An algebraic-perturbation variant of Barvinok’s algorithm

Jon Lee\textsuperscript{1,2}

Department of I\&OE
University of Michigan
Ann Arbor, MI, U.S.A.

Daphne Skipper\textsuperscript{3}

Department of Mathematics
Georgia Regents University
Augusta, GA, U.S.A.

Abstract

We give a variant of Barvinok’s algorithm for computing a short rational generating function for the integer points in $P := \{x \in \mathbb{R}^n : Ax \leq b\}$; a use of which is to count the number of integer points in $P$. We use an algebraic perturbation, replacing each $b_i$ with $b_i + \tau^i$, where $\tau > 0$ is an arbitrarily small indeterminate. Hence, our new right-hand vector has components in the ordered ring $\mathbb{Q}[\tau]$ of polynomials in $\tau$. Denoting the perturbed polyhedron by $P(\tau) \subset \mathbb{R}[\tau]^n$, we use the facts that: $P(\tau)$ is full dimensional, simple, and contains the same integer points as $P$.

Keywords: Barvinok’s algorithm; generating function; polyhedron; integer

\textsuperscript{1} Lee was partially supported by NSF grant CMMI-1160915; ONR grant N00014-14-1-0315.
\textsuperscript{2} Email: jonxlee@umich.edu
\textsuperscript{3} Email: dapskipper@gru.edu

Available online at www.sciencedirect.com
1 Introduction

Let $P := \{ x \in \mathbb{R}^n : Ax \leq b \}$ be a nonempty pointed polyhedron given by $m$ rational inequalities. In celebrated work, Barvinok gave an algorithm, which is polynomial-time when the dimension $n$ is fixed, for computing a short rational generating function for the integer points in $P$ (see [1, 2, 3]). One use of such an algorithm could be to compute integer edge-weighted graphs having some property (e.g., for given $c \in \mathbb{Z}_+^n$ and $b \in \mathbb{Z}_+^m$, the number of “perfect $c$-capacitated $b$-matchings” of a given graph on $n$ edges and $m$ vertices). Throughout, our viewpoint is that we are thinking about what is efficient/practical when the dimension $n$ is thought of as fixed, and the rational data $A$ and $b$ and number of constraints $m$ can be considered varying.

2 Perturbation variant

Before presenting our algorithm, we describe the setting to which we are adapting Barvinok’s algorithm. Next, we review the fundamental geometric results (concerning indicator functions of polyhedra and generating functions for the contained integer points) that Barvinok’s algorithm rests upon. Then, we present each step of our perturbation variant, as we describe its adaptation from a vanilla Barvinok algorithm.

The perturbation.

We make an algebraic perturbation of the right-hand side $b \in \mathbb{Q}^m$ of the inequalities, replacing each $b_i$ with $b_i + \tau^i$, for $i = 1, 2, \ldots, m$, where $\tau$ is considered to be an arbitrarily small positive real indeterminate. Hence, elements of our right-hand side vector become elements of the ordered ring $\mathbb{Q}[\tau]$ of polynomials in $\tau$ with rational coefficients. The ordering of polynomials in $\mathbb{Q}[\tau]$ is the usual one: $p(\tau) := \sum_{i=1}^{k} p_i \tau^i <_\tau q(\tau) := \sum_{i=1}^{k} q_i \tau^i$ if the least $l$ for which $p_l \neq q_l$ has $p_l < q_l$. Denoting the algebraically-perturbed polyhedron as $P(\tau) := \{ x \in \mathbb{R}[\tau]^n : Ax \leq \tau b + \tau \}$, where $\tau' := (\tau, \tau^2, \ldots, \tau^m)$, it is easy to see that: (i) $P(\tau)$ is always full dimensional, (ii) $P(\tau)$ is always simple, and (iii) $P(\tau)$ contains the same integer points as $P$.

Regarding (i), we assume that $P \neq \emptyset$, so there is a point $\hat{x} \in P$. Therefore, $A_i \hat{x} \leq b_i < b_i + \tau^i$, for $i = 1, 2, \ldots, m$. Hence, $P(\tau)$ is full dimensional.

Regarding (ii), a vertex of $P(\tau)$ is the unique solution $v \in \mathbb{R}[\tau]^n$ of $A_l v = b_l + \tau^l$, for $l \in \beta$, where $A_l$ is the $l$th row of $A$, $\beta \subset \{ 1, 2, \ldots, m \}$ and $|\beta| = n$, provided of course that such a unique solution exists and that it is in $P(\tau)$. Note that the components of $v$ are polynomials in $\tau$ and that all components
have no terms $\tau^j$ with nonzero coefficient having $j \notin \beta$. Therefore, $A_jv \neq b_j + \tau^j$ for $j \notin \beta$. Hence, $P(\tau)$ is always simple. We further note that $v \in \mathbb{Q}[\tau]^n$ when $A$ and $b$ are rational.

Regarding (iii), we wish to emphasize that the set of integers in $\mathbb{R}[\tau]$ is precisely the ordinary set of integers $\mathbb{Z}$.

Our Contribution.

Unlike other versions of Barvinok’s algorithm, we will have to do some arithmetic in $\mathbb{Q}[\tau]$. However, because of (i), we will not need to preprocess our input polyhedron if it is not full dimensional, and because of (ii), we will not need to triangulate tangent cones at the vertices of the polyhedron. So, in important respects, our variant is simpler.

The algorithm.

Before getting started, we lay out the fundamental result that Barvinok’s algorithm is based on. In what follows, $[\cdot]$ is the indicator function operator, and $\text{tcone}(P,F)$ denotes the tangent cone of polyhedron $P$ from its face $F$.

**Theorem 2.1 (Brion’s Theorem)** Let $P \subset \mathbb{R}^n$ be a polyhedron. Then

$$[P] \equiv \sum \{[\text{tcone}(P,v)] : v \text{ is a vertex of } P\} \quad (\text{modulo indicator functions of non-pointed cones}).$$

While this does not give us an exact identity for indicator functions, we do obtain an exact identity for the associated generating functions. We use the standard monomial notation where, for an $n$-vector of variables $z$ and an $n$-vector of integers $a$, we write $z^a := \prod_{i=1}^n z_i^{a_i}$. The generating function for the integer points of $P$ is defined as

$$g(P; z) := \sum_{a \in P \cap \mathbb{Z}^n} z^a.$$

**Theorem 2.2 (Brion’s Theorem for generating functions)** Let $P \subset \mathbb{R}^n$ be a polyhedron. Then

$$g(P; z) = \sum \{g(\text{tcone}(P,v); z) : v \text{ is a vertex of } P\}.$$ 

We do not seek to calculate an exact formula for $[P]$. Rather, we want a representation of the generating function $g(P; z)$ as a short rational function, and this is where Barvinok comes in.
1: **input** a rational pointed polyhedron \( P := \{ x \in \mathbb{R}^n : Ax \leq b \} \).

2: **output** a rational generating function for \( P \cap \mathbb{Z}^n \) in the form

\[
g(P; z) := \sum_{i \in I} \epsilon_i \sum_{a \in A_i} z^a \prod_{j=1}^n (1 - z^{b_{ij}}),
\]

where \( \epsilon_i \in \{ \pm 1 \} \), \( b_{ij} \in \mathbb{Z}^n \), and \( A_i \subset \mathbb{Z}^n \) for \( i \in I \), where \( I \) and the \( A_i \) are finite.

3: compute all vertices \( v \) and corresponding tangent cones \( C \) of \( P(\tau) \). Because \( P(\tau) \) is simple, each tangent cone \( C \) is simplicial in our setting. We organize this information as a set \( \mathcal{T} \) of triples \((C, v_C, \epsilon_C)\) (that we will update). In each triple, \( C \) is a pointed (simplicial) cone rooted at a vertex \( v_C \) of \( P(\tau) \), and we store an associated sign \( \epsilon_C \in \{ \pm 1 \} \). Initially, we let

\[
\mathcal{T} := \{ (C, v, +1) : v \text{ is a vertex of } P(\tau), \text{ and } C \text{ is the associated (simplicial) tangent cone of } P(\tau) \}.
\]

To compute the vertices, we can simply check all possible \((m^n) = \mathcal{O}(m^n)\) choices of sets of \( n \) basic rows from \( A \) (recall our assumption that \( n \) is fixed).

If \( v \in \mathbb{Q}[\tau]^n \) is the unique solution \( x \in \mathbb{R}[\tau]^n \) of \( A_l x = b_l + \tau l \), for \( l \in \beta \), then the simplicial cone \( C \) is the solution set in \( \mathbb{R}[\tau]^n \) of \( A_l x \leq 0 \), for \( l \in \beta \). Note that each such \( C \) is a rational cone — i.e., its generators are in \( \mathbb{Q}^n \).

4: the next step would normally be to triangulate each of the tangent cones \( C \), but in our setting each \( C \) is simplicial, so there is nothing to do here. Of course, our perturbation created more vertices for \( P(\tau) \) than for \( P \), effectively perturbing non-simple vertices to create many simple vertices. So we do not have to triangulate tangent cones, but we have more of them up front.

5: the index of a simplicial rational cone \( C := \text{cone}(\{h_1, h_2, \ldots, h_n\}) \), where each \( h_i \) is a primitive vector of the integer lattice \( \mathbb{Z}^n \) and \( H := [h_1, h_2, \ldots, h_n] \) is ind \( C := |\det H| \). In this step, we apply “signed decomposition”, possibly repeatedly, to any “high-index” cone \( C \) from a triple \((C, v_C, \epsilon_C)\) on the list \( \mathcal{T} \), replacing the associated triple with \( \leq n \) child triples (all with the same vertex \( v \) as the mother) — we need to give a recipe for calculating (the signs and cones of) the children. The least index of a cone is 1 (a unimodular cone), but we may be satisfied with cones with index below some threshold.
One iteration of Barvinok’s signed decomposition works as follows on a triple \((C, v_C, \epsilon_C)\): we decompose the simplicial rational cone \(C := \text{cone}\{h_1, h_2, \ldots, h_n\}\), based on a nonzero point \(w \in \mathbb{Z}^n\) such that \(\text{cone}\{h_1, h_2, \ldots, h_n, w\}\) is pointed. Let \(C_j := \text{cone}\{h_1, h_2, \ldots, h_{j-1}, w, h_{j+1}, \ldots, h_n\}\), Then there exist \(\epsilon_j \in \{0, \pm 1\}\), such that \([C] \equiv \sum_{j=1}^n \epsilon_j[C_j]\), \(\epsilon_j = 0\) if \(C_j\) is not full dimensional.

We will get \(w\) by finding a \(u\) related to a short nonzero vector in a particular lattice. Then we let \(w := Hu\). If \(\text{cone}\{h_1, h_2, \ldots, h_n, w\}\) is not pointed, then we simply substitute \(w\) with \(-w\), so it is easy to get the pointedness to hold. We compute the \(\epsilon_j \in \{0, \pm 1\}\) by simply letting \(\epsilon_j := \text{sign}(u_j)\), for \(j = 1, 2, \ldots, n\).

We have essentially specified how to get everything needed. Now we replace the mother \((C, v_C, \epsilon_C)\) with the set of children \(\{(C_j, v_C, \epsilon_C \cdot \epsilon_j) : \epsilon_j \neq 0\}\).

Finally, we explain how we get \(u\). The index of the cone \(C_j\) is
\[
\lambda_j := \det[h_1, h_2, \ldots, h_{j-1}, w, h_{j+1}, \ldots, h_n],
\]
for \(j = 1, 2, \ldots, n\). Employing Cramer’s rule, we have \(u = \lambda / \det H\), and we see that \(\lambda = \text{adj} H \cdot w\), where \(\text{adj} H := \det H \cdot H^{-1}\) is the well-known adjugate (or classical adjoint, or transposed matrix of cofactors) of \(H\). Note that \(\text{adj} H\) is integer when \(H\) is. So \(\lambda\) is a nonzero vector in the lattice \(\mathcal{L} := \text{adj} H \cdot \mathbb{Z}^n\). Our goal is to have the cones \(C_j\) all have low index. So it is natural to choose \(\lambda\) to be a short nonzero vector in the lattice \(\mathcal{L}\). Any norm will do, as long as we can reliably get within a constant factor of the shortest. So employing the \(\ell_2\)-norm and using the LLL algorithm is a natural choice. From \(\lambda\) we get \(u = \lambda / \det H\), and from \(u\) we get \(w = Hu\).

We end up with our list \(\mathcal{T}\) having all cones with low index. We also have
\[
g(P; z) = \sum_{(C, v_C, \epsilon_C) \in \mathcal{T}} \epsilon_C \cdot g(v_C + C; z),
\]
starting from Brion’s theorem for generating functions, and (repeatedly) substituting according to Barvinok’s signed decomposition, as we refine \(\mathcal{T}\).

6: now, for each \((C, v, \cdot) \in \mathcal{T}\), suppose that the simplicial cone \(C\) has the representation \(C := \{Hu : u \in \mathbb{R}^n_+\}\), with \(H \in \mathbb{Z}^{n \times n}\) and the columns of \(H\) being primitive integer vectors. We also have the associated point \(v \in \mathbb{Q}[\tau]^n\). We define the semi-open parallelepiped
\[
\Pi(v, H) := v + \{Hu : 0 \leq u < e\} \subset \mathbb{R}[\tau]^n.
\]
We want to enumerate the integer points in \( \Pi(v, H) \). Consider the Smith Normal Form of \( H \). That is, \( HU = WS \), where \( U \) and \( W \) are integer unimodular matrices and \( S = \text{diag}(s) \) is a diagonal integer matrix. Then

\[
\Pi(v, H) \cap \mathbb{Z}^n = \{ Ww - H[-H^{-1}(v - Ww)] : 0 \leq w < s, \ w \in \mathbb{Z}^n \}.
\]

Note that the round-down function \( \lfloor \cdot \rfloor \) is applied to polynomials in \( \tau \), since \( v \in \mathbb{R}[\tau]^n \). We have no difficulty in applying \( \lfloor \cdot \rfloor \), as the ordered ring \( \mathbb{Z} \) is a subset of the ordered ring \( \mathbb{R}[\tau] \). It is worth emphasizing that finally \( \Pi(v, H) \cap \mathbb{Z}^n \) is a finite set of integer points — no presence of \( \tau \).

**7:** write down the formula of step 2.

\[
g(P; z) := \sum_{(C, \nu_C, \epsilon_C) \in \mathcal{T}} \epsilon_C \cdot \frac{\sum_{a \in \Pi(v_C, H^C) \cap \mathbb{Z}^n} z^a}{\prod_{j=1}^n (1 - z h^C_j)},
\]

where \( H^C := [h^C_1, h^C_2, \ldots, h^C_n] \) is the matrix with columns generating \( C \). We have really calculated \( g(P(\tau); z) \), but observing that finally this expression does not depend on \( \tau \), we have as well calculated \( g(P; z) \).

**Theorem 2.3** The APB algorithm runs in polynomial time, in fixed dimension \( n \).

**Proof.** [sketch] The proof rests on the proof that Barvinok’s algorithm runs in polynomial time, in fixed dimension \( n \). We simply also observe that for the arithmetic that we do in \( \mathbb{R}[\tau] \), all of the polynomials arising have at most \( n \) nonzero terms, and the coefficients are (by Cramer’s rule) rational numbers having numerators and denominators bounded in bit size by a polynomial in the sizes of numbers in the input matrix \( A \).

**References**

