Analytical Computation of Ehrhart Polynomials: Enabling more Compiler Analyses and Optimizations

Sven Verdoolaege
Dept. of Computer Science
K.U.Leuven
sven@cs.kuleuven.ac.be

Rachid Seghir
ICPS-LSIIT UMR 7005
Université Louis Pasteur,
Strasbourg
seghir@icps.u-strasbg.fr

Kristof Beyls
Dept. of Electronics and
Information Systems
Ghent University
kristof.beyls@elis.UGent.be

ABSTRACT

Many optimization techniques, including several targeted specifically at embedded systems, depend on the ability to calculate the number of elements that satisfy certain conditions. If these conditions can be represented by linear constraints, then such problems are equivalent to counting the number of integer points in (possibly) parametric polytopes.

It is well known that this parametric count can be represented by a set of Ehrhart polynomials. Previously, interpolation was used to obtain these polynomials, but this technique has several disadvantages. Its worst-case computation time for a single Ehrhart polynomial is exponential in the input size, even for fixed dimensions. The worst-case size of such an Ehrhart polynomial (measured in bits needed to represent the polynomial) is also exponential in the input size. Under certain conditions this technique even fails to produce a solution.

Our main contribution is a novel method for calculating Ehrhart polynomials *analytically*. It extends an existing method, based on Barvinok's decomposition, for counting the number of integer points in a non-parametric polytope. Our technique always produces a solution and computes polynomially-sized Ehrhart polynomials in polynomial time (for fixed dimensions).

Categories and Subject Descriptors

G.2.1 [**Discrete Mathematics**]: Combinatorics—*Counting problems,Generating functions*; D.3.4 [**Programming Languages**]: Processors—*Compilers*

General Terms

Algorithms, Performance

Keywords

Barvinok's decomposition, compiler analysis, Ehrhart polynomial, parametric polytope, polyhedral model, quasi-polynomial, signed unimodular decomposition

1. INTRODUCTION

In many program analyses and optimizations, questions starting with "how many" need to be answered, e.g.

- How many memory locations are touched by a loop?[16]
- How many operations are performed by a loop?[23]
- How many cache lines are touched by a loop?[16]
- How many array elements are accessed between two points in time?[3]
- How many array elements are live at a given iteration (i, j)?[33]
- How many times is a statement executed before an iteration (i, j)?[32, 17]
- How many parallel processing elements can be used when executing a loop on an FPGA?[21]
- How many cache misses does a loop generate?[11, 9, 19]
- How much memory is dynamically allocated by a piece of code?[6]
- How many different array elements are accessed before element (x, y) is accessed?[25]

Answering these questions is often one of the corner stones in program analyses and optimizations important in embedded systems development, such as increasing parallelism[32], minimizing memory size[33, 32], estimating worst case execution time[23], increasing cache effectiveness[3], high-level transformations for DSP applications[17] and converting software loops into parallel hardware implementations[32, 21].

In all these compiler optimizations, the "counting" problems are modeled by linear inequalities, and the question becomes "How many integer points $x \in \mathbb{Z}^d$ satisfy the system of inequalities $Ax \geq b$?". Furthermore, the number of integer points often needs to be expressed as a function of a number of parameters. In some optimization techniques, the need for parameters depends on the problem instance, while in other techniques, e.g., [3, 6, 9, 16, 17, 23, 25, 32], the counting problem is intrinsically parametric.

```
S1 void s(int N, int M)
S2 {
S3   int i,j;
S4   for(i=max(0,N-M); i<=N-M+3; i++)
S5     for(j=0; j<=N-2*i; j++)
S6       S1;
S7 }
```

Figure 1: Example loop code

Recently, new techniques have been developed for such counting problems, but they either do not support parameters, e.g., [5, 28], or only to a very limited extent, e.g., `LattE` [14]. Earlier, Clauss [10] developed a technique for counting the number of integer points in general parametric sets of the form

$$P_{\mathbf{p}} = \left\{ \mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} \geq B\mathbf{p} + \mathbf{c} \right\}, \tag{1}$$

where A and B are integer matrices, $\mathbf{c}$ is an integer vector, and $\mathbf{p}$ is a vector of parameters. $P_{\mathbf{p}}$ is called a *parametric polytope*, when the number of points satisfying $A\mathbf{x} \geq B\mathbf{p}+\mathbf{c}$ is finite for each value of $\mathbf{p}$. Clauss showed that the number of integer points in $P_{\mathbf{p}}$ can be represented by a set of *Ehrhart polynomials*, each valid on a different *validity domain*. Consider, for example, the loop in Figure 1, and assume we wish to know how many times statement S1 executes, as a function of N and M. The answer to this question can be computed by counting the number of integer points in

$$P_{\binom{N}{M}} = \left\{ \binom{i}{j} \in \mathbb{Z}^2 \mid \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ -2 & -1 \end{pmatrix} \binom{i}{j} \geq \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 0 \\ -1 & 0 \end{pmatrix} \binom{N}{M} + \begin{pmatrix} 0 \\ 0 \\ -3 \\ 0 \\ 0 \end{pmatrix} \right\}. \tag{2}$$

The solution to this question is

$$\#P_{\binom{N}{M}} = \begin{cases} -4N+8M-8 & \text{if } M \leq N \leq 2M-7 \\ MN-2N-M^2+6M-8 & \text{if } N \leq M \leq N+3 \wedge N \leq 2M-7 \\ \frac{N^2}{4}+N+1-\frac{1}{2}\mathrm{frac}\left(\frac{N}{2}\right) & \text{if } 0 \leq N \leq M-1 \wedge 2M \leq N+6 \\ \frac{N^2}{4}-MN-N+M^2+2M+1-\frac{1}{2}\mathrm{frac}\left(\frac{N}{2}\right) & \text{if } M \leq N \leq 2M \leq N+6 \end{cases}$$

with $\mathrm{frac}(x)$ the fractional part of x. This solution contains 4 validity domains, each with a different Ehrhart polynomial. Although Clauss's technique intends to be general, it may still fail to produce a solution for some class of problems[32, 9]. Furthermore, for some other class of problems, it can produce exponentially sized Ehrhart polynomials, computed in a likewise exponential time, even for fixed dimensions, i.e., when the number of variables in the inequalities is fixed.

In this paper, we present a novel method of computing Ehrhart polynomials, which combines the decomposition into validity domains of [10] and a parametric version of the counting algorithm in [14]. The resulting algorithm

- handles general parametric polytopes of the form (1),
- always produces an answer,
- computes Ehrhart polynomials that are only polynomially large in terms of the input size (for fixed dimensions), by representing periodicity using fractional parts, and
- computes each of these polynomials in polynomial time, when the number of variables in the inequalities is fixed.

In Section 2, two examples are presented which highlight the limitations of Clauss's method, and give the motivation for using the presented method. In Section 3, our method for computing the number of integer solutions to parametric linear inequalities is presented. Section 4 evaluates and compares our method with Clauss's method, using a large set of linear inequalities generated by different compiler analyses. In Section 5, we give some implementation details. Finally, a comparison with related work is made in Section 6, followed by conclusions in Section 7.

2. CLAUSS'S METHOD

2.1 Ehrhart's Theory

To illustrate the deficiencies of Clauss's method, we first need to explain the general structure of the number of integer points in a parametric polytope $P_{\mathbf{p}}$ (1). Ehrhart [15] showed that this parametric count can be represented as a *quasi-polynomial* provided $P_{\mathbf{p}}$ can be represented as a convex combination of its parametric *vertices*, i.e.,

$$P_{\mathbf{p}} = \left\{ \mathbf{x} \in \mathbb{Q}^d \mid \mathbf{x} = \boldsymbol{\lambda} V(\mathbf{p}), 0 \leq \lambda_j, \sum_j \lambda_j = 1 \right\}, \tag{3}$$

with the columns of $V(\mathbf{p})$ the vertices of $P_{\mathbf{p}}$. Each vertex $\mathbf{v}_j(\mathbf{p})$ is an affine combination of the parameters with *rational* coefficients. Before we can define a quasi-polynomial, we need the concept of a *periodic number*.

DEFINITION 1. *An* n**-periodic number** $U(\mathbf{p})$ *is a function* $\mathbb{Z}^n \mapsto \mathbb{Z}$*, such that there exist periods* $\mathbf{q} = (q_1, \ldots, q_n) \in \mathbb{N}^n$ *such that* $U(\mathbf{p}) = U(\mathbf{p}')$ *whenever* $p_i \equiv p'_i \mod q_i$*, for* $1 \leq i \leq n$*. The lcm (least common multiple) of all* q_i *is called the period of* $U(\mathbf{p})$*.*

DEFINITION 2. *A* **quasi-polynomial** *in* n *variables is a polynomial in* n *variables over the* n*-periodic numbers.*

In other words, the coefficients of a quasi-polynomial depend periodically on the variables. In the remainder of this paper, we call the quasi-polynomials that represent the parametric count of a polytope simply *Ehrhart polynomials*. The periodic numbers can be represented by an n-dimensional lookup-table $U_{\mathbf{p}}$ such that $U(\mathbf{p}) = U_{\mathbf{p}}[p_1 \bmod q_1, \ldots, p_n \bmod q_n]$. The period in dimension i of such a periodic number is a divisor of the lcm of the denominators that appear in the coefficients of p_i in the affine expressions that define the vertices $\mathbf{v}_j(\mathbf{p})$ [10].

Loechner and Wilde [26] showed that the parametric vertices in (3) correspond to the n-faces of (1) when considered as a polyhedron in the $(d+n)$-dimensional combined data and parameter space.[1] Each of these faces may only correspond to a parametric vertex for a subset of the parameter values. Therefore, the parameter space has to be partitioned into *validity domains* D_k, each with a subset $\mathcal{V}_{D_k}(P_\mathbf{p}) \subset \mathcal{V}(P_\mathbf{p})$ of the total number of parametric vertices. Each validity domain D_k then also has a different representation (3), with the elements of $\mathcal{V}_{D_k}(P_\mathbf{p})$ as the columns of $V_{D_k}(\mathbf{p})$. It follows from [15] that in each validity domain the number of integer points in $P_\mathbf{p}$ can be represented by an Ehrhart polynomial $\mathcal{E}(P;\mathbf{p})$.

For example, in domain $0 \le N \le M-1 \wedge 2M \le N+6$, the parametric polytope in Equation (2) has vertices $(0,0)$, $(0,N)$ and $(\frac{N}{2},0)$. Taking into account the denominators, the period is 2 in the dimension of parameter N and 1 in the dimension of parameter M. Therefore, the Ehrhart-polynomial for this validity domain has the following form

$$[a,b]_N N^2 + [c,d]_N NM + [e,f]_N M^2 + [g,h]_N M + [i,j]_N N + [k,l]_N, \quad (4)$$

where unknowns $a,\ldots,l$ are rational numbers. After finding the correct values for $a,\ldots,l$, the solution can be written as:

$$\frac{1}{4}N^2 + N + \left[1,\frac{3}{4}\right]_N, \quad (5)$$

which equals $\frac{N^2}{4} + N + 1 - \frac{1}{2}\operatorname{frac}(\frac{N}{2})$.

2.2 Interpolation and Degenerate Domains

Based on the knowledge of the structure of the solution, Clauss and Loechner [10] calculate the number of points in a set of instances of $P_\mathbf{p}$ for fixed values of $\mathbf{p}$ in a given validity domain, called *initial countings*, and then calculate the Ehrhart polynomial for this validity domain through interpolation. During this calculation, they directly determine the elements in the lookup-tables representing the periodic numbers. To interpolate a d-dimensional Ehrhart polynomial with periods q_i their algorithm requires $\prod_{i=1}^{n}(d+1)q_i$ initial countings. For its initial countings, the algorithm searches for fixed parameter values located in a hyperrectangle, which ensures that the solution is uniquely determined. However, it is not always possible to find a hyperrectangle of the correct size that is completely inside a given parameter domain.

For example, consider domain $N \le M \le N+3 \wedge N \le 2M-7$ from the polytope in Equation (2). This domain is geometrically represented with •s in Figure 2. For this domain, the period in both dimensions is 1, and the implementation of Clauss's method in PolyLib[24] searches for a solution of the following form

$$aN^2M^2+bN^2M+cN^2+dNM^2+eNM+fN+gM^2+hM+i. \quad (6)$$

To find the nine unknown values, Clauss's method looks for a 3×3 rectangle in the validity domain where it can compute initial countings for. As is clear from Figure 2, however, no such rectangle can be found and the method fails to compute the solution. The validity domains where this problem occurs are known as *degenerate domains*. In principle, the algorithm could be adapted to search for points in configurations other than hyperrectangles, but a *systematic* way for finding such configurations is not obvious without resorting to the addition of an extra parameter [27].

[1] Intuitively speaking, an *n-face* is an n-dimensional "border" of the polytope.

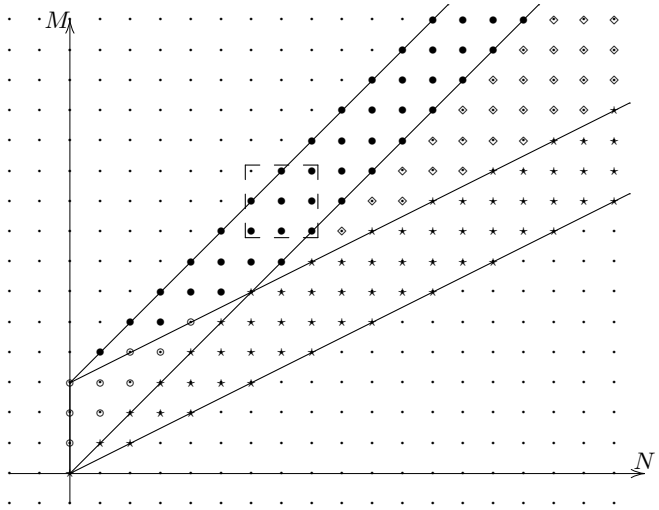


Figure 2: Geometrical representation of the validity domains of Equation (2). The points in the different validity domains are shown by different symbols. The domain marked by • is degenerate.

2.3 Large Solution Size

Since the periods q_i are bounded only by the *value* of the coefficients in the input, they can be *exponential* in terms of the input size and so the worst-case computation time for the Ehrhart polynomial in a single validity domain is exponential even for fixed dimension. Since Clauss's method is based on the lookup-table representation of periodic numbers, the output size is also exponential in the input size.

Consider, for example, the program in Figure 3 (matrix multiplication). Suppose we want to count the number of distinct TLB (Translation Lookaside Buffer) pages accessed between two consecutive accesses to the same TLB page. This count is an indication of the number of TLB page misses that can be expected and is called the reuse distance [3].

For simplicity, we'll assume that `A[i][k]` and `B[k][j]` access different TLB pages and we will concentrate on `A[i][k]`. We assume that `A` is 200×200 matrix, which is laid out in column major order, and starts at address zero. Furthermore, an element size of 4 bytes is assumed. As such, `A[i][k]` is located at address $4\times(200k+i)$.

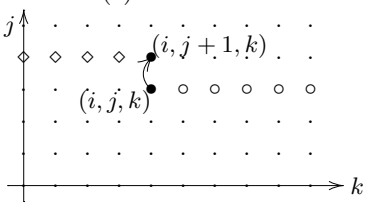
Iterations (i,j,k) and $(i,j+1,k)$ access the same array element `A[i][k]`. Figure 3b shows the iterations that are executed between these two iterations: iterations $(i,j,k+1\ldots199)$ ($\circ$ on the figure) and iterations $(i,j+1,0\ldots k-1)$ ($\diamond$ on the figure). The set of TLB pages accessed by the $\circ$-iterations can be described as

$$S_1 = \left\{ t \mid \exists k' : t = \left\lfloor \frac{800k'+4i}{L} \right\rfloor \wedge 0 \le i,j,k \le 199 \wedge k+1 \le k' \le 199 \right\},$$

where i, j and k are parameters. Assuming page size $L =$

```
do i = 0, 199
    do j = 0, 199
        s = 0
        do k = 0, 199
            s = s + A[i][k] * B[k][j]
        enddo
        C[i][k] = s
    enddo
enddo
```

(a) Source code



(b) Intermediate accesses

Figure 3: Matrix multiplication

4096, this can be written as a set of linear constraints:

$$S_1 = \{\, t \mid \exists k' : 1024t \le 200k' + i \le 1024t + 1023 \\ \wedge\, 0 \le i, j, k \le 199 \wedge k + 1 \le k' \le 199 \,\}$$

and further simplified to (e.g., using Omega [22])

$$S_1 = \{\, t \mid 0 \le i \wedge 1024t - 39800 \le i \le 199 \wedge 0 \le k \le 198 \\ \wedge\, 0 \le j \le 199 \wedge i + 200k \le 823 + 1024t \,\}$$

We obtain a similar expression S_2 for the $\diamond$-iterations. The total count of TLB pages is $\#(S_1 \cup S_2) = \#S_1 + \#(S_2 \setminus S_1)$. Concentrating on $S1$, we see that it is a one-dimensional polytope and using PolyLib we can find out that its vertices are

$$\frac{i}{1024} + 25\frac{k}{128} - \frac{823}{1024} \quad \text{and} \quad \frac{i}{1024} + \frac{4975}{128}.$$

Since the dimension of this polytope is $d = 1$ and the periods are $q_i = 1024$, $q_j = 1$ and $q_k = 128$, the interpolation method requires $2^3 \cdot 1024 \cdot 128$ initial countings. If we assume we need two bytes to represent a value, then we need up to 256KiB just to store a single periodic number in the output. Using our technique detailed in Section 3, which allows for a different representation of periodic numbers, we obtain the following equivalent, but much shorter solution in polynomial time (for fixed dimensions):

$$-\frac{25}{128}k - \mathrm{frac}\left(\frac{i+888}{1024}\right) + \mathrm{frac}\left(\frac{i+200k+199}{1024}\right) + \frac{40625}{1024},$$

with $\mathrm{frac}\,(x)$ the fractional part of x. Although the degeneracy problem can in principle be solved, the problem of a worst-case exponentially-sized output is intrinsic to this approach.

A number of the proposed compiler methods[3, 17, 25, 32] hard-code the resulting Ehrhart polynomial in the program they are optimizing. For these optimizations, large Ehrhart polynomials result in large binaries, making the optimizations less interesting in an embedded systems context. Using our method, the size of the resulting Ehrhart-polynomials remains small.

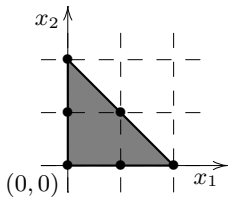


Figure 4: Barvinok example

In the next section we show how we have solved the degenerate domains and exponential complexity problems by taking an existing algorithm for counting the number of points in non-parametric polytopes based on Barvinok's decomposition and extending it to compute Ehrhart polynomials.

3. OUR METHOD

3.1 Barvinok's Algorithm

Without loss of generality (see Section 5), we assume that P is full-dimensional, i.e., that the dimension of P is d as well. Results cited from other sources have been adapted accordingly. When we refer to Barvinok's algorithm we mean the algorithm as outlined by Barvinok and further refined by De Loera et al.

The basic idea behind Barvinok's algorithm [1, 7, 14], is to consider the *generating function* of the integer points in a polytope P. This generating function is a formal power series with a term for each integer point in P, i.e.,

$$f(P; \mathbf{x}) = \sum_{\boldsymbol{\alpha} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\boldsymbol{\alpha}},$$

with $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$. Evaluating this function at $\mathbf{x} = \mathbf{1}$ yields the number of terms, which equals the desired number of points. The generating function is obviously not constructed by enumerating all the integer points in P, but rather as a signed sum of short rational functions that can be derived from the description of P.

Example 1. *Consider the polytope T shown in Figure 4: $T = \{\, \mathbf{x} \mid x_1 \ge 0 \wedge x_2 \ge 0 \wedge x_1 + x_2 \le 2 \,\}$. Its generating function is $f(T; \mathbf{x}) = 1 + x_1 + x_1^2 + x_2 + x_1x_2 + x_2^2$. Barvinok's algorithm, however, will produce this function in the following form:*

$$f(T; \mathbf{x}) = \frac{x_1^2}{(1 - x_1^{-1})(1 - x_1^{-1}x_2)} + \frac{x_2^2}{(1 - x_2^{-1})(1 - x_1x_2^{-1})} \\ + \frac{1}{(1 - x_1)(1 - x_2)}.$$

To construct the generating function as a signed sum of short rational functions, we consider the vertices $\mathbf{v}_i$ of P and the constraints that it saturates, i.e., the constraints $\langle \mathbf{a}, \mathbf{x} \rangle \ge b$ with $\langle \mathbf{a}, \mathbf{v}_i \rangle = b$, where $\langle \cdot, \cdot \rangle$ is the standard scalar product. The region in $\mathbb{Q}^d$ bounded by these constraints for a particular $\mathbf{v}_i$ is called the *supporting cone* $\mathrm{cone}(P, \mathbf{v}_i)$. E.g., the supporting cone of vertex $\mathbf{v}_1$ of the polytope (shaded area) in Figure 5 is shown in thick lines. It can be shown [1] that the generating function of P is equal to

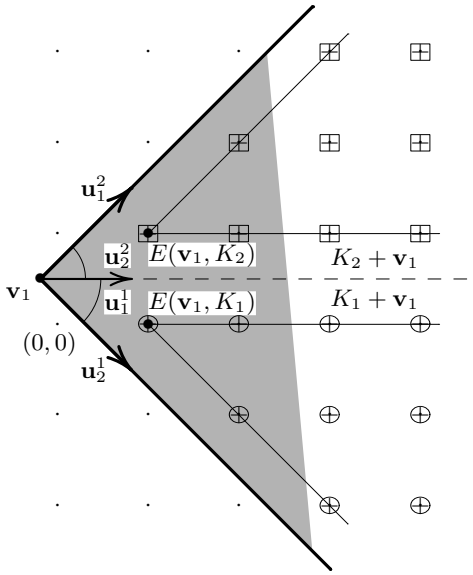


Figure 5: Decomposition of $\mathrm{cone}(P, \mathbf{v}_1)$**.**

the sum of the generating functions of its supporting cones. To construct the generating function of a supporting cone, we use Barvinok's decomposition into *unimodular cones*.

DEFINITION 3. *A* **cone** *with* **generators** $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k \in \mathbb{Z}^d$ *is a set of the form* $\{ \sum_i \lambda_i \mathbf{u}_i \mid \lambda_i \geq 0 \}$*. It is called* ***unimodular*** *if its generators form a basis of* $\mathbb{Z}^d$*.*

Note that using this definition, $\mathrm{cone}(P, \mathbf{v}_i)$ is not a cone itself, but the sum of $\mathbf{v}_i$ and some cone K. Barvinok proposed to decompose this cone into a signed "sum" of unimodular cones $\{ (\epsilon_i, K_i) \} = \mathcal{B}(K)$, with $\epsilon_i \in \{ -1, 1 \}$, the sign corresponding to unimodular cone K_i. Here, "sum" can be interpreted to mean that the generating function of K is the signed sum of the generating functions of the unimodular cones. It can be shown [1] that a simple explicit formula exists for the generating function of a unimodular cone:

$$f(K_i; \mathbf{x}) = \prod_{j=1}^{k} \frac{1}{(1 - \mathbf{x}^{\mathbf{u}_j^i})},$$

with $\mathbf{u}_j^i$ the generators of K_i. A key feature of this decomposition is that its computation time is polynomial in the input size (for fixed dimensions), where, as usual [30], the input size is the number of bits needed to represent the input polytope.

To obtain the final generating function, the generating functions corresponding to the unimodular cones K_i need to be translated to the vertex $\mathbf{v}$. If $\mathbf{v}$ is integer, then we simply need to add $\mathbf{v}$ to all the exponents in the generating function, which corresponds to a multiplication by $\mathbf{x}^{\mathbf{v}}$. If $\mathbf{v}$ is not integer, however, then we need to find another point $\mathbf{v}' = E(\mathbf{v}, K_i)$ such that $\mathbf{x}^{\mathbf{v}'} f(K_i; \mathbf{x})$ generates $K_i + \mathbf{v}$. Since K_i is unimodular, this point exists and is uniquely defined as the smallest integer linear combination of the generators of K_i that lies inside $K_i + \mathbf{v}$ [14]. I.e.,

$$E(\mathbf{v}, K_i) = \sum_j \lceil \lambda_j \rceil \mathbf{u}_j^i, \tag{7}$$

where $\boldsymbol{\lambda}$ is the rational solution to $\mathbf{v} = \sum_j \lambda_j \mathbf{u}_j^i$ and $\lceil . \rceil$ is the upper integer part. Note that if $\mathbf{v}$ is integer, then $E(\mathbf{v}, K_i) = \mathbf{v}$. The final generating function is then

$$f(P; \mathbf{x}) = \sum_{\mathbf{v} \in \mathcal{V}(P)} \sum_{i=1}^{|\mathcal{B}(K_\mathbf{v})|} \epsilon_i \frac{\mathbf{x}^{E(\mathbf{v}, K_i)}}{\prod_{j=1}^{d} (1 - \mathbf{x}^{\mathbf{u}_j^i})}. \tag{8}$$

For a proof of the correctness of this formula, see [1].

EXAMPLE 2. *Figure 5 shows a polytope P (shaded area) and its supporting cone (thick lines) at vertex $\mathbf{v}_1$. A possible signed unimodular decomposition for $\mathrm{cone}(P, \mathbf{v}_1) - \mathbf{v}_1$ is the pair $\{ (1, K_1), (1, K_2) \}$.*[2] *Let $\mathbf{v}_1' = E(\mathbf{v}_1, K_1)$ and $\mathbf{v}_1'' = E(\mathbf{v}_1, K_2)$. Since both signs are positive, we have*

$$f(\mathrm{cone}(P, \mathbf{v}_1); \mathbf{x}) = f(\mathbf{v}_1' + K_1; \mathbf{x}) + f(\mathbf{v}_1'' + K_2; \mathbf{x}).$$

The integer points in $\mathbf{v}_1'' + K_2$ are marked by $\boxplus$, whereas those in $\mathbf{v}_1' + K_1$ are marked by $\oplus$.

Note that each term in (8) has a pole at $\mathbf{x} = \mathbf{1}$. We can still evaluate this function at $\mathbf{x} = \mathbf{1}$ by computing the residue. De Loera [14] shows that, through a suitable variable substitution, each (multivariate) term in (8) can be written as the univariate

$$\epsilon_i' \frac{N(s)}{D'(s)} = \epsilon_i' \frac{(s+1)^{\langle \boldsymbol{\mu}, E(\mathbf{v}, K_i) \rangle - c}}{s^d D(s)}, \tag{9}$$

with $x_i = (s+1)^{\mu_i}$, $D(s)$ a polynomial with integer coefficients, independent of $\mathbf{v}$, $\boldsymbol{\mu}$ an integer vector not orthogonal to any generator and c the sum of the negative $\langle \boldsymbol{\mu}, \mathbf{u}_j^i \rangle$s. Evaluating (8) at $\mathbf{x} = \mathbf{1}$ is equivalent to summing the terms (9) evaluated at $s = 0$. This in turn can be accomplished by computing the coefficient of s^d in the Taylor expansion of $N(s)/D(s)$. Note that this only requires the first $(d+1)$ coefficients of $N(s)$. The (signed) sum of the coefficients of s^d in each of the terms then yields the desired number of points in the polytope.

EXAMPLE 3. *Consider once more the polytope T from Example 1. Since each of its supporting cones is unimodular, the generating function $f(T; \mathbf{x})$ has exactly one term for each vertex. To evaluate $f(T; \mathbf{x})$ at $\mathbf{1}$ one can take $\boldsymbol{\mu} = (1, -1)$ since it is not orthogonal to any generator. Substituting $x_1 = (s+1)^{\mu_1}$ and $x_2 = (s+1)^{\mu_2}$ in $f(T; \mathbf{x})$ we obtain:*

$$f(T; s) = \frac{(s+1)^2}{(1 - (s+1)^{-1})(1 - (s+1)^{-2})} + \frac{(s+1)^{-2}}{(1 - (s+1))(1 - (s+1)^2)} + \frac{1}{(1 - (s+1))(1 - (s+1)^{-1})}.$$

In order to obtain only positive powers in the denominators, we multiply the numerator and the denominator of each term by $(s+1)^{-c}$, with c the sum of the negative powers in the denominator. This returns $f(T; s)$ in the form:

$$f(T; s) = \frac{(s+1)^5}{(1 - (s+1))(1 - (s+1)^2)} + \frac{(s+1)^{-2}}{(1 - (s+1))(1 - (s+1)^2)} - \frac{(s+1)}{(1 - (s+1))(1 - (s+1))}. \tag{10}$$

[2] Note that this is not the decomposition that our implementation would produce, since, like De Loera [14], we perform the decomposition on the dual cone.

Algorithm 1 Parametric Barvinok

1. For each vertex $\mathbf{v}_i(\mathbf{p}) \in \mathcal{V}(P)$
   (a) Determine supporting cone $\text{cone}(P, \mathbf{v}_i(\mathbf{p}))$
   (b) Let $K = \text{cone}(P, \mathbf{v}_i(\mathbf{p})) - \mathbf{v}_i(\mathbf{p})$
   (c) Let $\{(\epsilon_j, K_j)\} = \mathcal{B}(K)$
   (d) For each K_j
      i. Determine $f(K_j; \mathbf{x})$
   (e) $f(\text{cone}(P, \mathbf{v}_i(\mathbf{p})); \mathbf{x}) = \sum_j \epsilon_j \mathbf{x}^{E(\mathbf{v}_i(\mathbf{p}), K_j)} f(K_j; \mathbf{x})$
2. For each validity domain D_k of P
   (a) $f_{D_k}(P; \mathbf{x}) = \sum_{\mathbf{v}_i \in \mathcal{V}_{D_k}(P)} f(\text{cone}(P, \mathbf{v}_i(\mathbf{p})); \mathbf{x})$
   (b) evaluate $f_{D_k}(P; \mathbf{1})$

Rewriting each term in the form $\frac{1}{s^2}\frac{N(s)}{D(s)}$ (through Taylor expansion and polynomial division) we obtain:

$$f(T;s) = \frac{1}{s^2}(\frac{1}{2}+\frac{9}{4}s+\frac{31}{8}s^2+\cdots)+\frac{1}{s^2}(\frac{1}{2}-\frac{5}{4}s+\frac{17}{8}s^2+\cdots) - \frac{1}{s^2}(1+s+0s^2+\cdots).$$

Finally, the number of integer points in the polytope T is given by the sum of the coefficients of s^2 in the polynomials $\frac{N(s)}{D(s)}$, i.e, $\frac{31}{8}+\frac{17}{8}-0=6$.

3.2 Computing Ehrhart Polynomials

Algorithm 1 shows our extension of Barvinok's method to parametric polytopes. The main idea behind this generalization is to consider Loechner and Wilde's decomposition of the parameter space and to apply Barvinok's algorithm to the fixed set of (parametric) vertices that belong to each validity domain. Thus, one parametric generating function is to be computed for each of these validity domains. We refer to [26] for details on how to compute these validity domains.

The generating function for the parametric polytope $P_{\mathbf{p}}$ on validity domain D is the parametric version of Equation (8):

$$f_D(P_{\mathbf{p}}; \mathbf{x}) = \sum_{\mathbf{v}(\mathbf{p}) \in \mathcal{V}_D(P_{\mathbf{p}})} \sum_{i=1}^{|\mathcal{B}(K_{\mathbf{v}})|} \epsilon_i \frac{\mathbf{x}^{E(\mathbf{v}(\mathbf{p}), K_i)}}{\prod_{j=1}^{d}(1-\mathbf{x}^{\mathbf{u}_j^i})}, \quad (11)$$

with $\epsilon_i \in \{-1, 1\}$ and $\mathbf{v}(\mathbf{p})$ a parametric vertex of the polytope $P_{\mathbf{p}}$. Each coordinate of $\mathbf{v}(\mathbf{p})$ is an affine function of the parameters. K_i is the ith unimodular cone in the signed unimodular decomposition of cone $K_{\mathbf{v}}$, the translation to the origin of the supporting cone at $\mathbf{v}(\mathbf{p})$. The supporting cone is again defined by the constraints that $\mathbf{v}(\mathbf{p})$ saturates, i.e., the constraints $\langle \mathbf{a}, \mathbf{x} \rangle \geq \langle \mathbf{b}, \mathbf{p} \rangle + c$ such that $\langle \mathbf{a}, \mathbf{v}(\mathbf{p}) \rangle = \langle \mathbf{b}, \mathbf{p} \rangle + c$. The correctness of (11) follows from the fact that the generators of K are independent of the parameters, which means that Barvinok's decomposition can be applied without change.

The exponent in the numerators of (11), which corresponds to the uniquely defined point inside the translated unimodular cone, is given by the parametric version of (7):

$$E(\mathbf{v}(\mathbf{p}), K_i) = \sum_{j=1}^{d} \lceil \lambda_j(\mathbf{p}) \rceil \mathbf{u}_j^i, \quad (12)$$

where the $\lambda_j(\mathbf{p})$s are rational affine functions of the parameters that solve $\mathbf{v}(\mathbf{p}) = \sum_{j=1}^{d} \lambda_j(\mathbf{p}) \mathbf{u}_j^i$. This form cannot be used directly to construct Ehrhart (quasi-)polynomials, since the upper integer part is not periodic. It can however be rewritten as

$$\lceil \lambda_j(\mathbf{p}) \rceil = -\lfloor -\lambda_j(\mathbf{p}) \rfloor = \lambda_j(\mathbf{p}) + \text{frac}(-\lambda_j(\mathbf{p})). \quad (13)$$

The second term on the right is now clearly a periodic number, say $U_j'(\mathbf{p})$, with period at most the common denominator of the coefficients that appear in $\lambda_j(\mathbf{p})$. Substituting the value of $\lceil \lambda_j(\mathbf{p}) \rceil$ in (12) we get

$$E(\mathbf{v}(\mathbf{p}), K_i) = \sum_{j=1}^{d} \lambda_j(\mathbf{p}) \mathbf{u}_j^i + \sum_{j=1}^{d} U_j'(\mathbf{p}) \mathbf{u}_j^i = \mathbf{v}(\mathbf{p}) + \mathbf{U}(\mathbf{p}), \quad (14)$$

with $\mathbf{U}(\mathbf{p})$ a vector of periodic numbers.

As in the non-parametric case, we can obtain the value of the parametric generating function (11) at $\mathbf{x} = \mathbf{1}$ by computing the residue. Again, the variable substitution proposed by De Loera [14] is independent of the numerator and hence of the parameters. Substituting (14) in (9), we obtain

$$N_{\mathbf{p}}(s) = (s+1)^{\langle \boldsymbol{\mu}, \mathbf{v}(\mathbf{p}) + \mathbf{U}(\mathbf{p}) \rangle - c} = (s+1)^{\Lambda(\mathbf{p})}, \quad (15)$$

with $\Lambda(\mathbf{p})$ an affine function of the parameters with a constant part that may be a periodic number. The coefficients of $N_{\mathbf{p}}(s)$ up to that of s^d (i.e., those required to compute the coefficient of s^d in $N_{\mathbf{p}}(s)/D(s)$) are

$$n_i(\mathbf{p}) = \binom{\Lambda(\mathbf{p})}{i} = \frac{\prod_{j=0}^{i-1}(\Lambda(\mathbf{p}) - j)}{i!} \quad \text{for } 0 \leq i \leq d. \quad (16)$$

Each coefficient $n_i(\mathbf{p})$ in the above formula is given by a product of at most d affine functions of the parameters with constant parts that may be periodic numbers. This implies that each of these coefficients is a multivariate polynomial of the parameters in which the coefficients may be periodic numbers and for which the sum of powers in each multivariate monomial is at most d. Since the coefficient of s^d in $N_{\mathbf{p}}(s)/D(s)$ is a linear combination of these $n_i(\mathbf{p})$, it conforms to the same property and so does the signed sum of all these terms. That is, the residue of (11), which is equal to the parametric number of points in $P_{\mathbf{p}}$, is an Ehrhart polynomial, as expected. I.e.,

$$\mathcal{E}_D(P; \mathbf{p}) = f_D(P_{\mathbf{p}}; \mathbf{1}) = \sum_{0 \leq i_1 + i_2 + \cdots + i_n \leq d} U_{\mathbf{i}}(\mathbf{p}) \mathbf{p}^{\mathbf{i}}, \quad (17)$$

with the $U_{\mathbf{i}}(\mathbf{p})$s periodic numbers and d the dimension of $P_{\mathbf{p}}$.

As explained in Section 2.1, such periodic number *can* be represented by a lookup-table. However, to avoid the exponential behavior of such lookup-tables, we keep the fractional parts from (13) instead. Since we require both additions (in (14), (15) and (16)) and multiplications (16) of periodic numbers, we represent all periodic numbers as linear combinations of products of such expressions:

$$U(\mathbf{p}) = \sum_i \alpha_i \prod_j \text{frac}\left(\beta_{ij0} + \sum_k \beta_{ijk} p_k\right),$$

with $\alpha_i \in \mathbb{Q}$ and $\beta_{ijk} \in \mathbb{Z}$. This representation is clearly closed under addition and multiplication.

EXAMPLE 4. *Consider the polytope*

$$P_{(p)} = \{\, \mathbf{x} \mid x_1 \geq 0 \wedge x_2 \geq 0 \wedge 2x_1 + 2x_2 \leq p \,\}.$$

This is a parametric version of the polytope in Examples 1 and 3 where p was set to 4. This polytope has a single validity domain $p \geq 0$ with parametric vertices $(0,0)$, $(\frac{p}{2},0)$ and $(0,\frac{p}{2})$. The generators of the corresponding supporting cones are $(1,0),(0,1)$, $(-1,0),(-1,1)$ and $(0,-1),(1,-1)$ respectively. Since all these supporting cones are unimodular, there is a single term in (11) for each vertex. Since $(\frac{p}{2},0) = -\frac{p}{2}(-1,0) + 0(-1,1)$, we conclude from (12) that the exponent in the numerator of the term for this vertex is

$$\left\lceil -\frac{p}{2} \right\rceil (-1,0) = \left(\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right), 0\right).$$

As in Example 3, we take $\boldsymbol{\mu} = (1,-1)$, a vector not orthogonal to any of the generators. The exponent of $(s+1)$ in (15) is

$$\left\langle (1,-1), \left(\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right), 0\right) \right\rangle + 3 = \frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right) + 3,$$

where $c = -3$ as in (10). The other vertices are handled similarly and we obtain

$$f(P;s) = \frac{(s+1)^{\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right) + 3}}{s^2(s+2)} + \frac{(s+1)^{-\frac{p}{2} + \operatorname{frac}\left(\frac{p}{2}\right)}}{s^2(s+2)} - \frac{s+1}{s^2},$$

where we simplified the denominators that we already calculated in (10). To evaluate $f(P;0)$, we calculate the coefficient of s^2 in

$$\frac{N(s)}{D(s)} = \frac{(s+1)^{\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right) + 3} + (s+1)^{-\frac{p}{2} + \operatorname{frac}\left(\frac{p}{2}\right)}}{s+2} - (s+1). \tag{18}$$

The second term does not contribute to s^2. The terms in the numerator of the first term can be expanded according to (16) as

$$\begin{aligned}(s+1)^{\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right) + 3} = 1 &+ \left(\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right) + 3\right) s + \\ &\left(\frac{p^2}{8} + \left(\frac{5}{2} - \operatorname{frac}\left(\frac{p}{2}\right)\right)\frac{p}{2} + \right. \\ &\left. \frac{1}{2}\left(\operatorname{frac}\left(\frac{p}{2}\right)\right)^2 - \frac{5}{2}\operatorname{frac}\left(\frac{p}{2}\right) + 3\right) s^2 \\ &+ r(s)s^3\end{aligned}$$

and

$$\begin{aligned}(s+1)^{\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right)} = 1 &+ \left(\frac{p}{2} - \operatorname{frac}\left(\frac{p}{2}\right)\right) s + \\ &\left(\frac{p^2}{8} + \left(\frac{1}{2} - \operatorname{frac}\left(\frac{p}{2}\right)\right)\frac{p}{2} + \right. \\ &\left. \frac{1}{2}\left(\operatorname{frac}\left(\frac{p}{2}\right)\right)^2 - \frac{1}{2}\operatorname{frac}\left(\frac{p}{2}\right)\right) s^2 \\ &+ r'(s)s^3.\end{aligned}$$

while the denominator yields

$$\frac{1}{s+2} = \frac{1}{2}\left(1 - \frac{1}{2}s + \frac{1}{4}s^2 + r''(s)s^3\right).$$

The coefficient of s^2 in (18), which equals $\mathcal{E}(P;p) = f(P;0)$, is therefore

$$\frac{p^2}{8} + \left(\frac{3}{4} - \frac{1}{2}\operatorname{frac}\left(\frac{p}{2}\right)\right)p + \frac{1}{2}\left(\operatorname{frac}\left(\frac{p}{2}\right)\right)^2 - \frac{3}{2}\operatorname{frac}\left(\frac{p}{2}\right) + 1.$$

Note that the Ehrhart polynomial in (17) can be computed for *any* validity domain, no matter its size or shape. Furthermore it is calculated *analytically*, which enables the use of fractional parts for periodic numbers. For fixed dimensions, the elementary operations on this representation can be performed in polynomial time. Furthermore, the number of such operations performed for each vertex is polynomial for fixed dimensions and so is Barvinok's decomposition [2]. Since the number of parametric vertices is also polynomial (for fixed dimensions) [26], we can compute a polynomially-sized Ehrhart polynomial in polynomial time for any given validity domain. To obtain a bound on the number of validity domains, consider the hyperplanes in the parameter space formed by the affine hulls of the $(m-1)$-dimensionsal intersections of pairs of validity domains.[3] These k hyperplanes partition the parameter space into a number of cells that is bounded by a polynomial in k (for fixed dimensions) [8]. Since (part of) these cells form a subdivision of the validity domains and since the k hyperplanes correspond (roughly) to the $(m-1)$-faces of P, which are themselves bounded in number by a polynomial in the input size (for fixed dimensions), the number of validity domains and hence also the total size is polynomial in the input size (for fixed dimensions). Note that in practice, the number of validity domains is typically very small.

4. EXPERIMENTS

During the past years, a number of researchers have observed degeneracy problems in Clauss's method while they were trying to count the number of points in the polytopes generated by their compiler analyses. Thanks to their collaboration, we have collected polytopes resulting from cache miss equations[19], exact analysis of cache behavior[9], converting software to hardware implementations[32], computing reuse distance equations[3] and a number of counting problems originating from as yet unpublished compiler analyses. For all these polytopes on which Clauss's method fails, our method successfully computes the solution. Our implementation and a set of such polytopes are available from `http://freshmeat.net/projects/barvinok/`.

Furthermore, we made a more extensive evaluation by comparing the results, the computation time and the size of the solution for parametric polytopes resulting from the following two analyses:

- Computation of reuse distances[3]. The reuse distance of a memory access to an array element a is the number of different array elements that are fetched since the previous access to a. First, the array elements accessed between use and reuse are represented as a union of parametric polytopes. Then, the reuse distance is computed by symbolically counting the number of integer points in the polytopes. In our experiments, none of these polytopes have periodic behavior and therefore

[3]The affine hull of a set S is the set $\{\, \lambda_1 x_1 + \cdots \lambda_k x_k \mid \{\, x_1, \ldots x_k \,\} \subset S, \sum_i \lambda_i = 1 \,\}$.

program	nr. of poly-topes	Clauss's method		our method
		#degen. domains	exec. time	exec. time
vpenta	6496	0	269.50s	165.84s
mxm	66	0	7.92s	1.98s
liv18	5296	6	248.68s*	135.43s
cholesky	76	0	6.12s	1.94s
jacobi	246	6	11.58s*	6.32s
gauss-jordan	308	0	19.01s	8.08s
tomcatv	8786	66	731.31s*	247.46s
total	21274	78	1294.12s*	567.05s

Table 1: Number of polytopes constructed by reuse distance calculation, number of degenerate domains using Clauss's method, and execution time of Clauss's and our method. The numbers marked by an * are partial since they only apply to the non-degenerate domains.

their period is always 1. The solution size is then polynomial for both fractional parts and table representation of periodic numbers. However, Table 1 shows that for 4 out of the 7 programs for which reuse distances are calculated, Clauss's method fails to compute the reuse distance, due to degenerate domains. Thanks to our method, the reuse distance can be calculated for all programs, which makes calculation of reuse distances practical and potentially useful to implement in industrial compilers. Furthermore, Table 1 shows that the computation times are reasonable and compared to Clauss's method, our method is about twice as fast for this set of polytopes. For 99% of these polytopes, there is only a single validity domain. The remaining 1% have two to four validity domains. The dimension d of these polytopes is 1 or 2, whereas the number of parameters ranges between 2 and 7.

- Computation of the number of cache lines and the number of memory pages accessed by a reference in a given execution of a loop nest[16]. In contrast to the polytopes resulting from the reuse distance calculation, some of these polytopes do have large periods. In Figure 6, the calculation time is plotted in function of the periodic behavior. In Figure 7, the size of the solution is shown as a function of the period of the solution. Remember that the period itself is exponential in the input size, even for fixed dimensions. Figure 6 shows that the computation time increases sharply with the period for Clauss's method. On the other hand, the computation time of our method is always less than 1 second, irrespective of the period. In comparison, the computation time using Clauss's method increases to more than 3 hours for one polytope. Figure 7 shows that the size of the solution using Clauss's method can increase to more than 33MB, whereas our method always produces solution sizes smaller than 9KB. For 99% of these polytopes, there is only a single validity domain. The remaining 1% have two validity domains. The dimension d of these polytopes is 1 or 2, whereas the number of parameters ranges between 0 and 3.

The experiments were performed on an otherwise idle

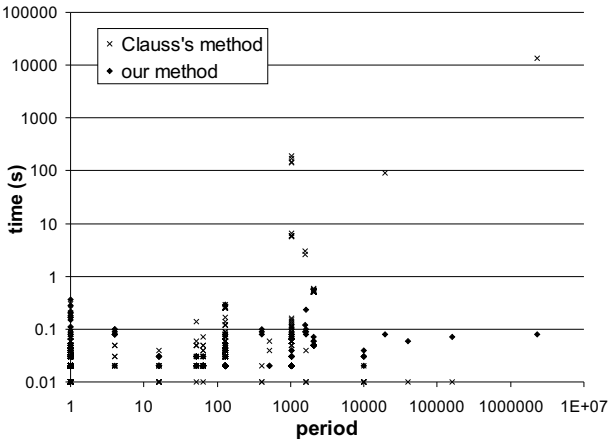


Figure 6: Execution time as a function of maximum period of computed polynomials.

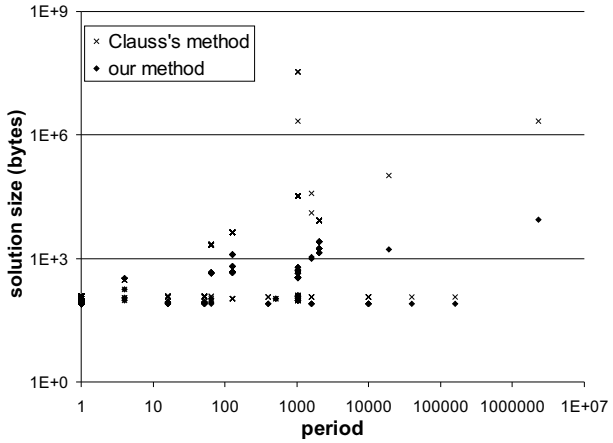


Figure 7: Solution size as a function of maximum period of computed polynomials.

2.66GHz Pentium4 machine running Linux. As explained in the next section, our implementation is built on top of the same library for performing polyhedral operations that is used by the implementation of Clauss's method. Furthermore, our implementation uses the same data structures to store polynomials. The size measure corresponds to the total memory size allocated to store the result. Note that the granularity is 4 bytes, since the experiments were performed on a 32 bits machine.

5. IMPLEMENTATION DETAILS

In Section 3, we assumed that P is full-dimensional. If P is of dimension $d-l$, with $l \geq 1$, then its description contains an equality $\langle \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{b}, \mathbf{p} \rangle + c$. Let $\mathbf{a}' = \mathbf{a}/g$, with g the gcd (greatest common divisor) of the elements in $\mathbf{a}$. $\mathbf{a}'^T$ can be extended to a unimodular matrix U [4].[4] Let $P' = UP$. Since U and U^{-1} are unimodular, there is one-to-one correspondence between the integer points in P and those in P' and so the number of points in both polytopes is the same, i.e., $\mathcal{E}(P') = \mathcal{E}(P)$. Furthermore, the first coordinate x'_1 of P' is independent of the other coordinates since $gx'_1 = g\langle \mathbf{a}', \mathbf{x} \rangle = \langle \mathbf{b}, \mathbf{p} \rangle + c$ by construction of U and so P' is the cross product of $P'' = \{ (\langle \mathbf{b}, \mathbf{p} \rangle + c)/g \}$ and some $P_1 \in \mathbb{R}^{d-1}$. Therefore we can factor P and compute the number of points in P as the product of those in P'' and P_1, i.e., $\mathcal{E}(P) = \mathcal{E}(P') = \mathcal{E}(P'') \cdot \mathcal{E}(P_1)$. The number of integer points in P'' is zero or one depending on the parameters and can be represented by a periodic number. Repeating the above l times, yields a $P_l \in \mathbb{R}^{d-l}$ of full dimension.

Even if P is full-dimensional, then we may in some cases still be able to write it as a cross product of two or more sets [20]. Since the dimension of each set is smaller than that of P, we can greatly reduce the computation time by calculating the number of points in each factor separately and multiplying the results afterward. Note that we also need to "multiply" the validity domains, i.e., the validity domains of the product are the intersections of the corresponding validity domains in the factors. Furthermore, if P or one of its factors is one-dimensional then it has two vertices $l(\mathbf{p}) \leq u(\mathbf{p})$ and we simply calculate $\lfloor u(\mathbf{p}) \rfloor - \lceil l(\mathbf{p}) \rceil + 1$ (again a periodic number) in each validity domain rather than using the algorithm of Section 3.

Our procedure for calculating Barvinok's decomposition into unimodular cones is an independent reimplementation of the corresponding procedure in `LattE` [13] as described in [14]. Like `LattE`, we use Shoup's implementation [31] of Lenstra, Lenstra and Lovasz' basis reduction algorithm and `GMP` [18] for computing in exact long integer arithmetic. Unlike `LattE`, however, we use `PolyLib` [24] for performing polyhedral operations, since this allows us to reuse the procedures for subdividing the parameter space into validity domains [26]. The disadvantage of using `PolyLib` is that it incurs a speed penalty by insisting on maintaining the dual representation for all polytopes and by its non-optimal use of the `GMP` library.

6. RELATED WORK

Two methods are often cited for counting the number of points in a parametric polytope: Clauss and Loechner (1998) [10] and Pugh (1994) [29]. As already explained in the previous sections, our technique and the first share the decomposition into validity domains, but whereas the first can produce exponentially sized Ehrhart polynomials or sometimes no polynomial at all, our technique always produces polynomially-sized Ehrhart polynomials in polynomial time (for fixed dimensions).

[4]A unimodular matrix is an integer matrix with determinant 1 or -1.

```
for(i=1;1000*i<=N;i++)
 for(j=1;1000*j<=N;j++)
  S1;
```

(a) How many times does `S1` execute?

$$\begin{cases} \frac{N^2}{1000000} & \text{if 1000 divides } N \\ \frac{(N-1)N}{1000000} - \frac{N-1}{1000000} & \text{if 1000 divides } N-1 \\ \dots & \\ \frac{(N-999)N}{1000000} - \frac{999N-998001}{1000000} & \text{if 1000 divides } N-999 \end{cases}$$

(b) solution computed by Pugh's method[29]

$$\left(\frac{N}{1000}\right)^2 - \frac{N}{500}\,\mathrm{frac}\left(\frac{N}{1000}\right) + \left(\mathrm{frac}\left(\frac{N}{1000}\right)\right)^2$$

(c) solution generated by our method

Figure 8: Example of an answer generated by Pugh's method. The number of different cases in Pugh's answer is as large as the factor of `i` and `j` in the program in (a) (1000 in this example). Therefore, the solution size of Pugh's method is exponentially large.

The technique of Pugh consists of a set of simplification and rewrite rules and the application of a set of standard summation formulas for some base cases. In contrast to our technique and that of Claus and Loechner, his technique does not appear to have ever been implemented. Furthermore, the description of the method in [29] fails to indicate which rewrite rules to use when several are applicable. We are therefore unable to systematically compare our results to those that would or would not be obtained using that method. Application by hand on the example in Figure 8 shows that even if it could be implemented, it would result in exponentially large solutions in the worst case, even for fixed dimensions.

Although `LattE` initially only counted the number of points in non-parametric polytopes, it has been extended to handle some form of parametric polytopes by what the authors call the homogenized Barvinok algorithm [12]. They only support a single parameter p, however. In particular, they only consider dilations pP of a non-parametric polytope P by a factor p. This means that all the vertices (see Eq. (3)) are of the form $p\mathbf{v}$ and that they do not need to consider validity domains, since the limited set of problems they handle always have a single fixed validity domain. Rather than Ehrhart polynomials, they compute Ehrhart *series* which are formal power series closely related to Ehrhart polynomials and seem to be more appropriate for their application domain. The main difference is that in an Ehrhart series the number of points in the dilatation pP is equal to the *coefficient* of the term t^p.

Recently some advances have been made towards automata-based counting [5, 28]. Although these techniques handle a larger class of problems (solutions to Presburger formulas), they do not support symbolic parameters. Preliminary experiments have shown that in the intersection of the application domains, i.e., for non-parametric polytopes, our method is as fast or faster (up to a factor 100 in some exceptional cases) than Parker's, except for polytopes with a large number (say thousands) of vertices. For such polytopes, the homogenized Barvinok algorithm as implemented by `LattE` would be more appropriate.[5]

7. CONCLUSION AND FUTURE WORK

Many compiler analyses and optimizations require the computation of the number of integer solutions to parametric systems of linear inequalities. This count can be represented by a set of Ehrhart polynomials, each valid in part of the parameter space. We have presented a new method, based on Barvinok's decomposition, for counting these polynomials analytically, resulting in the first implementation that handles *all* parametric polytopes of the form (1). By further using an alternative representation of periodic numbers using fractional parts, we compute *polynomially* sized Ehrhart polynomials—crucial for compiler optimizations that encode the Ehrhart polynomial in the resulting optimized program—in *polynomial* time, making a whole class of existing optimization techniques practically usable.

Our Ehrhart polynomials, though polynomially sized (for fixed dimensions), can in some cases still be relatively large. Further research into simplification of expressions containing fractional parts beyond what is applied already may reduce their size even further. We also plan to extend our technique to counting the number of points in integer projections of parametric polytopes, which would allow the parametric enumeration of solutions to Presburger formulas, after conversion to disjoint disjunctive normal form.

8. ACKNOWLEDGEMENTS

We thank Jesus A. De Loera, Jörg Rambau and the anonymous reviewers for their advice and Erin Parker for providing us with an implementation of her technique.

Kristof Beyls was supported by research projects GOA-12051002 and IWT-SB991147. Sven Verdoolaege was supported by FWO-Vlaanderen.

9. ADDITIONAL AUTHORS

Additional authors: Vincent Loechner (ICPS, LSIIT (UMR CNRS 7005), Université Louis Pasteur, Strasbourg, email: `loechner@icps.u-strasbg.fr`) and Maurice Bruynooghe (Dept. of Computer Science, K.U.Leuven, email: `maurice@cs.kuleuven.ac.be`).

10. REFERENCES

[1] A. Barvinok and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. *New Perspectives in Algebraic Combinatorics*, (38):91–147, 1999.

[2] A. I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. In *34th Annual Symposium on Foundations of Computer Science*, pages 566–572. IEEE, Nov. 1993.

[3] K. Beyls. *Software Methods to Improve Data Locality and Cache Behavior*. PhD thesis, Ghent University, 2004.

[4] A. J. C. Bik. *Compiler Support for Sparse Matrix Computations*. PhD thesis, University of Leiden, The Netherlands, 1996.

[5] B. Boigelot and L. Latour. Counting the solutions of Presburger equations without enumerating them. *Theoretical Computer Science*, (313):17–29, 2004.

[6] V. Braberman, D. Garbervetsky, and S. Yovine. On synthesizing parametric specifications of dynamic memory utilization. Technical report, Oct. 2003.

[7] M. Brion and M. Vergne. Residue formulae, vector partition functions and lattice points in rational polytopes. *J. Amer. Math. Soc.*, 10:797–833, 1997.

[8] R. Buck. Partition of space. *American Mathematical Monthly*, 50(9):541–544, 1943.

[9] S. Chatterjee, E. Parker, P. J. Hanlon, and A. R. Lebeck. Exact analysis of the cache behavior of nested loops. In *Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation*, pages 286–297. ACM Press, 2001.

[10] P. Clauss and V. Loechner. Parametric Analysis of Polyhedral Iteration Spaces. *Journal of VLSI Signal Processing*, 19(2):179–194, July 1998.

[11] P. D'Alberto, A. Veidembaum, A. Nicolau, and R. Gupta. Static analysis of parameterized loop nests for energy efficient use of data caches. In *Workshop on Compilers and Operating Systems for Low Power (COLP01)*, Sept. 2001.

[12] J. De Loera, D. Haws, R. Hemmecke, P. Huggins, B. Sturmfels, and R. Yoshida. Short rational functions for toric algebra and applications, July 2003. http://arxiv.org/abs/math.CO/0307350.

[13] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and R. Yoshida. A user's guide for latte v1.1, Nov. 2003. software package `LattE` is available at `http://www.math.ucdavis.edu/~latte/`.

[14] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. Effective lattice point counting in rational convex polytopes, Mar. 2003. http://www.math.ucdavis.edu/~latte/theory.html.

[15] E. Ehrhart. Polynômes arithmétiques et méthode des polyèdres en combinatoire. *International Series of Numerical Mathematics*, 35, 1977.

[16] J. Ferrante, V. Sarkar, and W. Thrash. On estimating and enhancing cache effectiveness. In U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua, editors, *Proceedings of the Fourth International Workshop on Languages and Compilers for Parallel Computing*, volume 589 of *Lecture Notes in Computer Science*, pages 328–343. Springer-Verlag, Aug. 1991.

[17] B. Franke and M. O'Boyle. Array recovery and high-level transformations for DSP applications. *ACM Transactions on Embedded Computing Systems*, 2(2):132–162, May 2003.

[18] Free Software Foundation, Inc. GMP. Available from `ftp://ftp.gnu.org/gnu/gmp`.

[19] S. Ghosh, M. Martonosi, and S. Malik. Cache miss

[5] This version embeds a polytope P in a single cone with a polar that has few rays if P has few facets [14].

equations: a compiler framework for analyzing and tuning memory behavior. *ACM Transactions on Programming Languages and Systems*, 21(4):703–746, 1999.

[20] N. Halbwachs, D. Merchat, and C. Parent-Vigouroux. Cartesian factoring of polyhedra in linear relation analysis. In *Static Analysis Symposium, SAS'03*, San Diego, June 2003. LNCS 2694, Springer Verlag.

[21] F. Hannig and J. Teich. Design space exploration for massively parallel processor arrays. In *Proceedings of the Sixth International Conference on Parallel Computing Technologies*, volume 2127 of *Lecture Notes in Computer Science*, pages 51–65, 2001.

[22] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott. The Omega calculator and library. Technical report, University of Maryland, Nov. 1996.

[23] B. Lisper. Fully automatic, parametric worst-case execution time analysis. In J. Gustafsson, editor, *Proc. Third International Workshop on Worst-Case Execution Time (WCET) Analysis*, pages 77–80, Porto, July 2003.

[24] V. Loechner. Polylib: A library for manipulating parameterized polyhedra. Technical report, ICPS, Université Louis Pasteur de Strasbourg, France, Mar. 1999.

[25] V. Loechner, B. Meister, and P. Clauss. Precise data locality optimization of nested loops. *J. Supercomput.*, 21(1):37–76, 2002.

[26] V. Loechner and D. K. Wilde. Parameterized polyhedra and their vertices. *International Journal of Parallel Programming*, 25(6):525–549, Dec. 1997.

[27] B. Nootaert. Een verbeterde methode voor de berekening van Ehrhart-polynomen. Master's thesis, Ghent University, 2004.

[28] E. Parker and S. Chatterjee. An automata-theoretic algorithm for counting solutions to Presburger formulas. In *Compiler Construction 2004*, volume 2985 of *Lecture Notes in Computer Science*, pages 104–119, Apr. 2004.

[29] W. Pugh. Counting solutions to Presburger formulas: How and why. In *SIGPLAN Conference on Programming Language Design and Implementation (PLDI'94)*, pages 121–134, 1994.

[30] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1986.

[31] V. Shoup. NTL. Available from `http://www.shoup.net/ntl/`.

[32] A. Turjan, B. Kienhuis, and E. Deprettere. A compile time based approach for solving out-of-order communication in Kahn Process Networks. In *IEEE 13th International Conference on Aplication-specific Systems, Architectures and Processors (ASAP'2002)*, July 2002.

[33] Y. Zhao and S. Malik. Exact memory size estimation for array computations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(5):517–521, October 2000.