Mathematical Programming Computation (2021) 13:415–442
https://doi.org/10.1007/s12532-020-00198-z

FULL LENGTH PAPER

Polytope volume by descent in the face lattice and applications in social choice

Winfried Bruns[1] · **Bogdan Ichim**[2,3]

Received: 19 August 2018 / Accepted: 28 October 2020 / Published online: 18 November 2020

Abstract
We describe the computation of polytope volumes by descent in the face lattice, its implementation in Normaliz, and the connection to reverse-lexicographic triangulations. The efficiency of the algorithm is demonstrated by several high dimensional polytopes of different characteristics. Finally, we present an application to voting theory where polytope volumes appear as probabilities of certain paradoxa.

Keywords Rational polytope · Face lattice · Volume · Triangulation · Voting theory

Mathematics Subject Classification 52B20 · 13F20 · 14M25 · 91B12

1 Introduction

The volume of a polytope is a geometric magnitude that has been studied since antiquity—formulas for the area of polygons or the volume of common 3-dimensional polytopes like pyramids were known in Babylonian and Egyptian mathematics. From a modern viewpoint one can say that these formulas are recursive: they reduce volume computations to the measurement of lengths.

The second author was partially supported by a grant of Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, Project Number PN-III-P4-ID-PCE-2016-0157, within PNCDI III.

✉ Bogdan Ichim
bogdan.ichim@fmi.unibuc.ro; bogdan.ichim@imar.ro

Winfried Bruns
wbruns@uos.de

1 FB Mathematik/Informatik, Universität Osnabrück, 49069 Osnabrück, Germany

2 Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, 010014 Bucharest, Romania

3 Simion Stoilow Institute of Mathematics of the Romanian Academy, Research Unit 5, C.P. 1-764, 010702 Bucharest, Romania

In toric algebra and geometry, volumes are almost a synonym for multiplicities, defined as leading coefficients of Hilbert (quasi)polynomials. For this classical connection see Teissier [34] or Bruns and Gubeladze [4, Section 6.E]. The package Normaliz [7] has contained options for the computation of Hilbert series and multiplicities from its beginnings. Until recently, the only approach to volumes was based on lexicographic (or placing) triangulations: the volume of a polytope is obtained as sum of simplex volumes, and the volume of a simplex is essentially computed as a determinant. (We refer the reader to [4] for unexplained algebraic notions and to Sect. 2 for geometric terminology.) The algorithm has been improved in several steps; see [6,8,10] for a description of the present state and [11] for a refined version computing integrals.

An attractive application of polytope volumes is probabilities of certain events in voting theory where results of elections with n candidates can be identified with lattice points in the positive orthant of $\mathbb{R}^N$, $N = n!$. Several classical phenomena, most prominently the Condorcet paradox, can be described by homogeneous linear inequalities. In a suitable probabilistic model, their probabilities for a large number of voters can be computed as lattice normalized volumes of rational polytopes. Already for $n = 4$ one has $N = 24$, so that these computations pose a challenging problem. Nevertheless, quite a number of interesting volumes and Hilbert series have been determined via lexicographic triangulations [9].

When more complicated computations of voting theory turned out inaccessible for the approach by lexicographic triangulations, we decided to add an algorithm for the volume of a polytope P that is based on descent in the face lattice of P. (The twofold meaning of "lattice" is an unfortunate, if customary clash of terminology.) In principle, it follows the classical formulas in reducing the computation of volumes to the measurement of lengths. However, we take a hybrid approach, and compute the volume of a simplex as a determinant. For this reason our descent algorithm behaves quite well also for simplicial polytopes.

We have not yet made precise what we mean by "volume". From our viewpoint, the central invariant for rational polytopes is *lattice (normalized) volume*, for which a unimodular lattice simplex Σ has volume 1 in the affine space spanned by Σ. Lattice normalized volume in $\mathbb{R}^n$ is invariant under the action of $\mathrm{GL}(n, \mathbb{Z})$. This group can be exactly represented in rational arithmetic. In contrast to Euclidean volume, which is invariant under the orthogonal group $O(n)$, it is a rational number also for lower dimensional polytopes (i.e., polytopes of dimension smaller than that of the ambient space). Therefore it is well-suited for precise computation by recursion to faces of a polytope. At the end, the Euclidean volume can be (and is) obtained by conversion from the lattice volume.

When the height $\mathrm{Ht}_F(v)$ of a point v over a facet is also measured by the lattice, then the recursion formula takes the simple form

$$\mathrm{Vol}(P) = \sum_{F \text{ facet of } P} \mathrm{Ht}_F(v)\,\mathrm{Vol}(F)$$

for a point $v \in P$. We call a subset $\mathscr{D}$ of the face lattice a *descent system* if it contains all faces that come up in the successive application of this formula. We try to keep $\mathscr{D}$ small by carefully choosing the points v, but the polytope should not have too many

non-simplex faces to keep the descent system from explosion. The largest descent system that has been computed for this paper has cardinality $> 6 \times 10^8$. This number makes it clear that a careful balance between memory usage and computation time is imperative if a wide range of applications is desired.

Though it is not visible straightaway, there is a triangulation or at least a similar decomposition in the background of the descent algorithm, namely a reverse lexicographic (or pulling) triangulation. The crucial point is *not* to compute it explicitly, but to distill the volume relevant information into a weight function associated with the descent system.

Normaliz is not the first package to exploit descent in the face lattice for volume computations. In Büeler and Enge's package Vinci [12] it appears as the *hot* algorithm, where "hot" is an acronym for "hybrid orthonormalization technique". Vinci is based on the article [13] by Büeler, Enge and Fukuda. In contrast to Normaliz, Vinci uses floating point arithmetic and computes only Euclidean volumes, and it does not contain convex hull computation or vertex enumeration, so that another package is needed for this auxiliary task.

In their very recent paper [20] Emiris and Fisikopoulos discuss probabilistic methods for estimating the volume of a polytope. On p. 38:2 they say that several packages, including Normaliz, "cannot handle general polytopes for dimension $d > 15$". Based on our experience, we cannot fully support this finding.

Section 2 introduces the geometric terminology used in the sequel. In Sect. 3 we discuss lattice volume, height and the linear algebra aspects of the algorithm. Section 4 describes the algorithm and its connection with reverse lexicographic triangulations. We give sample computations in Sect. 5. They are based on Normaliz 3.6.1[1] and include a wide range of polytopes with quite different characteristics and of dimensions between 15 and 55. The descent algorithm has staid essentially unchanged since version 3.6.1.

Section 5 contains also a comparison of the two Normaliz algorithms (triangulation and descent) to vinci's "hot" (descent) and "rch" (reverse lexicographic triangulation), and additionally to vinci's "rlass" (a revised version of Lassere's algorithm [28]) that is based on the same type of recursion, but has no Normaliz analogue. Despite of the same basic algorithmic approach of Normaliz' descent and vinci's hot, the implementations in Normaliz and vinci differ substantially, with significant consequences for applicability. A zip file with input for all computations of this paper can be downloaded from

https://www.normaliz.uni-osnabrueck.de/wp-content/uploads/2020/03/PolytopeVolumes.zip.

Finally, Sect. 6 discusses applications to voting theory. We have included it to demonstrate the efficiency of our algorithm in the field of voting theory. It provided many challenges for us without which this algorithm would not have been developed. We quote from [31]: *It seems, however, that the latest version of Normaliz is, at the present time, the most efficient software tool to obtain the IAC probabilities of electoral outcomes when more than three alternatives are in contention …*

[1] Released July 7, 2018.

2 Polytopes and cones

A *polytope* $P \subset \mathbb{R}^n$ is the convex hull of finitely many points $v_1, \dots, v_m \in \mathbb{R}^n$:

$$P = \operatorname{conv}(v_1, \dots, v_m) = \Big\{\alpha_1 v_1 + \cdots + \alpha_m v_m : 0 \le \alpha_i \le 1, i = 1 \dots, m, \sum_{i=1}^{m} \alpha_i = 1\Big\}.$$

By a fundamental theorem of Minkowski, a polytope can equivalently be described as a compact set that is the intersection of finitely many affine halfspaces:

$$P = \bigcap_{j=1}^{s} H_i^+,$$

where an *affine halfspace* is a set

$$H^+ = \{x \in \mathbb{R}^n : \lambda(x) \ge \beta\}$$

for a nonzero linear form $\lambda \in (\mathbb{R}^n)^*$ and $\beta \in \mathbb{R}$. (Not necessarily compact intersections of halfspaces are called *polyhedra*.) The halfspace H^+ is bounded by the hyperplane $H = \{x \in \mathbb{R}^n : \lambda(x) = \beta\}$.

The two descriptions of polytopes are often called *V-representations* and *H-representations*. If $v_1, \dots, v_m \in \mathbb{Q}^n$, then P is a *rational polytope*; equivalently one can choose the halfspaces to be *rational*: λ has coefficients in $\mathbb{Q}$ and $\beta \in \mathbb{Q}$. The *dimension* of P is the dimension of its affine hull $\operatorname{aff}(P)$. The polytope $P \subset \mathbb{R}^n$ is *full dimensional* if $\dim P = n$.

A *cone* C is the conical hull of finitely many vectors:

$$C = \big\{\alpha_1 v_1 + \cdots + \alpha_m v_m : \alpha_i \ge 0, i = 1 \dots, m\big\}.$$

Equivalently, it is the intersection of finitely many *linear halfspaces* in whose definition $\beta = 0$. The attribute *rational* is given in analogy with polyhedra. A cone is *pointed* if $\{0\}$ is the only vector subspace contained in C. For computations one usually represents a polytope $P \subset \mathbb{R}^n$ as the intersection of a pointed cone $C \subset \mathbb{R}^{n+1}$ with an affine hyperplane $H \cong \mathbb{R}^n$. We introduce this technique in Sect. 3; see the discussion surrounding Equation (2).

Let P be a polytope or cone (or a polyhedron in general). If P is contained in one of the two halfspaces bounded by a hyperplane S, then S is called a *support hyperperplane* of P. A subset $F \subset P$ is called a *face* if $F = P \cap S$ for a support hyperplane S of P. The polytope P is an improper face of P, as well as $\emptyset$ if P is a polytope. There are only finitely many faces, and every face is a polytope or cone, respectively. The maximal proper faces of P are called *facets*. The faces of P are partially ordered by inclusion. They actually form a lattice, the *face lattice* of P. Every face $F \ne P$ is the intersection of facets. The properties of the face lattice are important in the following; for the details see Ziegler [35]. The only point in a face of dimension 0 is a *vertex* of P. A face of dimension 1 is an *edge* if P is a polytope and an *extreme ray* if P is a

pointed cone. The difference $\operatorname{codim} F = \dim P - \dim F$ is called the *codimension* of F. We say that F is a *subfacet* if it has codimension 2. Often subfacets are called *ridges*.

The vertices of a polytope P form the unique minimal set V of points such that $P = \operatorname{conv}(V)$. Similarly a set E containing exactly one vector from each extreme ray is a unique minimal generating set of a pointed cone. It is clear that one wants to work with such minimal descriptions in computations.

In the following we use the term "support hyperplane" in a restricted sense: we additionally require that a support hyperplane S intersects P in a facet. This is justified: $P = \operatorname{aff}(P) \cap H_1^+ \cap \cdots \cap H_s^+$ where $H_1^+, \ldots, H_s^+$ are halfspaces whose support hyperplanes intersect P in pairwise different facets. The intersections $\operatorname{aff}(P) \cap H_i^+$ in such an irredundant representation are uniquely determined. In the fulldimensional case the halfspaces themselves are therefore uniquely determined.

A *simplex* is a polytope of dimension d with exactly $d + 1$ vertices: a triangle in dimension 2, a tetrahedron in dimension 3 etc. A polytope is *simplicial* if all its facets are simplices. If every vertex of a polytope P of dimension d is contained in exactly d facets, then P is *simple*. It follows that every face is contained in exactly $\operatorname{codim} F$ facets.

3 Lattice normalized volume

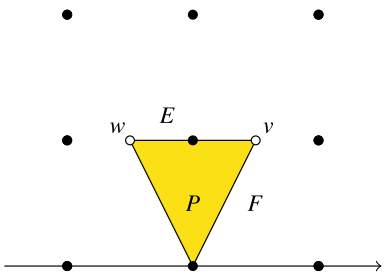
As mentioned in the introduction, we compute the lattice normalized volume of a rational polytope $P \subset \mathbb{R}^n$, i.e., a polytope with vertices in $\mathbb{Q}^n$. Let us explain this notion. The affine hull $A = \operatorname{aff}(P)$ is a rational affine subspace of $\mathbb{R}^n$. First assume that $0 \in A$. Then $L = \operatorname{aff}(P) \cap \mathbb{Z}^n$ is a subgroup of $\mathbb{Z}^n$ of rank $d = \dim P$ (and $\mathbb{Z}^n/L$ is torsionfree). Choose a $\mathbb{Z}$-basis $v_1, \ldots, v_d$ of L. The *lattice (normalized) volume* Vol on A is the Lebesgue measure on A scaled in such a way that the simplex $\operatorname{conv}(0, v_1, \ldots, v_d)$ has measure 1. The definition is independent of the choice of $v_1, \ldots, v_d$ since all invertible $d \times d$ matrices over $\mathbb{Z}$ have determinant ± 1. If $0 \notin A$, then we replace A by a translate $A_0 = A - w$, $w \in A$, and set $\operatorname{Vol}(X) = \operatorname{Vol}(X - w)$ for $X \subset A$. This definition is independent of the choice of w since Vol is translation invariant on A_0. Note that the polytope containing a single point $x \in \mathbb{Q}^n$ has lattice volume 1. (If desired, the definition of lattice volume can be extended to arbitrary measurable subsets of A.)

If P is a lattice polytope, i.e., a polytope with vertices in $\mathbb{Z}^n$, then $\operatorname{Vol}(P)$ is an integer. For an arbitrary rational polytope we have $\operatorname{Vol}(P) \in \mathbb{Q}$. As a consequence, $\operatorname{Vol}(P)$ can be computed precisely by rational arithmetic.

If P has full dimension n, then $\operatorname{Vol}(P) = n!\operatorname{vol}(P)$ where vol denotes the Euclidean volume. So it is only a matter of scaling by the integer $n!$ whether one computes the lattice volume or the Euclidean volume. However, if $\dim P < n$, then $\operatorname{vol}(P)$ need not be rational anymore: the diagonal in the unit square has Euclidean length $\sqrt{2}$, but lattice length 1.

A second invariant we need is the lattice height of a *rational* point x over a rational subspace $H \neq \emptyset$. More generally, one can consider points x such that $\operatorname{aff}(H, x)$ is again rational; for example, this is the case if H is a hyperplane in $\mathbb{R}^n$. If $x \in H$, we

Fig. 1 A rational polytope



set $\mathrm{Ht}_H(x) = 0$. Otherwise let $A = \mathrm{aff}(x, H)$ so that H is a hyperplane in A. Assume first that $0 \in A$.

Then H is cut out from A by an equation $\lambda(y) = \beta$ with a primitive $\mathbb{Z}$-linear form λ on $L = A \cap \mathbb{Z}^n$ and $\beta \in \mathbb{Q}$. That λ is *primitive* means that there exists $y \in L$ such that $\lambda(y) = 1$. Note that λ can be chosen such that it has integral values not only on L, but on the whole of $\mathbb{Z}^n$. In fact, L is not an arbitrary sublattice, but $\mathbb{Z}^n/L$ is torsionfree: L is the intersection of a vector subspace of $\mathbb{R}^n$ with $\mathbb{Z}^n$. This implies that L has a complement M in $\mathbb{Z}^n$ such that $\mathbb{Z}^n = L \oplus M$. So λ can be extended as a $\mathbb{Z}$-linear form on $\mathbb{Z}^n$ by the linear form 0 on M.

With this choice of λ, $\mathrm{Ht}_H(x) = |\lambda(x) - \beta|$ is called the *lattice height* of x over H. (There are exactly two choices for the pair (λ, β), differing by the factor -1.) If $0 \notin A$, then we choose an auxiliary point $v \in A$, replace H by $H - v$, A by $A - v$ and x by $x - v$. (In the algorithm we will only have to deal with the case $0 \in H$.) If P is a rational polytope and F is a facet or, more generally, a face of P, then $\mathrm{Ht}_F(x) = \mathrm{Ht}_H(x)$ where $H = \mathrm{aff}(F)$.

In order to compute the lattice height $\mathrm{Ht}_{\{0\}}(x)$ of a rational point x over the origin, one considers the ray from 0 through x and chooses u as the first nonzero integer point on this ray. Then $v = au$ for some $a \in \mathbb{Q}$, and $\mathrm{Ht}_{\{0\}}(x) = \mathrm{Ht}_{\{x\}}(0) = a$.

In Fig. 1 we have chosen $v = (1/2, 1)$ and $w = (-1/2, 1)$. The primitive linear forms defining E and F are $\lambda(x, y) = y$ and $\mu(x, y) = -2x + y$, respectively. Note that for measuring $\mathrm{Ht}_{\{v\}}(w)$ we must replace μ by $\mu/2$ since $\mu(s)$ and $\mu(t)$ differ by ± 2 for successive lattice points s and t on the line through w and v. Analogously, for measuring $\mathrm{Ht}_{\{v\}}(0)$ we must replace λ by $\lambda/2$ since $\lambda(s)$ and $\lambda(t)$ differ by ± 2 for successive lattice points s and t on the line through 0 and v. We obtain

$$\mathrm{Ht}_F(w) = 2, \quad \mathrm{Ht}_{\{v\}}(w) = 1, \quad \mathrm{Ht}_{\{v\}}(0) = 1/2, \quad \mathrm{Ht}_E(0) = 1,$$
$$\mathrm{Vol}(P) = 1, \quad \mathrm{Vol}(E) = 1, \quad \mathrm{Vol}(F) = 1/2.$$

Proposition 1 *Let $P \subset \mathbb{R}^n$ be a rational polytope, and $v \in P$. Then*

$$\mathrm{Vol}(P) = \sum_{F \text{ facet of } P} \mathrm{Ht}_F(v)\,\mathrm{Vol}(F). \tag{1}$$

Proof P is the union of the "pyramids" $\mathrm{conv}(v, F)$ where F runs through the facets of P. These pyramids intersect in lower dimensional polytopes $\mathrm{conv}(v, G)$ where G

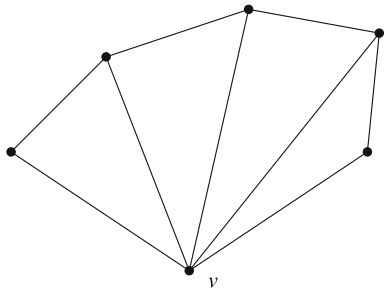


Fig. 2 Decomposition of a polygon into pyramids

is a face of codimension ≥ 2 of P. Since the intersections have measure 0 in the Lebesgue measure on $\mathrm{aff}(P)$ independently of the scaling, the proposition follows by the additivity of the measure, provided $\mathrm{Vol}(\mathrm{conv}(v, F)) = \mathrm{Ht}_F(v)\,\mathrm{Vol}(F)$ for all facets F of P.

To prove this claim, we can triangulate F, and use additivity again, thereby reducing it to the case of a pyramid over a simplex Δ, $\dim \Delta = \dim P - 1$. Then we choose a positive integer k such that kv and all vertices of $k\Delta$ have integer coordinates. This scales $\mathrm{Vol}(\mathrm{conv}(v, \Delta))$ as well as $\mathrm{Ht}_\Delta(v)\,\mathrm{Vol}(\Delta)$ by the factor k^d, $d = \dim P$. Therefore we can finally assume that v is a vertex of a lattice simplex P and $F = \Delta$ is its opposite facet. Under these hypotheses [4, 3.9] says exactly what we need: $\mathrm{Vol}(P) = \mathrm{Ht}_F(v)\,\mathrm{Vol}(F)$. In [4] the lattice volume of a lattice polytope is called its *multiplicity*; we will explain this terminology in Remark 3. For the convenience of the reader we include a proof of [4, 3.9] below. □

Figure 2 illustrates Proposition 1 in a simple case. It is clear that one should take v as a vertex of P in order to minimize the number of nonzero summands in Equation (1). Our choice of v will be discussed in the next section.

Remark 1 The proposition holds for all $v \in \mathrm{aff}(P)$, provided we replace Ht by its signed variant: In the definition choose the sign of λ in such a way that $\lambda(x) - \beta \geq 0$ for $x \in P$ and set $\mathrm{Ht}_F(y) = \lambda(y) - \beta$ for $y \in \mathrm{aff}(P)$. This is important if one wants to represent P by the signed decomposition into the pyramids $\mathrm{conv}(v, F)$. Normaliz does not use signed decompositions at present.

Together with the observation that a single point has lattice volume 1, the proposition constitutes a complete, recursive algorithm for the computation of lattice volumes, provided one can compute lattice height. In principle, this is the algorithm that we have implemented. However, an implementation of any practical value requires considerable care, as we will see in the next section.

The first useful modification is to stop the recursion if one hits a simplex face in the descent, and to compute the volume of a simplex directly.

Proposition 2 *Let $\Delta \subset \mathbb{R}^n$ be a rational simplex with vertices $v_0, \ldots, v_d$. Choose a basis $u_1, \ldots, u_d$ of the lattice $\mathrm{aff}(\Delta - v_0) \cap \mathbb{Z}^n$. Define the $d \times d$ matrix $T = (t_{ij})$ by the representations $v_i - v_0 = \sum_{j=1}^d t_{ij} u_j$, $i = 1, \ldots, d$. Then*

$$\mathrm{Vol}(\Delta) = |\det(T)|.$$

Proof This follows immediately from the substitution rule for Lebesgue integrals applied to the constant function $f = 1$. See [4, 2.C] for an algebraic proof. □

Remark 2 (a) With the help of Proposition 2 one can easily complete the proof of Proposition 1 without a reference to [4]. After a parallel translation we can assume that $v_0 = 0$ is a vertex of Δ and $v = v_d$ (v as as in Proposition 2). Let U be the $\mathbb{Q}$-vector subspace spanned by $v_1, \dots, v_d$ and U' its subspace spanned by $v_1, \dots, v_{d-1}$. Then we define sublattices $L \subset \mathbb{Z}^n$ and $L' \subset L$ by $L = \mathbb{Z}^n \cap U$ and $L' = \mathbb{Z}^n \cap U'$. Evidently L/L' is torsionfree. In other words, L' is a direct summand of L. This means, we can complete a $\mathbb{Z}$-basis $u_1, \dots, u_{d-1}$ of U' by $u_d \in L$ to a $\mathbb{Z}$-basis of L.

Let Δ' be the $(d-1)$-subsimplex $\operatorname{conv}(0, v_1, \dots, v_{d-1})$. With the notation of Proposition 2, we obtain $\operatorname{Vol}(\Delta) = |t_{dd}| \operatorname{Vol}(\Delta')$ by Laplace expansion of $\det(T)$ along the last column of T. For Proposition 1 it remains to show that $|t_{dd}| = \operatorname{Ht}_{\Delta'}(v_d)$. In fact, the primitive $\mathbb{Z}$-linear form λ on L that vanishes on L' has value ± 1 on u_d. So $|\lambda(v_d)| = |t_{dd}|$.

(b) As an anonymous referee pointed out, one can prove Proposition 1 by using the notion of determinant of an affine lattice; see Martinet [29, Prop. 1.3.4]. This approach reduces Proposition 1 to an assertion about Euclidean volume.

If we follow the definition of lattice height, then it is clear that we must choose a vertex of F as the origin of the coordinate system for every face F that comes up in the recursive application of Proposition 1. This complication disappears if $0 \in \mathbb{R}^n$ is a vertex of every face F involved. The reduction of the general case to the special situation is by the standard operation of homogenization.

Normaliz represents a rational polytope P in homogeneous coordinates as follows: C is a pointed cone generated by integral vectors $v_1, \dots, v_m$, δ is a primitive $\mathbb{Z}$-linear form on $\mathbb{Z}^n$ such that $\delta(x) > 0$ for all $x \in C$, $x \neq 0$, and

$$P = \{x \in C : \delta(x) = 1\}. \tag{2}$$

(In the terminology of [4], δ defines a grading on $\mathbb{Z}^n$.) If P is not already given in this form, we can easily realize it as such by introducing a homogenizing $(n+1)$-th coordinate: we replace $P \subset \mathbb{R}^n$ by $P' = P \times \{1\} \subset \mathbb{R}^{n+1}$, set $C = \mathbb{R}_+ P'$ and $\delta(x) = x_{n+1}$ for $x = (x_1, \dots, x_{n+1})$. Consequently one can directly assume that P is given by Equation (2). Then it is natural to pass to $\overline{P} = \operatorname{conv}(0, P)$. All faces of $\overline{P}$ have 0 as a vertex, except P and its faces. Under a mild condition we have $\operatorname{Vol}(P) = \operatorname{Vol}(\overline{P})$, and in the general case we can easily find the correcting factor:

Proposition 3 *Suppose that the rational polytope P is given as in Equation (2). If* $\operatorname{aff}(P)$ *contains an integral point, then* $\operatorname{Vol}(P) = \operatorname{Vol}(\overline{P})$. *More generally, if k is the smallest positive integer such that* $\operatorname{aff}(kP)$ *contains an integral point, then* $\operatorname{Vol}(P) = k \operatorname{Vol}(\overline{P})$.

Proof It is enough to discuss the general case. That the parallels $\operatorname{aff}(jP)$, $1 \leq j < k$, do not contain lattice points implies that $k \mid \delta(x)$ for all $x \in L = \operatorname{aff}(\overline{P}) \cap \mathbb{Z}^n$. On the other hand, $\delta(y) = k$ for some $y \in \operatorname{aff}(kP) \cap L$. Therefore δ/k is a primitive linear form on L. Clearly $\operatorname{Ht}_P(0) = 1/k$, and we can apply Proposition 1. □

If, in the situation of Proposition 3, $\dim P = n-1$ or P itself contains a lattice point, then evidently $\mathrm{Vol}(P) = \mathrm{Vol}(\overline{P})$.

The discussion above is summarized in the next proposition. It describes exactly the arithmetic of the practical computation.

Proposition 4 *Let $v_1, \dots, v_m \in \mathbb{Z}^n$ and suppose $C = \mathbb{R}_+ v_1 + \cdots + \mathbb{R}_+ v_m$ is a pointed cone, generated by $v_1, \dots, v_m$. Let δ be a primitive $\mathbb{Z}$-linear form on $\mathbb{Z}^n$ with $\delta(x) > 0$ for all nonzero $x \in C$. Set $P = \{x \in C : \delta(x) = 1\}$ and suppose that $d = \dim P \geq 1$. As above, we set $\overline{P} = \mathrm{conv}(0, P)$.*

1. *For $v \in C$, $v \neq 0$, one has*

$$\mathrm{Vol}(\overline{P}) = \frac{1}{\delta(v)} \sum_{F \text{ facet of } P} \mathrm{Ht}_{\overline{F}}(v)\, \mathrm{Vol}(\overline{F}).$$

2. *Let F be a facet of P and $\overline{F} = \mathrm{conv}(0, F)$ the corresponding facet of $\overline{P}$. Suppose that $\overline{F}$ is cut out from $\overline{P}$ by the $\mathbb{Z}$-linear form λ with $\lambda(x) \geq 0$ for $x \in \overline{P}$. Let $u_1, \dots, u_{d+1}$ be a $\mathbb{Z}$-basis of $\mathrm{aff}(\overline{P}) \cap \mathbb{Z}^n$ and set $g = \gcd(\lambda(u_1), \dots, \lambda(u_{d+1}))$. Then $\mathrm{Ht}_{\overline{F}}(v) = \lambda(v)/g$.*
3. *Suppose that $m = d+1$ and that $v_1, \dots, v_m$ are linearly independent. With $u_1, \dots, u_{d+1}$ as in (2), one has*

$$\mathrm{Vol}(\overline{P}) = \frac{1}{\delta(v_1) \cdots \delta(v_m)} |\det(T)|,$$

where $T = (t_{ij})$ is the $m \times m$-matrix defined by $v_i = \sum_{j=1}^m t_{ij} u_j$, $i = 1, \dots, m$.

Proof For (1) we observe that $v/\delta(v) \in P$. Therefore $\mathrm{Ht}_P(v/\delta(v)) = 0$ and we need to sum in Proposition 1 (applied to $v/\delta(v)$) only over the facets $\neq P$ of $\overline{P}$. They are exactly the facets of type $\overline{F}$ with F a facet of P. Since $0 \in \overline{F}$, the function $\mathrm{Ht}_{\overline{F}}$ is linear so that $\mathrm{Ht}_{\overline{F}}(v/\delta(v)) = \mathrm{Ht}_{\overline{F}}(v)/\delta(v)$.

For (2) we note that the primitive linear form on $\mathrm{aff}(\overline{P}) \cap \mathbb{Z}^n$ that computes $\mathrm{Ht}_{\overline{F}}$ is indeed λ/g.

(3) follows from Proposition 2 after scaling $v_1, \dots, v_m$ by $1/\delta(v_1), \dots, 1/\delta(v_m)$, respectively, and setting $v_0 = 0$. □

Remark 3 (a) The number k of Proposition 3 is called the *grading denominator* by Normaliz, for good reason as the proof shows. The user can choose whether Normaliz should compute $\mathrm{Vol}(P)$ or $\mathrm{Vol}(kP)$, together with the corresponding Euclidean volume.

(b) Suppose that $\mathrm{aff}(P)$ contains a lattice point. Then the Ehrhart function, the lattice point enumerator of iP, $i \in \mathbb{N}$, is a quasipolynomial $q(i)$ of degree $d = \dim P$, with constant leading coefficient $\mathrm{Vol}(P)/d!$. If the grading denominator is $k > 1$, then the components $q^{(j)}$ of the quasipolynomial are zero for $j \not\equiv 0\ (k)$, but the components $q^{(j)}$ for $j \equiv 0\ (k)$ again have constant leading coefficient $\mathrm{Vol}(P)/d!$. By the similarity with (or interpretation as) a Hilbert function it is justified to call

Vol(P) the *multiplicity* of P. This extends the standard usage of "multiplicity" for lattice polytopes (see [4, Section 6.E]).

(c) Normaliz contains all the linear algebra over $\mathbb{Z}$ that is necessary for the computations of Proposition 4. There are however aspects that deserve mentioning. For the linear algebra operations mentioned below, Normaliz uses the transformation of integer matrices to row echelon form by the Gaussian algorithm over $\mathbb{Z}$, which in its turn is based on the Euclidean algorithm for gcd computations. The determinant of a full rank square matrix is then the product of the diagonal elements of its rowechelon transform.

The basis $u_1, \ldots, u_{d+1}$ in Proposition 4(2) is obtained by saturating the sublattice of $\mathbb{Z}^n$ that is generated by $v_1, \ldots, v_m$. Since m can be extremely large, it saves a substantial amount of time to compute the saturation from a small subset that generates a sublattice of the same rank. Therefore Normaliz tries a random selection that is increased if the rank should not yet suffice.

The saturation itself is computed by a twofold orthogonalization. Let L be a sublattice of $\mathbb{Z}^n$. Then we compute the orthogonal sublattice

$$L^\perp = \{x \in \mathbb{Z}^n : \langle x, y\rangle = 0 \text{ for all } y \in L\}.$$

The saturation of L is given by $L^{\perp\perp}$, as it not hard to see. The computation of $L^\perp$ amounts to solving a homogeneous system of linear equations, and in its turn this task is reduced to the computation of an Hermite normal form [14, 2.4.3].

The computation of the primitive linear form λ amounts to solving a homogeneous system of linear equations as well.

It is difficult to control the size of intermediate results in linear algebra over $\mathbb{Z}$. Normaliz uses a twofold strategy to deal with overflows if the user tries a computation with 64 bit integers. The linear algebra operations are constantly monitored for overflows, and if such an overflow occurs for an intermediate result, the whole computation, for example the computation of an Hermite normal form, is repeated in GMP arithmetic. If an overflow occurs in a final result, then Normaliz starts from scratch in GMP arithmetic. Of the examples considered in Sect. 5 only one (par-24) uses GMP arithmetic.

The rational numbers that are computed as the volumes of rational polytopes often have very large numerators and denominators. They must not be interpreted as consequences of overflows of 64 bit integers in the linear algebra operations. They are inevitable because we must divide by the products of the degrees of the generators in Proposition 4(1) and (3) and further increase by the addition of these fractions.

4 Descent sytems

The discussion in the previous section has made it clear that we should compute Vol($\overline{P}$), given a rational polytope in homogenized coordinates. This is automatically taken care of by the use of Proposition 4. All faces of $\overline{P}$ that come up in the recursive use of Equation (1) are of type $\overline{F}$ where F is a face of P. Therefore we can work directly with P in the combinatorial description of the implementation. To simplify

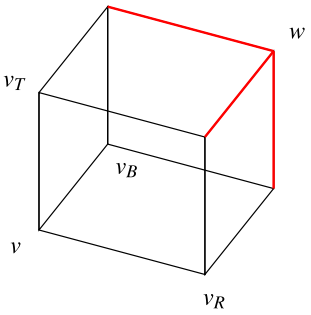


Fig. 3 A 3-dimensional cube

notation in this section, we assume that $\delta(v_i) = 1$ for all i (with the notation of Proposition 4). Otherwise $\mathrm{Ht}_{\dots}(v_i)$ must be divided by $\delta(v_i)$ whenever it appears. Moreover, we identify $\mathrm{Vol}(P)$ and $\mathrm{Vol}(\overline{P})$.

As an example let us discuss a 3-dimensional cube (or any polytope that is combinatorially equivalent to it).

All vertices of P itself have the same number of opposite facets. Let us choose v as in Fig. 3. Then the opposite facets of v are T, B and R, namely the top, the back and the right side of P. Thus

$$\mathrm{Vol}(P) = \mathrm{Ht}_T(v)\,\mathrm{Vol}(T) + \mathrm{Ht}_B(v)\,\mathrm{Vol}(B) + \mathrm{Ht}_R(v)\,\mathrm{Vol}(R).$$

A moment of thought shows that the best choice of the vertices of T, B and R are v_T, v_B and v_R respectively: only 3 edges appear as opposite facets, namely the 3 edges emanating from the vertex w that is antipodal to v. The edges are simplices so that we can compute their volumes as determinants, and no further recursion is necessary. Altogether we have constructed a 3-level system $\mathscr{D} = (\mathscr{D}_0, \mathscr{D}_1, \mathscr{D}_2)$

$$\mathscr{D}_0 = \{P\},\ \mathscr{D}_1 = \{T, B, R\},\ \mathscr{D}_2 = \{T \cap B, T \cap R, B \cap R\}$$

of faces and distinguished vertices v, v_T, v_B, v_R in the nonsimplex faces that allow the recursive computation of $\mathrm{Vol}(P)$.

Definition 1 A *descent system* $\mathscr{D} = (\mathscr{D}_i : i = 0, \dots, d-1)$ for a polytope P of dimension d is a family of sets $\mathscr{D}_i$ of faces F together with a map $F \mapsto v(F)$ assigning a vertex $v(F) \in F$ to every nonsimplex $F \in \bigcup_i \mathscr{D}_i$ such that the following conditions are satisfied for all i:

1. every $F \in \mathscr{D}_i$ is a $(d-i)$-dimensional face of P;
2. if G is a facet of the nonsimplex face $F \in \mathscr{D}_i$ and $v(F) \notin G$, then $G \in \mathscr{D}_{i+1}$;
3. if $G \in \mathscr{D}_{i+1}$, then there exists $F \in \mathscr{D}_i$ such that G is a facet of F not containing $v(F)$.

(There is no need to introduce $\mathscr{D}_d$ since all edges of P are simplices.)

It is immediately clear that a memoryless depth-first recursion would be a bad choice: it does not take into account that lower dimensional faces appear in a large

number of higher dimensional ones, and would therefore be computed over and over again. (Compare the numbers $\#\mathscr{D}$ and $\#\Sigma_{\mathcal{F}}$ in Table 1.)

We compute the descent system by generation: $\mathscr{D}_{i+1}$ is computed from $\mathscr{D}_i$, and if $\mathscr{D}_i = \emptyset$, the computation is complete; otherwise $\mathscr{D}_i$ is processed itself in a *parallelized* loop. It is enough to store only the consecutive layers $\mathscr{D}_i$ and $\mathscr{D}_{i+1}$ at any time since the recursive application of Equation (1) can be replaced by a forward transfer of the accumulated height information by means of a weight $w(F)$ that is assigned to each face in the descent system. The weight of P is 1, and the total volume $\mathrm{Vol}(P)$, initially set to 0, is accumulated step by step. For each face $F \in \mathscr{D}_i$ we perform the following operations:

1. Decide whether F is a simplex; if so, $w(F)\,\mathrm{Vol}(F)$ is added to $\mathrm{Vol}(P)$, and we are done with F.
2. Otherwise we must find the facets G of F,
3. select the vertex $v = v(F)$ (according to a rule explained below),
4. for each facet G of F not containing v
   (a) compute $\mathrm{Ht}_G(v)$,
   (b) insert G with $w(G) = 0$ into $\mathscr{D}_{i+1}$ if it has not yet been found by an already processed face $F' \in \mathscr{D}_i$,
   (c) increase $w(G)$ by $w(F)\,\mathrm{Ht}_G(v)$.

The implementation deviates from this description in the treatment of simplex faces; see Remark 4(e); however, the algorithm as described makes the theoretical analysis easier.

Proposition 5 *The algorithm computes* $\mathrm{Vol}(P)$ *correctly.*

Proof The only question could be whether the weight $w(F)$ is computed correctly. Let us say that the sequence $\mathcal{F} = (F_0, , \ldots, F_k)$ is a *flag* in $\mathscr{D}$ if $F_i \in \mathscr{D}_i$ for all i (and therefore $F_0 = P$) and F_{i+1} is a face of F_i not containing $v(F_i)$.

Let $F \in \mathscr{D}_k$ be a face of P. It follows from Equation (1) that $\mathrm{Vol}(F)$ contributes to the total volume $\mathrm{Vol}(P)$ with the weight

$$\sum_{\mathcal{F}} \mathrm{Ht}_{F_1}(v_0) \cdots \mathrm{Ht}_{F_k}(v_{k-1})$$

where $\mathcal{F}$ runs through all flags $(F_0, , \ldots, F_{k-1}, F)$ with $v_i = v(F_i)$, $i = 0, \ldots, k-1$. This weight is exactly $w(F)$ as computed by the algorithm. The proposition follows immediately by induction on k. □

Polytopes P are usually given as the convex hull of their vertices (V-description) or as the intersection of halfspaces (H-description), where the hyperplanes bounding the halfspaces define the facets of P. It is clear from Equation (1) that we need both descriptions. Regardless of which of them defines P, one must compute the other one. This is covered by the basic functionality of Normaliz (and many other packages). Once facets and vertices are known, one can compute the incidence matrix of facets and vertices. It is the basis of all combinatorial computations in the face lattice of P.

Since the number of faces in the descent system is potentially very large, the combinatorial details of the implementation are critical. We have tried to find a balance between computation time and memory usage. The main question is what to use as the signature of a face F in the descent system. Since the descent algorithm is meant for polytopes with a moderate number of facets and potentially many vertices, we use the set of facets of P that contain F and not the vertices of F. The set of facets is represented by a bitset.

For the "local" computations within $F \in \mathscr{D}_i$ the faces of F are identified by their vertices. These local computations consist of several steps: (i) selecting the vertices of F as those vertices of P that belong to all facets of P containing F, (ii) finding the facets of F by intersecting F with the facets of P not containing F, (iii) selecting the vertex $v(F)$, computing the heights of $v(F)$ over the facets of F not containing it, and finally (iv) pushing these facets, heights and $w(F)$ to $\mathscr{D}_{i+1}$. We describe step (ii) in more detail in Remark 4(a) below.

Among all candidates for $v(F)$ we choose a vertex v of F that (i) minimizes the set of "opposite" facets of F, and (ii) then minimizes the number of faces $F' \in \mathscr{D}_i$ containing v. While rule (i) is an obvious choice, rule (ii) tries to take v as "exterior" as possible in the set $\bigcup_{F' \in \mathscr{D}_i} F'$, so that the facets sent to $\mathscr{D}_{i+1}$ share as many subfacets as possible. (The choice of v_T, v_T, v_R for the cube illustrates this rule.) The introduction of rule (ii) has reduced the size of the descent systems typically by 20%.

Remark 4 (a) Every face of a polytope P is the intersection of a set of support hyperplanes H of P with P. This implies for a given face F that each facet F' of F is obtained as the intersection $F' = F \cap H'$ with a support hyperplane H' of P, $F \not\subset H'$. After the vertex set of F has been computed as the intersection of all $H \supset F$, we compute the intersections $F \cap H$, $F \not\subset H$. This operation yields a set of faces $\mathfrak{F}$ of F. In general, $\mathfrak{F}$ contains also non-facets of F, and a face $F'' \in \mathfrak{F}$ can be cut out by several hyperplanes H.

Despite of these observations, finding the facets F' of F in $\mathfrak{F}$ is a purely set theoretic task: we must find the maximal elements in $\mathfrak{F}$. These are exactly the facets of F, and each such facet F' satisfies $\dim F' = \dim F - 1$.

All the operations just mentioned use only the facet-vertex incidence vectors of P. There is no need to compute dimensions of faces which would be an alternative for selecting the facets among the elements of $\mathfrak{F}$.

(b) Normaliz tests whether the polytope P is simple. For simple polytopes the situation is simpler (!): If F is a face of P and $H \not\supset F$ a support hyperplane of P, then either $F \cap H = \emptyset$ or $G = F \cap H$ is a facet of F. Moreover, H is the only support hyperplane of P that cuts out G from F.

(c) The vertex sets of G are known for the facets G of $F \in \mathscr{D}_i$ that go into $\mathscr{D}_{i+1}$. Storing them with G would accelerate the computation somewhat, but would require considerably more memory, making computations for polytopes with large vertex sets impossible.

(d) One could modify rule (i) for the selection of $v(F)$ by counting only nonsimplex facets that contain v. Experiments have shown that this is not a good choice.

(e) Instead of sending simplex facets of $F \in \mathscr{D}_i$ into $\mathscr{D}_{i+1}$ the implementation computes them directly. This has almost no influence on computation time in general,

but reduces memory usage somewhat. For simplicial polytopes the gain is however tremendous.

(f) If the number of faces in $\mathscr{D}_i$ exceeds one million, $\mathscr{D}_i$ is processed in blocks of this size, and each block is freed when it is finished. This reduces memory usage further.

Remark 5 We give an overview of the complexity of the descent algorithm. It is proportional to the total number $\#\mathscr{D} = \sum_i \#\mathscr{D}_i$. With h denoting the number of facets of P and V the number of its vertices, the bit operations *per face* F can be estimated as follows:

1. $O(hV)$ bit operations for finding the vertices of F, the candidates for the facets of F, and selecting the vertex $v(F)$,
2. if P is not simple, $O(h^2V)$ bit operations for selecting the facets among the faces found in (2),
3. $O(h^2 \log \#\mathscr{D})$ bit operations for the insertion of the facets of $F \in \mathscr{D}_i$ into $\mathscr{D}_{i+1}$.

These are rough estimates that do not take into account that many bit operations are implemented as operations of bit sets represented by a vector of words of size 64.

The linear algebra methods over $\mathbb{Z}$ are described in Remark 3. Their complexity is very difficult to estimate. Even authoritative sources such as Cohen [14] do not contain bounds.

The role of the simplices in the descent system raises the suspicion that the algorithm implicitly uses a triangulation or at least a decomposition with similar properties. This is indeed the case. For each complete flag $\mathcal{F} = (F_0, \dots, F_k)$, in which F_k is necessarily a simplex, set

$$\Sigma_{\mathcal{F}} = \operatorname{conv}(v_0, \dots, v_{k-1}, F_k), \qquad v_i = v(F_i).$$

By the choice of $v_0, \dots, v_{k-1}$ and F_k this set is indeed a d-simplex, and one has

$$P = \bigcup_{\mathcal{F}} \Sigma_{\mathcal{F}}.$$

Moreover, the relative interiors of the simplices $\Sigma_{\mathcal{F}}$ are pairwise disjoint, and this property is good enough for volume computations. In general $\Sigma_{\mathcal{F}} \cap \Sigma_{\mathcal{F}'}$ is not a face of both simplices so that the decomposition is not a triangulation in the strong combinatorial sense. If a true triangulation is desired, one has to fix an order of the vertices $v_1, \dots, v_m$ of P beforehand, and for every nonsimplex face F select $v(F)$ as the first vertex that belongs to F. The triangulation constructed in this way is reverse-lexicographic in the sense of the Sturmfels correspondence or pulling in combinatorial terminology (for example, see [4, Section 7.C]) .

The primal algorithm of Normaliz that builds a cone (over a polytope) incrementally by successively adding generators produces a lexicographic (or placing) triangulation. Its construction is discussed in [8]. Lexicographic triangulations have many advantages and go very well with the Fourier-Motzkin elimination in convex hull computations.

However, if a cone or polytope is given by inequalities, then the reverse-lexicographic approach is more natural. Future versions of Normaliz may use it as well for the computations of triangulations. Nevertheless note that its success in volume computation is based on the fact that the number $w(F)$ captures the relevant information of the set of all flags ending in F. We will illustrate this effect by several sample computations in the next section.

5 Sample computations

5.1 The test polytopes

We demonstrate the power of the descent algorithm by some sample calculations. The following polytopes are used:

1. `Strict Borda` is the polytope underlying the computation of the probability of the strict Borda paradox in social choice; see [9] for the details.
2. `Condorcet` is one of the polytopes that appears in relation with the Condorcet's other paradox. It is discussed in Sect. 6.1, where it is labeled as $\mathfrak{Q}_1$.
3. `4 rules` comes from social choice as well. Again, it is discussed in Sect. 6.1.
4. `8x8-score` represents the monoid of "8×8 ordered score sheets" and was discussed in [25] .
5. `6x6-magic` represents the monoid of "6×6 magic squares", that is the monoid of squares of size 6×6 filled with nonnegative integers such that all rows, columns and the two main diagonals have the same sum called the "magic constant".
6. `d-par` is a parallelotope of dimension d produced as a test example.
7. `d-cube` is the unit cube of dimension d.
8. `bool mod` S_5 represents the boolean model for the symmetric group S_5 and `lin ord` S_6 is the linear order polytope for the symmetric group S_6; they belong to the area of statistical ranking, see [32] for example.
9. `A443` and `A543` are monoids defined by the 2-dimensional marginal distributions of the 3-dimensional contingency tables of sizes $4 \times 4 \times 3$ and $5 \times 4 \times 3$. In the classification of Ohsugi and Hibi [30] are listed as open cases and were closed in [5].
10. `cyclo60` represents the cyclotomic monoid of order 60 and was discussed by Beck and Hoşten in [3].
11. `d-cross` is the unit cross polytope of dimension d spanned by the unit vectors $\pm e_i, i = 1, \ldots, d$.

The first 9 polytopes in Table 1 are defined by inequalities and equations whereas the other 8 are lattice polytopes given by their vertices. The computation times in the "primal" and "descent" columns of Table 2 include the conversion from one representation to the other; for those in the "special" column it is superfluous. All computations can be (and were) done in 64 bit arithmetic, with the exception of `24-par` that needs GMP integers.

In Table 1 edim is the dimension of the space in which the polytope is computed – it is dim $P+1$. The number of vertices is denoted by #vert and that of support hyperplanes

Table 1 Numerical data

	edim	# supp	# vert	$\#\mathscr{D}$	# det	$\#\Sigma_{\mathscr{F}}$
`strict Borda`	24	33	6363	4,407,824	901,955	2×10^{10}
`Condorcet`	24	33	51,168	82,524,473	22,222,231	2.7×10^{12}
`4 rules`	24	36	233,644	652,216,133	177,513,245	17.9×10^{12}
`8x8-score`	56	63	6,725,600	6,725,550	343	4.2×10^{40}
`6x6-magic`	24	36	97,548	494,867,792	113,068,158	31.8×10^{12}
`20-par`	21	40	2^{20}	$2^{20} - 21$	380	20!
`24-par`	25	48	2^{24}	$2^{24} - 25$	552	24!
`20-cube`	21	40	2^{20}	$2^{20} - 21$	380	20!
`24-cube`	25	48	2^{24}	$2^{24} - 25$	552	24!
`bool mod` S_5	27	235	120	14,541,872	334,154	2×10^{10}
`lin ord` S_6	16	910	720	19,012,391	2,133,900	5.8×10^9
`A443`	30	4948	48	204,363	22,334	2,654,224
`A543`	36	29387	60	3,049,328	183,519	10^8
`cyclo60`	17	656100	60	1,712,752	149,253	9,188,100
`20-cross`	21	2^{20}	40	1	2^{19}	2^{19}
`24-cross`	25	2^{24}	48	1	2^{23}	2^{23}
`28-cross`	29	2^{28}	56	1	2^{27}	2^{27}

Table 2 Memory usage and times for parallelized volume calculations

	RAM in GB			Time		
	Primal	Special	Descent	Primal	Special	Descent
`strict Borda`	1.03		0.36	5:30:27 h		25.37 s
`Condorcet`			4.26			14:42 m
`4 rules`			34.39			4:50:52 h
`8x8-score`		9.92	12.78		6:02 m	4:38:12 h
`6x6-magic`			22.77			1:59:43 h
`20-par`		< 0.01	1.11		0.06 s	2:35 m
`24-par`		< 0.01	66.67		0.10 s	12:39:49 h
`20-cube`		< 0.01	1.11		< 0.01 s	2:26 m
`24-cube`		< 0.01	20.94		< 0.01 s	11:57:01 h
`bool mod` S_5	1.10		0.69	1:10:07 h		2:28 m
`lin ord` S_6	0.84		1.96	19:03 m		2:22 m
`A443`	0.68		0.05	1.55 s		18.35 s
`A543`	0.94		1.83	30.06 s		26:26 m
`cyclo60`	1.30		96.37	40.2 s		3:13:57 h
`20-cross`	1.96		1.07	12.90 s		11.79 s
`24-cross`	14.79		22.87	2:20 m		3:56 m
`28-cross`	203.43		354.47	57:07 m		1:47:08 h

by #supp. Moreover, $\#\mathscr{D}$ is the total size of the descent system, # det the number of determinants computed by the descent algorithm, and $\#\Sigma_{\mathscr{F}}$ the number of simplices in a decomposition that would be produced by the algorithm. (The triangulations used by the primal algorithm usually have somewhat different sizes, and also the number of computed determinants is most often different.)

5.2 Parallelized volume computations

The computation times in Table 2 are "wall clock times" taken on a Dell R640 system with two Intel™Xeon™Gold 6152 (a total of 44 cores) using 20 parallel threads (of the maximum of 88). The efficiency of the parallelization is discussed below. We define it as the quotient

$$\frac{T_1}{tT_t}$$

where T_1 is the time of the strictly sequential computation, t the number of threads and T_t the time of parallel computation with t threads. The times listed are for the descent algorithm discussed in this paper, the Normaliz primal algorithm using lexicographic triangulations, and special algorithms that can be applied in some cases; see Remark 6(b) and (e).

For the primal algorithm a missing entry in Table 2 means that the computation is not doable in a reasonable amount of time since the triangulation would have $> 10^{12}$ simplices. A missing entry in the 'special" column indicates that Normaliz has no special method for the particular example.

Remark 6 (a) A profiler run of the example `Strict Borda`, which we consider as a typical application of the descent algorithm, shows that $\approx 43\%$ of the computation time are spent on linear algebra, whereas the bitset operations take $\approx 26\%$. The rest goes into preparations and administration.

(b) Among the polytopes calculated by the primal algorithm, `strict Borda` is by no means the biggest (see [8] for much larger computations). However, among the polytopes calculated for [9] it is the largest since most others can be simplified by symmetrization (see [11]). Symmetrization can be applied very efficiently to `8x8-score`, and this is the special algorithm used for it in addition to descent. For it, the triangulation is approximately 10^{34} times as large as the descent system.

(c) Despite of the special algorithm for parallelotopes described in (e), we have run the descent algorithms on some of them since the results are predictable and can therefore be used as tests for correctness. Moreover, they are prototypes of simple polytopes with very few facets, but a large number of vertices.

The polytope `20-par` is an affine image of the `20-cube`. It is not hard to see that the selection rule for vertices in non-simplex faces produces the descent system $\mathscr{D}$ consisting exactly of the faces containing the vertex w antipodal to the vertex $v(P)$, as illustrated for the 3-cube by Figure 3. The simplex faces are the lines emanating from w. In this case the algorithm implicitly produces an affine image of the Knudsen-Mumford triangulation determined by the root system A_{20} (for example, see [4, Section 3.A]). Parallelotopes profit from the special handling of simple polytopes; see Remark 4(b).

(d) Analogous remarks apply to `24-cube` and `24-par`. If one compares the computation times of `24-cube` with its trivial arithmetic to that of `24-par` with substantially more complicated arithmetic, it becomes clear that the bulk of the computation time for these polytopes goes into the (identical) combinatorics. The handling of the very long bitsets representing the vertices in a face is the bottleneck in these computations, as becomes apparent also from `8x8-score`.

(e) As the column "special" shows, there is a tremendously faster approach to the parallelotopes: if P is a d-parallelotope, then $\mathrm{Vol}(P) = d!\,\mathrm{Vol}(\sigma)$ where σ is a "corner" simplex of P spanned by a vertex and its neighbors. If there should be any doubt: this follows from the transformation rule for volumes, once it has been observed for the unit cube.

The recognition of parallelotopes was added to Normaliz for the computation of lattice points in such polytopes, as they appear in numerical mathematics; see Kacwin, Oettershagen and Ullrich [26]. One could of course add a recognizer for cross d-polytopes as well. Again a single simplex would be sufficient: $\mathrm{Vol}(P) = 2^{d-1}\,\mathrm{Vol}(\sigma)$ for every simplex σ spanned by a vertex and an opposite facet. (It is enough to consider the unit cross polytope.)

(f) For the polytopes with a huge number of facets or vertices the transfer of these data between the components of the Normaliz system of course takes its toll.

Table 3 Efficiency of parallelization in %

# threads	1	2	4	8	16	32
R640	100	94	89	84	77	53
System 2	100	98	98	94	81	43

(g) For the polytopes defined by vertices the primal algorithm is usually more efficient, as shown by several of the last 8 polytopes. This was to be expected for those with a large number of non-simplex facets, but for the simplicial cross polytopes the difference is small.

The number of facets is moderate for `bool mod` S_5 and `lin ord` S_6; nevertheless it came as a surprise that the descent algorithm is significantly faster than the primal algorithm.

The cross polytopes are a class for which exact computation seems to be faster than probabilistic methods. The computation time for `18-cross` reported in [20, Table 1] is much higher than ours for `20-cross`. This is of course also true for parallelotopes if one uses the special approach explained in (e) above.

Table 3 documents the efficiency of parallelization on two different systems, the Dell R640 mentioned above and another system equipped with 2 Intel™Xeon™E5-2660 at 2.20GHz (a total of 16 cores and 32 threads). The test example is `Condorcet`. The computation times for a single thread are 213 min on the R640 and 370 min on system 2. Until 16 threads the efficiency of parallelization on both systems is almost equal and very acceptable.

5.3 Normaliz versus vinci

Tables 4 and 5 compare Normaliz, run with a single thread, to vinci [12], [13]. Among the algorithms offered by vinci we took the three that do not need third party software (except cdd) and do not a priori restrict the class of polytopes to which they can be applied. These are

1. rlass, a revised version of Lasserre's algorithm [28];
2. hot, "hybrid orthonormalization technique", a recursive algorithm that is very similar to Normaliz' descent;
3. rch, a revised version of Cohen and Hickey's combinatorial triangulation algorithm [15].

Remark 7 (a) As pointed out in the introduction, vinci computes only Euclidean volumes. Floating point computations may be sufficient for applications as in Sect. 6, and, as a posteriori comparisons with the exact rational computations of Normaliz show, the approximations computed by vinci are very good. However, we do not know of any a priori error bounds for volume computations in floating point.

(b) Not all computations were successful. The letter T in the tables indicates that the computation time exceeds 250 h. The letter R means that the RAM usage exceeds 500 GB. Moreover, LD stands for low dimension (see (d)), and H indicates that the

Table 4 Computation times Normaliz (single thread) versus vinci

	Normaliz 1x			Vinci			
	-s	Primal	Descent	cdd or *lrs*	rlass	hot	rch
`strict Borda`	0.99 s	50:15:12 h	4:33 m	2.38 s	1:25 m	1:18 m	28:32:03 h
`Condorcet`	1.84 s	T	4:18:56 h	26.83 s	R	50:05 m	T
`4 rules`	6.76 s	T	85:38:56 h	9:49 m	R	7:54:26 h	T
`8x8-score`	5:48 m	T	95:09:03 h	*9:54* m	E	T	T
`6x6-magic`	6.13 s	T	31:05:16 h	4:07 m	E	LD	T
`20-par`	12.91 s	T	44:29 m	2:26:38 h	1:53 m	R	T
`24-par`	11:33 m	T	216:09:45 h	*6:30* m	R	T	T
`20-cube`	7.29 s	T	42:55 m	3:20:38 h	33.47 s	R	T
`24-cube`	2:40 m	T	169:47:10 h	*6:38* m	R	T	T
`bool mod` S_5	0.18 s	5:50:36 h	32:57 m	1.69 s	T	LD	LD
`lin ord` S_6	30.98 s	1:48:36 h	40:05 m	2:33 m	H	5:15 m	59:31:08 h
`A443`	0.46 s	7.16 s	4:10 m	1.00 s	H	LD	LD
`A543`	13.65 s	4:55 m	12:14:55 h	2:20 m	H	LD	LD
`cyclo60`	1:32 m	2:20 m	89:09:53 h	35:21:10 h	H	44:05:45 h	85:57:56 h
`20-cross`	5.21 s	9.80 s	18.06 s	2:15:07 h	H	1:22:42 h	1:07:12 h
`24-cross`	1:57 m	2:22 m	5:53 m	*4:24* m	H	T	T
`28-cross`	54:34 m	1:22:34 h	2:28:17 h	*1:36:45* h	H	T	T

Table 5 Memory usage Normaliz (single thread) versus vinci

	Normaliz 1x		Vinci		
	Primal	Descent	rlass	hot	rch
`strict Borda`	0.75	0.3	58.3	1.37	0.003
`Condorcet`	T	4.32	R	42.5	T
`4 rules`	T	35.42	R	317.26	T
`8x8-score`	T	14.27	E	T	T
`6x6-magic`	T	23.06	E	LD	T
`20-par`	T	1.06	72.14	R	T
`24-par`	T	84.54	R	T	T
`20-cube`	T	1.06	26.94	R	T
`24-cube`	T	18.79	R	T	T
`bool mod` S_5	0.79	0.66	T	LD	LD
`lin ord` S_6	0.58	1.97	H	1.85	0.001
`A443`	0.50	0.4	H	LD	LD
`A543`	0.72	1.79	H	LD	LD
`cyclo60`	0.77	66.05	H	1.19	0.2
`20-cross`	0.26	0.93	H	0.32	0.32
`24-cross`	9.71	15.94	H	T	T
`28-cross`	159.39	271.76	H	T	T

number of facets exceeds the preset bound of 254 for rlass. The letter E indicates that vinci ended with an error message whose cause we could not find out.

(c) Convex hull computation and vertex enumeration are not contained in vinci. Instead we used cdd [22] for this step as recommended by the vinci documentation, except in five computations where cdd did not finish within 50 hours. These were done by lrs [2]. The times of lrs are marked by italics in the table. The time spent by cdd or lrs has not been added to the vinci computation times, whereas the Normaliz times contain convex hull and vertex enumeration. These times were measured in separate computations and appear in the `-s` column of Table 4.

The maximum Normaliz time with a single thread for this step is 54:34 m (for `28-cross`) and the second largest is 11:33 m (for `24-par`). Benchmarks for convex hull computations that include cdd and Normaliz can be found in [1] and [27]. Note that the lrs times have the same order of magnitude as the Normaliz times, whereas the cdd times are often much larger.

(d) We did not succeed to compute the volumes of lower dimensional polytopes, i.e., polytopes whose dimension is smaller than that of the ambient space, with vinci. The volume computed by vinci is then 0, which is of course correct with respect to the full dimensional volume of the ambient space, but this information is useless. In general, lower dimensional polytopes have no rational, isometric and full dimensional embedding, as the diagonal of the unit square shows.

The polytopes `strict Borda`, `Condorcet` and `4 rules` are lower dimensional, but for them there exists a workaround, for Euclidean as well as for lattice

volumes. The latter are needed for applications like those in Sect. 6. Let P be one of these polytopes. Then $H = \mathrm{aff}(P)$ is the affine hyperplane through the unit vectors. The lattice height of the origin over this hyperplane is 1, and the Euclidean height is $1/\sqrt{24}$. Set $\overline{P} = \mathrm{conv}(P, 0)$. It is enough to compute one of the volumes of $\overline{P}$ since

$$\mathrm{vol}(\overline{P}) = \frac{1}{24}\sqrt{24}\,\mathrm{vol}(P), \qquad \mathrm{Vol}(\overline{P}) = \mathrm{Vol}(P), \qquad \mathrm{Vol}(\overline{P}) = 24!\,\mathrm{vol}(\overline{P}).$$

The descent algorithm of Normaliz uses the same recursive approach as vinci's hot. But the implementations differ significantly. Normaliz' descent has ben designed for low memory usage, as pointed out in Sect. 4. This is a clear advantage for really large face lattices like those of `Condorcet`, `4 rules` or the parallelotopes, for which the memory usage differs by a factor of 10 or more. On the other hand, when there is no shortage of memory, hot is sometimes significantly faster than Normaliz with a single thread. Of course, the floating point arithmetic of vinci is a general advantage for computation time.

As the primal algorithm of Normaliz, rch evaluates a reverse lexicographic triangulation. On the whole, it is significantly slower than Normaliz lexicographic triangulation method. Both algorithms have modest memory requirements.

Normaliz has no equivalent of rlass. Therefore a direct comparison is not possible. While rlass is certainly ultrafast when it can be applied, its greediness for memory sets a rather tight bound for applications. It is our impression that the recursion tree of rlass is growing extremely fast. (For `bo5` it broke the time barrier without any result, for which we had expected 0 since the polytope is lower dimensional.)

(e) Several computations in the case of four candidates elections by Diss, Kamwa and Tlidi in [19] were done with Normaliz. We tried to repeat them with vinci, but did not succeed since both rlass and hot broke the memory barrier.

6 Application: computations of volumes in four candidates elections

The appearance of rational polytopes in social choice is fully discussed in [9, Section 2] and we use the same notations in the following. We refer the reader to [23] or [24] for extra details and a more extensive treatment. The basic assumption is that each voter has a linear preference order of the candidates in an election. If there are n candidates, then the number of preference orders is $N = n!$. The result of the election is the vector $(x_1, \ldots, x_N)$ that for every i lists the number of voters having preference order $i = 1, \ldots, N$. The further computations are based on the *Impartial Anonymous Culture* (IAC) assumption, see [9, Section 2] for details. (IAC) assumes that for a fixed number k of voters all election results have equal probability. This allows the computation of probabilities, as $k \to \infty$, of certain phenomena as lattice normalized volumes of rational polytopes.

6.1 Four voting rules, same winner

First, we recall four well-known voting rules.

1. The *plurality rule* (PR): the voters cast one vote for their preferred candidate. The *plurality winner* (PW) is the candidate which has the most first places in the preference orders of the voters.
2. The *negative plurality rule* (NPR): it requires the voters to cast one vote against their least preferred candidate. The *negative plurality winner* (NPW) is the candidate which has the fewest last places in the preference orders of the voters.
3. The *majority rule* (MR): all voters preferences are considered and we say that a candidate A "beats" a candidate B by *pairwise majority rule* if there are more voters which prefer A to B than voters that prefer B to A. The *Condorcet winner* (CW), i.e. the majority rule winner, is the candidate which beats all other candidates by the pairwise majority rule. As the Marquis de Condorcet [17] observed, the relation "beats" is nontransitive in general, and one must consider the possibility of Condorcet's paradox, namely an outcome without a Condorcet winner.
4. The *Borda rule* (BR): this is a weighted scoring rule which in the particular case of four candidates assigns 3 points to a candidate for each most-preferred ranking in a voter's preferences, 2 points for each second-place ranking, 1 point for each third-place ranking and zero points for each least-preferred ranking. The *Borda winner* (BW) is the candidate which cumulates the most points.

We want to compute the probability that all four voting rules deliver the same winner in four candidates elections as the numbers of voters k goes to ∞.

Let us choose a candidate A. The polytope $\mathcal{P}$ associated to the event that A is the winner of all four voting rules is cut by 36 inequalities and 1 equation from $\mathbb{R}^{24}$: 24 inequalities $x_i \geq 0$, 3 inequalities for each of the 4 rules fixing A as the winner, and the equation $x_1 + \cdots + x_{24} = 1$; see [9] for several related systems of equations. The combinatorial data of the polytope $\mathcal{P}$ and the computation time are listed in Tables 1 and 2, `4 rules`.

We have obtained

$$\operatorname{vol} \mathcal{P} = \frac{a}{b},$$

where

$$a = 154342951028970694926967872245875694933248780590692865565001570944662802210031839904092203533576766900008697462518883193615863751857064434519917747$$

and

$$b = 197348919941616942863689328362932710624415993010771743164635856670733662509278749736017422249308139949407199308434014022373196020318208000000000000000.$$

The probability that all four voting rules deliver the same winner in four candidates elections may then be computed as

$$4 \cdot \operatorname{vol} \mathcal{P} = \frac{4 \cdot a}{b} \approx 0.312833.$$

We were surprised by this rather small value: even if a Condorcet winner exists, the winner of the actual voting scheme is rather unpredictable. It would of course be possible to analyze the situation further by considering 3 rules versus the 4-th in each case. The computations need some hours, but they are well accessible.

6.2 On Condorcet's other paradox

In [18] Condorcet presents several examples of voting paradoxes that may appear in three candidates elections.

In particular we are interested in [18, Example 4, page 150], illustrating a voting situation in which the Condorcet winner is the same as the plurality winner, but not the Borda winner. We want to compute the probability that this phenomenon will appear in four candidates elections under (IAC).

Set candidate A to be both the plurality and the Condorcet winner. Since the Borda rule gives a total order of the candidates, we have four situations that may appear:

1. A is placed first by the Borda rule. We denote the corresponding polytope by $\mathcal{Q}_1$;
2. A is placed second by the Borda rule. We have to make a choice for the winning candidate. Assume that B beats A by the Borda rule and denote the corresponding polytope by $\mathcal{Q}_2$;
3. A is placed third by the Borda rule. We have to make a choice for the losing candidate. Assume that B and C beat A by the Borda rule (D is placed on last place) and denote the corresponding polytope by $\mathcal{Q}_3$;
4. A is placed last by the Borda rule (or in other words A is the *Borda loser*). We denote the corresponding polytope by $\mathcal{Q}_4$;

All polytopes of this family are cut by 33 inequalities and 1 equation in dimension 24. Not all the inequalities are relevant, however minimizing the number of inequalities does not make the problem easier to solve.

A fast computation shows that the polytope $\mathcal{Q}_4$ has empty absolute interior (i.e., the dimension is < 23), so its 23-dimensional volume is zero. In fact, it is known in general, see [21, Theorem 4], that the Condorcet winner cannot be the Borda loser. Further, there are two independent ways to compute the probability that the Condorcet winner is the same as the plurality winner, but not the Borda winner. It can be computed directly, with the formula:

$$12 \cdot (\operatorname{vol} \mathcal{Q}_2 + \operatorname{vol} \mathcal{Q}_3).$$

It can also be computed indirectly, using the fact that the probability that the Condorcet winner is the same as the plurality winner was computed previously in [33]. We recall from [9, Subsection 2.3] that the volume of the polytope associated is:

$$\operatorname{vol} \mathcal{E} = \frac{1065809825501191644931850 9}{6847565144260608000000000},$$

and the formula is:

$$4 \cdot (\operatorname{vol} \mathcal{E} - \operatorname{vol} \mathcal{Q}_1).$$

We have computed:

$$\operatorname{vol} \mathcal{Q}_1 = \frac{1551436593053676386582045146731502617111545979486042696852104222882000 09}{1102320838271070278766883635115881896290018550251848550368411648000000000},$$

$$\operatorname{vol} \mathcal{Q}_2 = \frac{8007917191946827148905632396266883808060150761021309697108559220076039}{1653481257406605418150325452673822844435027825377772825552617472000000000},$$

$$\operatorname{vol} \mathcal{Q}_3 = \frac{2072705500667484952215435851434572363770941977453049707343465792912717}{16534812574066054181503254526738228444350278253777728255526174720000000000}.$$

Both ways of computing the probability that the Condorcet winner is the same as the plurality winner, but not the Borda winner, deliver the same result, that is:

$$\frac{8215187742013575644127175981410341044437244958766614667842905799367 3107}{1377901047838837848458604543894852370362523187814810687960514560000000000} \approx 0.059621.$$

6.3 Condorcet efficiency of elimination

In this subsection we study the *Condorcet efficiency* of elimination procedures. This is the conditional probability that the Condorcet winner, provided that such winner exists, is elected by a certain voting scheme, as the number of voters $k \to \infty$. We consider the following two voting schemes.

First, *the plurality elimination rule*: this is an iterative procedure, in which, at each voting step, the candidate who obtained the minimum number of first place votes is eliminated. The last candidate non eliminated is the winner. Second, *the negative plurality elimination rule*: similarly, at each voting step the candidate with the maximum number of last place votes is eliminated. For four candidates elections both lead to three-stage elimination procedures, thus our study here completes the data presented in Table 7.4 of [24].

It seems that the simplest way to compute this probability is to consider the complementary event, that is the event that the Condorcet winner is eliminated either in the first or the second round. Notice that, if the Condorcet winner will pass through the first and the second round, then he or she will automatically win the third round, so the study of the third round is not needed.

The outcome that the Condorcet winner is eliminated in the first round of a certain voting scheme is called *the (reverse) strong Borda paradox* and its study was first introduced by the Chevalier de Borda in [16]. The occurrence of the (reverse) strong Borda paradox under both the plurality rule and the negative plurality rule was fully studied in subsection 2.5 of [9] to where we refer the reader for details and we only recall here that the volume of the polytope associated with the strong Borda paradox is

$$\operatorname{vol} \mathcal{B}_{\mathrm{Sg}} = \frac{325451674835828550681491}{6847565144260608000000000} = \operatorname{vol} \mathcal{B}_{\mathrm{RevNPR}},$$

while the volume of the polytope associated with reverse strong Borda paradox is

$$\operatorname{vol} \mathcal{B}_{\mathrm{SgRev}} = \frac{104898234852130241}{21035720123168587776} = \operatorname{vol} \mathcal{B}_{\mathrm{RevPR}}.$$

We also refer the reader to [9, Remark 3 (a)] for extra needed details, which will clarify the second notation used.

Let us denote by $\mathcal{F}$ the polytope corresponding to the event that candidate A is the Condorcet winner, candidate D is eliminated in the first round and candidate A is eliminated in the second round.

Then, the Condorcet efficiency of both elimination rules may be computed with the formula

$$\frac{p_{A=\mathrm{CW}} - \operatorname{vol}\mathcal{B}_{\mathrm{Rev}} - 3\operatorname{vol}\mathcal{F}}{p_{A=\mathrm{CW}}},$$

where $\operatorname{vol}\mathcal{B}_{\mathrm{Rev}}$ should be replaced by $\operatorname{vol}\mathcal{B}_{\mathrm{RevPR}}$ and $\operatorname{vol}\mathcal{B}_{\mathrm{RevNPR}}$.

We have obtained

$$\operatorname{vol}\mathcal{F}_{\mathrm{PR}} = \frac{65375080294032363232154095451613168794052651716 03}{19898897021667735198913285499098497024000000000000000},$$

so that the Condorcet efficiency of the plurality elimination rule turns out to be

$$\frac{12917831227518879529352235926668925725340723482839 7}{13902346267172648655816288737773486080000000000000000} \approx 0.929184,$$

while

$$\operatorname{vol}\mathcal{F}_{\mathrm{NPR}} = \frac{87391394898401644146716674012811354620163132417}{31091026140009682822081785811945799024640000000000},$$

so that the Condorcet efficiency of the negative plurality elimination rule turns out to be

$$\frac{20355237456037077623585217269678606595609864702 07}{21721717074542897707321950780888239308800000000000} \approx 0.937092.$$

It may come as a surprise the fact that the Condorcet efficiency of the negative plurality elimination rule is greater than the Condorcet efficiency of the plurality elimination rule. This is not totally unexpected, considering the data presented in Table 7.4 of [24] for three candidates two-rounds elimination procedures. However, in order to check our results, we have computed the probabilities for all ten possible results that the Condorcet winner may obtain in the three-rounds elimination procedures. The approximative numbers are contained in the tables below. For space reasons we did not include here the full exact data, which is available on request from the authors.

For the plurality elimination rule the probabilities are contained in Table 6.

For the negative plurality elimination rule the probabilities are contained in Table 7.

The entries in both tables should be read as follows: the entry at row i and column j represents an approximation of the conditional probability that the Condorcet winner obtains the i-th place in the first round and j-th place in the second round, under the assumption that such a winner exists. The missing number is the conditional probability that the Condorcet winner is eliminated in the first round, or in other words

Table 6 Probabilities under PR

1-st round	2-nd round		
	I	II	III
I	0.69605467532	0.04320695864	0.00335247384
II	0.06678615010	0.08902016651	0.01327777245
III	0.01396067951	0.02015490336	0.03039424802

Table 7 Probabilities under NPR

1-st round	2-nd round		
	I	II	III
I	0.46569938269	0.07611279571	0.00978979031
II	0.16256921634	0.11815379945	0.01272253146
III	0.04072126773	0.07383508505	0.01771994652

the probability of the (corresponding) reverse Borda paradox. Those probabilities have been computed in subsection 2.5 of [9] (they are also reported in [24, Table 7.5]). More precisely, the probability of the reverse strong Borda paradox under the plurality rule (respectively the negative plurality rule) is $\frac{104898234852130241}{4408976007260798976} \approx 0.02379$ (respectively $\frac{325451674835828550681491}{1435213544030208000000000} \approx 0.02268$). The exact numbers obtained add perfectly, in both cases studies the sum of all ten numbers equals 1.

Remark 8 The examples in this subsection are also computable by pyramid decomposition and symmetrization as discussed in [8,11]. However, this will take several weeks on a quite powerful system in place of minutes on a rather standard computer.

References

1. Assarf, B., Gawrilow, E., Herr, K., Joswig, M., Lorenz, B., Paffenholz, A., Rehn, T.: Computing convex hulls and counting integer points with polymake. Math. Program. Comput. **9**, 1–38 (2017)
2. Avis, D.: lrs: a revised implementation of the reverse search vertex enumeration algorithm. Available at http://cgm.cs.mcgill.ca/~avis/C/lrs.html
3. Beck, M., Hoşten, S.: Cyclotomic polytopes and growth series of cyclotomic lattices. Math. Res. Lett. **13**, 607–622 (2006)
4. Bruns, W., Gubeladze, J.: Polytopes, Rings and K-Theory. Springer, Berlin (2009)
5. Bruns, W., Hemmecke, R., Ichim, B., Köppe, M., Söger, C.: Challenging computations of Hilbert bases of cones associated with algebraic statistics. Exp. Math. **20**, 25–33 (2011)
6. Bruns, W., Ichim, B.: Normaliz: algorithms for affine monoids and rational cones. J. Algebra **324**, 1098–1113 (2010)
7. Bruns, W., Ichim, B., Römer, T., Sieg, R., Söger, C.: Normaliz. Algorithms for Rational Cones and Affine Monoids.https://doi.org/10.5281/zenodo.4246974. Available at http://normaliz.uos.de
8. Bruns, W., Ichim, B., Söger, C.: The power of pyramid decomposition in normaliz. J. Symbol. Comput. **74**, 513–536 (2016)
9. Bruns, W., Ichim, B., Söger, C.: Computations of volumes and Ehrhart series in four candidates elections. Ann. Oper. Res. **280**, 241–265 (2019)

10. Bruns, W., Sieg, R., Söger, C.: Normaliz 2013–2016. In: Böckle, G., Decker, W., Malle, G. (eds.) Algorithmic and experimental methods in algebra, geometry, and number theory, pp. 123–146. Springer, Berlin (2018)
11. Bruns, W., Söger, C.: Generalized Ehrhart series and integration in Normaliz. J. Symbol. Comput. **68**, 75–86 (2015)
12. Büeler, B., Enge, A.: Vinci. Package available from https://www.math.u-bordeaux.fr/~aenge/
13. Büeler, B., Enge, A., Fukuda, K.: Exact volume computation for polytopes: a practical study. In: Polytopes - combinatorics and computation (Oberwolfach, 1997), 131 – 154, DMV Sem., 29. Birkhäuser, Basel (2000)
14. Cohen, H.: A Course in Computational Number Theory. Springer, Berlin (1995)
15. Cohen, J., Hickey, T.: Two algorithms for determining volumes of convex polyhedra. J. Assoc. Comput. Mach. **26**, 401–414 (1979)
16. de Borda, J.-C.Chevalier: Mémoire sur les élections au scrutin. Histoire de'Académie Royale Des Sci. **102**, 657–665 (1781)
17. de Condorcet, N. M.: Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. Imprimerie Royale, Paris (1785)
18. de Condorcet, N.M.: On discovering the plurality will in an election. Appendix to On the constitution and functions of Provincial assemblies, (1788). In: McLean, I., Hewitt, F. (eds.) Condorcet: Foundations of Social Choice and Political Theory, pp. 148–156. Edward Elgar Publishing, Cheltenham (1994)
19. Diss, M., Kamwa, E., Tlidi, A.: The Chamberlin-Courant rule and the k-scoring rules: agreement and Condorcet committee consistency, https://doi.org/10.2139/ssrn.3198184. Preprint available from https://halshs.archives-ouvertes.fr/halshs-01817943/document
20. Emiris, I.Z., Fisikopoulos, V.: Practical polytope volume approximation. ACM Trans. Math. Softw. **44**, 38 (2018)
21. Fishburn, P., Gehrlein, W.V.: Borda's rule, positional voting, and Condorcet's simple majority principle. Public Choice **28**, 79–88 (1976)
22. Fukuda, K.: cddlib. Available at https://people.inf.ethz.ch/fukudak/cdd_home/
23. Gehrlein, W.V., Lepelley, D.: Voting Paradoxes and Group Coherence. Springer, Berlin (2011)
24. Gehrlein, W.V., Lepelley, D.: Elections, Voting Rules and Paradoxical Outcomes. Springer, Berlin (2017)
25. Ichim, B., Moyano-Fernández, J.J.: On the score sheets of a round-robin football tournament. Adv. Appl. Math. **91**, 24–43 (2017)
26. Kacwin, Ch., Oettershagen, J., Ullrich, T.: On the orthogonality of the Chebyshev–Frolov lattice and application. Monatsh. Math. **184**, 425–441 (2017)
27. Köppe, M., Zhou, Y.: New computer-based search strategies for extreme functions of the Gomory–Johnson infinite group problem. Math. Program. Comput. **9**, 419–469 (2017)
28. Lasserre, J.B.: An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$. J. Optim. Theory Appl. **39**, 363–377 (1983)
29. Martinet, J.: Perfect Lattices in Euclidean Spaces. Springer, Berlin (2003)
30. Ohsugi, H., Hibi, T.: Toric ideals arising from contingency tables. Ramanujan Math. Soc. Lect. Note Ser. **4**, 87–111 (2006)
31. Lepelley, D., Ouafdi, A., Smaoui, H.: Probabilities of electoral outcomes: from three-candidate to four-candidate elections. Theor. Decis. **88**, 205–229 (2020)
32. Sturmfels, B., Welker, V.: Commutative algebra of statistical ranking. J. Algebra **361**, 264–286 (2012)
33. Schürmann, A.: Exploiting polyhedral symmetries in social choice. Soc. Choice Welf. **40**, 1097–1110 (2013)
34. Teissier, B.: Monômes, volumes et multiplicités. In: Introduction à la théorie des singularités, II, pp. 127–141. Hermann, Paris (1988)
35. Ziegler, G.: Lectures on Polytopes. Springer, Berlin (1995)