Experiences with Enumeration of Integer Projections of Parametric Polytopes

Sven Verdoolaege[1], Kristof Beyls[2],
Maurice Bruynooghe[1], and Francky Catthoor[3]

[1] Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, B-3001 Leuven, Belgium
[2] Departement of Electronics and Information Systems, Ghent University – UGent,
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
[3] IMEC, Kapeldreef 75, B-3001 Leuven, Belgium; also at Katholieke Universiteit
Leuven, Department of Electrical Engineering

Abstract. Many compiler optimization techniques depend on the ability to calculate the number of integer values that satisfy a given set of linear constraints. This count (the enumerator of a parametric polytope) is a function of the symbolic parameters that may appear in the constraints. In an extended problem (the "integer projection" of a parametric polytope), some of the variables that appear in the constraints may be existentially quantified and then the enumerated set corresponds to the projection of the integer points in a parametric polytope.
This paper shows how to reduce the enumeration of the integer projection of parametric polytopes to the enumeration of parametric polytopes. Two approaches are described and experimentally compared. Both can solve problems that were considered very difficult to solve analytically.

1 Introduction

Many compiler optimization techniques require the enumeration of objects of a certain class. Examples include counting the number of calculations, accessed memory locations or statement executions in a loop nest or parts thereof [6, 21, 23, 28, 29, 38]; calculating the number of cache misses in a loop [12, 16, 24]; computing the number of dynamically allocated bytes [11]; enumerating the number of live array elements at a given iteration (i, j) [27, 42]; counting how many parallel processing elements can be used when executing a loop on an FPGA [5, 22, 25] and computing the amount of communication for a given schedule of parallel tasks on a distributed computing system [9, 26, 37].

These counts are used to drive optimizations such as increasing parallelism [38], minimizing memory size [1, 2, 27, 38, 42], estimating worst case execution time [28], increasing cache effectiveness [6, 16], high-level transformations for DSP applications [23], converting software loops into parallel hardware implementations [5, 18, 22, 25, 38] and minimizing communication overhead in distributed applications [9, 26, 37]. In many of these optimizations, the objects or events to be counted are modeled as the integer solutions to systems of linear

inequalities, i.e., as the elements of a set $S = \{\, \mathbf{x} \in \mathbb{Z}^d \mid A\mathbf{x} + \mathbf{c} \geq \mathbf{0} \,\}$, with $A \in \mathbb{Z}^{n \times d}$ and $\mathbf{c} \in \mathbb{Z}^n$. Furthermore, they often need the count in terms of a vector of parameters $\mathbf{p}$ (e.g., in the presence of symbolic loop bounds):

$$S_{\mathbf{p}} = \{\mathbf{x} \in \mathbb{Z}^d \mid A\mathbf{x} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0}\}. \tag{1}$$

A recent efficient algorithm that computes the function from specific values of $\mathbf{p}$ to the number of elements in $S_{\mathbf{p}}$ is presented in [40]. This paper considers the more general counting problem, where some of the variables can be existentially quantified. We propose a general solution for counting the number of elements (in terms of parameters $\mathbf{p}$) for sets that can be expressed in the form

$$\left\{ \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{y} \in \mathbb{Z}^{d'} : A\mathbf{x} + D\mathbf{y} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0} \right\}. \tag{2}$$

Computing the number of elements in such a set is, amongst others, needed in the program analyses described in [1, 2, 5, 6, 9, 12, 16, 25, 26, 37, 42]. Practical examples are discussed in an extended version of this paper, see [39].

Example 1. Consider an example adapted from [14] (Figure 1(a)). Assume we want to know the total number of array elements accessed by the statement in the inner loop as a function of the symbolic parameter p. This problem is equivalent to counting the number of elements in the set

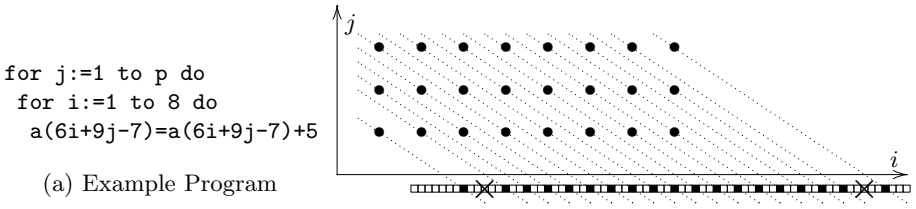
$$S_p = \{\, l \in \mathbb{Z} \mid \exists i, j \in \mathbb{Z} : l = 6i + 9j - 7 \wedge 1 \leq j \leq p \wedge 1 \leq i \leq 8 \,\}, \tag{3}$$

which can be written in the same form as (2):

$$\left\{ l \in \mathbb{Z} \mid \exists \begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{Z}^2 : \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} l + \begin{pmatrix} -6 & -9 \\ 6 & 9 \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} p + \begin{pmatrix} 7 \\ -7 \\ -1 \\ 0 \\ -1 \\ 8 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\}.$$

Figure 1(b) shows the array elements that are accessed for $p = 3$. These elements do not correspond to the integer points in a polytope. Even after scaling by 3 it still contains two "holes" (marked by $\times$ on the figure). These holes complicate the enumeration. For $p = 3$, the set S_p contains 19 points, see Figure 1(b). In general, the number of points in S_p can be described by the function Different polynomials represent the count in different regions of the parameter space. Following [36], we call these regions *chambers*. In general, the count in each chamber is described by a *step-polynomial*, as defined in Section 2.

The solution to a counting problem is called the enumerator of the set of constraints. Without parametric variables in the counting problem, the enumerator is an integer; otherwise the enumerator is a function that maps the values of the parametric variables to an integer. Different applications for different types of constraints has led to different proposals. Below, when we refer to the complexity of algorithms, we always mean for a fixed number of variables. Note that the enumeration of even parametric polytopes is NP-hard.



(a) Example Program

(b) Array elements accessed for $p = 3$

Fig. 1. Example adapted from [14].

Linear inequalities. Barvinok[3] was first to propose an algorithm for enumerating sets defined by linear inequalities that is polynomial-time.

Parametric linear inequalities. Ehrhart [19] showed that the general form of the enumerator for a certain form of parametric linear inequalities with a single parameter is a quasi-polynomial. Clauss et al. [15] extended this theory to handle the more general form shown in Equation (1), albeit in exponential time complexity. De Loera et al. [17] implemented Barvinok's [3] polynomial-time algorithm for enumerating sets defined by linear inequalities and its extension to compute the Ehrhart series corresponding to the dilation nP of a polytope P. Finally, Verdoolaege et al. [40] implemented a polynomial time algorithm for the counting problem of Equation (1).

Linear inequalities with existential variables. Barvinok and Woods [4] propose a polynomial time algorithm. No implementation has been reported, and the extension to parameters is not obvious.

Parametric linear inequalities with existential variables. Boulet [9] proposes to compute the enumerator of a set of parametric linear inequalities with existential variables in two steps. First, parametric integer programming (PIP) [20] is used to eliminate the existential variables, after which Clauss's [15] method is used to enumerate the resulting set of linear inequalities. However, no extensive evaluation has been reported and the appendix in [10] indicates that the method cannot compute the enumerator fully automatically. Meister [30] proposes a similar technique using his more general periodic polyhedra instead of PIP. No implementation has been reported. Clauss [13] proposed a method (recently implemented [35]) based on "thick facets" that works for a single existential variable.

Non-parametric Presburger formula. Presburger formulas consist of linear inequalities of integer variables, combined by existential and universal quantifiers, disjunction, conjunction and negation. ($\exists, \forall, \vee, \wedge, \neg$). Two recent methods [8, 31] represent the formula as a finite automaton to count the number of its integer solutions in exponential time.

Parametric Presburger formula. In [33], a number of rewrite rules are proposed to compute the enumerator of a parametric Presburger formula. However, the rules seem ad-hoc and no implementation has been reported, mak-

ing it hard to evaluate their usefulness in practice. In [41], a polynomial-time algorithm for enumerating sets as in (2) is proposed without implementation.

This paper investigates the combination of PIP with our method for parametric polytopes [40]. This combination can handle the parametric counting problems with existential variables reported in [2, 6, 9, 12] that were previously considered difficult or even unsolvable. Since PIP is worst-case exponential, we also investigate an alternative method that uses a number of simple polynomial rewriting rules to eliminate existential quantifiers. While all existential quantifiers are eliminated in our experiments on a wide range of practical applications, some could remain. In that case, PIP can be used as a back-up to solve the reduced problem. Theoretically, parametric Presburger formulas can be transformed into a disjoint union of sets of the form (2). For the majority of the parametric Presburger formulas we considered, this transformation could be performed efficiently and automatically by the Omega library.

Section 2 introduces background on parametric polytopes and enumerators and Section 3 two extensions for handling existential variables. An experimental evaluation is performed in Section 4, and concluding remarks follow in Section 5.

2 Parametric Polytopes

Before tackling integer projections of parametric polytopes, we review the results on enumeration of parametric polytopes. We refer to [40] for the details.

Definition 1. *A* rational polyhedron $P \in \mathbb{Q}^d$ *is a set of rational* d*-dimensional vectors* $\mathbf{x}$ *defined by linear inequalities*

$$P = \left\{ \mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} + \mathbf{c} \geq \mathbf{0} \right\}, \text{ with } A \in \mathbb{Z}^{m \times d} \text{ and } \mathbf{c} \in \mathbb{Z}^m. \tag{4}$$

A rational polytope *is a bounded rational polyhedron.*

Definition 2. *A* rational parametric polytope $P_{\mathbf{p}}$ *with* n *parameters* $\mathbf{p}$ *is a set of rational* d*-dimensional vectors* $\mathbf{x}$ *defined by linear inequalities on* $\mathbf{x}$ *and* $\mathbf{p}$

$$P_{\mathbf{p}} = \left\{ \mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0} \right\} \tag{5}$$

with $A \in \mathbb{Z}^{m \times d}$, $B \in \mathbb{Z}^{m \times n}$ *and* $\mathbf{c} \in \mathbb{Z}^m$, *and such that for each fixed value* $\mathbf{p}_0$ *of* $\mathbf{p}$, $P_{\mathbf{p}_0}$ *defines a (possibly empty) rational polytope in* $\mathbb{Q}^d$.

All the polyhedra in this paper are rational. If the parametrization of a polytope is clear from the context, subscript $\mathbf{p}$ is omitted. Note that the same equations that define a parametric polytope also define a potentially unbounded $(d+n)$-dimensional polyhedron in the combined data and parameter space.

$$P' = \left\{ (\mathbf{x}, \mathbf{p}) \in \mathbb{Q}^{d+n} \mid A\mathbf{x} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0} \right\}$$

Definition 3. *The* enumerator $c_P(\mathbf{p})$ *of a parametric polytope* $P_{\mathbf{p}}$ *is a function from the set of* n*-dimensional integer vectors* $\mathbb{Z}^n$ *to the set of natural numbers* $\mathbb{N}$.[4] *The function value at* $\mathbf{p}_0$*, denoted* $c_P(\mathbf{p}_0)$*, is the number of integer points in the polytope* $P_{\mathbf{p}_0}$.

$$\begin{aligned} c_P : \mathbb{Z}^n &\to \mathbb{N} \\ \mathbf{p}_0 &\mapsto c_P(\mathbf{p}_0) = \#\left(\mathbb{Z}^d \cap \left\{\, \mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} + B\mathbf{p}_0 + \mathbf{c} \geq \mathbf{0} \,\right\}\right) \end{aligned}$$

Definition 4. *A* step-polynomial $g : \mathbb{Z}^n \to \mathbb{Q}$ *of degree* d *is a function written in the form*

$$g(\mathbf{p}) = \sum_{j=1}^{m} \alpha_j \prod_{k=1}^{d_j} \lfloor \langle \mathbf{a}_{jk}, \mathbf{p} \rangle + b_{jk} \rfloor \,,$$

with $\alpha_j \in \mathbb{Q}$, $\mathbf{a}_{jk} \in \mathbb{Q}^n$, $b_{jk} \in \mathbb{Q}$, $\langle \cdot, \cdot \rangle$ *the inproduct, and* $\lfloor \cdot \rfloor$ *is the floor (greatest integer) function. A* piecewise step-polynomial $f : \mathbb{Z}^n \to \mathbb{Q}$ *consists of a subdivision of* $\mathbb{Z}^n$*, called the* chambers,[5] *each with an associated step-polynomial.*

Proposition 1 ([40]). *For fixed dimensions* d *and* n*, the enumerator of a parametric polytope can be computed as a piecewise step-polynomial in a time polynomial in the input size (the number of bits needed to represent the input [34]).*

Example 2. Consider the parametric polytope P_p

$$\{\, (x, y) \in \mathbb{Q}^2 \mid x + 3y \leq 8 \wedge x + 2y + 1 \leq 0 \wedge x + 2y + p \geq 0 \wedge x + 3p + 11 \leq 0 \,\}.$$

Figure 2 shows P_p for different values of p. The number of integer points is given by

$$c_P(p) = \begin{cases} 5 & \text{if } p \geq 3 \\ -\frac{3}{4}p^2 + \frac{15}{4}p + \frac{1}{2}\lfloor \frac{1}{2}p \rfloor & \text{if } 1 \leq p \leq 2 \end{cases}.$$

This enumerator has two chambers: $\{\, p \mid p \geq 3 \,\}$ and $\{\, p \mid 1 \leq p \leq 2 \,\}$. The step-polynomial associated with the first chamber is a constant. For the second chamber, we obtain a polynomial in the floors of p and $\frac{1}{2}p$. Note that this is only one of the possible representations of $c_P(p)$. For this particular example, a much simpler representation exists with chambers $\{\, p \mid p \geq 2 \,\}$ and $\{\, 1 \,\}$, and the constants 5 and 3 for the corresponding functions.

3 Existential Variables

This section considers the extension with existential variables. The general form of these counting problems, given in Equation 2, is equivalent to

$$\#\pi_d \left(\mathbb{Z}^{(d+d')} \cap \left\{\, (\mathbf{x}, \mathbf{y}) \in \mathbb{Q}^{(d+d')} \mid A\mathbf{x} + D\mathbf{y} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0} \,\right\}\right)$$

[4] In [40], the symbol $\mathcal{E}$ is used instead of c.
[5] Chambers are also called validity domains in some publications.

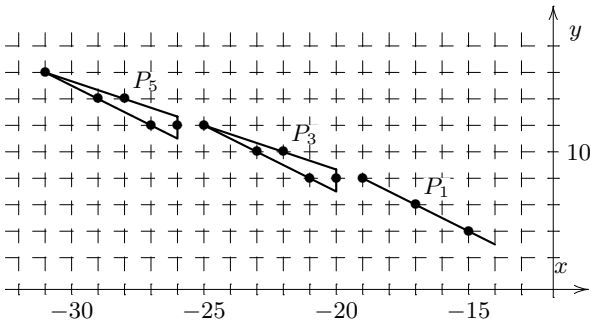


Fig. 2. Different instantiations of the parametric polytope from Example 2.

where π_d is the projection onto the first d dimensions. This parametric count corresponds to the number of points in the projection of the integer points in a parametric polytope, or integer projection of a parametric polytope for short.

Note that we cannot simply ignore the existential quantifier and count the number of points as if the set were a parametric polytope, since for any particular value of $\mathbf{x}$ there may be several values of $\mathbf{y}$ that satisfy the constraints. We also cannot simply project out the existential variables since there may exist values of $\mathbf{x}$ in this projection for which there is no *integer* value of $\mathbf{y}$ satisfying the constraints. E.g., if we project P_5 in Figure 2 onto the x-axis, then this projection will contain the value -30, while there is no integer y such that $(-30, y) \in P_5$.

We consider three techniques for eliminating existential variables; they are polynomial in the input size (for fixed dimensions) but not always applicable. In the latter case, one can fall back upon parametric integer programming to count the set. However, this is worst-case exponential, even for fixed dimensions.

3.1 Elimination

Unique Existential Variables The existential quantifiers introduced by tools that automatically extract counting problems from source code can sometimes be redundant. This occurs when for each $\mathbf{x}$ in the set, there is at most one integer value for y_i that satisfies the constraints. In such a case, the existential quantifier for y_i can be omitted without affecting the cardinality of the set.

Many cases can be detected when there is a constraint that involves y_i but none of the other existential variables $\overline{\mathbf{y}}$. Without loss of generality, we assume the constraint establishes a lower bound on the variable y_i, i.e., it is of the form

$$n_l y_i + \langle \mathbf{a}_l, \mathbf{x} \rangle + \langle \mathbf{b}_l, \mathbf{p} \rangle + c_l \geq 0 \tag{6}$$

with $n_l \in \mathbb{N}$. Combining this constraint with an upper bound

$$-n_u y_i + \langle \mathbf{a}_u, \mathbf{x} \rangle + \langle \overline{\mathbf{d}}_u, \overline{\mathbf{y}} \rangle + \langle \mathbf{b}_u, \mathbf{p} \rangle + c_u \geq 0 \tag{7}$$

we obtain

$$-n_u(\langle \mathbf{a}_l, \mathbf{x} \rangle + \langle \mathbf{b}_l, \mathbf{p} \rangle + c_l) \leq n_u n_l y_i \leq n_l(\langle \mathbf{a}_u, \mathbf{x} \rangle + \langle \overline{\mathbf{d}}_u, \overline{\mathbf{y}} \rangle + \langle \mathbf{b}_u, \mathbf{p} \rangle + c_u). \tag{8}$$

The number of distinct integer values for $n_u n_l y_i$ is given by the upper bound minus the lower bound plus one. If this number is smaller than $n_u n_l$, then the two constraints admit at most one integer value for y_i. That is, if

$$n_l(\langle \mathbf{a}_u, \mathbf{x} \rangle + \langle \overline{\mathbf{d}}_u, \overline{\mathbf{y}} \rangle + \langle \mathbf{b}_u, \mathbf{p} \rangle + c_u) + n_u(\langle \mathbf{a}_l, \mathbf{x} \rangle + \langle \mathbf{b}_l, \mathbf{p} \rangle + c_l) + 1 \leq n_l n_u \quad (9)$$

for all integer values that satisfy the constraints, then y_i is uniquely determined by $\mathbf{x}$ and $\mathbf{p}$ and can therefore be treated as a regular variable, without existential quantification. It is independent of the other existential variables because of our assumption that one of the constraints does not involve these other variables. Condition (9) can easily be checked by adding the negation to the existing set of constraints and testing for satisfiability. Note that it is sufficient to find one such pair to be able to drop the existential quantification of the variable.

Example 3. Consider the set S

$$\{\, x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : x + 3y \leq 8 \wedge x + 2y + 1 \leq 0 \wedge x + 2y + p \geq 0 \wedge x + 3p + 11 \leq 0 \,\}.$$

This is the same set that appeared in Example 2, except that y is now an existential variable. Since there is only a single existential variable, all constraints are independent of the "other existential variables". Using $x + 2y + p \geq 0$ and $-x - 3y + 8 \geq 0$ as constraints, condition (9) yields

$$x + 3p + 17 \leq 6. \quad (10)$$

All elements of the set satisfy this constraint so we can remove the existential quantification and the set S is then $P_p \cap \mathbb{Z}$, with P_p the set from Example 2.

Even if there is no single existential variable that is unique, some linear combination of existential variables may still be unique. To avoid enumerating all possible combinations, we only consider this case if we have two constraints that are "parallel in the existential space", i.e., such that $\mathbf{d}_l = n_l \mathbf{d}$ and $\mathbf{d}_u = -n_u \mathbf{d}$ for some positive integers n_l and n_u and an integer vector $\mathbf{d}$ with greatest common divisor (gcd) 1. We compute condition (9) from (6) and (7) with y_i replaced by $\langle \mathbf{d}, \mathbf{y} \rangle$ ($\overline{\mathbf{d}}_u$ is $\mathbf{0}$ in this case). If this condition holds, we perform a change of basis such that $y'_1 = \langle \mathbf{d}, \mathbf{y} \rangle$, which we now know to be unique. Such a change of basis can be obtained through transformation by the unimodular extension of $\mathbf{d}$ [7].

Example 4. Consider the set S (3) from Section 1. This set satisfies the equality $l = 6i + 9j - 7$, which means that $2i + 3j$ is unique. Transforming this set using the unimodular extension of $\mathbf{d} = (2, 3)$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

we obtain

$$S = \{\, l \in \mathbb{Z} \mid \exists x, y \in \mathbb{Z} : l = 3x - 7 \wedge -x - p \leq 2y \leq -x - 1 \wedge -x + 1 \leq 3y \leq -x + 8 \,\}.$$

Equation $l = 3x - 7$ provides an upper and a lower bound on x, hence Equation (9) is trivially satisfied, $\exists x$ can be removed and also l as it is now redundant.

$$S' = \{\, x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : -x - p \leq 2y \leq -x - 1 \wedge -x + 1 \leq 3y \leq -x + 8 \,\}. \quad (11)$$

Redundant Existential Variables Consider again a lower bound on the existential variable y_i: $n_l y_i + \langle \mathbf{c}_l, \mathbf{w} \rangle \geq 0$, where we used $\mathbf{c}_l := (\mathbf{a}_l, \overline{\mathbf{d}}_l, \mathbf{b}_l, c_l)$ and $\mathbf{w} := (\mathbf{x}, \overline{\mathbf{y}}, \mathbf{p}, 1)$ for brevity. Since we are only interested in integer values of y_i, this is equivalent to $n_u(n_l y_i + \langle \mathbf{c}_l, \mathbf{w} \rangle) + n_u - 1 \geq 0$, for any positive integer n_u. Similarly, for an upper bound we obtain $n_l(-n_u y_i + \langle \mathbf{c}_u, \mathbf{w} \rangle) + n_l - 1 \geq 0$. The range in (8) can therefore be expanded to

$$-n_u \langle \mathbf{c}_l, \mathbf{w} \rangle - n_u + 1 \leq n_u n_l y_i \leq n_l \langle \mathbf{c}_u, \mathbf{w} \rangle + n_l - 1.$$

If this range is larger than $n_u n_l$, i.e., if

$$n_l \langle \mathbf{c}_u, \mathbf{w} \rangle + n_u \langle \mathbf{c}_l, \mathbf{w} \rangle + n_l - 1 + n_u - 1 + 1 \geq n_l n_u, \quad (12)$$

then there is *at least* one integer value for each given value of the other variables. If this holds for all pairs of constraints, then variable y_i does not restrict the solutions in any way and can be eliminated (known as the Omega test [32]). Note that the constraints need not be independent of the other variables.

Example 5. Consider the set

$$S = \{\, x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : -x - p \leq 2y \leq -x - 1 \wedge x \leq -11 \wedge \\ -x + 1 \leq 3y \leq -x + 8 \wedge x + 3p + 10 \leq 0 \wedge p \geq 3 \,\}.$$

This set is shown (■) in Figure 3. Pairwise combining the two upper and two lower bounds to form condition (12), we obtain $2p + 1 \geq 4$, $26 \geq 9$, $-x - 1 \geq 6$ and $x + 20 + 3p \geq 6$. All of these are true in S. (In practice we would use the least common multiple of n_l and n_u instead of their product.) Variable y can therefore be eliminated and we obtain $S = \{\, x \in \mathbb{Z} \mid x \leq -11 \wedge p \geq 3 \wedge x + 3p + 10 \geq 0 \,\}$.

Independent Splits If neither of the two heuristics above apply, we can split the set into two or more parts by cutting the polyhedron in the combined space along a hyperplane. By considering hyperplanes that are independent of the existential variables, we ensure that the enumerator of the original set is the sum of the enumerators of the parts; otherwise we would obtain sets that may intersect, requiring the computation of a disjoint union.

In particular, we consider all pairs of a lower and an upper bound on an existential variable that do not depend on other existential variables, i.e., they are of the form (6). If neither condition (9) nor condition (12) is satisfied over the whole set, then we cut off that part of the set where condition (9) does hold. In the remaining part, condition (12) holds for this particular pair of constraints.

Since the number of pairs of constraints is polynomial in the input size, the number of sets we split off is also polynomial and so the whole technique, if it applies, is polynomial in the input size (for fixed dimension). As a special case, this technique always applies if there is only a single existential variable.

Example 6. Consider once more the set S' (11) from Example 4. The bottom of Figure 3 shows the projection of the corresponding polyhedron in the combined data-parameter space onto the xp-plane and the top shows the xy-slice at $p = 4$. The two constraints we considered in Example 3 also appear in this set. Condition (10) does not hold for the whole set, but instead is used to cut off the part that we considered in Example 3. This is the leftmost part (■) in Figure 3. Using the other constraints, we further split off $p \leq 2$ and $x \geq -10$. The remaining part is the set discussed in Example 5.

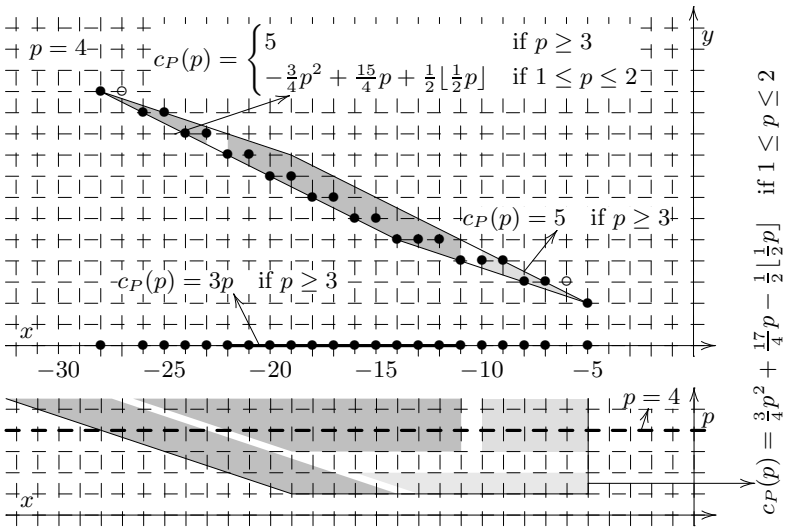


Fig. 3. Decomposition of the set from Example 6.

3.2 Parametric Integer Programming

Parametric integer programming (PIP) [20] is a technique for computing the lexicographical minimum of a parametric polytope as a function of the parameters. The solution is defined by rational linear expressions in both the original parameters and possibly some extra parameters, defined as the floors of rational linear expressions of other parameters. Different solutions may exist in different parts of the parameter space, each defined by linear inequalities in the parameters (both original and extra).

PIP can help in the enumeration of integer projections of parametric polytopes. Consider a set S (2) with d regular variables, d' existential variables and n

parameters. Compute the lexicographical minimum of the d' existential variables with the regular variables and the original parameters as parameters, i.e.,

$$\mathbf{y}^{\mathrm{m}}_{(\mathbf{x},\mathbf{p})} = \operatorname{lexmin} \left\{ \mathbf{y} \in \mathbb{Z}^{d'} \mid A\mathbf{x} + D\mathbf{y} + B\mathbf{p} + \mathbf{c} \geq \mathbf{0} \right\}.$$

Replacing $\mathbf{y}$ by $\mathbf{y}^{\mathrm{m}}_{(\mathbf{x},\mathbf{p})}$ in the definition of S does not change the number of solutions. However, $\mathbf{y}^{\mathrm{m}}_{(\mathbf{x},\mathbf{p})}$ is unique (it satisfies Equation (9)) and the quantifier can be dropped. The extra parameters that may appear in the solution can be handled by considering them as extra (unique) existential variables in the set S.

PIP always applies but is worst-case exponential, even for fixed dimension. It may decrease or increase the total dimension of the problem depending on the difference between the number of extra variables and the number of existential variables in the original problem. The dimension decreases by 1 for each such variable since PIP introduces an equality for each of them. The total dimension is important since the enumeration technique for parametric polytopes is only polynomial for fixed dimension.

Example 7. Consider again the set S' (11) from Example 4. We have:

$$\begin{aligned} y^{\mathrm{m}}_{(x,p)} &= \operatorname{lexmin} \{\, y \in \mathbb{Z} \mid -x - p \leq 2y \leq -x - 1 \wedge -x + 1 \leq 3y \leq -x + 8 \,\} \\ &= \begin{cases} 1 - x - \left\lfloor \frac{2-2x}{3} \right\rfloor & \text{if } x + 3p + 2 \geq 0 \\ -x - \left\lfloor \frac{p-x}{2} \right\rfloor & \text{otherwise} \end{cases}. \end{aligned}$$

Hence S' is the (disjoint) union of two sets $S_1 \sqcup S_2$. E.g., S_1 is defined as

$$\begin{aligned} S_1 = \{\, x \in \mathbb{Z} \mid \exists y, q \in \mathbb{Z} : y = 1 - x - q \wedge 2 - 2x \leq 3q \leq 4 - 2x \wedge \\ x + 3p + 2 \geq 0 \wedge -x - p \leq 2y \leq -x - 1 \wedge -x + 1 \leq 3y \leq -x + 8 \,\}, \end{aligned}$$

where q is the new "parameter" $q = \lfloor (2 - 2x)/3 \rfloor$. Each new set has exactly one extra (unique) existential variable, hence the total dimension remains constant.

4 Experiments

We count the number of integer points in formulas resulting from reuse distance equations [6], cache miss analysis [12], memory size estimation [2] and communication volume computation [9]. An overview of these problems and details on the specific versions of the `PolyLib` and `Barvinok` libraries we used are in [39].

4.1 Reuse Distances

We performed extensive experiments calculating reuse distances of a set of relatively small but representative test programs including matrix-matrix multiplication and Cholesky factorization. The second column of Table 1(a) shows the number of times a particular rule from Section 3.1 was used. The remaining columns are explained in Section 4.2. The row "Fixed" refers to the special case

type	RD	Chatterjee	Balasa	Boulet
Sets	19177	8+13	4	1
Fixed	3470	0+2	14	5
Change+Fixed	0	0+0	0	2
Unique	4890	8+9	0	0
Change+Unique	18	0+0	0	0
Redundant	684	0+0	2	1
Split	286	0+0	0	0
PIP	0	0+0	0	0

(a) Rule application distribution for polytopes originating from reuse distance equations (RD), cache miss analysis (Chatterjee), memory size estimation (Balasa) and communication volume computation (Boulet).

#EV	Dimension Decrease						
	?	-1	0	1	2	3	4
1			6186	527	25		
2		6	779	102	41	10	
3	2	2	122	66	11	6	
4			6	38	5		7
5			3	1	5		1
6	2					3	

(b) Dimension decrease induced by PIP in terms of the number of existential variables ($\#EV$).

Table 1. Tables with experimental results

of a unique existential variable determined by an equality; "Change" refers to a change of basis. In most of the tests we assume a cache line size of four words. Frequently, the matrix size is a multiple of the cache line size. The resulting enumerators for such cases were experimentally verified through a cache simulation. PIP was never needed in these experiments. Simply ignoring the existential quantifiers would have produced the wrong result, however, since we had to split some sets. Curiously, some sets contained redundant existential variables, even though they were created by Omega which should have removed them.

We also investigated the impact of the input size. For reuse distance calculation for matrix-matrix multiplication varying the sizes of the matrices, ranging from 20×20 to 640×640, produced no measurable increase in computation time. However, on tests where matrix sizes are not multiples of the cache line size, Omega fails to simplify the resulting Presburger formulas, and produces inexact formulas containing `Unknown`s. We were forced to devise a way that avoids Omega as much as possible. This modification increases the number of sets to enumerate. For matrices of size 19×19 and 41×41, some of the resulting sets proved too difficult to handle. For both sizes, we found at least one set where we had to abort `PolyLib` after one hour while it was calculating step-polynomials. Directly applying PIP also did not produce a result; moreover, PIP failed also on two other sets that were handled by our reduction rules.

Next, we compared the relative performance of PIP and our rules when combined with our polytope enumeration technique. A priori, we would expect that the method with PIP would perform worse since PIP itself is worst-case exponential and the use of PIP may significantly increase the dimension of the problem. Table 1(b) shows that this increase did not occur for our set of examples. Ignoring the 4 sets that failed to produce an answer (column "?") as well as

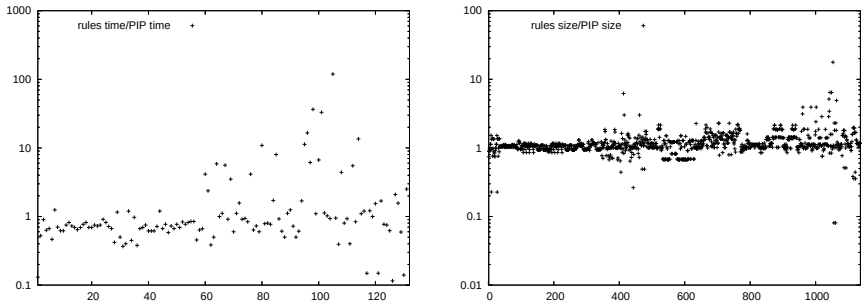


Fig. 4. Comparison between PIP and our rules

the 11355 sets without existential variables (not shown in the table), of the 7952 resulting polytopes, almost 90% have the same dimension as the original set. Furthermore, except for 8 polytopes which experience an increase in dimension, all others have a dimension that is smaller than that of the original set. There are even 35 polytopes with a decrease in dimension that is *larger* than the number of existential variables. The explanation for this phenomenon is that some of the sets allow a range of rational values in one of the dimension, but only a single integer value, e.g., $4 \leq 5i \leq 7$. Again, this is surprising since Omega should have discovered the corresponding equality. For the sets that PIP was able to handle, Figure 4 shows the relative execution time on the left, for sets with an execution time larger than 0.1s, and the relative size of the resulting enumerator on the right, for sets where this relative size is not exactly one. We conclude that for our set of examples, neither method has a clear performance gain over the other.

We previously reported [40] that our method for enumerating parametric polytopes is faster, sometimes significantly, than Clauss's method. Figure 5 provides further evidence of this improvement. The inputs are the parametric polytopes generated by PIP on the reuse distance sets. From a total of 18951 polytopes, 907 had a computation time of more than 0.1s. The implementation of Clauss's method failed to produce a complete result for 190 of these polytopes, due to "degenerate domains". The ratio of the execution times for the remaining polytopes is shown for the "raw" polytopes on the left and for the polytopes with redundant equalities removed on the right. For 17 polytopes on the left and 8 polytopes on the right, the computation with Clauss's method exceeds 10 minutes. The "ratio" for these polytopes is fixed to 100000 on the figures.

4.2 Other Applications

In this section, we mainly compare the combination of PIP with either Clauss's method or our method [40]. Applying Clauss's method on a problem for an 8×8 processor array presented in [10] leads to a computation time of 713s. The same problem for a 64×64 array, requires 6855s. Apparently, Clauss's method does not exploit equalities; first removing them reduces times to 0.04s and 1.43s. Using

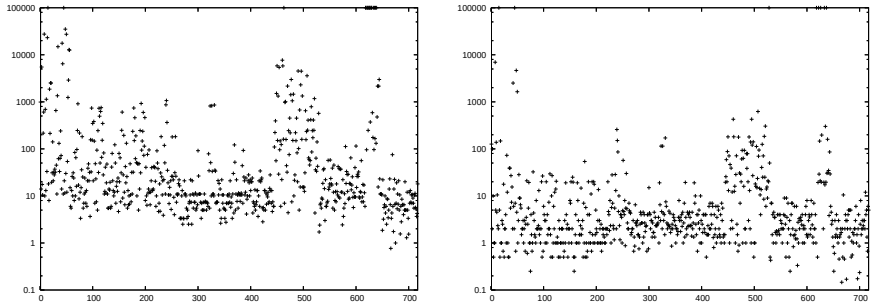


Fig. 5. Execution time ratio for Clauss's method compared to ours for the original polytopes on the left and preprocessed polytopes on the right.

our own method, which removes equalities automatically, we obtain both results in 0.01s. The applied rules are shown in column 5 of Table 1(a).

An example in [2] counts the number of array elements accessed by 4 references in a motion estimation loop kernel, for a number of different values of the symbolic loop bounds. We handled the symbolic loop bounds parametrically, thereby obtaining a single solution for all possible values of the symbolic loop bounds. Using Clauss's method (after removing equalities), counting takes respectively 1.38s, 0.01s, 1.41s and 1.41s. With our method, times are 0.06s, 0.01s, 0.07s and 0.04s. The applied rules are shown in column 4 of Table 1(a).

Finally, we considered a large formula from [12]. Computation times for the 8 disjuncts range from a couple of seconds to 1.5 minutes while Clauss's method for one of the parametric polytopes did not finish in 15 hours. To enumerate the whole formula, a disjoint union consisting of 13 sets was computed in less than a second using Omega. Their enumeration times are in the same range as those of the original disjuncts. The applied rules are shown in column 3 of Table 1(a), with the original disjuncts on the left and the disjoint sets on the right.

5 Conclusions

Many compiler analyses and optimizations require the enumeration of the integer projection of a parametric polytope. As shown, this problem can be reduced to a problem of enumerating parametric polytopes, either by using PIP or by applying a number of rewriting rules. This reduction, together with our polynomial method for enumerating parametric polytopes [40], yields a method that works well in practice and can solve many problems that were previously considered very difficult or even unsolvable. Although both approaches usually have comparable performance, there are some examples where PIP runs out of time. Since the applicability of the rules is easy to check, it seems appropriate to apply the rules first and to use PIP only when no complete reduction is achieved.

Acknowledgements Sven Verdoolaege was supported by FWO-Vlaanderen. Kristof Beyls was supported by research project GOA-12051002.

References

[1] S. Anantharaman and S. Pande. Compiler optimizations for real time execution of loops on limited memory embedded systems. In *RTSS*, 1998.

[2] F. Balasa, F. Catthoor, and H. De Man. Background memory area estimation for multidimensional signal processing systems. *IEEE Transactions on VLSI*, 3(2):157–172, 1995.

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

[4] A. Barvinok and K. Woods. Short rational generating functions for lattice point problems. *J. Amer. Math. Soc.*, 16:957–979, Apr. 2003.

[5] M. Bednara, F. Hannig, and J. Teich. Generation of distributed loop control. In *SAMOS*, volume 2268 of *LNCS*, pages 154–170, 2002.

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

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

[8] B. Boigelot and L. Latour. Counting the solutions of Presburger equations without enumerating them. *Theoretical Computer Science*, 313(1):17–29, Feb. 2004.

[9] P. Boulet and X. Redon. Communication pre-evaluation in HPF. In *EUROPAR'98*, volume 1470 of *LNCS*, pages 263–272. Springer Verlag, 1998.

[10] P. Boulet and X. Redon. Communication pre-evaluation in HPF. Technical report, Université des Sciences et Technologies de Lille, 1998. AS-182.

[11] V. Braberman, D. Garbervetsky, and S. Yovine. On synthesizing parametric specifications of dynamic memory utilization. Technical Report TR-2004-03, VERIMAG, Oct. 2003.

[12] S. Chatterjee, E. Parker, P. J. Hanlon, and A. R. Lebeck. Exact analysis of the cache behavior of nested loops. In *PLDI*, pages 286–297, 2001.

[13] P. Clauss. Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: Applications to analyze and transform scientific programs. In *International Conference on Supercomputing*, pages 278–285, 1996.

[14] P. Clauss. Handling memory cache policy with integer points counting. In *European Conference on Parallel Processing*, pages 285–293, 1997.

[15] P. Clauss and V. Loechner. Parametric analysis of polyhedral iteration spaces. *Journal of VLSI Signal Processing*, 19(2):179–194, July 1998.

[16] P. D'Alberto, A. Veidembaum, A. Nicolau, and R. Gupta. Static analysis of parameterized loop nests for energy efficient use of data caches. In *COLP*, 2001.

[17] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. Effective lattice point counting in rational convex polytopes. *The Journal of Symbolic Computation*, 38(4):1273–1302, 2004.

[18] S. Derrien, A. Turjan, C. Zissulescu, B. Kienhuis, and E. Deprettere. Deriving efficient control in Kahn process network. In *SAMOS*, 2003.

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

[20] P. Feautrier. Parametric integer programming. *Operationnelle/Operations Research*, 22(3):243–268, 1988.

[21] J. Ferrante, V. Sarkar, and W. Thrash. On estimating and enhancing cache effectiveness. In *LCPC*, volume 589 of *LNCS*, pages 328–343, 1991.
[22] D. Fimmel and R. Merker. Design of processor arrays for real-time applications. In *Euro-Par '98*, LNCS, pages 1018–1028, 1998.
[23] B. Franke and M. O'Boyle. Array recovery and high-level transformations for DSP applications. *ACM TECS*, 2(2):132–162, May 2003.
[24] S. Ghosh, M. Martonosi, and S. Malik. Cache miss equations: a compiler framework for analyzing and tuning memory behavior. *ACM Transactions on Programming Languages and Systems*, 21(4):703–746, 1999.
[25] F. Hannig and J. Teich. Design space exploration for massively parallel processor arrays. In *PaCT*, volume 2127 of *LNCS*, pages 51–65, 2001.
[26] F. Heine and A. Slowik. Volume driven data distribution for NUMA-machines. In *Euro-Par*, LNCS, pages 415–424, 2000.
[27] P. G. Kjeldsberg, F. Catthoor, and E. J. Aas. Data dependency size estimation for use in memory optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(7), July 2003.
[28] B. Lisper. Fully automatic, parametric worst-case execution time analysis. In *Workshop on Worst-Case Execution Time (WCET) Analysis*, pages 77–80, 2003.
[29] V. Loechner, B. Meister, and P. Clauss. Precise data locality optimization of nested loops. *J. Supercomput.*, 21(1):37–76, 2002.
[30] B. Meister. Projecting periodic polyhedra for loop nest analysis. In *CPC*, pages 13–24, 2004.
[31] E. Parker and S. Chatterjee. An automata-theoretic algorithm for counting solutions to Presburger formulas. In *Compiler Construction*, volume 2985 of *LNCS*, pages 104–119, 2004.
[32] W. Pugh. The Omega test: a fast and practical integer programming algorithm for dependence analysis. In *Conference on Supercomputing*, pages 4–13, 1991.
[33] W. Pugh. Counting solutions to Presburger formulas: How and why. In *PLDI*, pages 121–134, 1994.
[34] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.
[35] R. Seghir. Dénombrement des point entiers de l'union et de l'image des polyédres paramétrés. Master's thesis, ICPS, Strasbourg, France, June 2002.
[36] B. Sturmfels. On vector partition functions. *J. Comb. Theory Ser. A*, 72(2):302–309, 1995.
[37] E. Su and A. L. et al. Advanced compilation techniques in the PARADIGM compiler for distributed-memory multicomputers. In *ICS*, pages 424–433, 1995.
[38] A. Turjan, B. Kienhuis, and E. Deprettere. A compile time based approach for solving out-of-order communication in Kahn process networks. In *ASAP*, 2002.
[39] S. Verdoolaege, K. Beyls, M. Bruynooghe, and F. Catthoor. Experiences with enumeration of integer projections of parametric polytopes. Report CW 395, Department of Computer Science, K.U.Leuven, Leuven, Belgium, Oct. 2004.
[40] S. Verdoolaege, R. Seghir, K. Beyls, V. Loechner, and M. Bruynooghe. Analytical computation of Ehrhart polynomials: Enabling more compiler analyses and optimizations. In *CASES*, pages 248–258, 2004.
[41] S. Verdoolaege, K. M. Woods, M. Bruynooghe, and R. Cools. Computation and manipulation of enumerators of integer projections of +parametric polytopes. Report CW 392, Dept. of Computer Science, K.U.Leuven, Leuven, Belgium, 2005.
[42] Y. Zhao and S. Malik. Exact memory size estimation for array computations. *IEEE Transactions on VLSI Systems*, 8(5):517–521, October 2000.