,* $\forall \widehat{C}_\gamma \in \widehat{\mathfrak{S}'_\omega} \backslash \widehat{\mathfrak{S}}_\omega$, $\widehat{C}_\gamma = (\widehat{C}_1, \ldots, \widehat{C}_i, \ldots, \widehat{C}_r)$: $\widehat{\mathbf{x}} \in \widehat{C}_i$.

Proof. Due to the conservative lifting of $\mathbf{x}$, for each $\widehat{C}_\gamma \in \widehat{\mathfrak{S}}_\omega$, $\langle \cdot, \gamma \rangle$ still attains its minimum at $\widehat{C}_\gamma$, so $\widehat{C}_\gamma \in \widehat{\mathfrak{S}'_\omega}$. The proof of the second statement is trivial. □

Lemma 3.25. *Consider* $\mathbf{x}$ *with respect to* A_i. *Let* $\dim(A_i \cup \{\mathbf{x}\}) = n$. *Let* $\mathbf{x}$ *be lifted conservatively with respect to the* i*th component of* $\widehat{\mathfrak{S}}_\omega$ *and with respect to* $\widehat{S}_\omega^{(i)}$. *Then* $\widehat{\mathfrak{S}}_\omega \subseteq \widehat{\mathfrak{S}'_\omega}$ *and* $\widehat{S}_\omega^{(i)} \subseteq \widehat{S'}_\omega^{(i)}$. *If* $\mathbf{x} \in \operatorname{conv}(A_i)$, *then* $\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$, *otherwise,* $\forall \widehat{C}_\gamma \in \widehat{\mathfrak{S}'_\omega} \backslash \widehat{\mathfrak{S}}_\omega$, $\widehat{C}_\gamma = (\widehat{C}_1, \ldots, \widehat{C}_i, \ldots, \widehat{C}_r)$: $\widehat{\mathbf{x}} \in \widehat{C}_i$ *and* $\widehat{C}_i$ *belongs to the lower hull of* $\operatorname{conv}(\widehat{A}_i \cup \widehat{\mathbf{x}})$.

Proof. The statements concerning the mixed cells can be derived from Lemma 3.24. Because $\widehat{C}_\gamma$ belongs to the lower hull of $\operatorname{conv}(\widehat{A}_1 + \widehat{A}_2 + \cdots + \widehat{A}_r)$, its components $\widehat{C}_i$ belong to the lower hull of $\operatorname{conv}(\widehat{A}_i)$. □

Note that in special cases it may happen that no new mixed cells are obtained ($\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$) although $\mathbf{x} \notin \operatorname{conv}(A_i)$.

Lemma 3.25 uses the change of the normal cone complex $\mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ by adding a point $\mathbf{x}$. Observe that $\mathcal{N}_\vee^{k_j}(\widehat{A}_j)$, $j \neq i$, do not change. If $\mathbf{x} \notin \operatorname{conv}(A_i)$, then $\widehat{A'}_i = \widehat{A}_i \cup \{\mathbf{x}\}$ has new facets in the lower hull in comparison with $\widehat{A}_i$, which gives new cells in $S'^{(i)}_\omega$.

The new facets in the lower hull generate new k_i-faces. Some of them are k_i-faces of old facets. Others are new. The normal cone of one old k_i-face C is modified: $K(C, \widehat{A'}_i) \subset K(C, \widehat{A}_i)$. Instead of $K(C, \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ we have $K(C, \widehat{A'}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A'}_i)$. The normal cones of the new k_i-faces $K(C, \widehat{A'}_i)$ are new elements of $\mathcal{N}_\vee^{k_i}(\widehat{A'}_i)$.

The analogue definition for flattening corresponds to Definition 2.20.

Computing new mixed cells can be done by lifting the new point $\mathbf{x}$ conservatively and applying Algorithm 3.21, where the input for the ith component can be restricted to the new k_i-faces that contain $\widehat{\mathbf{x}}$. It is important to note that even a simple mixed subdivision cannot be guaranteed, because we are only dealing with the collection of mixed cells. See Section 3.7 for how to deal with this fact.

3.6. *Incremental Construction with Connectivity*

In this section the analogue pivoting mechanism of the dynamic lifting algorithm for the general case is described. Here we conjecture that in a regular subdivision, all mixed cells are connected. Therefore, we first investigate the modification of the normal cone complexes by the addition of a new point more carefully.

Assume that $\dim(A_j) = n$ for all $j = 1, \ldots, r$. There is a subdivision $S_\omega^{(i)}$ of A_i induced by ω_i. In addition to the normal cone complex $\mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ we consider, for each facet $\partial_\gamma \widehat{A}_i$ in the lower hull of $\widehat{A}_i$, the normal cone complex $\mathcal{N}_\vee^{k_i}(\partial_\gamma \widehat{A}_i)$. For each k_i-face C of $\partial_\gamma \widehat{A}_i$ there is a cone $K := K(C, \partial_\gamma \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\partial_\gamma \widehat{A}_i)$. If C is a common face of two facets $\partial_{\gamma_1} \widehat{A}_i$ and $\partial_{\gamma_2} \widehat{A}_i$, then $K(C, \partial_{\gamma_1} \widehat{A}_i) \cap K(C, \partial_{\gamma_2} \widehat{A}_i) \subset K(C, \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A}_i)$. We consider the root $\bigwedge_{K \in \mathcal{N}_\vee^{k_i}(\partial_\gamma \widehat{A}_i)} K$ of the normal cone complex. Since $\partial_\gamma \widehat{A}_i$ is a facet of the lower hull, the root is generated by the outer normal $-\gamma$ of $\partial_\gamma \widehat{A}_i$.

Adding a point $\mathbf{x} \notin \operatorname{conv}(A_i)$ to A_i gives rise to new facets $\partial_\gamma \widehat{A'}_i$ in the lower hull and thus gives rise to new normal cones $K(\partial_\gamma \widehat{A'}_i)$ and new normal cone complexes $\mathcal{N}_\vee^{k_i}(\partial_\gamma \widehat{A'}_i)$ and new roots. This process can be visualized nicely with an example.

Example 3.26. Consider again the same polytopes as in Fig. 8. On the left in Fig. 9 the normal cone complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^1(\widehat{P}_2)$ are drawn in a way that the intersection points are put above the corresponding mixed cells of the induced subdivision of (P_1, P_2). Since the induced subdivisions $S_{\omega_1}^{(1)}$ and $S_{\omega_2}^{(2)}$ consist only of one cell, respectively, the complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^2(\widehat{P}_2)$ are already the complexes $\mathcal{N}_\vee^1(\partial_\gamma \widehat{P}_i)$ for the cells $\partial_\gamma \widehat{P}_i$ in the subdivision of $\widehat{P}_i$. On the right the point $(3, 1)^t$ has been considered with respect to P_2 and amended after lifting it conservatively. A new cell C_2 is obtained, so $S_{\omega_2}^{\prime(2)} = \{C_1, C_2\}$. Thus two roots are drawn. Two cones $K_1 \in \mathcal{N}_\vee^1(\widehat{C}_1)$ and $K_2 \in \mathcal{N}_\vee^1(\widehat{C}_2)$ have a nonempty intersection which is drawn as the line between the two roots. The conservativeness of the lifting is reflected by the fact that the intersection points in the left picture also exist in the right picture.

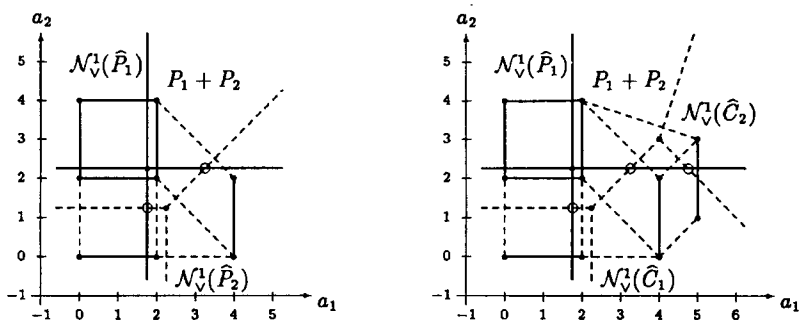


Fig. 9. The normal cone complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^1(\widehat{P}_2)$, with the same subdivision of $P_1 + P_2$ as in Fig. 8, are shown on the left. The roots of the normal cone complexes have been placed so that intersections between dashed and thick lines lie above the mixed cells. On the right, one point has been amended to P_2.

In Fig. 9 it is seen that the three mixed cells are connected to each other. The argument presented in the following sentence is due to Pedersen. By considering the corresponding construction with normal cone complexes, passing from one intersection point to another one becomes possible by going back to the roots. For a polytope it is obvious that all faces are *connected to each other*, i.e., for each pair of k-faces $\mathfrak{F}$, $\mathfrak{F}^*$, there is a path of k-faces $\mathfrak{F}^{(j)}$, $j = 0, \ldots, s$, such that $\mathfrak{F}^{(0)} = \mathfrak{F}$, $\mathfrak{F}^{(s)} = \mathfrak{F}^*$, $\dim(\mathfrak{F}^{(j)} \cap \mathfrak{F}^{(j+1)}) \geq k - 1$, $j = 0, \ldots, s - 1$. The notion of connectivity of mixed cells is more complicated.

Definition 3.27. Let S_ω be the induced subdivision of $\mathcal{A}$. Denote the collection of lifted mixed cells in $\widehat{S}_\omega$ by $\widehat{\mathfrak{S}}_\omega$. Two mixed cells $\widehat{C}, \widehat{D} \in \widehat{\mathfrak{S}}_\omega$ are *connected to each other* if there is a path from $\widehat{C}$ to $\widehat{D}$: a sequence of mixed cells $\widehat{C}^{(j)} \in \widehat{\mathfrak{S}}_\omega$, $j = 0, 1, \ldots, s$, exists such that $\widehat{C}^{(0)} = \widehat{C}$ and $\widehat{C}^{(s)} = \widehat{D}$, with

$$\dim(\widehat{C}_i^{(j)} \cap \widehat{C}_i^{(j+1)}) \geq k_i - 1, \qquad \forall i = 1, \ldots, r, \quad j = 0, \ldots, s - 1. \tag{20}$$

Ignoring the lifting values, the mixed cells C and D are said to be *connected to each other*. If all mixed cells in $\widehat{\mathfrak{S}}_\omega$ are connected to each other, then $\widehat{\mathfrak{S}}_\omega$ is *connected*. Ignoring the lifting values, $\mathfrak{S}_\omega$ is *connected* if all its mixed cells are connected to each other.

The following conjecture was first stated by Pedersen [42]:

Conjecture 3.28. *Let S_ω be the induced subdivision of $\mathcal{A}$ and let $\mathfrak{S}_\omega$ be the set of mixed cells. Then $\mathfrak{S}_\omega$ is connected.*

For $n = 2$, Pedersen has given a proof, based on normal fans. His idea has been shown on the left of Fig. 9.

Algorithm 3.29 allows the exploitation of the connectivity of the mixed cells. It is important to note that this only happens when $\widehat{\Delta}_\omega^{(i)} \neq \emptyset$, for full-dimensional polytopes. Also, as long as Conjecture 3.28 remains unproven, this part of the algorithm remains heuristic, but this can be directly switched off by omitting the test on $\widehat{\Delta}_\omega^{(i)} = \emptyset$ and applying Algorithm 3.21 with all lower k_i-faces instead of only the neighboring ones. Furthermore, note that, for Algorithm 3.29, it is sufficient that this conjecture holds for any *placeable* regular mixed subdivision.

Algorithm 3.29. The dynamic lifting algorithm to compute new mixed cells:

Input: $(k_1, k_2, \ldots, k_r)$, $n = \sum_{i=1}^r k_i$,	*type of mixture*
$(\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$,	*lifted point sets*
$(\widehat{\mathfrak{F}}_1, \widehat{\mathfrak{F}}_2, \ldots, \widehat{\mathfrak{F}}_r)$,	*k_i-faces of lower hull of* $\operatorname{conv}(\widehat{A}_i)$
$(\widehat{\Delta}_\omega^{(1)}, \widehat{\Delta}_\omega^{(2)}, \ldots, \widehat{\Delta}_\omega^{(r)})$,	*facets of lower hull of* $\operatorname{conv}(\widehat{A}_i)$
$\widehat{\mathfrak{S}}_\omega = \{ \widehat{C} \in \widehat{S}_\omega \mid V_n(C) > 0 \}$,	*collection of mixed cells*
$\mathbf{x}, i$.	*a point to add to A_i*
Output: $(\widehat{A}_1, \ldots, \widehat{A}_i \cup \{\widehat{\mathbf{x}}\}, \ldots, \widehat{A}_r)$,	*updated lifted point sets*
$(\widehat{\mathfrak{F}}_1, \ldots, \widehat{\mathfrak{F}}_i \cup \widehat{\mathfrak{F}}_i^{\mathbf{x}}, \ldots, \widehat{\mathfrak{F}}_r)$,	*updated k_i-faces*
$(\widehat{\Delta}_\omega^{(1)}, \ldots, \widehat{\Delta}_\omega^{(i)} \cup \widehat{\Delta}_\omega^{(i)\mathbf{x}}, \ldots, \widehat{\Delta}_\omega^{(r)})$,	*updated regular triangulations*
$\widehat{\mathfrak{S}}_\omega \cup \widehat{\mathfrak{S}}_\omega^{\mathbf{x}}$.	*updated collection of mixed cells*

COMPUTE $\widehat{\mathfrak{F}}_i^{\mathbf{x}}$:
  $\widehat{x}_{n+1} := \omega(\widehat{\mathfrak{S}}_\omega, \widehat{\Delta}_\omega^{(i)}, \mathbf{x}, i)$; *lift the point conservatively*
  $\widehat{A}_i := \widehat{A}_i \cup \{\widehat{\mathbf{x}}\}$; *update lifted points set*
  if $\widehat{\Delta}_\omega^{(i)} = \emptyset$ *update triangulation of* A_i
    then
      if $\dim(A_i) < n$
        then $\widehat{\mathfrak{F}}_i^{\mathbf{x}} := \mathit{Enumerate\_Faces}(\widehat{\mathcal{A}}_i, k_i, \widehat{\mathbf{x}})$; *new* k_i*-faces*
        else $\widehat{\Delta}_\omega^{(i)\mathbf{x}} := \mathit{Initial\_Facets}(\widehat{\mathcal{A}}_i, \widehat{\mathbf{x}})$; *initial facets of lower hull*
          $\widehat{\mathfrak{F}}_i^{\mathbf{x}} := \mathit{Enumerate\_Faces}(\widehat{\Delta}_\omega^{(i)\mathbf{x}}, k_i, \widehat{\mathbf{x}})$; *new* k_i*-faces*
          $\widehat{\Delta}_\omega^{(i)} := \widehat{\Delta}_\omega^{(i)} \cup \widehat{\Delta}_\omega^{(i)\mathbf{x}}$; *update the lower hull*
      end if;
    else $\widehat{\Delta}_\omega^{(i)\mathbf{x}} := \mathit{New\_Facets}(\widehat{\Delta}_\omega^{(i)}, \widehat{\mathbf{x}})$; *apply Algorithm 2.9*
      $\widehat{\mathfrak{F}}_i^{\mathbf{x}} := \mathit{Enumerate\_Faces}(\widehat{\Delta}_\omega^{(i)\mathbf{x}}, k_i, \widehat{\mathbf{x}})$; *new* k_i*-faces*
      $\widehat{\Delta}_\omega^{(i)} := \widehat{\Delta}_\omega^{(i)} \cup \widehat{\Delta}_\omega^{(i)\mathbf{x}}$; *update the lower hull*
  end if.

COMPUTE $\widehat{\mathfrak{S}}_\omega^{\mathbf{x}}$:
  if $\widehat{\Delta}_\omega^{(i)} = \emptyset$
    then $\widehat{\mathfrak{S}}^{\mathbf{x}} := \mathit{Algorithm\_3.21}(\widehat{\mathfrak{F}}_1, \ldots, \widehat{\mathfrak{F}}_i^{\mathbf{x}}, \ldots, \widehat{\mathfrak{F}}_r)$; *new mixed cells*
    else $\widehat{\mathfrak{S}}_\omega^{\mathbf{x}} := \emptyset$; *exploit neighborship relations*
      for all cells $C \in \mathfrak{S}_\omega$ do *apply the connectivity*
        $\widehat{\mathfrak{F}}_i'^{\mathbf{x}} := \mathit{Neighboring\_Faces}(C, \widehat{\mathfrak{F}}_i^{\mathbf{x}})$; *neighbored new* k_i*-faces*
        if $\widehat{\mathfrak{F}}_i'^{\mathbf{x}} \neq \emptyset$ then
          for all j in $\{1, \ldots, r\} \backslash \{i\}$ do
            $\widehat{\mathfrak{F}}_j' := \mathit{Neighboring\_Faces}(C, \widehat{\mathfrak{F}}_j)$; *apply the connectivity*
          end for;
          $\widehat{\mathfrak{S}}_\omega'^{\mathbf{x}} := \mathit{Algorithm\_3.21}(\widehat{\mathfrak{F}}_1', \ldots, \widehat{\mathfrak{F}}_i'^{\mathbf{x}}, \ldots, \widehat{\mathfrak{F}}_r')$; *new mixed cells*
          $\widehat{\mathfrak{S}}_\omega^{\mathbf{x}} := \widehat{\mathfrak{S}}_\omega^{\mathbf{x}} \cup \widehat{\mathfrak{S}}_\omega'^{\mathbf{x}}$; *update mixed cells*
        end if;
      end for;
  end if.

The idea is to apply Algorithm 2.9 to compute the new k_i-faces of $\widehat{A}_i \cup \{\widehat{\mathbf{x}}\}$ after adding a point $\mathbf{x}$ to A_i. The new mixed cells will be computed by repeated applications of Algorithm 3.21 which is called for each old mixed cell which is neighbored to a new k_i-face. Algorithm 3.21 is called with the following small input set: Let $C = (C_1, \ldots, C_r)$ be the old mixed cell. $\widehat{\mathfrak{F}}_i'^{\mathbf{x}}$ are those new k_i-faces which have a common $(k_j - 1)$-face with $\widehat{C}_i$. $\widehat{\mathfrak{F}}_j'$, $j \neq i$, are the k_j-faces which share a common $(k_j - 1)$-face with $\widehat{C}_j$. In practice, it turns out to be efficient to put the new faces $\widehat{\mathfrak{F}}_i'^{\mathbf{x}}$ in front of the argument list of Algorithm 3.21, in order to avoid to make face–face combinations which lead to other mixed cells, not containing any of the new faces.

In the beginning, when only the initial cell and some other cells exist, Algorithm 3.21 is called with all new k_i-faces. Note that, when $\dim(A_i)$ becomes n, due to the consideration of $\mathbf{x}$, *Initial_Facets* first computes one initial cell and then applies Algorithm 2.9 to all other points in A_i. Note that $\#A_i$ might be larger than $\dim(A_i)$. The new k_i-faces are

computed by *Enumerate_Faces*, which applies either to the whole list $\widehat{A_i}$ or to the new cells in $\Delta_\omega^{(i)\mathbf{x}}$.

Algorithm 3.29 combines the advantages of both the Cayley trick (Algorithm 3.8) and the generalization of Betke's idea (Algorithm 3.21), because it allows exploitation of the neighborship relations, so that many spurious face–face combinations can be prevented *a priori*, as opposed to the feasibility tests in Algorithm 3.21. This implies that Algorithm 3.29 provides an efficient solution to the unmixed, semimixed, and fully mixed case.

3.7. *Unfolding Cells with the Same Inner Normal*

For the unmixed case, we mentioned in Section 2.6 that, due to flattening, different cells could get the same inner normal. Because the cells are stored as before, this fact does not influence the computation of new cells or the computation of the volume. However, for static polyhedral continuation, i.e., the approach presented in [30], all cells need to have a different inner normal.

For dynamic polyhedral continuation, see Section 4.2, the following approach can be applied. The solutions which correspond to the existing cells have to be extended to the enlarged system, and therefore they can all use the *same* homotopy, generated by the *same* inner normal. In order to compute the solutions which correspond to the new cells, the points which belong to those new cells with the same inner normal can be relifted, by exploiting the data structure and by application of conservative lifting functions. In Fig. 10 the *unfolding* of cells is pictured on a small example.

The application to this unfolding method is straightforward when the Cayley trick is used to compute the mixed volume. However, when only the mixed cells are computed in Algorithm 3.29, we still have to rely on the recursion mechanism described in [50] to deal with the case when some mixed cells are not exactly of type $(k_1, k_2, \ldots, k_r)$. So, there is a tradeoff on the bound to be set for the lifting value. If the bound is low, flattening will occur frequently, which leads to relifting afterward.

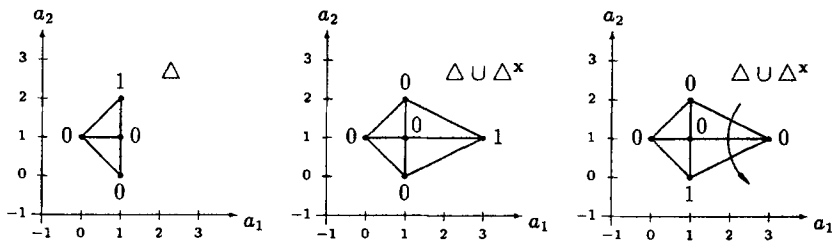


Fig. 10. A triangulation Δ is shown on the left. The big numbers indicate the values of the lifting function. In the middle $\widehat{\Delta}$ has been flattened and new cells which contain $\mathbf{x} = (3, 1)^t$ have been added. The two new cells have the same inner normal $\gamma = (-1, 0, 2)$. The modification of the lifting is displayed on the right. Here the two new cells have a different inner normal. The arrow indicates the order of traversing the new cells while relifting.

3.8. *Complexity and Cost Estimates*

Like volume computation, the problem of mixed-volume computation is known to be #P-hard, see [14] and [25]. The analogue result as in the unmixed case from computational convexity is given by the following proposition.

Proposition 3.30 [25]. *When the dimension n is fixed, the mixed volume of a tuple of polytopes can be computed in polynomial time in the input size.*

By application of the Cayley trick, Theorem 2.25 can be extended in a straightforward way. Again it is assumed that the problem is nondegenerate.

Theorem 3.31. *Given a tuple of points sets $\mathcal{A} = (A_1, A_2, \ldots, A_r)$. Denote $m = n + r - 1$. Let $\#\mathcal{A} = \sum_{i=1}^{r} \#A_i$ and $\mathcal{A}' = \mathcal{A} \backslash C$, with C a cell. The construction of a placeable fine-mixed subdivision S of $\mathcal{A}$ takes at least $O((\#S)m^3 + (\#\mathcal{A}')m^2)$, at most $O((\#S)m^3 + (\#S)(\#\mathcal{A}')m^2)$, and on average $O((\#S)m^3 + (\#S)(\#\mathcal{A}')m)$ arithmetical operations.*

Note that $\#S$ stands for the cardinality of the whole subdivision, with mixed as well as unmixed cells included. The Cayley trick becomes more expensive when r increases.

The cost of the static lifting method, based on the idea of Betke, has been investigated in great detail in [14] and in [17] and [18]. Following these approaches, a bound on the complexity of linear programming is needed. Therefore, the following result (see [24]) is used.

Theorem 3.32 (Karmarkar's Projective Algorithm Runs in Polynomial Time). *Karmarkar's algorithm can be adapted to solve the general linear programming problem in $O(nL)$ steps, where the average step-complexity is $O(n^{5/2}L)$. The required precision is $O(L)$.*

Fixed-precision calculation is assumed, so the factor L is omitted in what follows. Recall the following notations: $\widehat{\mathfrak{F}}_i$ denotes the k_i-faces of the lower hull of P_i and the collection of mixed cells in the subdivision S_ω is denoted by $\mathfrak{S}_\omega$.

Theorem 3.33. *Let $\varepsilon_i \in [0, 1]$ be the probability of success that a k_i-face in $\widehat{\mathfrak{F}}_i$ passes the feasibility test. On average, computing the collection of mixed cells $\mathfrak{S}_\omega$ requires $O(\prod_{i=1}^{r} (\#\widehat{\mathfrak{F}}_i)\varepsilon_i[in + (n-i)^{7/2}])$ arithmetical operations.*

Proof. In order to compute $\mathfrak{S}_\omega$ all face–face combinations need to be considered. This explains the product of the cardinalities of the k_i-faces with their respective probabilities. At level i, it takes time $O(in)$ to perform the elimination step in Algorithm 3.21. Applying Theorem 3.32, the average time needed to solve a linear programming problem in dimension $n - i$ is given by $O((n-i)^{7/2})$. □

Note the practical importance of this elimination step in Algorithm 3.21, as it allows reduction of the dimension of the second feasibility test. The asymptotic complexity

for the dynamic lifting algorithm is the same as given in Theorem 3.33, although in practice Algorithm 3.21 can benefit from sharing factorizations, i.e., at level i all face–face combinations of one branch can use the result of the same elimination step, while in Algorithm 3.29 these factorizations have to be computed again, each time the algorithm is invoked with a new point.

4. Impact on Polynomial System Solving

4.1. *The Theorem of Bernshteĭn*

Computation of the full solution set of a polynomial system $F = (f_1, f_2, \ldots, f_n)$ in n unknowns is often required in many applications. The third section provides some examples. Homotopy continuation methods have proven to be reliable for this purpose, see [36] for an introduction. The system to be solved is embedded into a family of systems, the so-called homotopy, which defines paths of solutions from known solutions to the desired solutions to be traced numerically by continuation methods. Recently, polyhedral homotopy methods have been presented [30], [52] for computation of all isolated roots of sparse Laurent polynomial systems in $\mathbb{C}_0^n$, with $\mathbb{C}_0 = \mathbb{C}\backslash\{0\}$.

Definition 4.1. A *Laurent polynomial* f is defined as

$$f(\mathbf{x}) = \sum_{c_\mathbf{a} \in A} c_\mathbf{a} \mathbf{x}^\mathbf{a}, \qquad c_\mathbf{a} \in \mathbb{C}_0, \quad \mathbf{x}^\mathbf{a} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \quad A \subset \mathbb{Z}^n, \quad \#A < \infty.$$

The set A is the *support* of f, $A = \mathrm{supp}(f)$. Its convex hull, $P = \mathrm{conv}(A)$, is the *Newton polytope* of f.

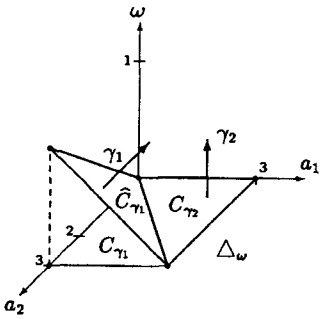
Definition 4.2. A *Laurent polynomial system* F is defined by a tuple of Laurent polynomials, $F = (f_1, f_2, \ldots, f_n)$. The tuples $\mathcal{A}$ and $\mathcal{P}$ collect respectively the supports and Newton polytopes:

$$\begin{array}{ll} \mathcal{A} = (A_1, A_2, \ldots, A_n), & A_j = \mathrm{supp}(f_j) \\ \mathcal{P} = (P_1, P_2, \ldots, P_n), & P_j = \mathrm{conv}(A_j) \end{array}, \qquad j = 1, 2, \ldots, n.$$

The relationship between tuples of polytopes and systems of polynomial equations has been given by Bernshteĭn, see [3].

Theorem 4.3. *Let F be a system of n Laurent polynomials with support $\mathcal{A}$ and the associated tuple of Newton polytopes $\mathcal{P}$. Then for almost all choices of the coefficients the system F has exactly as many roots in $\mathbb{C}_0^n$ as $V_n(\mathcal{P})$.*

Canny and Rojas presented a refined version of this theorem in [11]. They named the mixed volume of the Newton polytopes the BKK bound, named after its inventors, **B**ernshteĭn [3], **K**ushnirenko [33], and **K**hovanskiĭ [32]. See Chapter 4, Section 27, of [9] for more on the importance of mixed volumes in algebraic geometry. In [44] more refinements of the theorem are presented. Instead of the above definitions which might



$$F(x_1,x_2)=\begin{cases} x_1^3x_2^3+c_{11}x_1^3+c_{12}x_2^3+c_{13}=0\\ x_1^3x_2^3+c_{21}x_1^3+c_{22}x_2^3+c_{23}=0\end{cases}$$

$$\widehat{F}(x_1,x_2,t)=\begin{cases} x_1^3x_2^3t^0+c_{11}x_1^3t^0+c_{12}x_2^3t^1+c_{13}t^0=0\\ x_1^3x_2^3t^0+c_{21}x_1^3t^0+c_{22}x_2^3t^1+c_{23}t^0=0\end{cases}$$

$$\gamma_1=(1,-1,3)\qquad \gamma_2=(0,0,1)$$

Fig. 11. The regular triangulation is shown on the left. The corresponding polynomial system and the induced homotopy are displayed at the right.

look artificial, a Newton polytope can be related to a Laurent polynomial via an amoeba, see Chapter 6, Section 1, of [23], which models the asymptotic behavior of the roots. In [8] Newton polyhedra have been applied to compute the local uniformization for all branches of a curve defined by a system of equations in n-dimensional space.

4.2. *Incremental Polyhedral Continuation*

In [30] another constructive proof of Bernshteĭn's theorem has been presented, introducing the concept of polyhedral continuation for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$. Here we illustrate the idea of incremental polyhedral continuation with a small example.

Example 4.4. In Fig. 11 a regular triangulation of a polynomial system with the randomly chosen complex coefficients is shown. The triangulation Δ_ω also subdivides the system F in *initial form systems* F_{γ_i}, which are subsystems of F whose exponent vectors all belong to C_{γ_i}, $i = 1, 2, \ldots, \#\Delta_\omega$. The induced homotopy $\widehat{F}$ can be used directly to extend the solutions of the initial form system F_{γ_2} to the system F. In order to extend the solutions corresponding to the subsystem F_{γ_1}, a transformation needs to be used, defined by the components of the inner normal γ_1:

$$\gamma_1=(1,-1,3)=3\times\left(\frac{1}{3},\frac{-1}{3},1\right),\qquad x_1\leftarrow\widetilde{x}_1t^{1/3},\quad x_2\leftarrow\widetilde{x}_2t^{-1/3},\tag{21}$$

which leads to the homotopy

$$\mathcal{H}_{\gamma_1}(\widetilde{x}_1,\widetilde{x}_2,t)=\begin{cases}\widetilde{x}_1^3\widetilde{x}_2^3+c_{11}\widetilde{x}_1^3t^1+c_{12}\widetilde{x}_2^3+c_{13}=0,\\ \widetilde{x}_1^3\widetilde{x}_2^3+c_{21}\widetilde{x}_1^3t^1+c_{22}\widetilde{x}_2^3+c_{23}=0.\end{cases}\tag{22}$$

For $t = 0$, $\mathcal{H}_{\gamma_1}(\widetilde{\mathbf{x}},0) = F_{\gamma_1}(\widetilde{\mathbf{x}})$ and, for $t = 1$, $\mathcal{H}_{\gamma_1}(\widetilde{\mathbf{x}},1) = F(\widetilde{\mathbf{x}})$. So, by letting t vary from 0 to 1, the desired solutions can be computed.

This example illustrates the advantages of the induced homotopy obtained by the dynamic lifting algorithm: all lifting values are as low as possible which leads to a well-conditioned homotopy whose solution paths are weakly nonlinear and hence less expensive to track numerically than with a homotopy induced by a random lifting function.

4.3. *Computational Experiences*

After having done all this hard work ourselves, it is now time to put the computer to work. The algorithms presented above have been implemented in Ada, with the aid of the Verdix Ada System (VADS). All modules and programs are compiled and executed on a DECStation 5000/240. Except for the linear program solver, we have worked with integer calculus, because we are dealing with Newton polytopes.

4.3.1. *The Nine-Point Problem.* A long-standing problem in mechanism design has been the problem of finding all four-bar linkages whose coupler curve passes through nine prescribed points. Recently, in [55], for the first time a complete solution has been given. In [39] an efficient homotopy has been designed by exploiting its product-decomposition structure.

In [39] the unknowns of this system are grouped in two 6-tuples $\mathbf{z} = (n, \hat{n}, x, \hat{x}, a, \hat{a})$ and $\mathbf{w} = (m, \hat{m}, y, \hat{y}, b, \hat{b})$. The coordinates of the precision points are defined by the numbers δ_i and $\hat{\delta}_i$, for $i = 1, 2, \ldots, 8$. The ninth precision point lies in the origin. The first four equations are

$$n = a\hat{x}, \qquad \hat{n} = \hat{a}x, \qquad m = b\hat{y}, \qquad \hat{m} = \hat{b}y, \tag{23}$$

and the remaining eight equations have the following form:

$$f_i(\mathbf{z}, \mathbf{w}) = \gamma_i(\mathbf{z}, \mathbf{w})\hat{\gamma}_i(\mathbf{z}, \mathbf{w}) + \gamma_i(\mathbf{z}, \mathbf{w})\gamma_i^0(\mathbf{z}, \mathbf{w}) + \gamma_i^0(\mathbf{z}, \mathbf{w})\hat{\gamma}_i(\mathbf{z}, \mathbf{w}) = 0, \quad i = 1, 2, \ldots, 8, \tag{24}$$

where the polynomials $\gamma_i(\mathbf{z}, \mathbf{w})$, $\hat{\gamma}_i(\mathbf{z}, \mathbf{w})$, and $\gamma_i^0(\mathbf{z}, \mathbf{w})$ can be written in terms of linear ones:

$$\gamma_i(\mathbf{z}, \mathbf{w}) = q_i(\mathbf{z})r_i(\mathbf{w}) - q_i(\mathbf{w})r_i(\mathbf{z}), \tag{25}$$

$$\hat{\gamma}_i(\mathbf{z}, \mathbf{w}) = r_i(\mathbf{z})p_i(\mathbf{w}) - r_i(\mathbf{w})p_i(\mathbf{z}), \tag{26}$$

$$\gamma_i^0(\mathbf{z}, \mathbf{w}) = p_i(\mathbf{z})q_i(\mathbf{w}) - p_i(\mathbf{w})q_i(\mathbf{z}), \tag{27}$$

where p_i, q_i, and r_i are defined as

$$p_i(\mathbf{z}) = \hat{n} - \hat{\delta}_i x, \qquad q_i(\mathbf{z}) = n - \delta_i \hat{x}, \qquad r_i(\mathbf{z}) = \delta_i(\hat{a} - \hat{x}) + \hat{\delta}_i(a - x) - \delta_i\hat{\delta}_i, \tag{28}$$

$$p_i(\mathbf{w}) = \hat{m} - \hat{\delta}_i y, \qquad q_i(\mathbf{w}) = m - \delta_i \hat{y}, \qquad r_i(\mathbf{w}) = \delta_i(\hat{b} - \hat{y}) + \hat{\delta}_i(b - y) - \delta_i\hat{\delta}_i. \tag{29}$$

By the first four equations (23), the unknowns n, $\hat{n}$, m, and $\hat{m}$ can be replaced by their respective right-hand sides into the remaining eight equations (24). The resulting system

is unmixed and consists of eight equations of degree seven. Hence the total degree, i.e., the product of all degrees of the polynomials, equals 7^8 =5,764,801. This substitution blows up the total degree (originally equal to $2^4 4^8 = 1{,}048{,}576$), and the best m-homogeneous Bézout bound (from 286,720 to 645,120, $m = 4$), but leaves the BKK bound unchanged. The fact that all polytopes in the system are the same makes it easier to handle.

Now we give the results of our program on this eight-dimensional problem. In the system there are 259 terms, but only 158 of them lead to vertices. The time needed to verify this was less than 3 minutes (163 cpu sec.). For a random addition of the points, there are 13,339 simplices in the triangulation, computed in about 36 minutes (2162 cpu sec). The volume equals 83,977 divided by 8!. Computation of the volume, given the triangulation, costs about 1.3 minutes (75 cpu sec.).

The cost bounds for the dynamic lifting algorithm, as derived in Section 2.7, will be checked on this example. The time for the computation of the volume, given the triangulation Δ, corresponds to $O((\#\Delta)n^2)$. Multiplication by n gives $O((\#\Delta)n^3)$, yielding 600 cpu sec. (75 cpu sec. $\times$ 8) as the total cost for all factorizations. By dividing $O((\#\Delta)n^2)$ by $\#\Delta$ and multiplying by $\#A'$, the cost for computing the additional decompositions in the optimal case is given by 0.8 cpu sec. (75 cpu sec./13,339 $\times(158-9)$). In the worst case, multiplying $O((\#\Delta)n^2)$ by $\#A'$ yields 11,175 cpu sec. (75 cpu sec. $\times(158-9)$) as the total cost for computing all decompositions. In the average case, dividing $O((\#A')(\#\Delta)n^2)$ by n yields 1397 cpu sec. (11,175/8). After adding the factorization and decomposition costs the following inequality is obtained: $601 < 2{,}162 < 11{,}775$, while the average cost bound equals 1997 cpu sec., which is quite close to the actual computing time. Note that the latter time contains not only the arithmetic operations, but also the overhead caused by, e.g., memory management.

The purpose of this example is to demonstrate the complexity of volume computation, and not to claim that this leads to an efficient approach for solving this system. It is worth noting that the BKK bound 83,977 is less than 286,720, which is the bound for the number of the solutions which the brute-force technique in [55] was based on. As explained in [39], the product-decomposition structure of the system should be exploited in order to solve it more efficiently.

4.3.2. *A PUMA Robot.* The hand position and orientation of a PUMA robot can be modeled [37] by the following:

$$
\begin{cases}
x_1^2 + x_2^2 - 1 = 0, \\
x_3^2 + x_4^2 - 1 = 0, \\
x_5^2 + x_6^2 - 1 = 0, \\
x_7^2 + x_8^2 - 1 = 0, \\
0.004731x_1x_3 - 0.3578x_2x_3 - 0.1238x_1 - 0.001637x_2 - 0.9338x_4 + x_7 - 0.3571 = 0, \\
0.2238x_1x_3 + 0.7623x_2x_3 + 0.2638x_1 - 0.07745x_2 - 0.6734x_4 - 0.6022 = 0, \\
x_6x_8 + 0.3578x_1 + 0.004731x_2 = 0, \\
-0.7623x_1 + 0.2238x_2 + 0.3461 = 0.
\end{cases}
$$

The total degree of this system equals 128. By partitioning the set of unknowns in $Z = \{\{x_1, x_2\}, \{x_3, x_4, x_7, x_8\}, \{x_5, x_6\}\}$, the 3-homogeneous Bézout bound equals 16. The BKK bound equals 16, which is the exact number of isolated roots of this system. The remarkable fact of this system is that *any* initial mixed cell has volume 16.

4.3.3. *Camera Motion from Point Matches.* The following system models the displacement of a camera between two positions in a static environment, with the coordinates of the matched points as given in [17]. The coordinates of the frames have been scaled, i.e., all components have been divided by 1000. In [20] this problem has been formulated using epipolar geometry.

$$\begin{aligned}
-3.6d_1q_1 + 4.1d_1q_2 + 2.0d_1q_3 + 0.1d_1 + 4.1d_2q_1 + 1.8d_2q_2 + 3.7d_2q_3 - 0.2d_2 \\
+2.0d_3q_1 + 3.7d_3q_2 - 4.0d_3q_3 + 0.3d_3 + 0.1q_1 - 0.2q_2 + 0.3q_3 + 5.8 &= 0,\\
-2.140796d_1q_1 - 3.998792d_1q_2 + 3.715992d_1q_3 - 0.2828d_1 - 3.998792d_2q_1 \\
-1.575196d_2q_2 - 3.998792d_2q_3 + 3.715992d_3q_1 - 3.998792d_3q_2 - 2.140796d_3q_3 \\
+0.2828d_3 - 0.2828q_1 + 0.2828q_3 + 5.856788 &= 0,\\
0.3464d_1q_1 + 0.1732d_1q_2 - 5.999648d_1q_3 - 0.1732d_1 + 0.1732d_2q_1 - 5.999648d_2q_2 \\
-0.1732d_2q_3 + 0.3464d_2 - 5.999648d_3q_1 - 0.1732d_3q_2 - 0.3464d_3q_3 - 0.1732d_3 \\
-0.1732q_1 + 0.3464q_2 - 0.1732q_3 + 5.999648 &= 0,\\
-5701.3d_1q_1 - 2.9d_1q_2 + 3796.7d_1q_3 - 1902.7d_1 - 2.9d_2q_1 - 5698.7d_2q_2 \\
+1897.3d_2q_3 + 3803.3d_2 + 3796.7d_3q_1 + 1897.3d_3q_2 + 3803.3d_2 + 3796.7d_3q_1 \\
+1897.3d_3q_2 + 5703.1d_3q_3 + 0.7d_3 - 1902.7q_1 + 3803.3q_2 + 0.7q_3 + 5696.9 &= 0,\\
-6.8d_1q_1 - 3.2d_1q_2 + 1.3d_1q_3 + 5.1d_1 - 3.2d_2q_1 - 4.8d_2q_2 - 0.7d_2q_3 - 7.1d_2 \\
+1.3d_3q_1 - 0.7d_3q_2 + 9.0d_3q_3 - d_3 + 5.1q_1 - 7.1q_2 - q_3 + 2.6 &= 0,\\
-d_1q_1 - d_2q_2 - d_3q_3 + 1 &= 0.
\end{aligned}$$

The total degree of this system equals 64. By partitioning the set of unknowns as $\{\{d_1, d_2, d_3\}, \{q_1, q_2, q_3\}\}$, the 2-homogeneous Bézout bound 20 is obtained, which equals the mixed volume and the exact number of isolated solutions. The system is semimixed, i.e., there are only three different support sets. Therefore, it can be handled efficiently with the Cayley trick and, by exploiting the connectivity conjecture, the dynamic lifting algorithm computes the mixed volume more efficiently than the static lifting algorithm.

4.3.4. *An Inverse Position Problem.* This system occurs as Example 3.3 in [53] and has been described in [54]. It represents an inverse position problem for six-jointed robot arms.

$$\begin{aligned}
c_1^2 + z_{21}^2 + z_{22}^2 - 1 &= 0,\\
z_{31}^2 + z_{32}^2 + z_{33}^2 - 1 &= 0,\\
z_{41}^2 + z_{42}^2 + z_{43}^2 - 1 &= 0,\\
z_{51}^2 + z_{52}^2 + z_{53}^2 - 1 &= 0,\\
c_1z_{33} - c_2 + z_{21}z_{31} + z_{22}z_{32} &= 0,\\
-c_3 + z_{31}z_{41} + z_{32}z_{42} + z_{33}z_{43} &= 0,\\
-c_4 + z_{41}z_{51} + z_{42}z_{52} + z_{43}z_{53} &= 0,\\
-c_1 + z_{51}z_{61} + z_{52}z_{62} + z_{53}z_{63} &= 0,\\
-c_1e_2z_{32} + d_2z_{21} + d_3z_{31} + d_4z_{41} + d_5z_{51} - e_1z_{22} + e_2z_{22}z_{33} + e_3z_{32}z_{43} \\
-e_3z_{33}z_{42} + e_4z_{42}z_{53} - e_4z_{43}z_{52} + e_5z_{52}z_{63} - e_5z_{53}z_{62} - p_{61} &= 0,\\
c_1e_2z_{31} + d_2z_{22} + d_3z_{32} + d_4z_{42} + d_5z_{52} + e_1z_{21} - e_2z_{21}z_{33} - e_3z_{31}z_{43} \\
+e_3z_{33}z_{41} - e_4z_{41}z_{53} + e_4z_{43}z_{51} - e_5z_{51}z_{63} + e_5z_{53}z_{61} - p_{62} &= 0,\\
c_1d_2 + d_3z_{33} + d_4z_{43} + d_5z_{53} + e_2z_{21}z_{32} - e_2z_{22}z_{31} + e_3z_{31}z_{42} \\
-e_3z_{32}z_{41} + e_4z_{41}z_{52} - e_4z_{42}z_{51} + e_5z_{51}z_{62} - e_5z_{52}z_{61} - p_{63} &= 0.
\end{aligned}$$

The total degree of this system equals 1024. The 2-homogeneous Bézout number equals 320, with partition of the unknowns, computed in [53],

$$Z = \{\{z_{21}, z_{23}, z_{41}, z_{42}, z_{43}\}, \{z_{31}, z_{32}, z_{33}, z_{51}, z_{52}, z_{53}\}\}.$$

A better root count is provided by the BKK bound which equals 288. For any random complex choice of the parameters, there are only 16 finite regular solutions. This example illustrates the difficulty the dynamic lifting algorithm has when the factorizations cannot be shared (see the last paragraph of Section 3.8).

4.3.5. *A Heart-Dipole Problem.* The following problem has been presented as a heart-dipole problem, see [38] and [40]. The original problem description can be found in [41].

$$\begin{cases} a+b = 0.6325, \\ c+d = 0.8465, \\ ta+ub-vc-wd = 0.1245, \\ va+wb+tc+ud = 5.3452, \\ at^2-av^2-2ctv+bu^2-bw^2-2duw = 1.4352, \\ ct^2-cv^2+2atv+du^2-dw^2+2buw = 0.9896, \\ at^3-3atv^2+cv^3-3cvt^2+bu^3-3buw^2+dw^3-3dwu^2 = 0.3464, \\ ct^3-3ctv^2-av^3+3avt^2+du^3-3duw^2-bw^3+3bwu^2 = 3.1345. \end{cases}$$

The right-hand sides of the equations are the parameters of the system and have been chosen at random. The total degree of this system equals 576. When partitioning the set of unknowns into $Z = \{\{a, b, c, d\}, \{t, u, v, w\}\}$, the 2-homogeneous Bézout bound equals 193. The BKK bound equals 121. In [38] the number of solutions with a generic choice of the parameters, the so-called coefficient-parameter bound, is reported to equal 32. However, there is a type error in the formulation of the system, as presented in [38], so that for the original problem, presented in [41] and in [40], there can be only four regular solutions, for random right-hand sides. There are only two real symmetrical solutions. Note that in [40], this system has been reduced to a quadratic univariate equation.

4.3.6. *Butcher's Problem.* The next system belongs to the POSSO test suite, available at the site gauss.dm.unipi.it by anonymous ftp.

$$\begin{cases} zu+yv+tw-w^2-1/2w-\frac{1}{2} = 0, \\ zu^2+yv^2-tw^2+w^3+w^2-1/3t+4/3w = 0, \\ xzv-tw^2+w^3-1/2tw+w^2-1/6t+2/3w = 0, \\ zu^3+yv^3+tw^3-w^4-3/2w^3+tw-5/2w^2-1/4w-\frac{1}{4} = 0, \\ xzuv+tw^3-w^4+1/2tw^2-3/2w^3+\frac{1}{2}tw-7/4w^2-3/8w-\frac{1}{8} = 0, \\ xzv^2+tw^3-w^4+tw^2-3/2w^3+2/3tw-\frac{7}{6}w^2-1/12w-\frac{1}{12} = 0, \\ -tw^3+w^4-tw^2+3/2w^3-1/3tw+13/12w^2+7/24w+\frac{1}{24} = 0. \end{cases}$$

This example clearly illustrates the efficiency of the mixed volume as root count. The total degree equals 4608. The 4-homogeneous Bézout number equals 1361. With the algorithm presented in [49] a set structure, which yields the generalized Bézout number 605, can be obtained. The BKK bound equals 24 and can be computed in less than 1 minute. There are five isolated solutions and a component of solutions: $t = -1 = w$, $z = 0 = y$, and $u, v \in \mathbb{C}$.

Table 1. Characteristics of the polynomial systems.

Section	Description of the applications	Characteristics					
		n	r	d	B	$V_n(\mathcal{P})$	N
4.3.1	Nine point	8	1	5,764,801	645,120	83,977	8,652
4.3.2	PUMA robot	8	8	256	16	16	16
4.3.3	Camera	6	3	64	20	20	20
4.3.4	Inverse position	11	11	1,024	320	288	16
4.3.5	Heart dipole	8	8	576	193	121	4
4.3.6	Butcher	7	7	4,608	605	24	5
4.3.7	Cyclic n-roots	5	5	120	108	70	70
		6	6	720	504	156	156
		7	7	5,040	3,960	924	924
		8	8	40,320	20,352	2,560	1,152

4.3.7. *The Cyclic n-Roots Problem.* The following application belongs to a family of systems which have been presented in [2], [6], and [7]. The general formulation goes as follows:

$$\begin{cases} f_k(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{k} x_{(i+j) \bmod n}, & k = 1, 2, \ldots, n-1, \\ f_n(\mathbf{x}) = \prod_{j=1}^{n} x_j - 1. \end{cases}$$

In Table 1 the performance of the mixed volume as root count, compared with the Bézout bounds, can be seen. This application also demonstrated the #P-hardness of the problem of mixed-volume computation. Augmenting the dimension n leads to a significantly harder problem.

Note that when the system has to be solved, it is better to apply the following transformation: $y_i = x_i/x_n$, $i = 1, 2, \ldots, n-1$, after dividing the kth equation by x_n^k, for $k = 1, 2, \ldots, n$, as proposed in [17]. Here, the last unknown y_n only appears in the last equation, which means that the system can be solved more efficiently. However, the original formulation has been used here for solving the system.

4.3.8. *Execution Times.* In Table 1 the characteristics of each application are summarized. The meaning of the columns is as follows. The first and second columns provide a label and a short description of the application. The following columns respectively list the dimension n, the number of different polytopes r in the tuple, the total degree d, a generalized Bézout bound B, the mixed volume $V_n(\mathcal{P})$, and the number of isolated solutions N in $\mathbb{C}_0^n$.

Table 2 lists the number of mixed cells $\#\Delta$ and the cardinality $\#\lhd$ of the triangulation of the polytope used in the Cayley trick. The second part of the table contains the execution times for solving the applications. It should be stressed that these timings are only meaningful in relative comparison to each other and that they are only meant to give an idea of the performance of the current implementation of the algorithms. There are three stages in solving a polynomial system by polyhedral homotopy continuation. First, there is the computation of the mixed volume which can be done by either the

Table 2. Cardinalities and execution times for root counting and solving.

	Cardinalities		Times for root counting and solving (cpu. sec.)					
Section	#△	#◁	mvc	dmvc	Cayley	sphc	dphc	cont.
4.3.1	13,339	—	—	2,162.0	—	—	—	—
4.3.2	1	580	3.5	5.3	27.1	11.5	8.0	14.2
4.3.3	12	392	6.4	27.3	14.3	46.2	46.7	51.3
4.3.4	17	—	669.1	4,282.0	—	1,036.1	1,938.0	1,920.2
4.3.5	36	—	101.4	349.0	—	763.6	509.1	668.0
4.3.6	4	1,867	53.2	67.0	109.3	77.3	49.2	109.1
4.3.7	14	166	0.6	0.7	1.7	77.4	53.2	26.0
	25	1,109	5.5	6.8	28.7	337.2	217.6	181.7
	124	13,180	84.1	90.3	1,010.7	9,322.3	5,400.3	2,192.4
	268	—	853.7	882.9	—	—	—	—

static lifting algorithm (mvc), the dynamic lifting algorithm (dmvc), or the Cayley trick (Cayley). The second stage consists of solving a system with randomly chosen coefficients. Timings are given for the static (sphc) and dynamic (or incremental) polyhedral homotopy continuation (dphc) methods. Finally, timings for the third and last stage, the continuation to the target system (cont.), are listed. A "—" in the table indicates that the computations on our DS 5000/240 were too expensive to perform.

It can be seen that the dynamic lifting algorithm often requires more work than the static lifting algorithm. This is due to the fact that in the dynamic lifting algorithm, the factorizations cannot be shared (see the last paragraph of Section 3.8). However, in general this additional work pays off because the polyhedral continuation can be done more efficiently, as dynamic lifting is very capable in controlling the magnitude of the lifting values.

5. Conclusions

Three different algorithms have been investigated for computing mixed volumes by means of mixed subdivisions: the Cayley trick, static, and dynamic lifting. The key idea of the paper is the presentation of conservative lifting functions which allow the construction of regular triangulations without the randomness assumption, generally required by all other approaches. This has led to the construction of well-conditioned polyhedral homotopies for computing all isolated solutions to polynomial systems, which provides an important elaboration of the ideas presented in [30].

The Cayley trick is efficient when either it is desired to compute all cells, i.e., also the cells that do not contribute to the mixed volume, or when the system is semimixed and the total number of cells compared with the number of mixed cells is not exponentially large. The fact that the lifting function is again ruled out can be considered an advantage. The disadvantage of this approach is that, for fully mixed systems, the number of mixed cells is much less than the total number of cells. Note that the space complexity of the Cayley trick can be overcome by using reverse enumeration methods, developed by Avis and Fukuda, see [1]. However, these techniques cannot be applied to remove the

randomness assumption on the lifting function and hence offer less control on the growth of the lifting values than the dynamic lifting algorithm.

For fully mixed systems, the lifting method, based on Betke's idea, allows only the computation of the mixed cells and has to be preferred. The static lifting algorithm, with randomized lifting and with properly worked out feasibility tests, provides a very efficient way to compute mixed subdivisions and mixed volumes. It is used in the dynamic lifting algorithm which turns out to be less efficient, due to the fact that factorizations can no longer be shared. Nevertheless, the extra work done by the dynamic lifting algorithm pays off when it comes to constructing polyhedral homotopies for solving polynomial systems. The efficiency of both the static and dynamic lifting algorithms depends largely on the efficiency by which the feasibility tests can be worked out, which is determined by the efficiency of the linear programming solver. So, linear programming forms the computational bottleneck of both algorithms.

Another important conclusion of this project is that computing the mixed volume is in practice no harder than solving the system by tracking all solution paths. The incremental aspect of solving polynomial systems, in [51] applied for computing the solutions inside a bounded domain, also provides more insight into the complexity of homotopy continuation for this problem, see [47] for the complexity analysis of Bézout's theorem.

Finally, some important open problems can be mentioned. First, there is the proof of the conjecture on the connectivity of the mixed cells. Furthermore, from an algorithmical point of view, it would be interesting to develop algorithms, analogous to the flipping mechanisms proposed in [16], which transform placeable (mixed) subdivisions into any desired regular (mixed) subdivisions, with, e.g., either a minimum or a maximum number of cells. Last, but not least, the BKK bound does not provide an exact root count for many applications. It would be worthwhile to develop a systematic approach to reformulate problems to an equivalent formulation with a lower mixed volume, e.g., like was done with the cyclic n-roots problem.

Acknowledgments

The authors are grateful to Birkett Huber for explaining the Cayley trick and to Paul Pedersen for introducing the connectivity property by means of fans. This work also benefited from many valuable discussions with Pierre Verlinden. Last, but not least, the authors wish to thank the anonymous referees for their careful reading and many valuable comments.

References

1. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8(3):295–313, 1992.
2. J. Backelin and R. Fröberg. How we proved that there are exactly 924 cyclic 7-roots. *Proceedings of ISSAC-91*, pages 103–111. ACM, New York, 1991.
3. D. N. Bernshteĭn. The number of roots of a system of equations. *Functional Anal. Appl.*, 9(3):183–185, 1975. Translated from *Funktsional. Anal. i Prilozhen.*, 9(3):1–4, 1975.

4. U. Betke. Mixed volumes of polytopes. *Arch. Math.*, 58:388–391, 1992.
5. L. J. Billera and B. Sturmfels. Fiber polytopes. *Ann. of Math.*, 135(3):527–549, 1992.
6. G. Björk and R. Fröberg. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symbolic Comput.*, 12(3):329–336, 1991.
7. G. Björk and R. Fröberg. Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In M. Gyllenberg and L. E. Persson, editors, *Analysis, Algebra and Computers in Mathematical Research*, pages 57–70. Lecture Notes in Mathematics, volume 564. Dekker, New York, 1994.
8. A. D. Bryuno and A. Soleev. Local uniformization of branches of a space curve, and Newton polyhedra. *St. Petersburg Math. J.*, 3(1):53–82, 1992. Translated from *Algebra i Analiz*, 3(1):67–101, 1991.
9. Yu. D. Burago and V. A. Zalgaller. *Geometric Inequalities*, Grundlehren der mathematischen Wissenschaften, volume 285. Springer-Verlag, Berlin, 1988.
10. J. F. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. *Proceedings of the 10th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 89–104. Springer-Verlag, New York, 1993.
11. J. Canny and J. M. Rojas. An optimal condition for determining the exact number of roots of a polynomial system. *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, pages 96–101. ACM, New York, 1991.
12. K. L. Clarkson, K. Melhorn, and R. Seidel. Four results on randomized incremental constructions. In A. Finkel and M. Jantzen, editors, *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, Cachan, France, February 1992, pages 463–474. Lecture Notes in Computer Science, volume 577. Springer-Verlag, Berlin, 1992.
13. M. E. Dyer and A. M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.
14. M. Dyer, P. Gritzmann, and A. Hufnagel. On the complexity of computing mixed volumes. *SIAM J. Comput.*, to appear.
15. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. ETACS Monographs on Theoretical Computer Science, volume 10. Springer-Verlag, Berlin, 1987.
16. H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *Proceedings of the Eighth Annual Symposium on Computational Geometry*, pages 43–52. ACM, New York, 1992.
17. I. Z. Emiris. Sparse Elimination and Applications in Kinematics. Ph.D. thesis, Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1994.
18. I. Emiris and J. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. Technical Report 839, Computer Science Division, University of California, Berkeley, CA, 1994. Also in *J. Symbolic Comput.*, 11(1):1–33, 1996.
19. M. A. Facello. Implementation of a randomized algorithm for Delaunay and regular triangulations in three dimensions. *Comput. Aided Geom. Design*, 12(4):349–370, 1995.
20. O. D. Faugeras and S. Maybank. Motion from point matches: multiplicity of solutions. *Internat. J. Comput. Vision*, 4:225–246, 1990.
21. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, San Francisco, CA, 1979.
22. I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. Generalized Euler integrals and A-hypergeometric functions. *Adv. in Math.*, 84:255–271, 1990.
23. I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, Boston, MA, 1994.
24. P. Gritzmann and V. Klee. Mathematical programming and convex geometry. In P. M. Gruber and J. M. Wills, editors, *Handbook of Convex Geometry*, volume A, chapter 2.7, pages 627–674. North-Holland, Amsterdam, 1993.
25. P. Gritzmann and V. Klee. On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In R. Schneider, T. Bisztriczky, P. McMullen, and A. I. Weiss, editors, *Polytopes: Abstract, Convex and Computational*, pages 373–466. Kluwer, Boston, MA, 1994.
26. P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM J. Discrete Math.*, 6(2):246–269, 1993.

27. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics, volume 2. Springer-Verlag, Berlin, 1988.
28. L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.
29. B. Huber. Numerically solving sparse polynomial systems. Presented at AMS–IMS–SIAM Summer Research Conference on Continuous Algorithms and Complexity, Mount Holyoke College, South Hadley, MA, June 1994.
30. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541–1555, 1995.
31. M. M. Kapranov, B. Sturmfels, and A. V. Zelevinsky. Chow polytopes and general resultants. *Duke Math. J.*, 67(1):189–218, 1992.
32. A. G. Khovanskiĭ. Newton polyhedra and the genus of complete intersections. *Functional Anal. Appl.*, 12(1):38–46, 1978. Translated from *Funktsional. Anal. i Prilozhen.*, 12(1), 51–61, 1978.
33. A. G. Kushnirenko. Newton polytopes and the Bézout theorem. *Functional Anal. Appl.*, 10(3):233–235, 1976. Translated from *Funktsional. Anal. i Prilozhen.*, 10(3), 82–83, 1976.
34. S. R. Lay. *Convex Sets and Their Applications*. Wiley, New York, 1982.
35. C. W. Lee. Regular triangulations of convex polytopes. In P. Gritzmann and B. Sturmfels, editors, *Applied Geometry and Discrete Mathematics—The Victor Klee Festschrift*, pages 443–456. DIMACS Series, volume 4. AMS, Providence, RI, 1991.
36. A. Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
37. A. Morgan and V. Shapiro. Box-bisection for solving second-degree systems and the problem of clustering. *ACM Trans. Math. Software*, 13(2):152–167, 1987.
38. A. P. Morgan and A. J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989. Errata: 51:207, 1992.
39. A. P. Morgan, A. J. Sommese, and C. W. Wampler. A product-decomposition theorem for bounding Bézout numbers. *SIAM J. Numer. Anal.*, 32(4):1308–1325, 1995.
40. A. P. Morgan, A. Sommese, and L. T. Watson. Mathematical reduction of a heart dipole model. *J. Comput. Appl. Math.*, 27:407–410, 1989.
41. C. V. Nelsen and B. C. Hodgkin. Determination of magnitudes, directions, and locations of two independent dipoles in a circular conducting region from boundary potential measurements. *IEEE Trans. Biomed. Engrg.*, 28(12):817–823, 1981.
42. P. Pedersen. Private communication, 1994.
43. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, Berlin, 1985.
44. J. M. Rojas. A convex geometric approach to counting the roots of a polynomial system. *Theoret. Comput. Sci.*, 133(1):105–140, 1994.
45. R. Schneider. Polytopes and Brunn–Minkowski theory. In R. Schneider, T. Bisztriczky, P. McMullen, and A. I. Weiss, editors, *Polytopes: Abstract, Convex and Computational*, pages 273–300. Kluwer, Boston, MA, 1994.
46. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, Chichester, 1986.
47. M. Shub and S. Smale. Complexity of Bézout's theorem, I: Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.
48. B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3:207–236, 1994.
49. J. Verschelde and R. Cools. Symbolic homotopy construction. *Appl. Algebra Engrg. Commun. Comput.*, 4(3):169–183, 1993.
50. J. Verschelde and K. Gatermann. Symmetric Newton polytopes for solving sparse polynomial systems. *Adv. in Appl. Math.*, 16(1):95–127, 1995.
51. J. Verschelde and A. Haegemans. Homotopies for solving polynomial systems within a bounded domain. *Theoret. Comput. Sci.*, 133(1):141–161, 1994.
52. J. Verschelde, P. Verlinden, and R. Cools. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31(3):915–930, 1994.
53. C. W. Wampler. Bézout number calculations for multi-homogeneous polynomial systems. *Appl. Math. Comput.*, 51(2–3):143–157, 1992.

54. C. Wampler and A. Morgan. Solving the 6R inverse position problem using a generic case solution methodology. *Mech. Mach. Theory*, 26(1):91–106, 1991.
55. C. W. Wampler, A. P. Morgan, and A. J. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *ASME J. Mech. Design*, 114(1):153–159, 1992.
56. G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics, volume 152. Springer-Verlag, New York, 1995.

Received February 10, 1995, *and in revised form July* 5, 1995.