Effective Lattice Point Counting in Rational Convex Polytopes *

Jesús A. De Loera, Raymond Hemmecke
Jeremiah Tauzer, and Ruriko Yoshida

January 21, 2003

Abstract

This paper discusses algorithms and software for the enumeration of all lattice points inside a rational convex polytope: we describe `LattE`, a computer package for lattice point enumeration which contains the first implementation of A. Barvinok's algorithm [8].

We report on computational experiments with multiway contingency tables, knapsack type problems, rational polygons, and flow polytopes. We prove that this kind of symbolic-algebraic ideas surpasses the traditional branch-and-bound enumeration and in some instances `LattE` is the only software capable of counting. Using `LattE`, we have also computed new formulas of Ehrhart (quasi)polynomials for interesting families of polytopes (hypersimplices, truncated cubes, etc).

We end with a survey of other "algebraic-analytic" algorithms, including a "polar" variation of Barvinok's algorithm which is very fast when the number of facet-defining inequalities is much smaller compared to the number of vertices.

1 Introduction

Counting lattice points inside convex polyhedra is a truly fundamental and useful step in many mathematical investigations. It appears, for instance, in the context of Combinatorics [35, 43], Representation Theory [28, 40], Statistics [19, 22], and Number Theory [11, 37]. Lattices and polytopes are at the foundation of Discrete Optimization [25, 41]. This justifies the development of computer software that could count or list all lattice points on an arbitrary *rational* convex polyhedron.

In the 1980's H. Lenstra created an algorithm to **detect** integer points in polyhedra, based on the LLL-algorithm and the idea of short vectors (see [25, 33]). As a consequence, solving integer programming problems with a fixed number of variables can be done in time polynomial in the size of the input. We are not aware of any implementation of Lenstra's original algorithm; but there have been already efforts to investigate the practical value of these ideas. For example, Cook et al. [15] have implemented the integer programming algorithm by Lovász and Scarf [34], which is similar in structure to Lenstra's algorithm. In addition, Aardal and collaborators [1, 2, 3] have written fairly effective modifications of the LLL procedure for testing integer feasibility. In the 1990's, based on work by the

*Research supported by NSF Grant DMS-0073815 and by NSF VIGRE Grant No. DMS-0135345.

geometers Brion, Khovanski, Lawrence, and Pukhlikov, Barvinok created an algorithm to **count** integer points inside polyhedra that runs in polynomial time for fixed dimension (see [8, 9] and the references within). Shortly after Barvinok's breakthrough, Dyer and Kannan [20] modified the original algorithm of Barvinok, which originally relied on Lenstra's result, giving a new proof that integer programming problems with a fixed number of variables can be solved in polynomial time. In Section 2, extending the work initiated in [18], we describe the first ever implementation of Barvinok's algorithm valid for arbitrary rational polytopes; the program `LattE`.

In Section 3 we present some computational experience with our current implementation of `LattE`. We report on experiments with families of well-known rational polytopes: multiway contingency tables, knapsack type problems, and rational polygons. We demonstrate `LattE` competes with commercial branch-and-bound software and solves very hard instances, enumerating some examples that had never been done before. We also tested the performance in the case of two-way contingency tables and Kostant's partition function where special purpose software has been written already [6, 12, 18, 36]. In Section 4 we present formulas for the Ehrhart quasi-polynomials of several hypersimplices and truncations of cubes (e.g. the 24 cell). We show solid evidence that Barvinok's ideas are practical and can be used to solve non-trivial problems, both in integer programming and symbolic computing. In the last section of the paper we survey some other algorithms for lattice point enumeration. We describe *the polar Barvinok approach*. Like the original Barvinok's algorithm it runs in polynomial time when the dimension is fixed but it is in practice faster when the number of facet-defining inequalities is much smaller than the number of vertices.

2 `LattE`'s implementation of Barvinok's algorithm

In 1993 Barvinok [8] gave an algorithm that counts lattice points in convex rational polyhedra in polynomial time when the dimension of the polytope is fixed. In this section, we go through the steps of Barvinok's algorithm, showing how we implemented them in `LattE`. Barvinok's algorithm relies on two important new ideas: the use of rational functions as efficient data structures and the signed decompositions of cones into unimodular cones.

The input data are an $m \times d$ integral matrix M and an m-vector b. These data define a polyhedron $P = \{x \in \mathbb{R}^d | M_i x = b_i, \text{ for } i = 1, 2, \ldots s,\ M_i x \leq b_i, \text{ for } i = s+1, \ldots, m,\ M \in \mathbb{Z}^{m \times d}, \text{ and } b \in \mathbb{Z}^m\}$, where M_i represents the ith row vector of M and b_i represents the ith entry of b. The goal is to output a short formula for the multivariate generating function $f(P) = \sum_{a \in P \cap \mathbb{Z}^d} z^a$. Here, and throughout the paper, $z^a = z_1^{a_1} z_2^{a_2} \ldots z_d^{a_d}$. At the end, $f(P)$ will be written as a sum of "short" rational functions from which we can solve feasibility, counting, or even optimization questions, about the lattice points in P.

Note that when P is a polytope (i.e. a bounded polyhedron), the monomials of $f(P)$ are in bijection with the lattice points and thus $f(P)$ is a (Laurent) polynomial. Counting the lattice points in P is equivalent to evaluating the expression at the vector with all entries 1. Let v be a vertex of P. Then, the *supporting cone* $K(P, v)$ of P at v is $K(P, v) = v + \{u \in \mathbb{R}^d : v + \delta u \in P \text{ for all sufficiently small } \delta > 0\}$. Let $V(P)$ be the vertex set of P. One crucial component of Barvinok's algorithm is the ability to distribute the computation on the vertices of the polytope. This follows from the seminal theorem of Brion [13]:

Theorem 1 (See [13]) *Let P be a rational polyhedra and let $V(P)$ be the vertex set of P. Then,*

$$f(P) = \sum_{v \in V(P)} f(K(P, v)).$$

Example 2 *Consider the integral quadrilateral shown in Figure* 1.

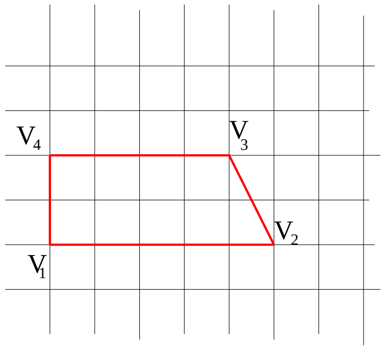

Figure 1: A quadrilateral in Example 2

We obtain four rational generation functions whose formulas are
$f(K_{V_1}) = \frac{1}{(1-z_1)(1-z_2)}, \quad f(K_{V_2}) = \frac{({z_1}^5+{z_1}^4 z_2)}{(1-{z_1}^{-1})(1-z_2^2{z_1}^{-1})},$
$f(K_{V_3}) = \frac{({z_1}^4 z_2+{z_1}^4 {z_2}^2)}{(1-{z_1}^{-1})(1-z_1 {z_2}^{-2})}, \quad f(K_{V_4}) = \frac{z_2^2}{(1-{z_2}^{-1})(1-z_1)}.$
Indeed, the result of adding the rational functions is equal to the polynomial
${z_1}^5 + {z_1}^4 z_2 + {z_1}^4 + {z_1}^4 {z_2}^2 + z_2 {z_1}^3 + {z_1}^3 + {z_1}^3 {z_2}^2 + z_2 {z_1}^2 + {z_1}^2 + {z_1}^2 {z_2}^2 + z_1 z_2 + z_1 + z_1 {z_2}^2 + {z_2}^2 + z_2 + 1.$ $\square$

In order to use Brion's theorem for counting lattice points in convex polyhedra, we need to know how to compute the rational generating function of convex rational pointed cones. For polyhedral cones this generating function is a rational function whose numerator and denominator have a well-understood geometric meaning (see Chapter 4 in [43] and [44] for a clear explanation). We already have a "simple" formula when the cone is simplicial: Let $\{u_1, u_2, \ldots, u_k\}$ be a set of linearly independent integral vectors of $\mathbb{R}^d$, where $k \leq d$. Let K be a cone which is generated by $\{u_1, u_2, \ldots, u_k\}$, in other words, $K = \{\lambda_1 u_1 + \lambda_2 u_2 + \ldots + \lambda_k u_k,$ for some $\lambda_i \geq 0$ and $i = 1, 2, \ldots, k\}$. Consider the parallelepiped $S = \{\lambda_1 u_1 + \lambda_2 u_2 + \ldots + \lambda_k u_k,\ 0 \leq \lambda_i < 1,\ i = 1, 2, \ldots, k\}$.

It is well-known [43] that the generating function for the lattice points in K equals

$$\sum_{\beta \in K \cap Z^d} z^\beta = \Big(\sum_{\tau \in S \cap Z^d} z^\tau\Big) \prod_{i=1}^{k} \frac{1}{1 - z^{u_i}}. \quad (*)$$

Thus, to derive a formula for arbitrary pointed cones one could decompose them into simplicial cones, via a triangulation, and then apply the formula above and the inclusion-exclusion principle (see Proposition 1.2 in [44]). Instead, Barvinok's idea is that it is more efficient to further decompose each simplicial cone into simplicial unimodular cones. A *unimodular* cone is a simplicial cone with generators $\{u_1, \ldots, u_k\}$ that form an integral basis for the lattice $\mathbb{R}\{u_1, \ldots, u_k\} \cap \mathbb{Z}^d$. Note that in this case the numerator of the formula has a single monomial, in other words, the parallelepiped has only one lattice point.

2.1 Simplicial signed decompositions

We now focus our attention on how the cone decomposition is done. To decompose a cone into simplicial cones the first step is to do a triangulation (*triangulation* of a cone C in dimension d is a collection of d-dimensional simplicial cones whose union is C, their interiors are disjoint and any pair intersect in a (possibly empty) common face). There are efficient algorithms, when the dimension is fixed, to carry a triangulation (see [5, 32] for details). In `LattE` we use the well-known Delaunay triangulation which we compute via a convex hull calculation. The idea is to "lift" the rays of the cone into a higher dimension of paraboloid by adding a new coordinate which is the sum of the squares of the other coordinates, take the lower convex hull of the lifted points, and then "project" back those simplicial facets. We use Fukuda's implementation in `CDD` [24] of this lift-and-project algorithm. This is not the only choice of triangulation, and definitely not the smallest one. In Section 5 we discuss some situations when the choice of triangulation in fact gives a better rational function.

In principle, one could at this point list the points of the fundamental parallelepiped, for example, using a fast Hilbert bases code such as `MLP` [26] or `NORMALIZ` [14], and then use formula $(*)$ for a general simplicial cone. Theoretically this is bad because the number of lattice points in the parallelepiped is exponentially large already for fixed dimension. In practice, this can often be done and in some situations is useful. Barvinok instead decomposes each simplicial cone as a (signed) sum of simplicial *unimodular* cones. To be more formal, for a set $A \subset \mathbb{R}^d$, the indicator function $[A] : \mathbb{R}^d \to \mathbb{R}$ of A is defined as

$$[A](x) = \begin{cases} 1 \text{ if } x \in A, \\ 0 \text{ if } x \notin A. \end{cases}$$

We want to express the indicator function of a simplicial cone as an integer linear combination of the indicator functions of simplicial cones. There is a nice valuation from the algebra of indicator functions of polyhedra to the field of rational generating functions (see [9]), and many of its properties can be used in the calculation. For example, the valuation is zero when the polyhedron contains a line.

Theorem 3 (see Theorem 3.1 in [9]) *There is a valuation f from the algebra of indicator functions of rational polyhedra into the field of multivariate rational functions such that for any polyhedron P, $f([P]) = \sum_{\alpha \in P\cap\mathbb{Z}^d} x^\alpha$.*

Therefore once we have a unimodular cone decomposition, the rational generating function of the original cone is a signed sum of "simplicial" rational functions. Next we focus on how to decompose a simplicial cone into unimodular cones.

Let $u_1, u_2, \ldots, u_d$ be linearly independent integral vectors which generate a simplicial cone K. We denote the *index* of K by $\mathrm{ind}(K)$ which tells how far K is from being unimodular. That is, $\mathrm{ind}(K) = |\det(u_1|u_2|\ldots|u_d)|$ which is the volume of the parallelepiped spanned by $u_1, u_2, \ldots, u_d$. It is also equal to the number of lattice points inside the half-open parallelepiped. K is unimodular if and only if the index of K is 1. Now we discuss how we implemented the following key result of Barvinok:

Theorem 4 (see Theorem 4.2 in [9]) *Fix the dimension d. Then, there exists a polynomial time algorithm with a given rational polyhedral cone $K \subset \mathbb{R}^d$, which computes unimodular cones K_i, $i \in I = \{1, 2, \ldots, l\}$, and numbers $\epsilon_i \in \{-1, 1\}$ such that*

$$[K] = \sum_{i\in I} \epsilon_i [K_i].$$

Let K be a rational pointed simplicial cone. Consider the closed parallelepiped

$$\Gamma = \{\alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_d u_d : \ |\alpha_j| \leq (\mathrm{ind}(K))^{-\frac{1}{d}},\ j = 1, 2, \ldots, d\}.$$

Note that this parallelepiped Γ is centrally symmetric and one can show that the volume of Γ is 2^d. Minkowski's First Theorem [41] guarantees that because $\Gamma \subset \mathbb{R}^d$ is a centrally symmetric convex body with volume $\geq 2^d$, there exists a non-zero lattice point w inside of Γ. We will use w to build the decomposition.

We need to find w explicitly. We take essentially the approach suggested by Dyer and Kannan [20]. We require a subroutine that computes the smallest vector in a lattice. We use Lenstra Lenstra Lovasz' (LLL) basis reduction algorithm [25, 41] to find a short vector. Given A, an integral $d \times d$ matrix whose columns generate a lattice, LLL's algorithm outputs A', a new $d \times d$ matrix, spanning the same lattice generated by A. The column vectors of A', $u'_1, u'_2, ..., u'_d$, are short and nearly orthogonal to each other, and each u'_i is an approximation of the shortest vector in the lattice, in terms of Euclidean length. It is well-known [41] that there exists a unique unimodular matrix U such that $AU = A'$.

The method proposed in [20] to find w is the following: Let $A = (u_1|u_2|\ldots|u_d)$, where the u_i are the rays of the simplicial cone we wish to decompose. Compute the reduced basis of A^{-1} using the LLL algorithm. Let A' be the reduced basis of A^{-1}. Dyer and Kannan observed that we can find the smallest vector with respect to the l^∞ norm by searching over all linear integral combinations of the column vectors of A' with small coefficients. We call this search the *enumeration step*. Let λ be the smallest vector in the lattice spanned by A' with respect to the l^∞ norm. We know that there exists a unique unimodular matrix U such that $A' = A^{-1}U$. Minkowski's theorem for the l^∞ norm implies that for the non-singular matrix A', there exists a non-zero integral vector z such that $\lambda = \|A'z\|_\infty \leq |\det(A')|^{1/d}$, where $\|.\|$ is the infinity norm of the vector space $\mathbb{R}^d$. See statement 23 in page 81 in [41]. We can set

$$\|\lambda\|_\infty \leq |\det(A')|^{1/d} = |\det(A^{-1}U)|^{1/d} = |\det(A^{-1})\det(U)|^{1/d}$$

$$= |\det(A^{-1})|^{1/d} = |\det(A)|^{-1/d} = |\mathrm{ind}(K)|^{-1/d}.$$

Since A^{-1} and A' span the same lattice, there exists an integral vector $w \in \mathbb{R}^d$ such that $\lambda = A^{-1}w$. Then, we have

$$w = A\lambda.$$

Note that w is a non-zero integral vector which is a linear integer combination of the generators u_i of the cone K *with possibly negative coefficients*, and with coefficients at most $|\mathrm{ind}(K)|^{-1/d}$. Therefore, we have found a non-zero integral vector $w \in \Gamma$. In `LattE`, we try to avoid the enumeration step because it is very costly. Instead, we choose λ to be the shortest of the columns in A'. This may not be the smallest vector, but for practical purposes, it often decreases the $|\mathrm{ind}(K)|$ just like for the shortest vector. Experimentally we have observed that we rarely use the enumeration step.

In the next step of the algorithm, for $i = 1, 2, \ldots, d$, we set

$$K_i = \mathrm{cone}\{u_1, u_2, \ldots, u_{i-1}, w, u_{i+1}, \ldots, u_d\}.$$

Now, we have to show that for each i, $\mathrm{ind}(K_i)$ is smaller than $\mathrm{ind}(K)$. Let $w = \sum_{i=1}^d \alpha_i u_i$. Then, we have

$$\mathrm{ind}(K_i) \quad = \quad |\det((u_1|u_2|\ldots|u_{i-1}|w|u_{i+1}|\ldots|u_d))|$$

$$
\begin{aligned}
&= |\alpha_i| |\det((u_1|u_2|\ldots|u_{i-1}|u_i|u_{i+1}|\ldots|u_d))| \\
&= |\alpha_i| \mathrm{ind}(K) \leq (\mathrm{ind}(K))^{\frac{d-1}{d}} .
\end{aligned}
$$

There is one more technical condition that w needs to satisfy. This is that w and $u_1, \ldots, u_d$ belong to an open half-space (see Lemma 5.2 [8]). This is easy to achieve as either the w we found or $-w$ satisfy this condition. We can now decompose the original cone K into cones K_i for $i = 1, 2, \ldots, d$, of smaller index, $[K] = \sum \pm [K_i]$. This sum of indicator functions carries signs which depend on the position of w with respect to the interior or exterior of K. We iterate this process until K_i becomes a unimodular cone for $i = 1, 2, \ldots, d$. For implementing Barvinok's decomposition of cones, we use the package `NTL` by Victor Shoup [42] to compute the reduced basis of a cone and to compute with matrices and determinants. All our calculations were done in exact long integer arithmetic using the GNU-package `GMP` [23] or the routines integrated in `NTL`. Here is the pseudocode of the algorithm and an example.

Algorithm 5 *(Barvinok's Decomposition of a Simplicial Cone)*
Input: *A simplicial cone $C = cone\{u_1, u_2, \ldots, u_d\}$ given by its generators.*
Output: *A list of unimodular cones and numbers ϵ_i as in Theorem 4.*
Set two queues Uni and NonUni.
`If` *C is unimodular, then $Uni = Uni \cup \{C\}$.*
`Else` *$NonUni = NonUni \cup \{C\}$.*
`While` *NonUni is not empty* `do`
    Take a cone $K \in NonUni$ and set $A = (u_1, u_2, \ldots, u_d)$, be a matrix whose columns are the rays of K.
    Compute the smallest vector λ in the lattice, with respect to
        l^∞, which is spanned by the column vectors of A^{-1}.
    Find a non-zero integral vector z such that $\lambda = A^{-1}z$.
    Check whether vectors $z, u_1, u_2, \ldots, u_d$ are in an open half plane.
        `if` *not, set $z := -z$.*
    Set $K_i = cone\{u_1, \ldots, u_{i-1}, z, u_{i+1}, \ldots, u_d\}$
        and set $A_i = (u_1, \ldots, u_{i-1}, z, u_{i+1}, \ldots, u_d)$ for $i = 1, 2, \ldots, d$.
    `For` *$i = 1, 2, \ldots, d$,*
        `if` *$\det(A_i)$ and $\det(A)$ have the same sign,*
            assign $\epsilon_{K_i} = \epsilon_K$,
        `else` *$\epsilon_{K_i} = -\epsilon_K$.*
    `For` *$i = 1, 2, \ldots, d$,*
        `if` *K_i is unimodular, $Uni = Uni \cup \{K_i\}$,*
        `else` *$NonUni = NonUni \cup \{K_i\}$.*
`Print` *all elements in Uni.*

It is very important to remark that, in principle, one also needs to keep track of lower dimensional cones present in the decomposition for the purpose of writing the inclusion-exclusion formula of the generating function $f(K)$. For example in Figure 2 we have counted in twice a ray, thus it needs to be removed.

But this is actually not necessary thanks to a *Brion's polarization trick* (see remark 4.3 in [9]): Let K^* be the dual cone to K. Apply the iterative procedure above to K^* instead of K, ignoring the lower dimensional cones. This can be done because once we polarize the result back, the contribution of the lower dimensional cones is zero with respect to the valuation that assigns to an indicator function its generating function counting the

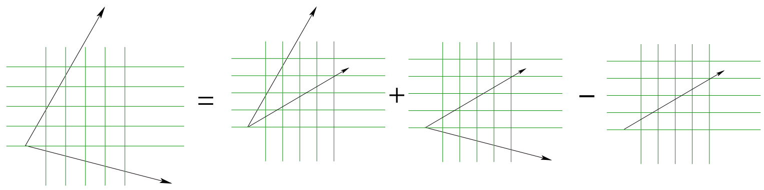

Figure 2: Contribution of lower dimensional cones

lattice points (see Corollary 2.8 in [9]). In the current implementation of `LattE` we do the following:

1. Find the vertices of the polytope and their defining supporting cones.
2. Compute the polar cone to each of the cones.
3. Apply the Barvinok decomposition to each of the polars.
4. Polarize back the cones to obtain a decomposition, into full-dimensional unimodular cones, of the original supporting cone at each vertex.
5. Recover the generating function of each cone and, by Brion's theorem, for the whole polytope.

.

Here is an example of how we carry out the decomposition.

Example 6 *Let K be a cone generated by $(2,7)^T$ and $(1,0)^T$. Let*

$$A = \begin{pmatrix} 2 & 1 \\ 7 & 0 \end{pmatrix}.$$

Then, we have $\det(A) = -7$ *and*

$$A^{-1} = \begin{pmatrix} 0 & \frac{1}{7} \\ 1 & \frac{-2}{7} \end{pmatrix}.$$

The reduced basis A' of A^{-1} and the unimodular matrix U for the transformation from A^{-1} to A' are: $A' = \begin{pmatrix} \frac{1}{7} & \frac{3}{7} \\ \frac{-2}{7} & \frac{1}{7} \end{pmatrix}$, and $U = \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix}$. By enumerating the column vectors, we can verify that $(\frac{-2}{7}, \frac{1}{7})^T$ is the smallest vector with respect to l^∞ in the lattice generated by the column vectors of A^{-1}. So, we have $z = (1,0)^T$. Then, we have two cones:

$$\begin{pmatrix} 2 & 0 \\ 7 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The second cone is unimodular of index -1 which is the same sign as the determinant of A. Thus, $Uni =\ Uni \cup \{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\}$, and assign to it $\epsilon = 1$. The first cone has determinant 2. So, we assign $\epsilon = -1$. Since the first cone is not unimodular, we have $NonUni = NonUni$ $\cup\{\begin{pmatrix} 2 & 0 \\ 7 & 1 \end{pmatrix}\}$. Set

$$A = \begin{pmatrix} 2 & 0 \\ 7 & 1 \end{pmatrix}.$$

Then, we have $\det(A) = 2$ *and*

$$A^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{-7}{2} & 1 \end{pmatrix},\ A' = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{2} & \frac{1}{2} \end{pmatrix} \text{ and } U = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}.$$

Since $\lambda = (\frac{1}{2}, \frac{-1}{2})^T$ *is the smallest vector with respect to* l^∞*, we have* $z = (1,3)^T$*. So, we get two cones:*

$$\begin{pmatrix} 2 & 1 \\ 7 & 3 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}.$$

The first matrix has negative determinant which is not the same sign as the determinant of its parent matrix A*. Since* $\epsilon_A = -1$*, we assign to the first cone* $\epsilon = 1$ *and the second one has positive determinant, so we assign to it* $\epsilon = 1$*. Since both of them are unimodular, we take them into Uni and since NonUni is empty, we end while loop and print all elements in Uni.*

This gives a full decomposition:

$$cone\{\begin{pmatrix} 2 \\ 7 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\}$$

$$= \ominus cone\{\begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\} \oplus cone\{\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\} \oplus cone\{\begin{pmatrix} 2 \\ 7 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix}\}.$$

□

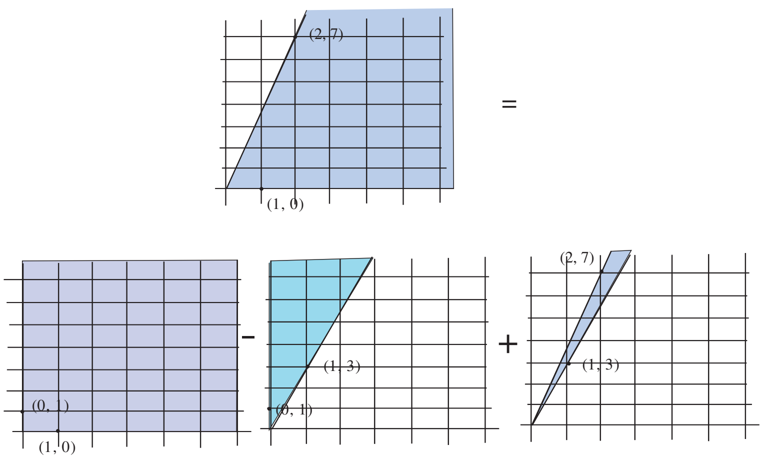

Figure 3: Example of Barvinok's decomposition

From the previous example, we notice that the determinant of each cone gets much smaller in each step. This is not an accident as Theorem 4 guarantees that the cardinality of the index set I of cones in the decomposition is bounded polynomially in terms of the determinant of the input matrix. We have looked experimentally at how many levels of iteration are necessary to carry out the decomposition. We observed experimentally that it often grows linearly with the dimension. We tested two kinds of instances. We used random square matrices whose entries are between 0 and 9, thinking of their columns as

Dimension	Height of tree	# of cones	| determinant|	Time (seconds)
2	1.33	2.53	11.53	0
3	2.87	12.47	55.73	0.005
4	3.87	65.67	274.667	0.153
5	5.87	859.4	3875.87	0.25
6	7.47	10308	19310.4	3.67
7	8.53	91029.4	72986.3	41.61
8	10.67	2482647.533	1133094.733	2554.478

Table 1: Averages of 15 random matrices for computational experiences

Dimension	# of vertices	# of unimodular cones at a vertex cone	Time (seconds)
$B_3 = 4$	6	3	0.05
$B_4 = 9$	24	16	0.15
$B_5 = 16$	120	125	0.5
$B_6 = 25$	720	1296	7.8

Table 2: The numbers of unimodular cones for the Birkhoff polytopes

the generators of a cone centered at the origin. We tested from 2×2 matrices all the way to 8×8 matrices, and we tested fifteen random square matrices for each dimension. We show the results in Table 1. For computation, we used a 1 GHz Pentium PC machine running Red Hat Linux.

The second set of examples comes from the Birkhoff polytope B_n of doubly stochastic matrices [41]. Each vertex of the polytope is a permutation matrix which is a 0/1 matrix whose column sums and row sums are all 1 [41]. We decompose the cone with vertex at the origin and whose rays are the $n!$ permutation matrices. The results are reported in Table 2.

2.2 From cones to rational functions and counting

Once we decompose all cones into simplicial unimodular cones, it is easy to find the generating function attached to the ith cone K_i. In the denominator there is a product of binomials of the form $(1 - z^{B_{ij}})$ where B_{ij} is the jth ray of the cone K_i. Thus the denominator is the polynomial $\prod(1 - z^{B_{ij}})$. How about the numerator? The cone K_i is unimodular, thus it must have a single monomial z^{A_i}, corresponding to the unique lattice point inside the fundamental parallelepiped of K_i. Remember that the vertex of K_i is one of the vertices of our input polytope. If that vertex v has all integer coordinates then $A_i = v$, else v can be written as a linear combination $\sum \lambda_j B_{ij}$ where all the λ_i are rational numbers and can be found by solving a system of equations (remember the B_{ij} form a vector space basis for $\mathbb{R}^d$). The unique lattice point inside the parallelepiped of the cone K_i is simply $\sum \lceil \lambda_j \rceil B_{ij}$ (see Lemma 4.1 in [9]).

Brion's theorem says the sum of the rational functions coming from the unimodular cones at the vertices is a polynomial with one monomial per lattice point inside the input polytope. One can think that to compute the number of lattice points inside of a given convex polyhedron one substitutes the value of 1 at each of the variables. Unfortunately, this is a singularity of all the rational functions. Instead we discuss now the method used in `LattE` to compute this value. The typical generating function of lattice points inside a unimodular cone forms:

$$E[i]\frac{z^{A_i}}{\prod(1-z^{B_{ij}})},$$

where z^a is multinomial of dimension d, each A_i (cone vertex) and B_{ij} (a generator of cone i) are integer vectors of length d, i ranges over all cones given, j ranges over the generators of cone i, and $E[i]$ is 1 or -1. Adding these rational functions and simplifying would yield the polynomial function of the lattice point of the polytope. Now this is practically impossible as the number of monomials is too large. But calculating the number of monomials in this polynomial is equivalent to evaluating the limit as z_i goes to 1 for all i.

We begin by finding an integer vector λ and making the substitution $z_i \to t^{\lambda_i}$. This is with the intention of obtaining a univariate polynomial. To do this, λ must be picked such that there is no zero denominator in any cone expression, i.e. no dot product of λ with a B_{ij} can be zero. Barvinok showed such λ can be picked in polynomial time by choosing points on the moment curve. Unfortunately, this method gives large values on the entries of λ. Instead we try random vectors with small integer entries, allowing small increments if necessary, until we find λ. Since we are essentially trying to avoid a measure zero set, this process terminates very quickly in practice.

After substitution, we have expressions of the form $\pm t^{num_i}/\prod(1-t^{den_{ij}})$, where num_i, and den_{ij} are integers. Notice that this substitution followed by summing these expressions yields the same polynomial as would result from first summing and then substituting. This follows from the fact that we can take Laurent series expansions, and the sum of Laurent series is equal to the Laurent series of the sum of the original expressions.

Also, note that we have the following identity:

$$\sum_{\alpha\in P\cap\mathbb{Z}^d} z^\alpha = \sum_{i=1}^{\#\ of\ cones} E[i]\frac{z^{A_i}}{\prod(1-z^{B_{ij}})}.$$

After substitution we have the following univariate (Laurent) polynomial such that:

$$\sum_{\alpha\in P\cap\mathbb{Z}^d} t^{\sum_{i=1}^d \lambda_i\alpha_i} = \sum_{i=1}^{\#\ of\ cones} E[i]\frac{t^{num_i}}{\prod(1-t^{den_{ij}})}.$$

With the purpose of avoiding that the exponents at the numerators get too large we factor out a power of t, say t^c. Now we need to evaluate the sum of these expressions at $t=1$, but we cannot evaluate these expressions directly at $t=1$ because each has a pole there. Consider the Laurent expansion of the sum of these expressions about $t=1$. The expansion must evaluate at $t=1$ to the finite number $\sum_{\alpha\in P\cap\mathbb{Z}^d} 1$. It is a Taylor expansion and its value at $t=1$ is simply the constant coefficient. If we expand each expression about $t=1$ individually and add them up, it will yield the same result as adding the expressions and then expanding (again the sum of Laurent expansions is the Laurent expansion of the sum of the expressions). Thus, to obtain the constant coefficient of the sum, we add up the constant coefficients of the expansions about $t=1$ of each summand. Computationally, this is accomplished by substituting $t=s+1$ and expanding about $s=0$ via a polynomial division. Summing up the constant coefficients with proper accounting for $E[i]$ and proper decimal accuracy yields the desired result: the number of lattice points in the polytope.

Before the substitution $t = s + 1$ we rewrite each rational function in the sum (recall t^c was factored to keep exponents small).

$$\sum E[i] \frac{t^{num_i - c}}{\prod(1 - t^{den_{ij}})} = \sum E'[i] \frac{t^{num'_i}}{\prod(t^{den'_{ij}} - 1)},$$

in such a way that $den'_{ij} > 0$ for all i, j. This requires the powers of t at each numerator to be modified and the sign $E'[i]$ is also adjusted. Then the substitution $t = s + 1$ yields

$$\sum E'[i] \frac{(1+s)^{num'_i}}{\prod((1+s)^{den'_{ij}} - 1)},$$

where it is evident that, in each summand, the pole $s = 0$ has an order equal to the number of factors in the denominator. This is the same as the number of rays in the corresponding cone and we denote this number by d.

Thus the summand for cone i can be rewritten as $E'[i]s^{-d}P_i(s)/Q_i(s)$ where $P_i(s) = (1+s)^{num_i}$ and $Q_i(s) = \prod^d((1+s)^{den'_{ij}} - 1)/s)$. $P_i(s)/Q_i(s)$ is a Taylor polynomial whose s^d coefficient is the contribution we are looking for (after accounting for the sign $E'[i]$ of course). The coefficients of the quotient $P_i(s)/Q_i(s)$ can be obtained recursively as follows: Let $Q_i(s) = b_0 + b_1 s + b_2 s^2 + \ldots$ and $P_i(s) = a_0 + a_1 s + a_2 s^2 + \ldots$ and let $\frac{P_i(s)}{Q_i(s)} = c_0 + c_1 s + c_2 s^2 + \ldots$. Therefore, we want to obtain c_d which is the coefficient of the constant term of P_i/Q_i. So, how do we obtain c_d from $Q_i(s)$ and $P_i(s)$? We obtain this by the recurrence relation as the following:

$$c_0 = \frac{a_0}{b_0},$$

$$c_k = \frac{1}{b_0}(a_k - b_1 c_{k-1} - b_2 c_{k-2} - \ldots - b_k c_0) \text{ for } k = 1, 2, \ldots.$$

In order to obtain c_d, only the coefficients $a_0, a_1, \ldots, a_d$ and $b_0, b_1, \ldots, b_d$ are required.

Example 7 *(A triangle). Let us consider three points in 2 dimension such that $V_1 = (0,1)$, $V_2 = (1,0)$, and $V_3 = (0,0)$. Then, the convex hull of V_1, V_2, and V_3 is a triangle in 2 dimension. We want to compute the number of lattice points by using the residue theorem. Let K_i be the vertex cone at V_i for $i = 1,2,3$. Then, we have the rational functions:*

$$f(K_1) = \frac{y}{(1-y^{-1})(1-xy^{-1})}, f(K_2) = \frac{x}{(1-x^{-1})(1-x^{-1}y)}, f(K_3) = \frac{1}{(1-x)(1-y)}.$$

We choose a vector λ such that the inner products of λ and the generators of K_i are not equal to zero. We choose $\lambda = (1,-1)$ in this example. Then, reduce multivariate to univariate with λ, so that we have:

$$f(K_1) = \frac{t^{-1}}{(1-t)(1-t^2)}, f(K_2) = \frac{t}{(1-t^{-1})(1-t^{-2})}, f(K_3) = \frac{1}{(1-t)(1-t^{-1})}.$$

We are simplifying more. We want to have all the denominators have positive exponents. We simplify them in order to eliminate negative exponents in the denominators with simple algebra. Then, we have:

$$f(K_1) = \frac{t^{-1}}{(1-t)(1-t^2)}, f(K_2) = \frac{t^4}{(1-t^1)(1-t^2)}, f(K_3) = \frac{-t}{(1-t)(1-t)}.$$

We factor out t^{-1} from each rational function, so that we obtain:

$$f(K_1) = \frac{1}{(1-t)(1-t^2)}, f(K_2) = \frac{t^5}{(1-t^1)(1-t^2)}, f(K_3) = \frac{-t^2}{(1-t)(1-t)}.$$

We substitute $t = s+1$ and simplify them to the form $\frac{P(s)}{s^d Q(s)}$:

$$f(K_1) = \frac{1}{s^2(2+s)}, f(K_2) = \frac{1+5s+10s^2+10s^3+5s^4+s^5}{s^2(2+s)}, f(K_3) = \frac{-(1+2s+s^2)}{s^2}.$$

Now we use the recurrence relation to obtain the coefficient of the constant terms. Then, we have for $f(K_1)$, $c_2 = \frac{1}{8}$. For $f(K_2)$, we have $c_2 = \frac{31}{8}$. For $f(K_3)$, we have $c_2 = -1$. Thus, if we sum up all these coefficients, we have 3 which is the number of lattice points in this triangle. $\square$

`LattE` produces the generating function of the lattice points of an input polytope. This generating function is a multivariate polynomial of finite degree. As we saw in Subsection 2.2 it is possible to count the number of lattice points without expanding the rational functions into the sum of monomials. Suppose that instead of wanting to know the number of lattice points we simply wish to *decide* whether there is one lattice point inside the polytope or not. The integer feasibility problem is an important and difficult problem [3, 41]. Obviously, one can simply compute the residues and then if the number of lattice points is non-zero, clearly, the polytope has lattice points. But something faster and more elementary can be done if we just test for the existence of lattice points. We are simply testing whether the polynomial has any monomials at all, or whether the polynomial is the zero polynomial.

Remember that *all* the coefficients of the polynomial are positive, and in fact equal to one. If we find an specific vector α of positive values whose substitution gives us a nonzero answer, then we are sure the polynomial has monomials. On the other hand if the answer is zero, the polynomial *must be* the zero polynomial since there is no cancellation of monomial values. Hence a single test on a non-zero vector, that avoids poles, evaluated at the rational functions decides integer feasibility. To implement this, one has to take care of how to deal with large integers. Another alternative is to substitute not just any vector, but a vector whose entries are roots of unity, thus it reduces the complexity.

Before we end our description of `LattE`, we must comment on how we deal with polytopes that are not full-dimensional (e.g. transportation polytopes). Given the lower-dimensional polytope $P = \{x \in \mathbb{R}^n : Ax = a, Bx \leq b\}$ with the $d \times n$ matrix A of full row-rank, we will use the equations to transform P into a polytope $Q = \{x \in \mathbb{R}^{n-d} : Cx \leq c\}$ in fewer variables, whose integer points are in one-to-one correspondence to the integer points of P. This second polytope will be the input to the main part of `LattE`. The main idea of this transformation is to find the general integer solution $x = x_0 + \sum_{i=1}^{n-d} \lambda_i g_i$ to $Ax = a$ and to substitute it into the inequalities $Bx \leq b$, giving a new system $Cx \leq c$ in $n-d$ variables $\lambda_1, \ldots, \lambda_{n-d}$.

It is known that the general integer solution $Ax = a$ can be found via the Hermite normal form $H = (R|0)$ of A [41]. Here, R is a lower-triangular matrix and $H = AU$ for some unimodular matrix U. Moreover, as A is supposed to have full row-rank, R is a non-singular $d \times d$ matrix. Let U_1 be the matrix consisting of the first d columns of U and U_2 consisting of the remaining $n-d$ columns of U. Now we have $AU_1 = R$ and $AU_2 = 0$

and the columns of U_2 give the generators $\{g_1,\ldots,g_{n-d}\}$ of the integer null-space of A. Thus, it remains to determine a special integer solution x_0 to $Ax=a$.

To do this, first find an integer solution y_0 to $Hy=(R|0)y=a$, which is easy due to the triangular structure of R. With $x_0=Uy_0$, we get $Ax_0=AUy_0=Hy_0=a$ and have found all pieces of the general integer solution $x=x_0+\sum_{i=1}^{n-d}\lambda_i g_i$ to $\{x\in\mathbb{Z}^n : Ax=a\}$.

3 Computational experience and performance

`LattE` provides an interactive web page `www.math.ucdavis.edu/~latte` where any user can freely submit a problem to be tested. You can also find there the files of all the experiments presented in this section. If the reader is interested in a copy of the code, please write to the first author. At the moment we can handle polytopes of dimension 30 and up to a several thousands vertices. For all computations, we used a 1 GHz Pentium PC machine running Red Hat Linux. Here is a short description of how to use `LattE`. Suppose we want to count the number of lattice points inside of a polytope $P\subset\mathbb{R}^d$ such that $P=\{x\in\mathbb{R}^d|Ax\le b,\ A\in\mathbb{Z}^{m\times d},\ b\in\mathbb{Z}^m\}$.
The input format for `LattE` is as in `CDD` such that:

$$\begin{array}{ll} m & d+1 \\ b & -A. \end{array}$$

For example. If we want to count the number of the lattice points inside of the unit standard cube in 3 dimension, the input format is the following:

$$\begin{array}{rrrr} 6 & 4 & & \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}$$

There are 6 inequalities in 3 variables +1 entry corresponding to the right hand side (which is 4 entries) in this example. Now suppose we want to solve problems that are not full-dimensional. We want to count the number of lattice points inside of a polytope $P\subset\mathbb{R}^d$ such that $P=\{x\in\mathbb{R}^d|A_ix=b_i$, for $i=1,2,\ldots s$, $A_ix\le b_i$, for $i=s+1,\ldots,m$, $A\in\mathbb{Z}^{m\times d}$, and $b\in\mathbb{Z}^m\}$, where A_i represents the ith row vector of A and b_i represents the ith element of b.

The input format for `LattE` is as in `CDD` such that:

$$\begin{array}{llllll} m & d+1 & & & & \\ b & -A & & & & \\ s & 1 & 2 & \ldots & s. \end{array}$$

For example, if we want to count the number of the lattice points inside of the polytope of the small knapsack problem $\{x+2y+3z=6,\ x\ge 0,\ y\ge 0,$ and $z\ge 0\}$, the input format must be the following:

4	4		
6	−1	−2	−3
0	1	0	0
0	0	1	0
0	0	0	1
1	1		

There are 4 inequalities in 3 variables +1 entry (which is 4 entries) in this example.

First we report on computation with convex rational polytopes. For computation, we used a 1 GHz Pentium PC machine running Red Hat Linux. We begin with the class of *multiway contingency tables*. A *d-table of size* $(n_1, \ldots, n_d)$ is an array of non-negative integers $v = (v_{i_1,\ldots,i_d})$, $1 \leq i_j \leq n_j$. For $0 \leq m < d$, an *m-marginal* of v is any of the $\binom{d}{m}$ possible m-tables obtained by summing the entries over all but m indices. For instance, if $(v_{i,j,k})$ is a 3-table then its 0-marginal is $v_{+,+,+} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} v_{i,j,k}$, its 1-marginals are $(v_{i,+,+}) = (\sum_{j=1}^{n_2} \sum_{k=1}^{n_3} v_{i,j,k})$ and likewise $(v_{+,j,+})$, $(v_{+,+,k})$, and its 2-marginals are $(v_{i,j,+}) = (\sum_{k=1}^{n_3} v_{i,j,k})$ and likewise $(v_{i,+,k})$, $(v_{+,j,k})$.

Such tables appear naturally in statistics and operations research under various names such as *multi-way contingency tables*, or *tabular data*. We consider the **table counting problem**: *given a prescribed collection of marginals, how many d-tables are there that share these marginals?* Table counting has several applications in statistical analysis, in particular for independence testing, and has been the focus of much research (see [19] and the extensive list of references therein). Given a specified collection of marginals for d-tables of size $(n_1, \ldots, n_d)$ (possibly together with specified lower and upper bounds on some of the table entries) the associated *multi-index transportation polytope* is the set of all non-negative *real valued* arrays satisfying these marginals and entry bounds. The counting problem can be formulated as that of counting the number of integer points in the associated multi-index transportation polytope. We begin with a small example of a three-dimensional table of format $2 \times 3 \times 3$ given below. The data displayed in Table 3 have been extracted from the 1990 decennial census and is used in [22]. For the 2-marginals implied by these data we get the answer of 441 in less than a second.

We present now an example of a $3 \times 3 \times 3$ table with fairly large 2-marginals. They are displayed in Table 4. `LattE` took only 19.67 seconds of CPU time. The number of lattice points inside of this polytope is

2249847900174017152559270967589010977293.

Next we present an example of a $3 \times 3 \times 4$ table with large 2-marginals. The 2-marginals are displayed in Table 5. The CPU time for this example was 44 minutes 42.22 seconds. The number of lattice points inside of this polytope is

4091700129572445106288079361219676736812805058988286839062994.

The next family of examples are some hard knapsack-type problems. Suppose we have a set of positive relatively prime integers $\{a_1, a_2, \ldots, a_d\}$. Denote by a the vector $(a_1, a_2, \ldots, a_d)$. Consider the following problem: does there exist a non-negative integral vector x satisfying $ax = a_0$ for some positive integer a_0? We take several examples from [1] which have been found to be extremely hard to solve by commercial quality branch-and-bound software. This is very surprising since the number of variables is at most 10. It is not very difficult to see that, if the right-hand-side value a_0 is large enough, the equation will surely have a non-negative integer solution. The *Frobenius number* for a knapsack problem is the largest value a_0 such that the knapsack problem is infeasible. Aardal and Lenstra [1] solved them using the reformulation in [3]. Their method works significantly better than branch-and-bound using `CPLEX 6.5`. Here we demonstrate that

Gender = Male

Race	Income Level $\leq \$10,000$	$> \$10000$ and $\leq \$25000$	$> \$25000$	Total
White	96	72	161	329
Black	10	7	6	23
Chinese	1	1	2	4
Total	107	80	169	356

Gender = Female

Race	Income Level $\leq \$10,000$	$> \$10000$ and $\leq \$25000$	$> \$25000$	Total
White	186	127	51	364
Black	11	7	3	21
Chinese	0	1	0	1
Total	197	135	54	386

Table 3: Three-way cross-classification of gender, race, and income for a selected U.S. census tract. *Source*: 1990 Census Public Use Microdata Files.

our implementation of Barvinok's algorithm is fairly fast and, on the order of seconds, we resolved the first 15 problems in Table 1 of [1] and verified all are infeasible except **prob9**, where there is a mistake. The vector $(3480, 1, 4, 4, 1, 0, 0, 0, 0, 0)$ is a solution to the right-hand-side 13385099. In fact, using `LattE` we know that the exact number of solutions is 838908602000. For comparison we named the problems exactly as in Table 1 of [1]. We present our results in Table 6. Of course, we can achieve more. It is very interesting to know the number of lattice points if we add 1 to the Frobenius number for each problem. In Table 7, we find the number of solutions if we add 1 to the Frobenius number on each of the (infeasible) problems. The speed is practically the same as in the previous case. In fact the speed is the same regardless of the right-hand-side value a_0.

Already counting the lattice points of large width convex polygons is a non-trivial task

164424	324745	127239
262784	601074	9369116
149654	7618489	1736281

163445	49395	403568
1151824	767866	8313284
1609500	6331023	1563901

184032	123585	269245
886393	6722333	935582
1854344	302366	9075926

Table 4: 2-Marginals for the $3 \times 3 \times 3$ example.

273510	273510	273510	191457
273510	273510	547020	191457
273510	547020	273510	191457

464967	273510	273510
547020	273510	464967
410265	601722	273510

273510	273510	273510
410265	547020	136755
547020	136755	410265
191457	191457	191457

Table 5: 2-Marginals for the $3 \times 3 \times 4$ example.

if one uses brute-force enumeration (e.g. list one by one the points in a bounding box of the polygon and see whether it is inside the polygon). Here we experiment with very large convex *almost* regular n-gons. Regular n-gons cannot have rational coordinates, but we can approximate them to any desired accuracy by rational polygons. In the following experiment we take regular n-gons, from $n = 5$ to $n = 12$ centered at the origin (these have only a handful of lattice points). We take a truncation of the coordinates up to 3, 9, and 15 decimal digits, then we multiply by a large enough power of 10 to make those vertex coordinates integral and we count the number of lattice points in the dilation. All experiments take less than a second.

The next two sets of examples are families that have been studied quite extensively in the literature and provide us with a test for speed. In the first case we deal with *two-way contingency tables.* The polytope defined by a two-way contingency table is called the *transportation polytope.* We present the results in Table 9. The second family consists of flow polytopes for the complete 4 digraph and the complete 5 digraph. Consider the directed complete graph K_l for $l \in \mathbb{N}$ and $l \geq 3$. We assign a number to each node of the graph. Then, we orient the arcs from the node of smaller index to the node of bigger index. Let N be the node set of the complete graph K_l, let w_i be a weight assigned to node i for $i = 1, 2, \ldots, l$, and let A be the arc set of K_l. Then, we have the following constraints, with as many variables as arcs:

$$\sum_{(j,i)\text{arc enters} i} x_{ji} - \sum_{(i,j)\text{arc has tail} i} x_{ij} = w_i,$$

$$x_{ij} \geq 0 \ \forall (i,j) \in A.$$

These equalities and inequalities define a polytope and this polytope is a special case *flow polytope.* The results for the complete graphs K_4 and K_5, with different weight vectors, are shown in Tables 10 and 11 respectively.

These two families of polytopes have been studied by several authors [7, 12, 18, 36] and thus are good for testing performance of `LattE`. We used several examples of transportation polytopes, as presented in the table below. In general, `LattE` runs at comparable performance to the software of [7, 12] for generic vectors (a, b) but is slower for degenerate inputs (those that do not give a simple polytope). The reason seems to be that at each non-simplex vertex `LattE` needs to triangulate each cone which takes considerable time in

											Frobenius #	Time (m, s)
cuww1	12223	12224	36674	61119	85569						89643481	0.55s
cuww2	12228	36679	36682	48908	61139	73365					89716838	1.78s
cuww3	12137	24269	36405	36407	48545	60683					58925134	1.27s
cuww4	13211	13212	39638	52844	66060	79268	92482				104723595	2.042s
cuww5	13429	26850	26855	40280	40281	53711	53714	67141			45094583	16.05s
pro1	25067	49300	49717	62124	87608	88025	113673	119169			33367335	47.07s
prob2	11948	23330	30635	44197	92754	123389	136951	140745			14215206	1m0.58s
prob3	39559	61679	79625	99658	133404	137071	159757	173977			58424799	1m28.3s
prob4	48709	55893	62177	65919	86271	87692	102881	109765			60575665	59.04s
prob5	28637	48198	80330	91980	102221	135518	165564	176049			62442884	1m41.78s
prob6	20601	40429	40429	45415	53725	61919	64470	69340	78539	95043	22382774	3m45.86s
prob7	18902	26720	34538	34868	49201	49531	65167	66800	84069	137179	27267751	2m57.64s
prob8	17035	45529	48317	48506	86120	100178	112464	115819	125128	129688	21733990	8m29.78s
prob10	45276	70778	86911	92634	97839	125941	134269	141033	147279	153525	106925261	4m24.67s

Table 6: Infeasible knapsack problems.

problem	RHS	# os lattice points.
cuww1	89643482	1
cuww2	89716839	1
cuww3	58925135	2
cuww4	104723596	1
cuww5	45094584	1
prob1	33367336	859202692
prob2	14215207	2047107
prob3	58424800	35534465752
pro4	60575666	63192351
pro5	62442885	21789552314
pro6	22382775	218842
pro7	27267752	4198350819898
pro8	21733991	6743959
pro10	106925262	102401413506276371

Table 7: The number of lattice points if we add 1 to the Frobenius number.

	10^3 (seconds)	10^9 (seconds)	10^{15} (seconds)
5gon	2371673(0.136)	2377641287748905186(0.191)	2377641290737895844565559026875(0.289)
6gon	2596011(0.153)	2598076216000000011(0.193)	2598076211353321000000000000081(0.267)
7gon	2737110(0.175)	2736410188781217941(0.318)	2736410188638105174143840143912(0.584s)
8gon	2820021(0.202)	2828427120000000081(0.331)	2828427124746200000000000000201(0.761)
9gon	2892811(0.212)	2892544245156317460(0.461)	2892544243589428566861745742966(0.813)
10gon	2931453(0.221)	2938926257659276211(0.380)	2938926261462380264188126524437(0.702)
11gon	2974213(0.236)	2973524496796366173(0.745)	2973524496005786351949189500315(1.858)
12gon	2997201(0.255)	3000000004942878881(0.466)	3000000000000005419798779796241(0.696)

Table 8: The numbers of the approximated regular polygons. We show the number of lattice points in different dilation factors (powers of ten) and time of computation.

problems of high dimension.

4 New Ehrhart (quasi-)polynomials

Given a rational polytope $P \subset \mathbb{R}^d$, the function

$$i_P(t) := \#\left(tP \cap \mathbb{Z}^d\right),$$

for a positive integer t, was first studied by E. Ehrhart [21] and has received a lot of attention in combinatorics. It is known to be a polynomial when all vertices of P are integral and it is a quasi-polynomial for arbitrary rational polytopes. It is called the *Ehrhart quasi-polynomial* in honor of its discoverer (see chapter 4 [43]). A function $f : \mathbb{N} \to \mathbb{C}$ is a quasi-polynomial if there is an integer $N > 0$ and polynomials $f_0, \ldots, f_{N-1}$ such that $f(s) = f_i(s)$ if $s \equiv i \bmod N$. The integer N is called a *quasi-period* of f. Therefore, by counting the number of lattice points for sufficiently many dilations of a rational polytope, we can interpolate its Ehrhart quasi-polynomial.

Using `LattE`, we have calculated the Ehrhart polynomials and quasi-polynomials for polytopes that are slices or nice truncations of the unit d-cube. To the best of our knowledge these values were not known before. For example, the 24-cell polytope centered at the origin with smallest integer coordinates has Ehrhart polynomial $i_{24\_cell}(s) = 8s^4 + \frac{32s^3}{3} + 8s^2 + \frac{16s}{3} + 1$. In Table 12, we see the Ehrhart polynomials for the hypersimplices $\Delta(n,k)$. They are defined as the slice of the n-cube by the hyperplane of equation $\sum x_i = k$ with $k \leq n$. Note that $\Delta(n,k) = \Delta(n, n-k)$ because of the symmetries of the regular cube. The hypersimplices form one of the most famous families of 0/1-polytopes. It is known that hypersimplices are *compressed polytopes* [?]. This means that their Ehrhart polynomials can be recovered from the f-vectors of any of their lexicographic triangulations. Instead, here we recovered them explicitly for the first time using `LattE` and interpolation.

For some truncated unit cubes, we have their Ehrhart quasi-polynomials.

Proposition 8 *The Ehrhart quasi-polynomial for the truncated unit cube in Figure 4, where its vertices are at 1/3 and 2/3 of the way along edges of the cube, is given by:*

$$i_{tru\_cube1}(s) = \begin{cases} \frac{77s^3}{81} + \frac{23s^2}{9} + \frac{19s}{9} + 1 \ \textit{if}\ s \equiv 0 \ \textit{mod}\ 3, \\ \frac{77s^3}{81} + \frac{61s^2}{27} - \frac{7s}{27} - \frac{239}{81} \ \textit{if}\ s \equiv 1 \ \textit{mod}\ 3, \\ \frac{77s^3}{81} + \frac{65s^2}{27} + \frac{29s}{27} - \frac{31}{81} \ \textit{if}\ s \equiv 2 \ \textit{mod}\ 3. \end{cases}$$

Proposition 9 *The Ehrhart quasi-polynomial for the cuboctahedron (Figure 5) is:*

$$i_{tru\_cube2}(s) = \begin{cases} \frac{5s^3}{6} + 2s^2 + \frac{5s}{3} + 1 \ \textit{if}\ s \equiv 0 \ \textit{mod}\ 2, \\ \frac{5s^3}{6} + \frac{3s^2}{2} - \frac{5s}{6} - \frac{3}{2} \ \textit{if}\ s \equiv 1 \ \textit{mod}\ 2. \end{cases}$$

Proposition 10 *The Ehrhart quasi-polynomial for the truncated regular simplex, where the vertices are at 1/3 and 2/3 of the way along the simplex edges (see Figure 6), is given by:*

$$i_{tru\_simplex}(s) = \begin{cases} \frac{23s^3}{81} + \frac{7s^2}{9} + \frac{13s}{9} + 1 \ \textit{if}\ s \equiv 0 \ \textit{mod}\ 3, \\ \frac{23s^3}{81} + \frac{19s^2}{27} + \frac{5s}{27} - \frac{95}{81} \ \textit{if}\ s \equiv 1 \ \textit{mod}\ 3, \\ \frac{23s^3}{81} + \frac{17s^2}{27} + \frac{23s}{27} + \frac{41}{81} \ \textit{if}\ s \equiv 2 \ \textit{mod}\ 3. \end{cases}$$

Margins	# of lattice points	Time (seconds)
[220, 215, 93, 64], [108, 286, 71, 127]	1225914276768514	1.048
[109, 127, 69, 109], [119, 86, 108, 101]	993810896945891	1.785
[72, 67, 47, 96], [70, 70, 51, 91]	25387360604030	1.648
[179909, 258827, 224919, 61909], [190019, 90636, 276208, 168701]	1357102606340183816466829663506589992315207 9	1.954
[229623, 259723, 132135, 310952], [279858, 170568, 297181, 184826]	64691139545929664520000400080400324337115486 2	1.765
[249961, 232006, 150459, 200438], [222515, 130701, 278288, 201360]	31972024969011143788722925548784784531046347 5	1.854
[140648, 296472, 130724, 309173], [240223, 223149, 218763, 194882]	32277356082100885641727027595059910706126362 5	1.903
[65205, 189726, 233525, 170004], [137007, 87762, 274082, 159609]	6977523720740024241056075121611021139576919	1.541
[251746, 282451, 184389, 194442], [146933, 239421, 267665, 259009]	86131634328064904959323613215503919068202761 4	1.880
[138498, 166344, 187928, 186942], [228834, 138788, 189477, 122613]	6331319141434282775456653136453337858898646 7	1.973
[20812723, 17301709, 21133745, 27679151], [28343568, 18410455, 19751834, 20421471]	665711555567792389878908993624629379187969880179721169068827951	2.917
[15663004, 19519372, 14722354, 22325971], [17617837, 25267522, 20146447, 9198895]	632927044239416550802939713953488488074542532047205264724620 15	3.161
[13070380, 18156451, 13365203, 20567424], [12268303, 20733257, 17743591, 14414307]	430753571461735704921172916856016048305446437692528313373425 57	2.990

Table 9: Testing for 4×4 transportation polytopes.

Weights on nodes	# of lattice points	Time (seconds)
[-6, -8, 5, 9]	223	0.288
[-9, -11, 12, 8]	330	0.286
[-1000, -1, 1000, 1]	3002	0.287
[-4383, 886, 2777, 720]	785528058	0.287
[-4907, -2218, 3812, 3313]	20673947895	0.288
[-2569, -3820, 1108, 5281]	14100406254	0.282
[-3842, -3945, 6744, 1043]	1906669380	0.281
[-47896, -30744, 46242, 32398]	19470466783680	0.282
[-54915, -97874, 64165, 88624]	106036300535520	0.281
[-69295, -62008, 28678, 102625]	179777378508547	0.282
[-3125352, -6257694, 926385, 8456661]	34441480172695101274	0.509
[-2738090, -6701290, 190120, 9249260]	28493245103068590026	0.463
[-6860556, -1727289, 934435, 7653410]	91608082255943644656	0.503

Table 10: Testing for the complete graph K_4.

Weights on nodes	# of lattice points	secs
[-12, -8, 9, 7, 4]	14805	0.319
[-125, -50, 75, 33, 67]	6950747024	0.325
[-763, -41, 227, 89, 488]	222850218035543	0.325
[-11675, -88765, 25610, 64072, 10758]	563408416219655157542748	0.319
[-78301, -24083, 22274, 19326, 60784]	1108629405144880240444547243	0.336
[-52541, -88985, 1112, 55665, 84749]	3997121684242603301444265332	0.331
[-71799, -80011, 86060, 39543, 26207]	160949617742851302259767600	0.316
[-45617, -46855, 24133, 54922, 13417]	15711217216898158096466094	0.285
[-54915, -97874, 64165, 86807, 1817]	102815492358112722152328	0.277
[-69295, -62008, 28678, 88725, 13900]	65348330279808617817420057	0.288
[-8959393, -2901013, 85873, 533630, 11240903]	6817997013081449330251623043931489475270	0.555
[-2738090, -6701290, 190120, 347397, 8901863]	277145720781272784955528774814729345461	0.599
[-6860556, -1727289, 934435, 818368, 6835042]	710305971948234346520365668331191134724	0.478

Table 11: Testing for the complete graph K_5. Time is given in seconds

n	k	the Ehrhart polynomial $P(s)$
4	1	$\frac{s^3}{6}+s^2+\frac{11s}{6}+1$
4	2	$\frac{2x^3}{3}+2s^2+\frac{7s}{3}+1$
5	1	$\frac{s^4}{24}+\frac{5s^3}{12}+\frac{35s^2}{24}+\frac{25s}{12}+1$
5	2	$\frac{11s^4}{24}+\frac{25s^3}{12}+\frac{85s^2}{24}+\frac{35s}{12}+1$
6	1	$\frac{s^5}{120}+\frac{s^4}{8}+\frac{17s^3}{24}+\frac{15s^2}{8}+\frac{137s}{60}+1$
6	2	$\frac{13s^5}{60}+\frac{3s^4}{2}+\frac{47s^3}{12}+5s^2+\frac{101s}{30}+1$
6	3	$\frac{11s^5}{20}+\frac{11s^4}{4}+\frac{23s^3}{4}+\frac{25s^2}{4}+\frac{37s}{10}+1$
7	1	$\frac{s^6}{720}+\frac{7s^5}{240}+\frac{35s^4}{144}+\frac{49s^3}{48}+\frac{203s^2}{90}+\frac{49s}{20}+1$
7	2	$\frac{19s^6}{240}+\frac{63s^5}{80}+\frac{49s^4}{16}+\frac{287s^3}{48}+\frac{763s^2}{120}+\frac{56s}{15}+1$
7	3	$\frac{151s^6}{360}+\frac{161s^5}{60}+\frac{256s^4}{36}+\frac{21s^3}{2}+\frac{3199s^2}{360}+\frac{259s}{60}+1$
8	1	$\frac{s^7}{5040}+\frac{s^6}{180}+\frac{23s^5}{360}+\frac{7s^4}{18}+\frac{967s^3}{720}+\frac{469s^2}{180}+\frac{363s}{140}+1$
8	2	$\frac{s^7}{42}+\frac{29s^6}{90}+\frac{53s^5}{30}+\frac{91s^4}{18}+\frac{49s^3}{6}+\frac{343s^2}{45}+\frac{283s}{70}+1$
8	3	$\frac{397s^7}{1680}+\frac{359s^6}{180}+\frac{281s^5}{40}+\frac{245s^4}{18}+\frac{1273s^3}{80}+\frac{2051s^2}{180}+\frac{2027s}{420}+1$
8	4	$\frac{151s^7}{315}+\frac{151s^6}{45}+\frac{463s^5}{45}+\frac{161s^4}{9}+\frac{862s^3}{45}+\frac{574s^2}{45}+\frac{533s}{105}+1$
9	1	$\frac{s^8}{40320}+\frac{s^7}{1120}+\frac{13s^6}{960}+\frac{9s^5}{80}+\frac{1069s^4}{1920}+\frac{267s^3}{160}+\frac{29531s^2}{10080}+\frac{761s}{280}+1$
9	2	$\frac{247s^8}{40320}+\frac{121s^7}{1120}+\frac{763s^6}{960}+\frac{253s^5}{80}+\frac{14203s^4}{1920}+\frac{1667s^3}{160}+\frac{88721s^2}{10080}+\frac{1207s}{280}+1$
9	3	$\frac{477s^8}{4480}+\frac{1311s^7}{1120}+\frac{1731s^6}{320}+\frac{1107s^5}{80}+\frac{13899s^4}{640}+\frac{3477s^3}{160}+\frac{15419s^2}{1120}+\frac{1473s}{280}+1$
9	4	$\frac{15619s^8}{40320}+\frac{3607s^7}{1120}+\frac{11311s^6}{960}+\frac{1991s^5}{80}+\frac{63991s^4}{1920}+\frac{4669s^3}{160}+\frac{166337s^2}{10080}+\frac{1599s}{280}+1$
10	1	$\frac{s^9}{362880}+\frac{s^8}{8064}+\frac{29s^7}{12096}+\frac{5s^6}{192}+\frac{3013s^5}{17280}+\frac{95s^4}{128}+\frac{4523s^3}{2268}+\frac{6515s^2}{2016}+\frac{7129s}{2520}+1$
10	2	$\frac{251s^9}{181440}+\frac{31s^8}{1008}+\frac{1765s^7}{6048}+\frac{37s^6}{24}+\frac{42863s^5}{8640}+\frac{481s^4}{48}+\frac{115205s^3}{9072}+\frac{4993s^2}{504}+\frac{5729s}{1260}+1$
10	3	$\frac{913s^9}{22680}+\frac{1135s^8}{2016}+\frac{5071s^7}{1512}+\frac{179s^6}{16}+\frac{3128s^5}{135}+\frac{2999s^4}{96}+\frac{63041s^3}{2268}+\frac{8069s^2}{504}+\frac{3553s}{630}+1$
10	4	$\frac{44117s^9}{181440}+\frac{2489s^8}{1008}+\frac{66547s^7}{6048}+\frac{683s^6}{24}+\frac{409361s^5}{8640}+\frac{2543s^4}{48}+\frac{363947s^3}{9072}+\frac{10127s^2}{504}+\frac{7883s}{1260}+1$
10	5	$\frac{15619s^9}{36288}+\frac{15619s^8}{4032}+\frac{94939s^7}{6048}+\frac{3607s^6}{96}+\frac{101311s^5}{1728}+\frac{11911s^4}{192}+\frac{25394s^3}{567}+\frac{21689s^2}{1008}+\frac{1627s}{252}+1$
11	1	$\frac{s^{10}}{3628800}+\frac{11s^9}{725760}+\frac{11s^8}{30240}+\frac{121s^7}{24192}+\frac{7513s^6}{172800}+\frac{8591s^5}{34560}+\frac{341693s^4}{362880}+\frac{84095s^3}{36288}+\frac{177133s^2}{50400}+\frac{7381s}{2520}+1$
11	2	$\frac{1013s^{10}}{3628800}+\frac{5533s^9}{725760}+\frac{2189s^8}{24192}+\frac{14795s^7}{24192}+\frac{447689s^6}{172800}+\frac{246697s^5}{34560}+\frac{14597s^4}{1134}+\frac{543763s^3}{36288}+\frac{91949s^2}{8400}+\frac{1199s}{252}+1$
11	3	$\frac{299s^{10}}{22680}+\frac{16621s^9}{72576}+\frac{41591s^8}{24192}+\frac{88693s^7}{12096}+\frac{170137s^6}{8640}+\frac{604109s^5}{17280}+\frac{3043997s^4}{72576}+\frac{308473s^3}{9072}+\frac{60929s^2}{3360}+\frac{15059s}{2520}+1$
11	4	$\frac{56899s^{10}}{453600}+\frac{565631s^9}{362880}+\frac{205733s^8}{24192}+\frac{326491s^7}{12096}+\frac{2400629s^6}{43200}+\frac{1348787s^5}{17280}+\frac{5535695s^4}{72576}+\frac{468655s^3}{9072}+\frac{1185701s^2}{50400}+\frac{16973s}{2520}+1$
11	5	$\frac{655177s^{10}}{1814400}+\frac{336083s^9}{90720}+\frac{2078791s^8}{120960}+\frac{287639s^7}{6048}+\frac{7525771s^6}{86400}+\frac{95557s^5}{864}+\frac{35914087s^4}{362880}+\frac{1125575s^3}{18144}+\frac{443179s^2}{16800}+\frac{17897s}{2520}+1$
12	1	$\frac{s^{11}}{39916800}+\frac{s^{10}}{604800}+\frac{s^9}{20736}+\frac{11s^8}{13440}+\frac{10831s^7}{1209600}+\frac{1903s^6}{28800}+\frac{242537s^5}{725760}+\frac{139381s^4}{120960}+\frac{341747s^3}{129600}+\frac{190553s^2}{50400}+\frac{83711s}{27720}+1$
12	2	$\frac{509s^{11}}{9979200}+\frac{169s^{10}}{100800}+\frac{551s^9}{22680}+\frac{2057s^8}{10080}+\frac{332249s^7}{302400}+\frac{18997s^6}{4800}+\frac{876959s^5}{90720}+\frac{80179s^4}{5040}+\frac{244681s^3}{14175}+\frac{150293s^2}{12600}+\frac{68591s}{13860}+1$
12	3	$\frac{50879s^{11}}{13305600}+\frac{6979s^{10}}{86400}+\frac{60271s^9}{80640}+\frac{32153s^8}{8064}+\frac{5483809s^7}{403200}+\frac{897259s^6}{28800}+\frac{11875111s^5}{241920}+\frac{185339s^4}{3456}+\frac{451173s^3}{11200}+\frac{338503s^2}{16800}+\frac{58007s}{9240}+1$
12	4	$\frac{1093s^{11}}{19800}+\frac{62879s^{10}}{75600}+\frac{20893s^9}{3780}+\frac{10813s^8}{504}+\frac{684323s^7}{12600}+\frac{340967s^6}{3600}+\frac{5258s^5}{45}+\frac{38819s^4}{378}+\frac{1202029s^3}{18900}+\frac{42218s^2}{1575}+\frac{1103s}{154}+1$
12	5	$\frac{1623019s^{11}}{6652800}+\frac{882773s^{10}}{302400}+\frac{1908073s^9}{120960}+\frac{1028401s^8}{20160}+\frac{7395023s^7}{67200}+\frac{2401619s^6}{14400}+\frac{4398559s^5}{24192}+\frac{8661917s^4}{60480}+\frac{12163441s^3}{151200}+\frac{782969s^2}{25200}+\frac{8861s}{1155}+1$
12	6	$\frac{655177s^{11}}{1663200}+\frac{655177s^{10}}{151200}+\frac{5507s^9}{252}+\frac{336083s^8}{5040}+\frac{6898277s^7}{50400}+\frac{1430341s^6}{7200}+\frac{3152491s^5}{15120}+\frac{1200463s^4}{7560}+\frac{30291s^3}{350}+\frac{68321s^2}{2100}+\frac{18107s}{2310}+1$

Table 12: The Ehrhart polynomials for the hypersimplices $\Delta(n,k)$

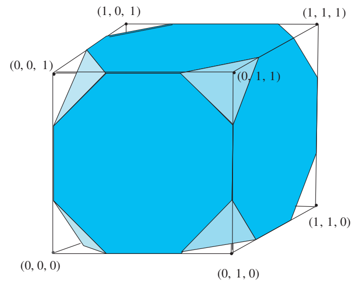

Figure 4: The truncated cube.

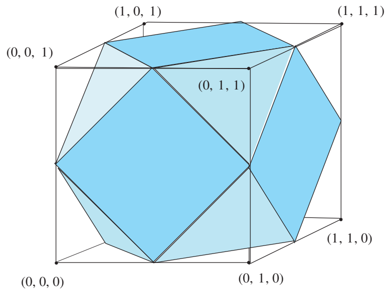

Figure 5: The cuboctahedron.

5 Other enumeration algorithms and future work

We have demonstrated the practical relevance of Barvinok's cone decomposition approach for counting lattice points and deriving formulas. Several other algorithms are available to carry out the same kind of enumeration. It is important to implement them all in the same computer system for comparison of performance and to corroborate that the answers are correct. Some problems are solvable by some methods but not by others. To close this article we quickly review some of the algorithms available to date that will appear in the future versions of `LattE`.

We have established that the major practical bottleneck of Barvinok's algorithm is the fact that a polytope may have too many vertices. Since we visit all vertices to compute the rational function the result can be costly. For example, in the case of multiway transportation polytopes, the number of vertices is much larger than the number of facet-defining inequalities. For example, the well-known polytope of magic cubes in the $4 \times 4 \times 4$ case has over two million vertices, but only 64 inequalities describe the polytope. This is the same with other classical challenges such as the 5×5 magic square matrices (see [4] for details on these examples). In such cases we propose the following simple variation of Barvinok's algorithm. In a forthcoming paper we will use it to solve several very large problems of combinatorial interest:

Algorithm 11 (Dealing with polytopes with few facets) *1. Position the d-dimensional polytope P inside $\mathbb{R}^{d+1}$, by embedding the polytope at level $x_{d+1} = 1$.*

2. Consider the $(d+1)$-dimensional cone over P; call this cone K. Compute the polar

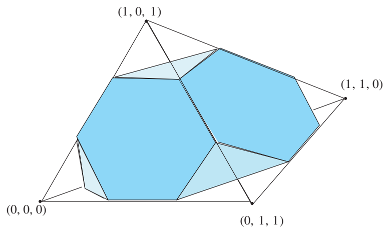

Figure 6: The truncated simplex.

K^* of this cone. Since the number of facets of P is small compared to its vertices the number of rays of the cone K^* is small.

3. *Apply Barvinok's decomposition of K^* into unimodular cones. Polarize back each of these cones. It is known, e.g. Corollary 2.8 in [9], that by dualizing back we get a unimodular cone decomposition of K. From it we can retrieve a signed sum of rational functions that has all the lattice points of K as monomials.*

4. *The issue is now, how to extract just the points of P. This can be done by a suitable monomial substitution that gives a coarser generating function graded into levels for the cone K. In other words, the polytope P is by construction at level $x_{d+1} = 1$, thus the monomials associated with the lattice points in P are of the form $z_1^{a_1} z_2^{a_2} \ldots, z_d^{a_d} t$. We want to group together all such monomials. The problem is that the substitution may be a pole of one or more of the rational functions. We need to know the coefficient of t when the variables z_i tend to 1. This can be done by the Laurent series calculations described before (see also Theorem 2.6 in [10]).*

   We have discovered that there is a strong dependence of the poles of the rational function with the way we apply the decomposition. Roughly speaking, this depends on choosing a good initial triangulation of the cone.

Another successful counting algorithm (and one that can be merged into the polar Barvinok algorithm) is based on Gröbner and Hilbert bases. Let A be a $m \times d$ integral matrix. Consider a convex pointed polyhedral cone $C = \{x | Ax = 0,\ x \geq 0\}$. We wish to study $C \cap \mathbb{Z}^d$. With any rational pointed polyhedral cone $C = \{Ax = 0, x \geq 0\}$ and a field k we associate a *semigroup ring*, $R_C = k[x^a : a \in C \cap \mathbb{Z}^d]$. A *Hilbert basis* of the cone C is a finite set of vectors in S_C such that every other element of S_C is a non-negative integer combination of these elements. The main theorem is that R_C equals $k[x_1, x_2, \ldots, x_N]/I_C$ where I_C is the toric ideal generated by binomial relationships holding among the N Hilbert basis elements (see [16, 45]). It turns out that R_C is a graded k-algebra. A graded k-algebra has a decomposition $R_C = \bigoplus R_C(i)$, where each $R_C(i)$ collects all elements of degree i and it is a k-vector space (with $R_C(0) = k$). The function $H(R_C, i) = \dim_k(R_C(i))$ is the *Hilbert function* of R_C. The *Hilbert-Poincaré series* of R_C is $H_{R_C}(t) = \sum_{i=0}^{\infty} H(R_C, i) t^i$.

The Hilbert-Poincaré series can be computed from the knowledge of the Gröbner bases of I_C. Here is the reason why we want this series:

Lemma 12 *Let R_C be the semigroup ring obtained from the minimal Hilbert basis of a cone C. The number of distinct lattice points of degree s equals the Hilbert function $H(R_C, s)$.*

Several "analytic" algorithms have been proposed by many authors [6, 12, 30, 35, 39]. A couple of these methods have been implemented and appear as the fastest for unimodular polyhedra. None of them has been implemented for arbitrary rational polytopes. Consider, for example, Beck's method: Let M_i denote the columns of the matrix M. We can interpret $P(M,b) \cap \mathbb{Z}^d$ as the Taylor coefficient of z^b for the function $\Pi_{j=1}^{d} \frac{1}{(1-z^{M_j})}$. One approach to obtain the particular coefficient is to use the residue theorems. For example, it was seen in [11] that if M_i denotes the i-th column of the defining matrix M, then

$$P(M,b) \cap \mathbb{Z}^d = \frac{1}{(2\pi i)^m} \int_{|z_1|=\epsilon_1} \cdots \int_{|z_m|=\epsilon_m} \frac{z_1^{-b_1-1} \cdots z_m^{-b_m-1}}{(1-z^{M_1}) \cdots (1-z^{M_d})}\, dz \ .$$

Here $0 < \epsilon_1, \ldots, \epsilon_m < 1$ are different numbers such that we can expand all the $\dfrac{1}{1-z^{M_k}}$ into the power series about 0. It is possible to do a partial fraction decomposition of the integrand into a sum of simple fractions. This was done very successfully to carry out very hard computations regarding the Birkhoff polytopes [12]. Vergne and collaborators have recently developed a powerful general theory about the multivariate rational functions $\Pi_{j=1}^{d} \frac{1}{(1-z^{M_j})}$ [6, 46]. Experimental results show it is a very fast method for unimodular polytopes [7]. Pemantle and Wilson [39] have pursued an even more general computational theory of rational generating functions where the denominators are not necessarily products of linear forms.

Acknowledgements: We thank A. Barvinok, D. Bertsimas, B. Sturmfels, and M. Vergne for several suggestions and useful conversations.

References

[1] Aardal, K., Lenstra, A.K., and Lenstra, H.W. Jr., *Hard equality constrained integer knapsacks,* preliminary version in W.J. Cook and A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization: 9th International IPCO Conference, Lecture Notes in Computer Science vol. 2337, Springer-Verlag, 2002, pp 350-366.

[2] Aardal, K. Weismantel, R., and Wolsey, L.A. *Non-standard approaches to integer programming.* Workshop on Discrete Optimization, DO'99 (Piscataway, NJ). Discrete Appl. Math. 123 (2002), no. 1-3, 5–74.

[3] Aardal, K., Hurkens, C.A.J., and Lenstra, A.K. *Solving a linear diophantine equations with lower and upper bounds on the variables.* In R.E Bixby, E.A Boyd, R.Z. Rios-Mercado (eds) "Integer Programming and Combinatorial Optimization", 6th International IPCO conference. Lecture notes in Computer Science 1412 Springer Verlag, 1998, 229-242.

[4] Ahmed, M., De Loera, J., and Hemmecke, R. *Polyhedral cones of magic cubes and square.* to appear in "New Directions in Combinatorial Geometry", (eds. Aronov et al.), Springer Verlag.

[5] Aurenhammer, F. and Klein, R. *Handbook of Computational Geometry* (Ed. J.-R. Sack and J. Urrutia). Amsterdam, Netherlands: North-Holland, pp. 201-290, 2000.

[6] Baldoni-Silva, W. and Vergne, M. *Residues formulae for volumes and Ehrhart polynomials of convex polytopes.* manuscript 81 pages. available at math.ArXiv, CO/0103097.

[7] Baldoni-Silva, W., De Loera, J., and Vergne, M. *Counting integral flows on Networks* in preparation.

[8] Barvinok, A.I. *Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math of Operations Research 19 (1994) 769 - 779.

[9] Barvinok, A.I. and Pommersheim, J. *An algorithmic theory of lattice points in polyhedra*, in: New Perspectives in Algebraic Combinatorics (Berkeley, CA, 1996-1997), 91-147, Math. Sci. Res. Inst. Publ. 38, Cambridge Univ. Press, Cambridge, 1999.

[10] Barvinok, A.I. and Woods, K. *Short rational generating functions for lattice point problems* manuscript 2002, available at arXiv.math.CO.0211146.

[11] Beck, M.*Counting lattice points by means of the residue theorem* Ramanujan Journal, 4, no. 3 (2000), 299-310.

[12] Beck, M. and Pixton, D. *The Ehrhart polynomial of the Birkhoff polytope*, manuscript 2002.

[13] Brion, M. *Points entiers dans les polyèdres convexes*, Ann. Sci. École Norm. Sup. 21 (1988) 653-663.

[14] Bruns, W. and Koch, R. `NORMALIZ`*, computing normalizations of affine semigroups*, Available via anonymous ftp from ftp//ftp.mathematik.uni-onabrueck.de/pub/osm/kommalg/software/

[15] Cook, W., Rutherford, T., Scarf, H.E., Shallcross, D. *An implementation of the generalized basis reduction algorithm for integer programming*, ORSA Journal of Computing, 5, 1993, 206–212.

[16] Cox, D., Little, J., and O'Shea, D. *Ideals, varieties, and Algorithms*, Springer Verlag, Undergraduate Text, 2nd Edition, 1997.

[17] Cox, D., Little, J., and O'Shea, D. *Using Algebraic Geometry*, Springer Verlag, Undergraduate Text, 2nd Edition, 1997.

[18] De Loera, J. and Sturmfels, B. *Algebraic unimodular counting*, to appear Math. Programming Ser. B. Preprint available at (`arXiv:math.CO/0104286`), 2001.

[19] Diaconis, P. and Gangolli, A. *Rectangular arrays with fixed margins*, Discrete probability and algorithms (Minneapolis, MN, 1993), 15–41, IMA Vol. Math. Appl., 72, Springer, New York, 1995.

[20] Dyer, M. and Kannan, R. *On Barvinok's algorithm for counting lattice points in fixed dimension*, Math of Operations Research 22 (1997) 545 - 549.

[21] Ehrhart, E. *Polynomes arithmétiques et methode des polyédres en combinatoire,* International Series of Numerical mathematics, vol 35., Birhäuser, Basel 1977.

[22] Fienberg, S.E. and Makov, U.E. and Meyer, M.M. and Steele, R.J. *Computing the Exact Conditional Distribution for a Multi-Way Contingency Table Conditional on its Marginal Totals*, in Data Analysis From Statistical Foundations, 145–166, Nova Science Publishers, 2001, A. K. Md. E. Saleh, Huntington, NY.

[23] Free Software Foundation, Inc. `GMP`, Available via `http://www.gnu.org/copyleft/lesser.html`.

[24] Fukuda, K. `cdd` and `cdd+`, *The CDD and CDD Plus*, Available via `http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`.

[25] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, second edition. Algorithms and Combinatorics, 2, Springer-Verlag, Berlin, 1993.

[26] Hemmecke, R. *On the Computation of Hilbert Bases of Cones*, Proceedings of First International Congress of Mathematical Software, Beijing 2002. Software implementation `MLP` available from `http://www.testsets.de`.

[27] Henrici, P. *Applied and computational complex analysis*, John Wiley & Sons, Inc., (1974), 243 - 245.

[28] Kirillov, A. N. *Ubiquity of Kostka polynomials*, in Physics and Combinatorics, Proceedings Nagoya 1999, edited by A.N. Kirillov, A. Tsuchiya and H. Umemura, World Scientific, 2001. Also available at `http://front.math.ucdavis.edu/math.QA/9912094`.

[29] Lasserre, J.B. *La valeur optimale des programmes entiers* C.R. Acad. Sci. Paris, Ser. I 335 , (2002) 1–4.

[30] Lasserre, J.B. and Zeron, E.S. *Solving the knapsack problem via Z-transform.* Oper. Res. Letters 30, (2002), 394–400.

[31] Lawrence, J. *Rational-function-valued valuations on polyhedra* in "Discrete and Computational Geometry" (New Brunswick, NJ, 1989/1990), 199–208 DIMACS Ser. Discrete Mathematics and Theoretical Computer Science, 6, American Mathematical Soc., Providence RI, 1991.

[32] Lee, C.W. *Subdivisions and triangulations of polytopes* in Handbook of Discrete and Computational Geometry, 271-290, (Goodman J.E. and O'Rourke J. eds.) , CRC Press, New York, 1997.

[33] Lenstra, H.W. *Integer Programming with a fixed number of variables* Mathematics of Operations Research, 8, 538–548

[34] Lovász, L. and Scarf, H.E. *The generalized basis reduction algorithm* Math. of Operations Research, 17, 1992, 751–764.

[35] MacMahon, P.A. *Combinatorial Analysis*, Vol. I and II, Chelsea, 1960, reprint of 1915 edition.

[36] Mount, J. *Fast unimodular counting*, Combinatorics, Probability, and Computing 9 (2000) 277–285.

[37] Nijehuis, A. and Wilf, H. *Representations of integers by linear forms in nonnegative integers* J. Number Theory 4 (1972), 98–106.

[38] Ohsugi, H. and Hibi, T. *Convex polytopes all of whose reverse lexicographic initial ideals are square-free* Proc. of the AMS, vol. 129, 9, (2001), 2541–2546.

[39] Pemantle, R. and Wilson, M. *Asymptotics of multivariate sequences, part I: smooth points of the singular variety.* J. Comb. Th. Ser. A. , to appear.

[40] Schmidt, J.R. and Bincer, A. *The Kostant partition function for simple Lie algebras*, J. Mathematical Physics 25 (1984) 2367–2373.

[41] Schrijver, A. *Theory of Linear and Integer Programming.* Wiley-Interscience, 1986.

[42] Shoup, V. `NTL`*, A library for doing Number Theory*, Available via anonymous URL := `http://shoup.net/ntl/`.

[43] Stanley, R.P. *Enumerative Combinatorics*, Volume I, Cambridge, 1997.

[44] Stanley, R.P. *Decompositions of rational convex polytopes* Annals of Discrete Math. 6, (1980), 333-342.

[45] Sturmfels, B. *Gröbner bases and convex polytopes*, university lecture series, vol. 8, AMS, Providence RI, (1996).

[46] Szenes, A. and Vergne, M. *Residue formulae for vector partitions and Euler-MacLaurin sums.* preprint (2002), 52 pages. Available at math.ArXiv, CO/0202253.