Counting integer points in a convex rational polytope

Problem

We have a convex rational polytope and want to count the integer points inside it, that is given $P = \{x \in \mathbb{Z}^n, Ax \leq B\}$, we want $\#P$. The algorithm described is polynomial in the number of inequalities with the dimension n fixed.

Preface

In a period of downtime, I got interested in algebraic number theory and ended up having to count vectors with coefficients limited in size in the kernel of a matrix. Thus I discovered Barvinok's algorithm. At least two solvers already exist but reinventing the wheel is so fun.

This algorithm can be implemented from scratch (I only ended up using fplll library, did not try to plug my pseudo-working LLL implementation) while using some beautiful ideas and neat tricks. The presented version is primal half-open variant only, which to me was more easily understandable as it does without the whole notion of polarity.

The purpose of this document is to give a rather informal presentation of the algorithm skimming over many proofs, focusing instead on the intuition. Several implementation details that are either completely unmentioned or glossed over in the litterature are also presented. It should be fairly comprehensive so as to allow the reader to implement the full algorithm. That's because I'm redacting this partly for documentation purposes, not too long after finishing the implementation.

Basic algebra and convex optimization notions are not explained. The reader unfamiliar with such notions should refer to documents such as [1].

Overview

Given a convex rational polytope $P = \{x \in \mathbb{Z}^n, Ax \leq B\}$, find $\#P$. The algorithm steps are:

- view the polytope as a signed sum of (pointed) tangent cones about a vertex and of tangent cones of faces.
- The generating function view allows us to consider only the tangent cones about a vertex as the others are non-pointed.
- Decompose each cone as a bunch of simplicial cones. The inclusion or exclusion of each face matters so as not to count points multiple times.
- Compute the generating function of a simplicial cone.
  1. Recursively express a simplicial cone as a signed sum of simplicial cones so that fundamental parallelepiped of the cone can be enumerated.
  2. List the points in the fundamental parallelepiped
- Evaluate the generating function at $z = 1^n$

The generating function in practice is not explicitely computed. The evaluation is directly updated as soon as more terms of the generating function are obtained. This however transforms the algorithm to a randomized one but as we will see, failures do not matter as they happen on a measure zero set.

The main algorithm is first described for the simplest configuration: a non degenerate polytope with integer vertices. In the last section, we address how these generalizations are handled.

Toward a representation of the polytope as a signed sum of pointed cones

We use the usual indicator function of a convex set C, defined as:

$$[C] : \mathbb{R}^n \mapsto \mathbb{R}, [C](x) = \begin{cases} 1 & \text{if } x \in C \\ 0 & \text{otherwise} \end{cases}$$

Brion's theorem states that:

$$[P] = \sum_{\emptyset \neq F \text{ faces of } P \text{ (including P itself)}} (-1)^{\dim F} [tcone(P, F)]$$

, where $tcone(P, F)$ is the tangent cone at any point $x_F \in F$, <u>in the interior of F</u> (e.g. the tangent cone of an edge of a convex 2D polygon is the half-plane containing P, supported by the edge)

The generating function of a polytope P is:

$$f(x, P) = \sum_{z \in \mathbb{Z}^n} [P](z) \times z = \sum_{z \in P \cap \mathbb{Z}^n} z$$

For a simplicial pointed cone at vertex v with rays $\bigcup_i r_i$, the generating function is

$$f(x, Cv) = z^v \frac{\sum_{u \in (\Pi \cap \mathbb{Z}^n)} z^u}{\prod_i (1 - z^{r_i})}$$

This formula is very important and we'll make use of it later. Nice explanatory pictures are found in [2]. The nice thing with this representation is that the generating function of non-pointed cones is 0. An easy example showing this is the generating function for the x-axis line:

$$\sum_{i=-\infty}^{\infty} x^i = \left(\sum_{i=0}^{\infty} x^i + x^{-i} \right) - x^0 = \frac{1}{1-x} + \frac{1}{1-x^{-1}} - 1 = \frac{(1-x^{-1}) + (1-x) - (2-x-x^{-1})}{2-x-x^{-1}} = 0$$

Thus we can discard terms in the Brion formula where $\dim F \neq 0$ (because all these terms are non pointed cone). We are left with $f(x, P) = \sum_{v \in vertex(P)} f(x, cone(P, v))$

Suppose we have found the generating function $f(x, P)$. Evaluating this function at 1^n gives us the number of integer points in P, what we are looking for.

Going from cones to simplicial cones

We must go in 3D to find a non-simplicial cones we can have to handle coming from a polytope. A pyramid has one non simplicial cone. A simplicial cone in $n-$dimensions has n rays that span the whole space. Finding a decomposition of a cone into simplicial ones is easy.

Splitting the cone with a hyperplane

A non-simplicial cone has at least $n+1$ rays. There exists a linear hyperplane formed by $n-1$ rays such that rays lie on both side of the hyperplane. Starting with k rays, we end up with a left and a right set each at most k-1 rays. Moreover, the left and right cones only intersect on the splitting hyperplane. Repeating this procedure, we eventually end up with a simplicial decomposition.

A sketch of a proof of why such a splitting linear hyperplane exists: Select n rays r_i spanning the whole space. Choose any remaining ray u. if $u \notin cone(r_i)$, then obviously a face of $cone(r_i)$ will fit as the splitting hyperplane. It remains the case for $u \in cone(r_i)$. We have $u = \sum \lambda_i r_i$, with $\lambda_i \geq 0$. Moreover, say $\lambda_1, \lambda_2 > 0$ (at least two are nonzero as it does not make sense to consider a cone with two collinear rays). Thus

$$\lambda_1 r_1 = u - \lambda_2 r_2 - \sum_{i>2} \lambda_i r_i$$
$$r_1 = \mu_u u + \mu_2 r_2 + \sum_{i>2} \mu_i r_i$$

, with $\mu_2 < 0$. Thus $r_1 \notin cone(u, r_2, ..., r_n)$. We are back in the previous case, where a face of this new cone can be chosen as the splitting hyperplane. $\square$

The simplicial cones are not disjoint

The only problem with this splitting is that the cones are sharing faces. It means that we might double count integer points. The solution for this is to be able to compute the generating function of a cone where we additionally decide to either include or exclude faces. It requires two things.

- decide how the including-excluding for the faces is done.
- how the generating function is computed with this additional constraint taken into account.

We'll discuss the first problem here while we defer the second problem to the next section.

Attributing the include-exclude property to each face

To sum up the situation, a cone is split into multiple simplicial cones. The union is the original cone but it is not disjoint. To make it disjoint, we need to play with the inclusion-exclusion of faces. More formally:

- an extended simplicial cone is defined as $C_k = C_{raw,k} \cup \bar{F}_k$, where $C_{raw,k} = cone\left(\bigcup_{i \in I_k} r_i\right)$ and F_k is the set of faces that should be included ($\bar{F}_k$ excluded) for the cone C_k. A face has a one-to-one mapping to a ray: we associate to r_a the face $cone(\bigcup_{i \neq a} r_i)$.
- Denote by F_0 the set of faces in the original cone and by F_1 the set of inner faces, created by the split (i.e. the set of faces of the simplicial cones minus F_0).
- We must have $\bigcup_k F_k = F_0 \cup F_1$, as well as the condition that $\forall (f_a, f_b) \in (F_0 \cup F_1)^2, f_a \cap f_b = \dot{\bigcup}_k (F_k \cap (f_a \cap f_b))$, with the emphasis on the disjoin union.

That will prevent the double counting. To find an assignement in a simple manner, one needs to notice that F_0 contributions comes from the faces of one simplicial cones and faces F_1 comes from exactly two simplicial cones. Thus each condition rests on at most two variables, the inclusion-exclusion of a face for at most two simplicial cones. Hello 2-SAT. I don't have a non-tedious way to present the 2-SAT conditions so I'll leave it as an exercise. As for the existence of a solution, there is one (in the literrature it's not achieved with 2-SAT but with a constructive algorithm). As a side note, the existence relies on the all-inclusivity of the faces of the original cone, as some pattern exists where no solution can be found (eg 1-0-1-0 for the pyramid).

This 2-SAT representation will also be used in the next section for face control when splitting simplicial cones in search for cones with lower index.

Computing the generating function of a simplicial cone

Here we are given a simplicial cone about a vertex and a boolean list representing whether to include or exclude each face. The generating function for a cone about a vertex v, with all faces included, is: $f(x, Cv) = z^v \frac{\sum_{u \in \Pi} z^u}{\prod_i (1 - z^{r_i})}$. Pictures in [2] help undertanding the formula.

Enumerating integer points in the fundamental parallelepiped

The fundamental parallelepiped is $\Pi = \{\sum \lambda_i r_i, 0 \leq \lambda_i < 1\}$ .The solution to account for the inclusion-exclusion of faces is obvious here. We just have to play on the strict-or-not inequalities. So for example, if f_0 is excluded and f_1 is included, we have $\Pi = \{\sum \lambda_i r_i, 0 < \lambda_0 \leq 1, 0 \leq \lambda_1 < 1\}$. That's all. We now have to to enumerate the points in the parametrized fundamental parallelepiped.

Smith normal form of the lattice matrix helps a lot

First, we observe how the number of integer points in the fundamental parallelepiped is related to the determinant of the matrix with the lattice vectors as columns.

$$\begin{aligned} \#\left(\Pi_A \cap \mathbb{Z}^n\right) &= |\det(A)|, \text{ with } A = (r_1|...|r_n), r_i \text{ the cone rays} \\ &= \operatorname{ind} cone(r_1, ..., r_n) \end{aligned}$$

To see this, put A in the Smith normal form, $A = UDV$, (U, D, V integer matrices, D diagonal (plus a condition on the diagonal elements, not relevant here), U and V unimodal). This form always exists and can be computed easily by alternating row and column reduction. There is a bijection between elements of $\mathcal{L}(A)$ and $\mathcal{L}(D)$ (the map $\phi(x) = U^{-1}x$ maps elements of $\mathcal{L}(A)$ to elements of $\mathcal{L}(D)$. Bijective map because V unimodular, and the inverse of an integer unimodular matrix is also integer). Integer points of the fundamental parallelepipeds of $\mathcal{L}(A)$ and $\mathcal{L}(D)$ also are in a one-to-one mapping.

Enumerating points using D and U

Enumerating the points in the fundamental parallelepiped of $\mathcal{L}(D)$ is trivial as D is diagonal. It simply consists of the vectors $\mathbf{0} \leq \mathbf{x} < \mathrm{diag}(D)$. Given $x \in \Pi_D$, Ux corresponds to a point in the fundamental parallelepiped in Π_A, modulo L(A).

Finding the class representative in Π_A

We are gien a point y, and we want to find it's equivalent in Π_A. Compute the inverse A^{-1} over the $\mathbb{Q}$. $A^{-1}y = (\lambda_1 ... \lambda_n)$. $z = A(\bar{\lambda_1} ... \bar{\lambda_n})$ with $\bar{\lambda_i} = k + \lambda_i, k \in \mathbb{Z}, 0 \lesseqgtr \bar{\lambda_i} \lesseqgtr 1$ is the corresponding point in the face parametrized fundamental parallelepiped. The weird symbol $\lesseqgtr$ is a reminder that we play here on strict-or-not inequalities here for the inclusion-exclusion of faces.

Decomposing a simplicial cone as signed sum of simpler simplicial cones

Complexity-wise, one cannot enumerate a given simplicial cone as its size is exponential in the coordinates of the polytope. The solution is to decompose the simplicial cone as a signed sum of simpler simplicial cones. This procedure is repeated until we obtain simplicial cones of manageable index. The branching factor is at most n (n as in $\mathbb{Z}^n$) but the decrease of the index compensates more than enough so that the depth of the tree ends up being $O(\log \log \mathrm{ind}\, B)$, with B the original simplicial cone (treated nicely in [1], 7.1.9).

A signed decomposition by choosing one vector

To achieve this, we will select a particular vector w and consider the cones $K_i = cone(..., r_{i-1}, w, r_{i+1}, ...; \theta), \theta$ indicating a particular parametrization for the inclusion-exclusion of the faces. One condition for this to work is that $cone(r_1, ..., r_n, w)$ must be pointed. This will be true iff $w \notin -cone(r_1, ..., r_n)$ (proof is easy). The condition gives an efficient algorithm to detect such cases. Then one can use $-w$.

Finding the sign for each cone

$$[K] = \sum \varepsilon_i [K_i]$$

$$, \text{with } \varepsilon_i = \begin{cases} 1 & \text{if } r_i \text{ and } w \text{ are on the same side of the hyperplane containing} \{0, r_1, ..., r_{i-1}, r_{i+1}, r_n\} \\ -1 & \text{if on different sides} \\ 0 & \text{if } w \text{ is on the hyperplane. We can drop all such cones, they won't contribute} \end{cases}$$

Setting the include-exclude property of faces to avoid double counting

The inclusion-exclusion of faces is determined by modelling the problem as a satisfibility problem, with clauses of at most 2 variables. 2-SAT again. Denote by $X_{ij} = 1$ if the face j of K_i is included and similarly, Y_j the condition for the input cone K.

$$X_{ii} * \varepsilon_i = \begin{cases} 1 & \text{if } Y_i \\ -1 & \text{otherwise} \end{cases}$$

$$X_{ij}\varepsilon_i + X_{ji}\varepsilon_j = \begin{cases} 1 & \text{if } Y_i \vee Y_j \\ 0 & \text{otherwise} \end{cases}, j \neq i,$$

Choosing the vector w guiding the decomposition

As the cone K is of full dimension, we have $w = A\alpha$. (A the matrix of K (columns are r_i)) Let

$$\begin{aligned} U &= (\mathrm{ind}\, K_1, ... \mathrm{ind}\, K_n) \\ &= (\det(\sum r_i \alpha_i, r_2, ..., r_n), ..., \det(r_1, r_2, ..., , \sum r_i \alpha_i)) \\ &= (\alpha_1 \,\mathrm{ind}\, K, ..., \alpha_n \,\mathrm{ind}\, K) \\ &= \alpha \times \mathrm{ind}\, K \end{aligned}$$

We want to find the shortest vector α (relative to the norm $\|\|_\infty$), with w integer.

$$\alpha = A^{-1}w = ((A^{-1} \times \det(A))w)/\det(A)$$

The matrix $(A^{-1} \times \det(A))$ is an integer matrix (Cramer's rule). Finding the minimum α is then equivalent to finding the shortest vector of the lattice spanned by the columns of $(A^{-1} \times \det(A))$. The LLL algorithm can do that. w is recovered easily from α.

Note: one should always take the gcd of the coefficients for each rays, otherwise it will be hard to reduce the index.

Evaluating the generating function at $z = 1$

The full generating function looks like:

$$\begin{aligned} f(z, P) &= \sum_{cone(C_i, v_i, \theta_i)} \varepsilon_i z^{v_i} \frac{\sum_{u \in \Pi_{C_i, \theta_i}} z^u}{\prod_{j=1}^{n} (1 - z^{r_{C_{i,j}}})} \\ &= \sum_i \varepsilon_i \frac{z^{a_i}}{\prod_{j=1}^{n} (1 - z^{b_{ij}})} \end{aligned}$$

Warning: mathematical soundness of what follows is probably garbage; I'm just lowly software engineer that had just a few analysis class ages ago. Anyways.

From multivariate limit to univariate

Looking at the formula we see that z=1 is a pole, so it's not that simple to evaluate. As the number of poles is finite (one for each rays), the function is meromorphic on $\mathbb{R}^n$. We can evaluate the function at z=1 using limits from whichever direction. Let $z_\tau = (e^{\tau\lambda_1}, ..., e^{\tau\lambda_n})$. Then

$$\begin{aligned} \lim_{z \to 1} f(z, P) &= \lim_{\tau \to 0} f(z_\tau, P) \\ &= \lim_{\tau \to 0} \sum_i \varepsilon_i \frac{e^{\tau < \lambda, a_i >}}{\prod_{j=1}^{n} (1 - e^{\tau < \lambda, b_{ij} >})} \\ &= \lim_{\tau \to 0} \sum_i \varepsilon_i \tau^{-n} e^{\tau < \lambda, a_i >} \prod_{j=1}^{n} \frac{\tau}{(1 - e^{\tau < \lambda, b_{ij} >})} \end{aligned}$$

If $\forall i, j, < \lambda, b_{ij} > \neq 0$, we still have a meromorphic function and the limit approach remains valid.

Removing the limit by Taylor expansions

The last change in the previous equations was done because the function $h(x) = x/(1 - e^{-x})$ is analytical in the neighborhood of 0, meaning that there is a taylor expansion at 0.

$$\begin{aligned} \#P &= \lim_{\tau \to 0} \sum_i \varepsilon_i \tau^{-n} e^{\tau < \lambda, a_i >} \frac{\tau}{\prod_{j=1}^{n} (1 - e^{\tau < \lambda, b_{ij} >})} \\ &= \lim_{\tau \to 0} \sum_i \varepsilon_i \tau^{-n} f_1(\tau, a_i) \prod_{j=1}^{n} f_2(\tau, b_{ij}) \\ &= \sum_i \varepsilon_i \left(f_1(x, a_i) \prod_{j=1}^{n} f_2(x, b_{ij}) \right) [n], \text{ where } [n] \text{ means getting the coefficient of } x^n \end{aligned}$$

Cool, we only need to compute the taylor expansion of f_1 and f_2 with $n + 1$ terms. Note that for efficiency, the polynomial multiplications should be done mod x^{n+1} so that useless terms are never computed.

Computing the Taylor expansions

f_1 is trivial.
$f_2(x, b_{ij}) = \frac{h(-\tau < \lambda, b_{ij} >)}{-< \lambda, b_{ij} >}$ We now need the taylor expansion of h(x). It's easy to compute the expansion of $1/h(x) = (1 - e^{-x})/x$ (remove first term of $-e^{-x}$, then shift by 1). Then,

$$1 = h(x) \times (1/h(x)) = (\sum a_i x^i)(\sum b_i x^i)$$

. A decent recurrence formula is not hard to obtain from this([1] chapter 7 covers this in depth).

Choosing λ

Papers use a rather involved method using the moment curve. If you're fine with a randomized algorithm then just select a random vector and you'll do ok. The reason for this is that the set of bad vectors is measure zero. That's because you have a finite number of rays which must not be orthogonal to your λ. A finite union of measure zero sets is a measure zero set (the subspace orthogonal to a nonzero vector of dimension n is of dimension n-1, hence of measure zero).

Handling complications

Rational vertices

Our generating function is a Laurent serie thus the variables have integer coefficients. That does not sit well with z^{v_i} nor Π_{C_i,θ_i} in the formula. This is handled by shifting the cone to a nearby integer point and enumerating the points of the fundamental parallelepiped so that these points are in the fundamental parallelepiped of the cone at the rational vertex by playing on the lattice coefficients of the shift (so that $0 \lesseqgtr \lambda_i \lesseqgtr 1$ in the view of the original rational cone).

Degenerate polytopes

A degenerate polytope is defined as $P = \{x | Ax \leq B, Cx = D\}$. If the polytope is not full dimensional, everything breaks down as lattice determinant are valid only for full dimensional lattices. The solution is to project the polytope on a lower dimensional space such that there is a bijection between the integer points of the original and projected polytope. This is done by projecting iteratively on a subspace of dimension one less.

Let r be the first column of C, with $\gcd(r) = 1$ (important, and if can't simplify with D_1, no solution exist). There exists an integer unimodular matrix M such that $rM = (1, 0, ..., 0)$ (hint: as $\gcd = 1$, r can be reduced to $(1, 0, ..., 0)$ by unimodular operations. Representing these operations in a matrix gives M). Since M is integer unimodular, integer points $\mathbb{Z}^n$ and $M \times \mathbb{Z}^n$ are in bijection. Additionally, let $x \in P$. Then

$$r^t x = (x^t M^{-1})_1 = D_1, \text{ (M^{-1} has r as its first row)}$$

In other words, $x^t M^{-1} = \begin{pmatrix} D_1 \\ y \end{pmatrix}$ and $x = M \begin{pmatrix} D_1 \\ y \end{pmatrix}$ (y integer because M is integer and unimodular). The new polytope hyperplanes in the subspace are computed next. As a reminder, $P = \{x | Ax \leq B, Cx = D\}$. Consider one relation:

$$\begin{aligned} A_i^t x &\leq B_i \\ A_i^t \left(M \begin{pmatrix} D_1 \\ y \end{pmatrix} \right) &\leq B_i \\ A\prime_i^t y &\leq B\prime_i \\ \text{, where } A\prime_i^t &= (A_i^t M)_{2...n} \\ B\prime_i &= B_i - (A_i^t M)_1 \times D_1 \end{aligned}$$

Similarly, relations $Cx = D$ are updated.

Switching between cone and planar representation of a polytope

Degenerate polygons forces us to have primitives to switch the representation of a polytope. The simplest way to achieve this is finding submatrices of full rank from the constraints to find vertices and computing hyperplanes from n-1 vertices. Standard stuff, not much to talk about.

Closing words

All done. There's plenty more to talk about the applications of this algorithm but the references are there for that. Anyway, the algorithm in itself is worth the journey. I find especially pretty how the representation as a polynomial allows us to drop non pointed cones and then how the evaluation of the polynomial is done through series expansions.

I might release the C++ code if there is interest (< 2000 lines). However it would either be unusable as is or I would need to release my personal codebase (hurray single repo :)).

Until then,

References

[1] Jesús A. De Loera, Raymond Hemmecke, and Matthias Köppe. Algebraic and Geometric Ideas in the Theory of Discrete Optimization. Society for Industrial and Applied Mathematics, dec 2012.

[2] Ruriko Yoshida. Barvinok's enumeration algorithm and applications to Statistics. ppt available at http://polytopes.net/pdf/ISM.pdf.