CONCURRENCY: PRACTICE AND EXPERIENCE, VOL. 8(6), 445–460 (JULY–AUGUST 1996)

Scanning parameterized polyhedron using Fourier–Motzkin elimination

MARC LE FUR
IRISA, Campus de Beaulieu
35042 Rennes Cedex
France

SUMMARY

The paper presents two algorithms for computing a control structure whose execution enumerates the integer vectors of a parameterized polyhedron defined in a given context. Both algorithms reconsider the *successive projection method*, based on Fourier–Motzkin pairwise elimination, defined by Ancourt and Irigoin. The way redundant constraints are removed in their algorithm is revisited in order to improve the computation time for the enumeration code of higher order polyhedrons as well as their execution time. The algorithms presented here are at the root of the code generation in the HPF compiler PANDORE developed at IRISA, France; a comparison of these algorithms with the one defined by Ancourt and Irigoin is given in the class of polyhedrons manipulated by the PANDORE compiler.

1. INTRODUCTION

The polyhedron scanning problem (PSP) consists of computing a control structure whose execution enumerates the integer vectors of a polyhedron defined by a set of affine constraints. The motivations for studying the PSP can be found in parallelization and loop transformation [1] as well as in the compilation of regular loops — loops with affine bounds and array subscripts — for hierarchical shared memory parallel computers [2] or for distributed memory parallel computers[3,4,5]. Usual loop transformations (reversal, permutation, skewing) can be seen as the application of an unimodular linear transformation to a polyhedral iteration space; the resulting space is still a polyhedron whose integer vectors can be enumerated in the lexicographic order by a perfectly nested loop. As regards the compilation of regular loops for hierarchical shared memory or for distributed memory parallel computers, the computation codes and the data exchange codes (between the local cache and the shared memory and between the distributed memories respectively) can be modelled as polyhedrons whose scanning codes are at the root of the code generated on each processor.

With the compilation of HPF regular loops on parallel computers, the algorithms designed for solving the PSP deal with polyhedrons that turn out to be more complex in terms of dimension and number of constraints. This complexity obviously depends on the compilation scheme but is mainly due to HPF data distribution directives that find their expression in new variables and therefore new inequalities in the polyhedrons that found code generation. The problem is more delicate in the case of the generation of the communication code — related to a given right-hand side reference of an assignment — since both the distribution of the right-hand side reference and that of the left-hand side are translated into the polyhedron associated with the communication code. For instance, the static analysis shown in [3] relies on a polyhedron defined by a system with six variables, two equalities

CCC 1040–3108/96/060445–16 Received November 1995

and 12 inequalities in order to produce the communication code for the following parallel loop:

```
REAL X(100,100), Y(100,100)

!HPF$ PROCESSORS P(5)
!HPF$ DISTRIBUTE X(CYCLIC(10),*) ONTO P
!HPF$ DISTRIBUTE Y(BLOCK,*) ONTO P

DO I = 1, 100
  DO J = 1, 100
    X(I,J) = Y(J,I)
  END DO
END DO
```

The complexity of the polyhedrons involved in the compilation process can be considerably increased if array distributions are specified by way of alignment on a template. For the parallel nested loop given in [5] for instance:

```
REAL X(0:42,0:42), Y(0:42,0:42)

!HPF$ TEMPLATE T(0:150,0:150)
!HPF$ ALIGN X(I,J) WITH T(3*I,3*J)
!HPF$ ALIGN Y WITH X
!HPF$ PROCESSORS P(0:3,0:3)
!HPF$ DISTRIBUTE T(CYCLIC(4),CYCLIC(4)) ONTO P

DO I = 1, 14
  DO J = 0, 14
    X(3*I,3*J) = Y(3*I,3*J) + Y(3*I-1,3*J)
  END DO
END DO
```

the communication code shown in [5] is based on polyhedrons with eight variables, ten equalities and 32 constraints. So, mature HPF compilers will have to incorporate efficient algorithms for solving the PSP.

The most commonly used method for solving the PSP is based on Fourier–Motzkin pairwise elimination[6,7]. Ancourt and Irigoin first designed an algorithm that used the pairwise elimination [8]; it consists of a series of projections of the polyhedron along the different axes, followed by an elimination of the redundant inequalities introduced by the projections. Two other solutions to the PSP have been proposed; their common characteristic lies in the fact they do not generate redundant constraints. Feautrier suggests the use of a parameterized simplex (PIP: parametric integer programming) to compute the lower and upper bounds in each dimension of the polyhedron [9]. The use of PIP is complemented by a simplification of the bounds generated [10]. The second technique, shown in [11], is based on Chernikova's algorithm [12]. The method also proceeds with projections but relies on the computation of both the constraints and the rays/vertices of the polyhedron.

This paper reconsiders the *successive projection method* described by Ancourt and Irigoin and lays emphasis on the removal of redundant inequalities. The first aim is to improve the computation time for the enumeration codes so that the successive projection method can be applied on polyhedrons of higher dimension and of higher complexity, such as those encountered in the compilation of HPF loops for parallel computers. The second aim is to reduce the execution time of the scanning codes. Two algorithms are presented in this paper. The first one reviews the second phase in the Ancourt–Irigoin algorithm, whereas the second proposes to interlace projections and redundant constraint eliminations to make it possible to generate an enumeration code when the previous algorithms turn out to be inapplicable.

The organization of the paper is as follows. Section 2. reviews the definition of pairwise elimination and its main properties. The elimination of redundant constraints in a system is discussed in Section 3.. Then we recall Ancourt and Irigoin's algorithm and present the two improvements implemented in the PANDORE compiler; a comparison of the three methods is shown. Finally, an extension of these algorithms to parameterized polyhedrons is presented in Section 5..

2. DEFINITIONS AND PROPERTIES

The following definitions and properties are given in the $\mathbb{Q}^n$ space even though they remain valid in $\mathbb{R}^n$.

Let $S = A\,(x_1 \dots x_n)^T + b \geq 0$ be a system of affine constraints (A is an integer matrix and b is an integer column-vector) and let $P = \{x = (x_1 \dots x_n)^T \;/\; Ax + b \geq 0\}$ be the polyhedron associated with S. Fourier–Motzkin pairwise elimination[6,7] aims at computing the projection of the polyhedron P along a given x_q axis. Let us denote P_{x_q} the resulting polyhedron and S_{x_q} the system of inequalities defining P_{x_q}:

$$S_{x_q} = Nil(x_q) \cup Elim(x_q, Min(x_q), Max(x_q))$$

in which $(Min(x_q), Max(x_q), Nil(x_q))$ is the partition of S:

- $Min(x_q) = \{c_i x_q + f_i((x_r)_{r \neq q}) \geq 0\}_{i \in I}$, where the f_is are affine functions and $\forall i \in I \;\; c_i > 0$; the inequalities in $Min(x_q)$ are termed *minimizing constraints* for x_q
- $Max(x_q) = \{c_j x_q + f_j((x_r)_{r \neq q}) \geq 0\}_{j \in J}$, where $\forall j \in J \;\; c_j < 0$: the set of *maximizing constraints* for x_q
- $Nil(x_q) = \{f_k((x_r)_{r \neq q}) \geq 0\}_{k \in K}$: the set of constraints in S where the coefficient of x_q is zero

and where $Elim(x_q, Min(x_q), Max(x_q))$ stands for the set of pairwise eliminations of the variable x_q for each pair of inequalities in $Min(x_q) \times Max(x_q)$:

$$Elim(x_q, Min(x_q), Max(x_q)) = \begin{cases} \emptyset \text{ if } |I|.|J| = 0 \\ \{-c_j f_i((x_r)_{r \neq q}) + c_i f_j((x_r)_{r \neq q}) \geq 0\}_{(i,j) \in I \times J} & \text{otherwise} \end{cases}$$

Figure 1 illustrates a variable removal in a set of inequalities with the Fourier–Motzkin elimination.

The projection of a polyhedron using a pairwise elimination is known for its strong spatial complexity: an elimination can produce $(m/2)^2$ inequalities from a system containing m constraints. More generally, the projection of a polyhedron along l axes can produce $m^{2^l}/2^{2^{l+1}-2}$ constraints. So the projection process with a pairwise elimination must be

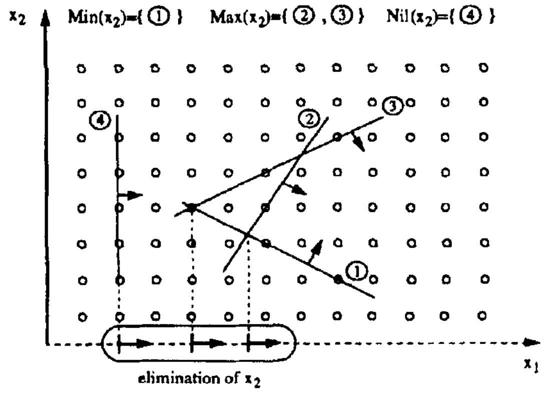

Figure 1. Variable elimination in a set of constraints

used with care. Fourier–Motzkin elimination has other remarkable properties ($E_1 \uplus E_2$ denotes the disjoint union of the sets E_1 and E_2):

Property 1 $\tilde{x} = (\tilde{x_1} \ldots \tilde{x_n})$ *satisfies* S *iff* $\tilde{x}$ *satisfies* $S_{x_q} \uplus Min(x_q) \uplus Max(x_q)$.

Property 2 *S is feasible iff all the constraints $e_i \geq 0$ ($e_i \in \mathbb{Z}$) resulting from the elimination of all the variables in S are obviously feasible, i.e. such that $e_i \in \mathbb{N}$.*

Finally, it can be observed that most of the constraints computed during the pairwise elimination process are redundant (implicit) even though the system in which the variable is removed is non-redundant. Figure 2 illustrates this property. The removal of redundant inequalities is a crucial problem when using the pairwise elimination to project a polyhedron along an axis.

3. ELIMINATING REDUNDANT INEQUALITIES

In order to synthesize the code enumerating the integer vectors of a polyhedron, it is necessary to define the notion of a non-redundant subset stemming from the system $S = Ax + b \geq 0$ in a given context $C = Mx + h \geq 0$. This notion is specified in Section 3.1.. The computation of this non-redundant subset is based on a unitary redundancy test briefly presented in Section 3.2..

3.1. Non-Redundant Subset in a Given Context

The non-redundant subset of $S = Ax + b \geq 0$ in the context $C = Mx + h \geq 0$ (S and C are assumed to be disjoint) computed by the algorithm given in Figure 3 is termed here as *elimination of the redundant constraints of S in the context C*. The parameter S_nred in the function *elim_red_ctxt_acc* acts as an accumulator; it represents the non-redundant subset extracted from the inequalities already inspected in S. S_red stands for the constraints in S not yet considered. The *elimination of almost all the redundant constraints of S in the context C* denotes the result of the algorithm in Figure 3 in which we keep at least one constraint of S (if $S\_red = \{ax + \beta \geq 0\}$ and $S\_nred = \emptyset$, the result is $\{ax + \beta \geq 0\}$).

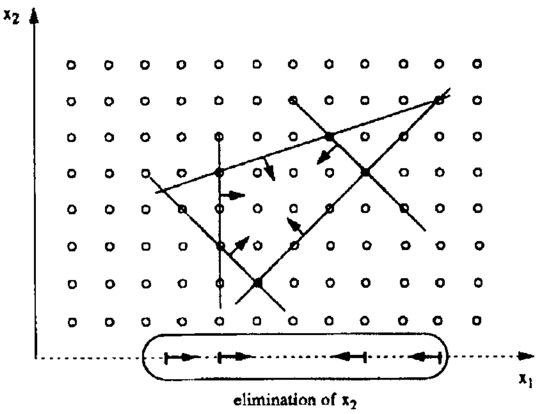

Figure 2. Pairwise elimination generates redundant constraints

elim_red_ctxt(S, C) = elim_red_ctxt_acc($\emptyset$, S, C)

elim_red_ctxt_acc(S_nred, S_red, C) =
  if $S\_red = \emptyset$
  then S_nred
  else **let** $ax + \beta \geq 0$ be a constraint of S_red **and** $S\_red' = S\_red - \{ax + \beta \geq 0\}$
    in **if** $ax + \beta \geq 0$ is redundant in $S\_nred \uplus S\_red' \uplus C$
      then elim_red_ctxt_acc(S_nred, S_red', C)
      else elim_red_ctxt_acc($S\_nred \uplus \{ax + \beta \geq 0\}$, S_red', C)

Figure 3. Non-redundant subset in a given context

Note that the non-redundant subset computed by the algorithm is not unique and depends on the order in which the constraints of S are visited; indeed, the removal of a redundant inequality $ax + \beta \geq 0$ in a system S can make other redundant constraints in S non-redundant in $S - \{ax + \beta \geq 0\}$[6]. For the set of inequalities in Figure 4, for instance, the elimination of the redundant constraints (in the context $\emptyset$) returns the set $\{2,3,4\}$ if the inequalities are visited in the order 1,2,3,4 and returns $\{1,3,4\}$ in the order 2,1,3,4.

3.2. Redundancy test used in Pandore

The problem we want to solve here is to decide whether a constraint $ax + \beta \geq 0$ in a system $S = Ax + b \geq 0$ is redundant in S; that is, whether $\forall \tilde{x} \in \mathbf{Z}^n \; A'\tilde{x} + b' \geq 0 \Longrightarrow a\tilde{x} + \beta \geq 0$, where $A'x + b' \geq 0$ stands for the system $S - \{ax + \beta \geq 0\}$.

From the following criterion[13]: $ax + \beta \geq 0$ is redundant in $Ax + b \geq 0$ iff $A'x + b' \geq 0 \cup \{ax + \beta < 0\}$ is not feasible, we can define a redundancy test which is non-exact in $\mathbf{Z}^n$: in $\mathbf{Z}^n$, $ax + \beta \geq 0$ is redundant in $Ax + b \geq 0$ if $A'x + b' \geq 0 \cup \{ax + \beta \leq -1\}$ is not feasible in $\mathcal{Q}^n$.

The feasibility problem underlying this redundancy test can be implemented using Fourier–Motzkin elimination according to property 2. This method[8] turns out to be viable

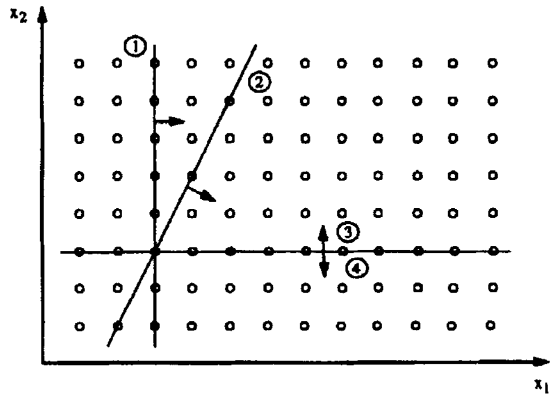

Figure 4. A redundant set of constraints

for small dimensions but is inapplicable to the problems encountered in the compilation of HPF loops for parallel computers. To solve this feasibility problem, the first phase of the *simplex method* [14,6] is better suited. The simplex method aims at solving the linear program $min\{cx+\delta \mathbin{/} Ax+b \geq 0\}$ (or $max\{cx+\delta \mathbin{/} Ax+b \geq 0\}$) in rational numbers. It comprises two phases: phase I consists in determining a feasible solution to $Ax+b \geq 0$ and phase II in finding an optimal solution. These two parts are solved by only one algorithm: the *simplex algorithm*. The worst-case behavior of the algorithm has proven exponential on dummy examples but its average behavior is known to be polynomial. Furthermore, the algorithm uses a constant amount of memory (the simplex algorithm successively applies pivoting operations on the *coefficient matrix* associated with the linear program), unlike the Fourier–Motzkin pairwise elimination.

4. SCANNING NON-PARAMETERIZED POLYHEDRONS

Let $P = \{x = (x_1 \ldots x_n)^T \mathbin{/} Ax + b \geq 0\}$ (where A is an integer matrix and b an integer vector) be the polyhedron whose scanning code must be computed and let $S = Ax + b \geq 0$ be the system of inequalities associated with P.

4.1. Ancourt–Irigoin's algorithm

The algorithm described in [8] proceeds in two main phases.

Phase I: Computation of a system equivalent to S resulting from the iterative application of pairwise elimination in property 1 (for $x_n, x_{n-1}, \ldots, x_2$ in this order):

$$S' = Min(x_1) \uplus Max(x_1) \biguplus \ldots \biguplus Min(x_n) \uplus Max(x_n)$$

Phase II: Elimination of redundant inequalities in S'. This phase produces a system equivalent to S' (and thus to S):

$$S'' = Min'(x_1) \uplus Max'(x_1) \biguplus \ldots \biguplus Min'(x_n) \uplus Max'(x_n)$$

where $Min'(x_i)$ (resp. $Max'(x_i)$) is the system of constraints stemming from the removal of redundant inequalities in $Min(x_i)$ (resp. $Max(x_i)$). The computation of S'' starts with the elimination of redundant constraints in $Min(x_n)$ and $Max(x_n)$ then in $Min(x_{n-1})$ and $Max(x_{n-1})$ and so on, up to the simplification of the sets $Min(x_1)$ and $Max(x_1)$. Let S'_nred_i be the system $Min'(x_n) \uplus Max'(x_n) \uplus \ldots \uplus Min'(x_{i+1}) \uplus Max'(x_{i+1})$ resulting from the elimination of redundant inequalities in $Min(x_n) \uplus Max(x_n) \uplus \ldots \uplus Min(x_{i+1}) \uplus Max(x_{i+1})$ and let S'_red_i be the subset of S' $Min(x_{i-1}) \uplus Max(x_{i-1}) \uplus \ldots \uplus Min(x_1) \uplus Max(x_1)$ composed of the constraints of S' not yet considered. For $i = n, n-1, \ldots, 2$ in this order, the sets $Min'(x_i)$ and $Max'(x_i)$ are computed sequentially in the following way:

(a) $Min'(x_i)$ = elimination of almost all the redundant constraints of $Min(x_i)$ in the context $S'\_nred_i \uplus S'\_red_i \uplus Max(x_i)$

(b) $Max'(x_i)$ = elimination of almost all the redundant constraints of $Max(x_i)$ in the context $S'\_nred_i \uplus S'\_red_i \uplus Min'(x_i)$

or

(a) $Max'(x_i)$ = elimination of almost all the redundant constraints of $Max(x_i)$ in the context $S'\_nred_i \uplus S'\_red_i \uplus Min(x_i)$

(b) $Min'(x_i)$ = elimination of almost all the redundant constraints of $Min(x_i)$ in the context $S'\_nred_i \uplus S'\_red_i \uplus Max'(x_i)$.

The sequence selected depends on whether the elimination of (almost all) the redundant inequalities of $Min(x_i)$ is preferred before those of $Max(x_i)$ or not. According to the observation formulated in Section 3.1., one can note that the two previous sequences do not lead to the same result. The sets $Min(x_1)$ and $Max(x_1)$ being of the form $Min(x_1) = \{c_j x_1 - \alpha_j \geq 0\}_{j \in J}$ and $Max(x_1) = \{-c_k x_1 + \alpha_k \geq 0\}_{k \in K}$ ($c_j, c_k > 0$), the computation of the singletons $Min'(x_1)$ and $Max'(x_1)$ is more trivial: $Min'(x_1) = \{c_{j_0} x_1 - \alpha_{j_0} \geq 0\}$, $Max'(x_1) = \{-c_{k_0} x_1 + \alpha_{k_0} \geq 0\}$ with $\alpha_{j_0}/c_{j_0} = max\{\alpha_j/c_j \ / \ j \in J\}$ and $\alpha_{k_0}/c_{k_0} = min\{\alpha_k/c_k \ / \ k \in K\}$.

4.2. Extracting Bounds from Constraints

The last step of the algorithm is common to most of the methods used to synthesize the scanning code of a polyhedron. It performs an extraction of the lower and upper bounds Low_i and Upp_i from the sets of inequalities $Min'(x_i) = \{c_q x_i + f_q(x_1, \ldots, x_{i-1}) \geq 0\}_{q \in Q_i}$ and $Max'(x_i) = \{c_r x_i + f_r(x_1, \ldots, x_{i-1}) \geq 0\}_{r \in R_i}$. This Section shows how these bounds are determined in the PANDORE compiler.

In his thesis[15], Irigoin gives the following equivalences: $\alpha\beta + \gamma \geq 0 \iff \beta \geq \mathbf{div}(-\gamma + \alpha - 1, \alpha)$ and $-\alpha\beta + \gamma \geq 0 \iff \beta \leq \mathbf{div}(\gamma, \alpha)$ where α is a positive integer. According to these equivalences, the sets of lower and upper integer bounds for each x_i can be deduced: $Low_i = \{div(-f_q(x_1, \ldots, x_{i-1}) + c_q - 1, c_q)\}_{q \in Q_i}$ and $Upp_i = \{div(f_r(x_1, \ldots, x_{i-1}), -c_r)\}_{r \in R_i}$. Thus, the control structure enumerating the integer vectors of P is given by the perfectly nested loop

for $x_1 = $ **max** Low_1, **min** Upp_1

$\ddots$

for $x_n = $ **max** Low_n, **min** Upp_n

In order to reduce the size of the coefficients in the constraints stemming from a pairwise elimination, and in the end to produce the simplest possible bounds, each inequality $\sum_i a_i x_i + \beta \geq 0$ produced by an elimination is replaced by the equivalent constraint (in terms of integer vectors) $\sum_i (a_i/g)x_i + div(\beta - g + 1, g) \geq 0$ in which g denotes the *gcd* of the a_is (this simplification is a consequence of the equivalences described above).

4.3. First Improvement: the Top-Down Algorithm

A first reading of Ancourt–Irigoin's algorithm shows that the removal of redundant inequalities in $Min(x_i)$ and $Max(x_i)$ allows for the constraints in S'_nred_i, that is, the inequalities associated with the innermost loop indices. Depending on the nature of pairwise elimination itself — most of the minimizing and maximizing constraints for x_i are positive combinations and thus consequences of inequalities in $Min(x_n) \uplus Max(x_n) \biguplus \ldots \biguplus Min(x_{i+1}) \uplus Max(x_{i+1})$ — this leads to the removal of most of the redundant constraints in $Min(x_i)$ and $Max(x_i)$, which simplifies the loop bounds of x_i accordingly. On the other hand, the method removes minimizing and maximizing constraints for x_i which are non-redundant in the context S'_red_i associated with the outer loop indices. This finds its expression in the generation of loops containing *holes*, that is loops

$$\begin{array}{l} \textbf{for } x_1 = \alpha_1, \beta_1 \\ \quad \ddots \\ \qquad \textbf{for } x_n = \alpha_n, \beta_n \end{array}$$

in which $\exists\ i \in 1..n\ \exists\ (x_1 \ldots x_{i-1})\ \alpha_1 \leq x_1 \leq \beta_1\ \ldots\ \alpha_{i-1} \leq x_{i-1} \leq \beta_{i-1}$ such that $\alpha_i > \beta_i$. Henceforth, such a loop will be termed *a loop containing a hole located at depth* x_i. For instance, let us consider the polyhedron possessing 451800 integer vectors defined by the system

$$\left\{ \begin{array}{lll} i & -i+7 & j \\ -j+13 & k & -k+19 \\ l-1 & -l+300 & m-1 \\ 10*l-m+1 & -500*i+5*l & 500*i-5*l+499 \\ -300*j+l+m & 300*j-l-m+299 & -l-m+3999 \\ -200*k+2*l+m & 200*k-2*l-m+199 & \end{array} \right. \geq 0$$

which comprises five variables i, j, k, l, m and 17 inequalities. For this polyhedron, Ancourt–Irigoin's algorithm produces the loop

```
for i = 0 , 3
  for j = div(i-5,3) , div(-i+43,3)
    for k = div(3*j,11) , 19
      for l = max(100*i,1) , min(100*i+99,300)
        for m = max(1,300*j-1,200*k-2*l) ,
                min(10*l+1,300*j-l+299,200*k-2*l+199
```

which contains 88108 holes, all located at depth m, with the result that many useless bound computations are performed at run time.

The second observation formulated on Ancourt–Irigoin's algorithm concerns the order in which the constraints of the system S' stemmed from phase I are simplified. The elimination of redundant constraints is performed *bottom-up*, that is, from the minimizing and maximizing constraints for x_n up to those computed for x_1. By definition, the system S'_red_i

associated with the loops surrounding x_i comprises many inequalities, hence making the simplification of the minimizing and maximizing constraints for x_i very costly.

These two remarks lead to the *top-down* algorithm that improves the second phase in Ancourt–Irigoin's algorithm. The phase still produces a system equivalent to S' and thus to S

$$S''' = Min''(x_1) \uplus Max''(x_1) \biguplus \ldots \biguplus Min''(x_n) \uplus Max''(x_n)$$

where $Min''(x_i)$ and $Max''(x_i)$ are the sets of minimizing and maximizing constraints for x_i coming from the elimination of redundant inequalities in $Min(x_i)$ and $Max(x_i)$ respectively. First, the sets $Min(x_1)$ and $Max(x_1)$ are simplified, as shown in section 4.1., to yield $Min''(x_1)$ and $Max''(x_1)$. Then, for $i = 2,3,\ldots,n$ in this order, the sets $Min''(x_i)$ and $Max''(x_i)$ are computed independently as follows:

- $Min''(x_i)$ = elimination of the redundant constraints of $Min(x_i)$ in the context $Min''(x_1) \uplus Max''(x_1) \biguplus \ldots \biguplus Min''(x_{i-1}) \uplus Max''(x_{i-1})$
- $Max''(x_i)$ = elimination of the redundant constraints of $Max(x_i)$ in the context $Min''(x_1) \uplus Max''(x_1) \biguplus \ldots \biguplus Min''(x_{i-1}) \uplus Max''(x_{i-1})$.

4.4. Second Improvement: the Interlaced Algorithm

Due to the properties of pairwise elimination, the system S' produced in phase I of Ancourt–Irigoin's algorithm may contain many inequalities. This can make the elimination of redundant constraints in phase II excessively complex or even impossible if S' does not fit in the memory; for the system presented in Section 4.3., for instance, S' is composed of 324 constraints. This observation justifies the last algorithm, called *top-down algorithm*, incorporated in the PANDORE compiler: the projections along an axis and the eliminations of redundant inequalities are interlaced in order to avoid the possible combinational explosion of the number of constraints in phase I. As in the top-down algorithm, the elimination of redundant minimizing or maximizing inequalities for x_i does not allow for the constraints associated with the innermost loop indices. Figure 5 describes the computation of a system $S'''' = eq\_sys(S,\ n)$ equivalent to S of the form

$$S'''' = Min'''(x_1) \uplus Max'''(x_1) \biguplus \ldots \biguplus Min'''(x_n) \uplus Max'''(x_n)$$

in which $Min'''(x_i)$ (resp. $Max'''(x_i)$) is a set of minimizing (resp. maximizing) constraints for x_i that will provide the bounds in the nested loop scanning the polyhedron.

As in Ancourt–Irigoin's algorithm, one can notice that the order in which the sets Min'''_x_i and Max'''_x_i are computed can be reversed in the function *eq_sys* (the two sequences do not lead to the same result):

(a) Max'''_x_i = elim_red_ctxt($Max\_x_i,\ Min\_x_i \uplus Nil\_x_i$)

(b) Min'''_x_i = elim_red_ctxt($Min\_x_i,\ Max'''\_x_i \uplus Nil\_x_i$).

4.5. Comparison of the Algorithms

In order to compare the three algorithms in the class of polyhedrons described in [3], we have implemented Ancourt–Irigoin's algorithm in the interpreted functional language CAML

eq_sys(S, i) =
let $Min\_x_i \uplus Max\_x_i \uplus Nil\_x_i$ be the partition of S
in if $i = 1$
then $Min'''\_x_1 \uplus Max'''\_x_1$
where $Min'''\_x_1 = \{c_{j_0}x_1 - \alpha_{j_0} \geq 0\}$ with $\alpha_{j_0}/c_{j_0} = max\{\alpha_j/c_j \ / \ j \in J\}$
if $Min\_x_i = \{c_j x_1 - \alpha_j \geq 0\}_{j \in J}$ $(c_j > 0)$
and $Max'''\_x_1 = \{-c_{k_0}x_1 + \alpha_{k_0} \geq 0\}$ with $\alpha_{k_0}/c_{k_0} = min\{\alpha_k/c_k \ / \ k \in K\}$
if $Max\_x_i = \{-c_k x_1 + \alpha_k \geq 0\}_{k \in K}$ $(c_k > 0)$
else let Min'''_x_i = elim_red_ctxt(Min_x_i, $Max\_x_i \uplus Nil\_x_i$)
in let Max'''_x_i = elim_red_ctxt(Max_x_i, $Min'''\_x_i \uplus Nil\_x_i$)
in let Nil'''_x_i = elim_red_ctxt(Nil_x_i, $Min'''\_x_i \uplus Max'''\_x_i$)
in let $S\_x_i = Elim(x_i, Min'''\_x_i, Max'''\_x_i) \cup Nil'''\_x_i$
in eq_sys(S_x_i, $i - 1$) $\biguplus$ $Min'''\_x_i \uplus Max'''\_x_i$

Figure 5. Interlaced Algorithm in the Non-Parameterized Case

[16], in which the PANDORE compiler has been developed. The three implementations use the unitary redundancy test presented in Section 3.2. and arbitrary large integers to eliminate overflow problems that may occur during pairwise elimination and the pivoting operations performed in the simplex calls. The measures that will allow us to compare the scanning codes produced by the different algorithms are the following:

- the ratio of the time spent in the enumeration code generation with our implementation of Ancourt–Irigoin's algorithm to the one obtained with the top-down or the interlaced algorithm
- the number of elementary operations (min and max with two arguments, div, $+$, $-$, $*$, loop index increment) performed during the execution of the scanning code
- the user time required to execute the nested loop, measured on a Sun SparcStation 10 after compilation with `gcc -O2`
- the holes that may appear in the nested loops.

The complexity of each polyhedron is specified by way of its number of vertices, computed with the polyhedral library[11].

4.5.1. *Polyhedron P_1*

Set of constraints This is composed of 12 inequalities and four variables i,j,k,l; the associated polyhedron P_1 comprises 502500 integer vectors and 25 rational vertices:

$$\left\{ \begin{array}{lll} i & -i + 19 & j \\ -j + 19 & k - 1 & -k + 1000 \\ -k + l & 2 * k - l + 1 & -200 * i + k + l \\ 200 * i - k - l + 199 & -200 * j - k + 2 * l & 200 * j + k - 2 * l + 199 \end{array} \right. \geq 0$$

Scanning code produced with Ancourt–Irigoin's algorithm

```
for i = 0 , 15
  for j = div(i,3) , 19
    for k = 1 , 1000
       for l = max(k,200*i-k,div(200*j+k+1,2)) ,
               min(2*k+1,200*i-k+199,div(200*j+k+199,2))
```

Scanning code produced with the top-down algorithm

```
for i = 0 , 15
  for j = max(div(i,2),2*i-15) , min(i+1,15)
    for k = max(1,div(200*j,3),div(400*i-200*j-197,3),div(200*i+1,3)) ,
            min(1000,div(400*i-200*j+398,3),100*i+99)
      for l = max(k,200*i-k,div(200*j+k+1,2)) ,
              min(2*k+1,200*i-k+199,div(200*j+k+199,2))
```

Scanning code produced with the interlaced algorithm

```
for i = 0 , 15
  for j = max(2*i-15,div(i,2)) , min(15,i+1)
    for k = max(div(200*i+1,3),div(400*i-200*j-197,3),div(200*j,3),1) ,
            min(100*i+99,200*j+199,div(400*i-200*j+398,3),1000)
      for l = max(div(200*j+k+1,2),200*i-k,k) ,
              min(div(200*j+k+199,2),200*i-k+199,2*k+1)
```

Evaluation of the scanning codes

Algorithm	Ratio	# elementary op.	Execution time × 10, s	# Holes
Ancourt–Irigoin	—	7710317	22.470	276510 at depth l
Top-down	6.2	2207086	11.120	10 at depth k
Interlaced	4.3	2215789	11.010	10 at depth k

4.5.2. *Polyhedron P_2*

Set of constraints This comprises 17 inequalities and 5 variables i,j,k,l,m; the polyhedron P_2 defined by the system possesses 451800 integer vectors and 58 rational vertices:

$$\left\{\begin{array}{lll} i & -i+7 & j \\ -j+13 & k & -k+19 \\ l-1 & -l+300 & m-1 \\ 10*l-m+1 & -500*i+5*l & 500*i-5*l+499 \\ -300*j+l+m & 300*j-l-m+299 & -l-m+3999 \\ -200*k+2*l+m & 200*k-2*l-m+199 & \end{array}\right. \geq 0$$

Scanning code produced with Ancourt–Irigoin's algorithm

```
for i = 0 , 3
  for j = div(i-5,3) , div(-i+43,3)
    for k = div(3*j,11) , 19
      for l = max(100*i,1) , min(100*i+99,300)
        for m = max(1,300*j-l,200*k-2*l) ,
                min(10*l+1,300*j-l+299,200*k-2*l+199)
```

Scanning code produced with the top-down algorithm

```
for i = 0 , 3
  for j = div(i,3) , min(div(-i+37,3),div(11*i+10,3))
    for k = max(i,div(i+3*j,2)) ,
            min(18,6*i+5,div(3*j+5,2),div(i+3*j+3,2))
      for l = max(100*i,1,-300*j+200*k-299,div(50*k+2,3),div(300*j+9,11)) ,
              min(100*i+99,300,300*j+298)
        for m = max(1,300*j-l,200*k-2*l) ,
                min(10*l+1,300*j-l+299,200*k-2*l+199)
```

Scanning code produced with the interlaced algorithm

```
for i = 0 , 3
  for j = max(div(2*i-3,3),0) , min(div(-i+37,3),div(11*i+10,3))
    for k = max(div(i+3*j,2),i) ,
            min(div(i+3*j+3,2),div(3*j+5,2),6*i+5,18)
      for l = max(div(300*j+9,11),div(50*k+2,3),-300*j+200*k-299,1,100*i) ,
              min(300*j+298,100*k+99,-300*j+200*k+199,300,100*i+99)
        for m = max(200*k-2*l,300*j-l,1) ,
                min(200*k-2*l+199,300*j-l+299,10*l+1)
```

Evaluation of the scanning codes

Algorithm	Ratio	# elementary op.	Execution time × 100, s	# Holes
Ancourt–Irigoin	—	3206140	27.100	88108 at depth m
Top-down	58	1443244	18.000	0
Interlaced	103.2	1451956	17.270	0

4.5.3. *Polyhedron P_3*

The system defining P_3 is composed of 16 inequalities and five variables i,j,k,l,m; the polyhedron contains 376250 integer vectors and 72 rational vertices.

$$\left\{\begin{array}{lll} i & -i+11 & j \\ -j+7 & k & -k+15 \\ l-1 & -l+500 & -l+m-1 \\ 4*l-m+1 & -300*i+3*l+m+3 & 300*i-3*l-m+296 \\ -500*j+l+m & 500*j-l-m+499 & -200*k+2*l+m \\ 200*k-2*l-m+199 & & \end{array}\right. \geq 0$$

In this example, only the top-down and the interlaced algorithms produced an enumeration code. In our implementation of Ancourt–Irigoin's algorithm, the coefficient matrix of the simplex associated with the elimination of the first redundant constraint does not fit in the memory (the system S' stemming from phase I comprises 448 constraints).

Scanning code produced with the top-down algorithm

```
for i = 0 , 11
  for j = max(div(3*i-11,5),div(3*i-1,10),div(6*i,35)) , div(3*i+2,7)
    for k = max(div(9*i,14),div(9*i-1,8),div(3*i-6,2),3*j) ,
            min(div(3*i-j+2,2),div(15*j+14,4),div(3*i+5*j+7,4),
                div(5*j+9,2))
      for l = max(1,-500*j+200*k-499,div(100*k+2,3),300*i-200*k-202,
                  150*i-250*j-251,div(300*i+2,7)) ,
              min(500,300*i-200*k+296,-500*j+200*k+199,150*i-250*j+148,
                  div(200*k+198,3),250*j+249,75*i+73)
        for m = max(l+1,300*i-3*l-3,500*j-l,200*k-2*l) ,
                min(4*l+1,300*i-3*l+296,500*j-l+499,200*k-2*l+199)
```

Scanning code produced with the interlaced algorithm

```
for i = 0 , 11
  for j = max(div(6*i,35),div(3*i-1,10),div(3*i-11,5)) , div(3*i+2,7)
    for k = max(3*j,div(3*i-6,2),div(9*i-1,8)) ,
            min(div(5*j+9,2),div(3*i+5*j+7,4),div(15*j+14,4),
                div(3*i-j+2,2))
      for l = max(div(300*i+2,7),150*i-250*j-251,300*i-200*k-202,100*j,
                  div(100*k+2,3),-500*j+200*k-499,1) ,
              min(75*i+73,250*j+249,div(200*k+198,3),150*i-250*j+148,
                  -500*j+200*k+199,300*i-200*k+296,500)
        for m = max(200*k-2*l,500*j-l,300*i-3*l-3,l+1) ,
                min(200*k-2*l+199,500*j-l+499,300*i-3*l+296,4*l+1)
```

The enumeration code generation here is 3.1 times faster with the interlaced version than with the top-down version. The loops generated with both algorithms do not contain any hole.

4.5.4. *Polyhedron P_4*

The system defining P_4 comprises five variables i,j,k,l,m and 17 inequalities:

$$\left\{\begin{array}{lll} i & -i+5 & j \\ -j+13 & k & -k+12 \\ l-1 & -l+500 & -l+m-1 \\ 4*l-m+1 & -500*i+3*l+m+3 & 500*i-3*l-m+496 \\ -300*j+l+m & 300*j-l-m+299 & -l-m+3999 \\ -200*k+2*l+m & 200*k-2*l-m+199 & \end{array}\right. \geq 0$$

P_4 possesses 74 rational vertices and 357781 integer vectors. Our implementation of Ancourt–Irigoin's algorithm as well as the top-down version did not allow us to synthesize a scanning code for this polyhedron (the system S' resulting from their first phase did not fit in the memory). The nested loop obtained with the interlaced algorithm is the following:

```
for i = 0 , 5
  for j = max(div(10*i,21),div(5*i-1,6),div(5*i-11,3)) ,
          div(25*i+24,21)
    for k = max(div(9*j,5),div(5*i-6,2),div(5*i+3*j-1,4),div(15*i-1,8)) ,
            min(div(3*j+7,2),div(5*i+3*j+7,4),div(9*j+8,4),
                div(15*i+14,7))
      for l = max(div(500*i+2,7),250*i-150*j-151,500*i-200*k-202,60*j,
                  div(100*k+2,3),-300*j+200*k-299,1) ,
              min(125*i+123,150*j+149,div(200*k+198,3),250*i-150*j+248,
                  -300*j+200*k+199,500*i-200*k+496,500)
        for m = max(200*k-2*l,300*j-l,500*i-3*l-3,l+1) ,
                min(200*k-2*l+199,300*j-l+299,500*i-3*l+496,4*l+1)
```

This nested loop does not contain any hole.

5. Extension for Scanning Parameterized Polyhedrons

Let $P(y) = \{x = (x_1 \dots x_n)^T \;/\; Ay + Bx + c \geq 0\}$ be a parameterized polyhedron in the context $Q = \{y \;/\; My + h \geq 0\}$ (A, B, M are integer matrices and c, h integer vectors). Now let $S = Ay + Bx + c \geq 0$ and $C = My + h \geq 0$ be the systems of inequalities associated with these polyhedrons. The top-down and the interlaced algorithms can be naturally extended in order to synthesize the code scanning $P(y)$ in the context Q. For both algorithms, the method now consists in computing a system equivalent to S (in the context C) of the form

$$\Lambda(y) \biguplus Min(x_1) \uplus Max(x_1) \biguplus \dots \biguplus Min(x_n) \uplus Max(x_n)$$

where $Min(x_i)$ and $Max(x_i)$ are sets of minimizing and maximizing constraints for x_i and $\Lambda(y)$ a system of inequalities only depending on the vector of parameters y. If Low_i (resp. Upp_i) is the set of lower (resp. upper) bounds for x_i resulting from $Min(x_i)$ (resp.

$Max(x_i)$), the control structure enumerating the vectors of $P(y)$ in the context Q thus becomes:

$$\begin{array}{l} \textbf{for } x_1 = \textbf{max } Low_1, \textbf{ min } Upp_1 \\ \quad \ddots \\ \quad\quad \textbf{for } x_n = \textbf{max } Low_n, \textbf{ min } Upp_n \end{array}$$

if $\Lambda(y) = \emptyset$ and

$$\begin{array}{l} \textbf{if } \bigwedge_{i=1}^{i=r} t_i y + g_i \geq 0 \\ \textbf{then for } x_1 = \textbf{max } Low_1, \textbf{ min } Upp_1 \\ \quad\quad \ddots \\ \quad\quad\quad \textbf{for } x_n = \textbf{max } Low_n, \textbf{ min } Upp_n \\ \textbf{else skip} \end{array}$$

if $\Lambda(y) = \{t_1 y + g_1 \geq 0, \ldots, t_r y + g_r \geq 0\}$ (the integer vectors of the context Q that do not satisfy the constraints in $\Lambda(y)$ define empty instances of $P(y)$).

5.1. The Parameterized Top-down Algorithm

The two phases of the top-down algorithm become:

Phase I: Computation of a system equivalent to S resulting from the iterative application of pairwise elimination in property 1 (for $x_n, x_{n-1}, \ldots, x_1$ in this order):

$$S' = \Lambda(y) \biguplus Min(x_1) \uplus Max(x_1) \biguplus \ldots \biguplus Min(x_n) \uplus Max(x_n)$$

Phase II: Elimination of redundant constraints in S'. The system S'' resulting from this phase is of the form:

$$S'' = \Lambda'(y) \biguplus Min'(x_1) \uplus Max'(x_1) \biguplus \ldots \biguplus Min'(x_n) \uplus Max'(x_n)$$

in which $\Lambda'(y)$ is the elimination of the redundant constraints of $\Lambda(y)$ in the context C and for $i = 1, 2, \ldots, n$ in this order, the sets $Min'(x_i)$ et $Max'(x_i)$ are computed independently as follows:

- $Min'(x_i)$ = elimination of the redundant constraints of $Min(x_i)$ in the context $C \biguplus \Lambda'(y) \biguplus Min'(x_1) \uplus Max'(x_1) \biguplus \ldots \biguplus Min'(x_{i-1}) \uplus Max'(x_{i-1})$
- $Max'(x_i)$ = elimination of the redundant constraints of $Max(x_i)$ in the context $C \biguplus \Lambda'(y) \biguplus Min'(x_1) \uplus Max'(x_1) \biguplus \ldots \biguplus Min'(x_{i-1}) \uplus Max'(x_{i-1})$

5.2. The Parameterized Interlaced Algorithm

The system equivalent to S that leads to the generation of the conditional statement enumerating the integer vectors of $P(y)$ in the context Q is now given by the call $prm\_eq\_sys(S, n, C)$ to the function described in Figure 6.

prm_eq_sys(S, i, C) =
  if $i = 0$
  then elim_red_ctxt(S, C)
  else **let** $Min\_x_i \uplus Max\_x_i \uplus Nil\_x_i$ be the partition of S
    in **let** Min'_x_i = elim_red_ctxt(Min_x_i, $Max\_x_i \uplus Nil\_x_i \uplus C$)
      in **let** Max'_x_i = elim_red_ctxt(Max_x_i, $Min'\_x_i \uplus Nil\_x_i \uplus C$)
        in **let** Nil'_x_i = elim_red_ctxt(Nil_x_i, $Min'\_x_i \uplus Max'\_x_i \uplus C$)
          in **let** $S\_x_i = Elim(x_i, Min'\_x_i, Max'\_x_i) \cup Nil'\_x_i$
            in prm_eq_sys(S_x_i, $i - 1$, C) $\biguplus$ $Min'\_x_i \uplus Max'\_x_i$

Figure 6. Interlaced algorithm in the parameterized case

6. CONCLUSION

In this paper, we have presented the two algorithms implemented in the HPF compiler PANDORE that permit the synthesis of the code enumerating the integer vectors of a parameterized polyhedron in a given context. These algorithms make Ancourt-Irigoin's method applicable to the polyhedrons used in the compilation of HPF loops for parallel computers. Indeed, when applied to these polyhedrons, the first phase of the *successive projection method* may produce many inequalities, thus making the elimination of redundant constraints in phase II excessively complex or even impossible if the system produced in phase I does not fit in the memory.

Both algorithms lay emphasis on the way redundant inequalities are removed; they rely on an unitary redundancy test based on the simplex algorithm and take the properties of pairwise elimination into account in order to improve both the computation time and the execution time of the scanning codes. The first algorithm presented in the paper reconsiders the second phase in Ancourt-Irigoin's algorithm. In PANDORE, this version is applied on polyhedrons whose system computed in phase I can be produced in memory and possesses a *reasonable* size, that is a size that makes the removal of redundant inequalities applicable. The second version shown in the paper makes possible the generation of a scanning code for the other polyhedrons; it interlaces projections and eliminations of redundant constraints in order to avoid the possible combinational explosion of the number of constraints in the first version.

ACKNOWLEDGMENTS

I would like to thank Andrew Tagoe for his careful reading over the paper.

REFERENCES

1. M. E. Wolf and M. S. Lam, 'A loop transformation theory and an algorithm to maximize parallelism', *IEEE Transactions on Parallel and Distributed Systems*, **2**, (10) (1991).
2. C. Ancourt, *Génération automatique de codes de transfert pour multiprocesseurs à mémoires locales*, PhD thesis, Université de Paris VI, March 1991.
3. F. André, M. Le Fur, Y. Mahéo and J.-L. Pazat, 'The Pandore data-parallel compiler and its portable runtime', in *High-Performance Computing and Networking*, LNCS 919, Springer Verlag, Milan, Italy, May 1995.
4. S. M. Amarasinghe and M. Lam, 'Communication optimization and code generation for distributed memory machines', in *ACM SIGPLAN'93 Conference on Programming Language Design and Implementation*, June 1993.

5. C. Ancourt, F. Coelho, F. Irigoin and R. Keryell, 'A linear algebra framework for static HPF code distribution', in *Fourth International Workshop on Compilers for Parallel Computers*, Delft, The Netherlands, December 1993.
6. A. Schrijver, *Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization*, Wiley, 1986.
7. R. J. Duffin, 'On Fourier's analysis of linear inequality systems', *Mathematical Programming Study*, **1**, 71–95 (1974).
8. C. Ancourt and F. Irigoin, 'Scanning polyhedra with DO loops', in *Third ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, April 1991.
9. P. Feautrier, 'Semantical analysis and mathematical programming application to parallelization and vectorization', in M. Cosnard *et al.* (eds), *Parallel and Distributed Algorithms*, Elsevier Science Publishers B.V. (North Holland), 1989, pp. 309–320.
10. J.-F. Collard, P. Feautrier and T. Risset, *Construction of DO Loops from Systems of Affine Constraints*, Research Report 93-15, LIP, Lyon, France, May 1993.
11. H. Le Verge, V. Van Dongen and D. K. Wilde, *Loop Nest Synthesis Using the Polyhedral Library*, Research Report 2288, INRIA, France, May 1994.
12. H. Le Verge, *A Note on Chernikova's Algorithm*, Research Report 1662, INRIA, France, April 1992.
13. M.C. Cheng, 'General criteria for redundant and nonredundant linear inequalities', *Journal of Optimization Theory and Applications*, **53** (1), 37–42 (1987).
14. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*, Prentice-Hall, 1982.
15. F. Irigoin, *Partitionnement de boucles imbriquées. Une technique d'optimisation pour les programmes scientifiques*. Ph.D. thesis, Université Pierre et Marie Curie, June 1987.
16. M.-V. Aponte, A. Lavaille, M. Mauny, A. Suarez and P. Weis, *The CAML Reference Manual*. Technical Report 121, INRIA, France, September 1990.