Analytical Computation of Ehrhart Polynomials and its Applications for Embedded Systems

Sven Verdoolaege Kristof Beyls Maurice Bruynooghe
Rachid Seghir Vincent Loechner

Abstract

Many optimization techniques, including several targeted specifically at embedded systems, depend on the ability to calculate the number of points in a parametrized polytope. It is well known that this parametrized count can be represented by an Ehrhart polynomial, which is usually computed through interpolation. In some cases, however, this interpolation fails and in some other cases it can take a very long time to compute. By extending an existing method, based on Barvinok's decomposition, to count the number of points in an non-parameterized polytope we show that we can compute the Ehrhart polynomial *analytically*, solving these problems to a large extent.

1 Introduction

Many optimization techniques involve a substep that counts the number of points in a set S. Typically, this set is a subset of $\mathbb{Z}^d$ that can be described by a set of linear constraints, i.e., $S = \{\mathbf{x} \in \mathbb{Z}^d | A\mathbf{x}+\mathbf{b} \geq 0\}$ or $S = \{\mathbf{x} \in \mathbb{Q}^d | A\mathbf{x}+\mathbf{b} \geq 0\} \cap \mathbb{Z}^d$, i.e., S is the intersection of $\mathbb{Z}^d$ and a *rational polyhedron* (Schrijver 1986). The problem of counting the number of elements in S is therefore equivalent to counting the number of integer points in a rational polyhedron. Usually, this polyhedron is bounded and called a *polytope*, which implies that the count is finite. The extremal points of such a polytope are called its *vertices*. In some cases the linear constraints describing S involve not only the set variables $\mathbf{x}$, but also an independent set of variables $\mathbf{p}$ called the parameters, i.e., $S = \{\mathbf{x} \in \mathbb{Z}^d | A\mathbf{x} + C\mathbf{p} + \mathbf{b} \geq 0\}$. Finding the number of points in S is now equivalent to counting the number of integer points in a *parametrized polytope* and the resulting count will also depend on $\mathbf{p}$ and can be represented by an *Ehrhart polynomial* (Ehrhart 1977). This is the case that interests us here.

The problem of counting the number of points frequently occurs in the context of cache analysis. Malik et al. (1997) mention the CME (Cache Miss Equations) (Ghosh et al. 1999) as one of the characteristics to be considered for analyzing embedded software and solving these CME involves a parametrized counting problem. Chatterjee et al. (2001) propose a technique, which, in contrast to the use of CME, is exact and show an example involving an Ehrhart polynomial with as variable the starting address of an array. Beyls and D'Hollander (2001) propose a technique based on reuse distances using Ehrhart polynomials. Their main application is the generation of cache hints for EPIC architectures (Beyls and D'Hollander 2002). Zhao et al. (2003) use a similar technique based on reuse distances to predict the impact of optimizations for embedded systems. D'Alberto et al. (2001) apparently also use Ehrhart polynomials to obtain efficient use of data caches, without going into too much detail. Kim et al. (2003) use Ehrhart polynomials to count the number of page breaks as part of their analysis of SDRAM energy consumption. Finally, Clauss (1996) was the first to show the applicability of Ehrhart polynomials in computer science and includes an example that extends the techniques of Ferrante et al. (1991) for counting the number of cache lines accessed by a loop to parametrized loops.

Parametrized counting has also found its use outside the domain of cache analysis. Lisper (2003) uses parametrized counting in his WCET (Worst-Case Execution Time) analysis to obtain safe upper bounds of execution times in the context of real-time systems. In his example, he uses the techniques by Pugh (1994), but an actual implementation is likely to use Ehrhart polynomials. Also in the context of real-time systems, Braberman et al. (2003) use Ehrhart polynomials to automatically determine the size of the memory region associated with a scope in function of the arguments of the `java` method that defines the scope. Scoped memory (Bollella and Gosling 2000) is used here to avoid the unpredictable behaviour of regular garbage collection on real-time systems.

In the context of process networks, Rijpkema et al. (1999) use Ehrhart polynomials during the linearization step of their method for converting parametrized `Matlab` programs to process networks. This linearization indicates the order in which tokens are used. In the same context, Turjan et al. (2002) use Ehrhart polynomials to represent their rank function, which returns the number of iteration points of a for-statement executed before a given iteration point. A similar idea is used by Loechner et al. (2002) to perform data layout transformations. They try to order the data according to the order in which it is accessed, which is represented by an Ehrhart polynomial.

It is clear that Ehrhart polynomials are a very useful concept both inside and outside the domain of embedded systems. However, there are some problems associated with the technique used in the current implementation (Clauss and Loechner 1998) to construct Ehrhart polynomials. One of these is the problem of "degenerate domains", which has been explicitly mentioned by Turjan et al. (2002). Basically, the technique simply fails in some cases. In other cases, the computation time can be (sometimes extremely) large. We will show that by extending an existing method, based on Barvinok's decomposition (Barvinok 1993), to compute the number of integer points in a non-parametrized polytope, we can obtain Ehrhart polynomials analytically and solve these problems to a large extent.

In the remaining sections, we will first explain how the previous method worked. Then, we will briefly discuss Barvinok's algorithm in section 3 and explain how this can be used to compute Ehrhart polynomials in section 4. We will then apply this on a realistic example in section 5 and show some further experimental results in section 6. After discussing related work in section 7, we finish off with conclusions and future work.

2 Ehrhart Polynomials

Ehrhart (1977) showed that the number of integer points in the dilatation $pP = \{p\mathbf{x} | \mathbf{x} \in P\}$ of a rational polytope P can be represented by a pseudo-polynomial, aka *Ehrhart polynomial*, $\mathcal{E}(P)$ in p. The degree of $\mathcal{E}(P)$ is less than or equal to the dimension of P. The coefficients of $\mathcal{E}(P)$ are *periodic numbers*, which means that they depend periodically on the parameter p. Basically, a periodic number u_p is a lookup-table with $u_p = \mathtt{u}[p \bmod s]$, where s is called the *period* of u_p. The periods of the coefficients are all less than or equal to the lcm (least common multiple) of the denominators of the vertices of P.

Clauss and Loechner (1998) extended this research to parametrized polytopes with multiple parameters. The number of integer points in a parametrized polytope P with n parameters and of dimension d, is an Ehrhart polynomial of degree at most d in all the parameters, with coefficients that may be periodic in its n parameters, i.e., $u_{\mathbf{p}} = \mathtt{u}[p_1 \bmod s_1][p_2 \bmod s_2] \ldots [p_n \bmod s_n]$. The period s_i of the coefficients for a given parameter p_i is less than or equal to the lcm of denominators of the coefficients in p_i in the description of the vertices of P. An extra complication is that the parameter space may need to be divided into separate regions called *validity domains*, each with an associated Ehrhart polynomial. The reason for this is that some of the vertices only exist for a subset of the possible parameter values.

In their implementation, they use this knowledge about the structure of the Ehrhart polynomials to compute them through interpolation. That is, they calculate the number of integer points in a number of instances of P for fixed values of the parameters $\mathbf{p}$—these instances are then non-parametrized polytopes—and interpolate the Ehrhart polynomial from these *initial countings*. To be able to interpolate a "regular" polynomial of degree d, $d + 1$ instances are needed. To interpolate an Ehrhart polynomial of degree d in n parameters with period $\mathbf{s}$, $\prod_{i=1}^{n}(d + 1)s_i$ instances are needed.

This leads to the first problem of the current implementation: *degenerate domains*. If any validity domain does not contain a subregion with ds_i consecutive values in each dimension, then it may be very difficult or even impossible to find a complete set of appropriate instances for interpolation. Although the current implementation incorporates some heuristics to remedy this problem, it will still fail fairly often if this problem arises. In theory, it might be possible to find a solution for all cases, but it would be rather involved.

The remaining problems are related to the time complexity of their algorithm. First, their method for calculating the number of integer points in non-parametrized polytopes basically enumerates all points (except for the final dimension), so if any of the instances of P contains a large number of points, the computation time will rise accordingly. Second, if the periods are large, then the number of instances will be very large and the interpolation itself will also take a long time. Furthermore, the output size will be relatively large in this case as well.

3 Barvinok's Algorithm

Barvinok and Pommersheim (1999) describe an algorithm for counting the number of integer points in a non-parametrized polytope. This algorithm was later refined by De Loera et al. (2003c), especially with respect to a practical implementation. Although the algorithm is the culmination of research by many authors, a crucial part of the algorithm is Barvinok's decomposition into unimodular cones (Barvinok 1993), which ensures the overall polynomial time complexity for fixed dimensions.

We will not explain the algorithm in detail, but the basic idea is to consider a polynomial $f(P;\mathbf{x})$, called the *generating function*, which is the sum of a number of monomials $\mathbf{x}^{\mathbf{z}} = \prod_i x_i^{z_i}$, one for each integer point $\mathbf{z}$ in the polytope P. The number of points in P is then simply the number of monomials in the generating function, which can be calculated by evaluating the generating function at $\mathbf{1}$, i.e., $\#P = f(P;\mathbf{1})$.

Obviously, the generating function is not constructed by enumerating all points in the polytope. Rather, in can be shown that the generating function of a polytope is equal to the sum of the generating functions of the *supporting cones* at each vertex. The supporting cone at a vertex is the polyhedron defined by the constraints that are saturated by the vertex, i.e., those for which equality holds for the vertex. The generating function of a supporting cone can in turn be expressed as the sum of short rational functions through application of Barvinok's decomposition into unimodular cones.

The final generating function is then also the sum of short rational functions. To evaluate this function, we cannot simply substitute $\mathbf{1}$ for $\mathbf{x}$ since $\mathbf{1}$ is a pole of all the short rational functions, but rather we need to compute the residues of the rational functions. We refer to De Loera et al. (2003c) for a detailed explanation.

Algorithm 1 summarizes the algorithm. One interesting aspect that will be needed later is that in case a vertex $\mathbf{v}_i$ is non-integer, then we need to find a specific integer point in the neighbourhood of $\mathbf{v}_i$. This is encapsulated by the function E in step 1e.

4 Computing Ehrhart Polynomials using Barvinok's Algorithm

The first idea that might come to mind could be to use Barvinok's algorithm to perform the initial countings needed for interpolation. This would make these countings more efficient for large polytopes, solving one of the problems mentioned in section 2. It turns out, however, that, with some modifications, we can use Barvinok's algorithm on parametrized polytopes directly to obtain

Algorithm 1 Barvinok's algorithm

1. For each vertex $\mathbf{v}_i$ of P
   (a) Determine supporting cone cone$(P, \mathbf{v}_i)$
   (b) Let $K = \text{cone}(P, \mathbf{v}_i) - \mathbf{v}_i$
   (c) Decompose $[K]$ into unimodular cones $\sum_j \epsilon_j [K_j]$
   (d) For each K_j
      i. Determine $f(K_j; \mathbf{x})$
   (e) $f(\text{cone}(P, \mathbf{v}_i); \mathbf{x}) = \sum_j \epsilon_j \mathbf{x}^{E(\mathbf{v}_i, K_j)} f(K_j; \mathbf{x})$
2. $f(P; \mathbf{x}) = \sum_i f(\text{cone}(P, \mathbf{v}_i); \mathbf{x})$
3. evaluate $f(P; \mathbf{1})$

the Ehrhart polynomial *analytically*, obviating the need for interpolation and solving *all* of the problems mentioned above.

Algorithm 2 shows the parametrized version of algorithm 1. We will leave a detailed discussion of the algorithm to a forthcoming full paper. We will note, though, that apart from taking into account validity domains, the overall structure of the algorithm remains the same. There are only three substeps which need to take into account the parametrization, viz. the construction of the supporting cones, the construction of the numerators of the rational functions and the computation of the residues. The last two also need to handle periodic numbers.

Since we no longer need to interpolate, it is obvious that the problem of degenerate domains has been removed as well. The reader may then wonder what happens in the cases where the previous implementation would encounter such degenerate domains. To answer this question, we must look at the location where the periodic numbers are introduced, which is the function E in step 1e.

According to De Loera et al. (2003c), the integer point $E(\mathbf{v}_i, K_j)$ we are looking for is of the following form: $\sum_k \lceil \lambda_k \rceil B_{jk}$, where $\boldsymbol{\lambda}$ is the rational solution of $\mathbf{v}_i = \sum_k \lambda_k B_{jk}$, with B_{jk} some set of integer vectors related to K_j that form a basis for $\mathbb{Z}^d$.[1] Generalizing to the parametrized case, we need $E(\mathbf{v}_i(\mathbf{p}), K_j)$ and we have that $\boldsymbol{\lambda}(\mathbf{p})$ is the rational solution of $\mathbf{v}_i(\mathbf{p}) = \sum_k \lambda_k(\mathbf{p}) B_{jk}$ and therefore $E(\mathbf{v}_i(\mathbf{p}), K_j) = \sum_k \lceil \lambda_k(\mathbf{p}) \rceil B_{jk}$ with (Graham et al. 1989)

$$\lceil \lambda_k(\mathbf{p}) \rceil = \lambda_k(\mathbf{p}) + \frac{(-m\lambda_k(\mathbf{p})) \bmod m}{m}. \tag{1}$$

In equation (1), m is the lcm of the denominators of all fractions that appear in any of the λ_k. This equation can be evaluated for a number of fixed values for the parameters to obtain a periodic number where each period s_i is at most m.

[1] In fact B_{jk} are the rays of K_j.

Algorithm 2 Parametrized Barvinok

1. For each vertex $\mathbf{v}_i(\mathbf{p})$ of P
   (a) Determine supporting cone $\mathrm{cone}(P, \mathbf{v}_i(\mathbf{p}))$
   (b) Let $K = \mathrm{cone}(P, \mathbf{v}_i(\mathbf{p})) - \mathbf{v}_i$
   (c) Decompose $[K]$ into unimodular cones $\sum_j \epsilon_j [K_j]$
   (d) For each K_j
      i. Determine $f(K_j; \mathbf{x})$
   (e) $f(\mathrm{cone}(P, \mathbf{v}_i(\mathbf{p})); \mathbf{x}) = \sum_j \epsilon_j \mathbf{x}^{E(\mathbf{v}_i(\mathbf{p}), K_j)} f(K_j; \mathbf{x})$
2. For each validity domain D of P
   (a) $f(P; \mathbf{x}) = \sum_{\mathbf{v}_i \in D} f(\mathrm{cone}(P, \mathbf{v}_i(\mathbf{p})); \mathbf{x})$
   (b) evaluate $f(P; \mathbf{1})$

Unlike the case of interpolation, we do not need to worry here whether the values of the parameters actually belong to the validity domain. To obtain the jth entry of each periodic number for a given parameter n_i, $d + 1$ valid instantiations with $n_i \bmod s_i = j$ are needed for the interpolation. If fewer than $d+1$ such instantiations exist in the validity domain, then the interpolation will fail, whereas our method will produce the correct result. If *no* such instantiation exists in the validity domain, then our method will still produce (meaningless but harmless) entries in the periodic number. They will never be used since the Ehrhart polynomial associated with this validity domain will never be evaluated for a value of the parameter with $n_i \bmod s_i = j$.

This minor inelegance can be removed by using a different representation for the coefficients of the Ehrhart polynomial. Rather than using periodic numbers, we can represent the coefficients by linear expressions in products of the modulo expressions that appear in equation (1). That is, rather than converting the modulo expressions to periodic numbers, we simply propagate them through the remaining calculations. This new representation also solves the third and final problem mentioned in section 2, i.e., the problem of the large periods.

5 Example

As an example we will use the technique of Beyls and D'Hollander (2002) to compute the reuse distance of access `A[i][k]` in the program in figure 1 (matrix multiplication). The reuse distance is the number of distinct "elements" that are accessed between two subsequent accesses to the same array element. In this example we will count the number of distinct TLB pages.

Iterations (i, j, k) and $(i, j + 1, k)$ access the same array element `A[i][k]`. We want to count the number of distinct TLB pages accessed between these

```
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
```

Figure 1: Matrix multiplication

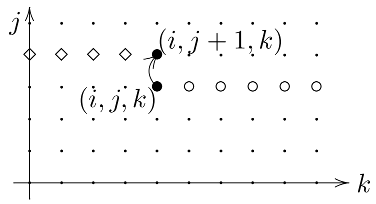

Figure 2: Intermediate accesses

two accesses. For simplicity, we will 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 layed 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 $4 \times (200k + i)$.

Figure 2 shows the iterations that are executed between (i, j, k) and $(i, j+1, k)$: iterations $(i, j, k+1 \ldots 199)$ ($\circ$ 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 follows

$$S_1 = \left\{ p | \exists k' : p = \left\lfloor \frac{800k' + 4i}{L} \right\rfloor \wedge 0 \leq i, j, k \leq 199 \wedge k + 1 \leq k' \leq 199 \right\},$$

where i, j and k are parameters. This can be written as a set of linear constraints:

$$S_1 = \{p | \exists k' : 1024p <= 200k' + i <= 1024p + 1023 \\ \wedge 0 <= i, j, k <= 199 \wedge k + 1 <= k' <= 199)\},$$

where we assume page size $L = 4096$, and simplified to (e.g., using Omega; Kelly et al. 1996)

$$S_1 = \{p | 0, 1024p - 39800 <= i <= 199 \wedge 0 <= k <= 198 \\ \wedge 0 <= j <= 199 \wedge i + 200k <= 824 + 1024p\}.$$

Table 1: Experimental Results

	#VD	#DD	interpolation	analytical	modulo
e16	4	0	16.076s	0.775s	0.622s
isnm	2	0	5.153s	0.074s	0.064s
g14	6	2	0.771s	0.194s	0.194s
RD1	2	1	2m31.524s	0.210s	0.070s
RD2	1	0	26.920s	7.120s	0.040s
CME	5	?	∞	∞	2.205s

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

$$\frac{i}{1024} + 25\frac{k}{128} - \frac{103}{128} \qquad \text{and} \qquad \frac{i}{1024} + \frac{4975}{128}.$$

This means we can expect the coefficients of the one-dimensional Ehrhart polynomial to be periodic numbers with period 1024 in i and period 128 in k and this is indeed the result of the interpolation method and the analytical method with periodic number representation. Using the modulo representation, however, we obtain the simpler

$$-\frac{25}{128}k - \frac{(i + 888) \bmod 1024}{1024} + \frac{(i + 200k + 199) \bmod 1024}{1024} + \frac{40625}{1024}.$$

As we will show in the next section, the computation time to obtain this result is also much lower than that needed by either of the methods that use periodic numbers.

6 Experimental Results

We have implemented the algorithm discussed in this paper and we have obtained some preliminary results. Selected results comparing the previous implementation to our own are shown in table 1. The first column shows the number of validity domains, the second the number of degenerate domains (only relevant for the interpolation method) and the remaining columns show the computation times for the interpolation method, the analytical method with periodic number representation and the analytical method with modulo representation. For all three methods, computations were performed in exact long integer arithmetic using `GMP`.

The first three polytopes can be found in the PolyLib distribution (Loechner 1999). The first two contain some moderate-sized periodic numbers. The analytical method is clearly faster than interpolation and the modulo representation is also slightly faster than the periodic number representation. The third polytope results in a few degenerate domains for the interpolation method. Again the analytical method is faster even though it produces more results than the interpolation method. The next two polytopes appear in the context of reuse distances. The speed improvements are even more dramatic in these cases. RD2 is the example discussed in the previous section. The final polytope is based on the CME. For this polytope, both the interpolation method and the analytical method with periodic number representation had to be aborted because they slowly used up all available memory. The experiments were performed on an Athlon MP 1500+ with 512MiB internal memory.

7 Related Work

Two methods are often cited for counting the number of points in a parametrized polytope: Clauss and Loechner (1998) and Pugh (1994). We have already highlighted the main differences between our technique and the first of these two, i.e., the use of an extended version of Barvinok's algorithm instead of interpolation. Other than that, the two techniques are very similar. For example, in our implementation we have reused their method of subdividing the parameter space in validity domains (Loechner and Wilde 1997).

The technique of Pugh (1994) consists of a set of simplification rules and the application of a set of standard summation formulas for some base cases. If he cannot find an explicit formula, he resorts to "splintering" the polytope. Since he mentions neither the use of periodic numbers or modulo expressions, this splintering is likely to be compute intensive and to produce complicated results. In contrast to our technique and that of Clauss and Loechner (1998), his technique does not appear to have been implemented yet.

Our implementation of Barvinok's algorithm draws heavily from the description by De Loera et al. (2003c) of the non-parametrized version. De Loera et al. (2003a) extended the algorithm to what they call the homogenized Barvinok algorithm, which allows them to compute Ehrhart *series*. An Ehrhart series is a formal power series that is closely related to an Ehrhart polynomial; they are in fact polynomially intercovertible. 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. It does not appear to be as directly usable in our context, but it has interesting mathematical properties. They apparently only handle simple dilatations with a single parameter. The source of their `LattE` tool (De Loera et al. 2003b) is now also available.

8 Conclusions and Future Work

We have presented a new method for calculating Ehrhart polynomials that has significant advantages over the previous implementation. First, by using an extended version of Barvinok's algorithm, we can dispense with the use of interpolation, removing the problem of degenerate domains. Second, by using an alternative representation for Ehrhart polynomials using modulo expressions, we can produce a more compact output in case of large periods in a significantly reduced time. We assume that the computation time inherits the polynomial execution time (for fixed dimensions) behaviour from Barvinok's algorithm, but we still need to proof this formally.

We have noticed, however, that the output of our current implementation can sometimes be further simplified, so for future work, we will be investigating methods for automatically simplifying modulo expressions. Another point of future work is the extension to counting the number of point in integer projections of parametrized polytopes. This problem occurs when the description of the set whose elements we want to count contains existential variables. We currently do not handle this situation.

The problem of counting the number of points in an integer projection has been investigated by Clauss (1997), but as far as we know his proposal has not been implemented yet and may not be very efficient. Seghir (2002) describes a technique that works for projecting out a single dimension, i.e., it would work for a description with a single existential variable. It has been implemented as part of PolyLib, but it is not clear how easy it would to extend the technique to multiple existential variables. Possibly the most promising approach is that of Barvinok and Woods (2002), who work directly on generating functions.

References

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

Barvinok, A. and K. Woods (2002, November). Short rational generating functions for lattice point problems. http://arxiv.org/abs/math.CO/0211146.

Barvinok, A. I. (1993, November). A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. In *34th Annual Symposium on Foundations of Computer Science*, pp. 566–572. IEEE.

Beyls, K. and E. D'Hollander (2001). Reuse distance as a metric for cache behavior. In *IASTED conference on Parallel and Distributed Computing and Systems 2001 (PDCS01)*, pp. 617–662.

Beyls, K. and E. D'Hollander (2002, November). Compile-time cache hint generation for EPIC architectures. In *2nd Workshop on Explicitly Parallel Instruction Computing Architecture and Compilers (EPIC-2)*.

Bollella, G. and J. Gosling (2000). The real-time specification for Java. *Computer 33*(6), 47–54.

Braberman, V., D. Garbervetsky, and S. Yovine (2003, October). On synthesizing parametric specifications of dynamic memory utilization. Technical report.

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

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

Clauss, P. (1997). Handling memory cache policy with integer points counting. In *European Conference on Parallel Processing*, pp. 285–293.

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

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

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

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

De Loera, J. A., R. Hemmecke, J. Tauzer, and R. Yoshida (2003c, March). Effective lattice point counting in rational convex polytopes. http://www.math.ucdavis.edu/˜latte/theory.html.

Ehrhart, E. (1977). Polynomes arithmétiques et méthode des polyèdres en combinatoire. *International Series of Numerical Mathematics 35*.

Ferrante, J., V. Sarkar, and W. Thrash (1991, August). On estimating and enhancing cache effectiveness. In U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua (Eds.), *Lecture Notes in Computer Science*, Volume 589, pp. 328–343. Springer-Verlag.

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

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

Graham, R. L., D. E. Knuth, and O. Patashnik (1989). *Concrete Mathematics*. Addison-Wesley.

Kelly, W., V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott (1996, November). The Omega calculator and library. Technical report, University of Maryland.

Kim, H. S., N. Vijaykrishnan, M. Kandemir, E. Brockmeyer, F. Catthoor, and M. J. Irwin (2003). Estimating influence of data layout optimizations on SDRAM energy consumption. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pp. 40–43. ACM Press.

Lisper, B. (2003, April). Fully automatic, parametric worst-case execution time analysis. MRTC report, Dept. of Computer Science and Engineering, Mälardalen University.
`http://www.mrtc.mdh.se/publ.php3?id=0531`.

Loechner, V. (1999, March). Polylib: A library for manipulating parameterized polyhedra. Technical report, ICPS, Université Louis Pasteur de Strasbourg, France.

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

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

Malik, S., M. Martonosi, and Y.-T. S. Li (1997). Static timing analysis of embedded software. In *Proceedings of the 34th annual Conference on Design Automation Conference*, pp. 147–152. ACM Press.

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

Rijpkema, E., E. Deprettere, and B. Kienhuis (1999). Compilation from matlab to process networks. In *Second International Workshop on Compiler and Architecture Support for Embedded Systems (CASES'99)*.

Schrijver, A. (1986). *Theory of linear and integer programming*. John Wiley & Sons.

Seghir, R. (2002, June). Dénombrement des point entiers de l'union et de l'image des polyédres paramétrés. Master's thesis, ICPS, Université Louis Pasteur de Strasbourg, France.

Turjan, A., B. Kienhuis, and E. Deprettere (2002, July). 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)*.

Zhao, M., B. Childers, and M. L. Soffa (2003). Predicting the impact of optimizations for embedded systems. In *Proceedings of the 2003 ACM SIGPLAN conference on Language, Compiler, and Tool for Embedded Systems*, pp. 1–11. ACM Press.