The `Z_Polyhedra` library in MAPLE

Rui-Juan Jing[1] and Marc Moreno Maza[2]

[1] University of Western Ontario, `rjing8@uwo.ca`,
[2] University of Western Ontario, `moreno@csd.uwo.ca`.

Abstract. The $\mathbb{Z}$-Polyhedra is a library written in MAPLE and dedicated to solving problems dealing with the integer points of polyhedral sets. Those problems include decomposing the integer points of polyhedral sets, solving parametric integer programs, performing dependence analysis in for-loop nests and determining the validity of certain Presburger formulas. This article discusses the design of the $\mathbb{Z}$-Polyhedra library and provides numerous illustrations of its usage.

1 Introduction

Solving systems of linear equations is a well-studied and fundamental problem in mathematical sciences. When the input system includes equations as well as inequalities, the algebraic complexity of this problem increases from polynomial time to single exponential time with respect to the number of variables. When, in addition, the solution points with integer coordinates are the *only* ones of interest, the problem becomes even harder and is still actively investigated.

The integer points of polyhedral sets are, indeed, of interest in many areas of mathematical sciences, see for instance the landmark textbooks of A. Schrijver [13] and A. Barvinok [3], as well as the compilation of articles [4]. One of these areas is the analysis and transformation of computer programs. For instance, integer programming [5] is used by P. Feautrier in the scheduling of for-loop nests [6] while Barvinok's algorithm [2] (for counting integer points in polyhedra) is adapted by M. Köppe and S. Verdoolaege in [10] to answer questions like how many memory locations are touched by a for-loop nest. In [11], W. Pugh proposes an algorithm, called the *Omega Test*, for testing whether a polyhedron has integer points. In the same paper, W. Pugh shows how to use the Omega Test for performing dependence analysis [11] in for-loop nests.

In [12], W. Pugh also suggests, without stating a formal algorithm, that the Omega Test could be used for quantifier elimination on Presburger formulas. This observation has motivated our papers [7, 8], where we propose a new approach for computing the integer points of systems of linear equations and inequalities. Here, solving means decomposing the solution set into geometrically meaningful components and providing compact representations of those components. Moreover, the proposed algorithm runs in polynomial time when the dimension of the ambient space is fixed and the input system satisfies mild assumptions. This work produced a MAPLE library, originally called `Polyhedra` and presented at ISSAC 2017 as a software demonstration. To emphasize the fact that this library is primarily dedicated to the integer points of polyhedral sets, we re-baptized it the `ℤ-polyhedra` library. Improved design, new features and a faster core-solver (the command `IntegerSolve`) leads us to the present paper.

Section 2.1 discusses the implementation of the mathematical concepts involved in the manipulation of $\mathbb{Z}$-polyhedra. Section 3 gives an overview of the user-interface and the main solvers implemented in the `ℤ-polyhedra` library. Section 4 illustrates the usage of the library through examples taken from the literature. We note that the new algorithm (to be reported in a soon coming article) behind the command `IntegerSolve`) has reduced the execution of some problems from minutes to fractions of a second.

The `ℤ-library` is publicly available from the web site of the RegularChains library at `www.regularchains.org`. A comparison with related software can be found in the last section of [9].

2 Mathematical concepts and their implementation

In this section, we review the basic concepts of polyhedral geometry that are involved in the specifications of the commands of our `Z_Polyhedra` library. We also discuss the implementation of those concepts, in particular their adaptation to the context of effective computations. Section 2.1 is dedicated to the notion of a polyhedral set while Section 2.3, 2.2 and 2.3 focus on lattices, $\mathbb{Z}$-Polyhedra and parametric $\mathbb{Z}$-Polyhedra.

Notation 1 We use bold letters, e.g. $\mathbf{v}$, to denote vectors and we use capital letters, e.g. A, to denote matrices. Also, we assume that vectors are column vectors. For row vectors, we use the transposition notation, that is, A^t for the transposition of a matrix A. As usual, we denote by $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ the ring of integers, the field of rational numbers and the field of complex numbers. Unless specified otherwise, all matrices and vectors have their coefficients in $\mathbb{Z}$.

2.1 Polyhedra

A subset $P \subseteq \mathbb{Q}^n$ is called a *convex polyhedron* (or simply a *polyhedron*) if $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ holds, for a matrix $A \in \mathbb{Q}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{Q}^m$, where n, m are positive integers; we call the linear system $\{A\mathbf{x} \leq \mathbf{b}\}$ a *representation* of P. Hence, a polyhedron is the intersection of finitely many half-spaces.

An inequality of the system $A\mathbf{x} \leq \mathbf{b}$ is *redundant* whenever it is implied by all the other inequalities in $A\mathbf{x} \leq \mathbf{b}$. A representation of a polyhedron is *minimal* if no inequality of that representation is redundant.

An inequality $\mathbf{a}^t\mathbf{x} \leq b$ (with $\mathbf{a} \in \mathbb{Q}^n$ and $b \in \mathbb{Q}$) is an implicit equation of the inequality system $A\mathbf{x} \leq \mathbf{b}$ if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$. The *dimension* of the polyhedron P, denoted by $\dim(P)$, is $n-r$, where n is dimension[3] of the ambient space (that is, $\mathbb{Q}^n$) and r is the maximum number of implicit equations defined

[3] Of course, this notion of dimension coincides with the topological one, that is, the maximum dimension of a ball contained in P.

by linearly independent vectors. We say that P is *full-dimensional* whenever $\dim(P) = n$ holds. In other words, P is full-dimensional if and only if it does not have any implicit equations.

The article [9] presents an efficient algorithm for computing a minimal representation of the polyhedron P from any representation of P. This algorithm builds upon ideas proposed by Egon Balas in [1]; it is implemented in the `Z_Polyhedra` library by the command `MinimalRepresentation` of the module `PolyhedraTools`.

Let p, q be two positive integers such that $p + q = n$ holds. We rank the coordinates $(x_1, \ldots, x_n)$ of an arbitrary point $\mathbf{x}$ as $x_1 > \cdots > x_n$ and we denote by $\mathbf{u}$ (resp. $\mathbf{v}$) the first p (last q) coordinates of $\mathbf{x}$. We denote by $\mathsf{proj}(P; \mathbf{v})$ the *projection of* P *on* $\mathbf{v}$, that is, the subset of $\mathbb{Q}^q$ defined by:

$$\mathsf{proj}(P; \mathbf{v}) = \{\mathbf{v} \in \mathbb{Q}^q \mid \exists\ \mathbf{u} \in \mathbb{Q}^p,\ (\mathbf{u}, \mathbf{v}) \in P\}.$$

Fourier-Motzkin elimination (FME for short) is an algorithm computing the projection $\mathsf{proj}(P; \mathbf{v})$ of the polyhedron of P by successively eliminating the $\mathbf{u}$-variables from a representation of P.

Consider a representation R of P and a positive integer i such that $1 \leq i \leq n$. Denote by $R^{(x_i)}$ the inequalities in R whose largest variable is x_i. A *projected representation* of P induced by R is a representation of P consisting of

1. $R^{(x_1)}$, if $n = 1$,
2. $R^{(x_1)}$ and a projected representation of $\mathsf{proj}(P; (\mathbf{x}_2, \ldots, x_n))$, otherwise.

The article [9] presents an efficient algorithm for computing a minimal projected representation of the polyhedron P from any R representation of P. This algorithm is implemented in the `Z_Polyhedra` library by the command `MinimalProjectedRepresentation` of the module `PolyhedraTools`. In particular, this command provides a much more efficient way of performing FME than the command `Project` of the `PolyhedralSets` library in MAPLE, as illustrated by the comparative implementation reported in [9].

2.2 Lattices

The n-dimensional *integer lattice*, namely $\mathbb{Z}^n$, is the lattice in the Euclidean space $\mathbb{R}^n$ whose lattice points are all n-tuples of integers. More generally, a lattice of $\mathbb{R}^n$ consists of all linear combinations with integer coefficients of a basis of $\mathbb{R}^n$ (as a vector space). The data-type `Lattice` of the `Z_Polyhedra` library implements lattices in that latter sense, with some adaptation to support our purpose of studying the integer points of polyhedra. This adaptation is actually taken from the article [14]. To be precise,

1. we restrict the basis vectors, given by $n \times n$ matrix A, to have integer coefficients,
2. we allow a shift of the origin by a vector $\mathbf{b} \in \mathbb{Z}^n$.

Therefore, we call an *integer lattice* of $\mathbb{Z}^n$ any set of the form

$$\{A\mathbf{x} + \mathbf{b} \mid \mathbf{x} \in \mathbb{Z}^n\}$$

where $A \in \mathbb{Z}^{n \times n}$ is a full-rank matrix and $\mathbf{b} \in \mathbb{Z}^n$ is a vector; such a set is denoted by $\mathcal{L}(A, \mathbf{b})$.

2.3 $\mathbb{Z}$-Polyhedra

Following [14] here again, we call a $\mathbb{Z}$-*Polyhedron* the intersection of a polyhedron with an integer lattice. The purpose of this notion is, for us, to support the description of the integer points of a polyhedron $P \subseteq \mathbb{Q}^n$, that is, the description of the set $P \cap \mathbb{Z}^n$. This leads us to some preliminary remarks.

Consider first the problem of solving a Diophantine equation over $\mathbb{Z}$, say in 2 variables x and y. For instance, consider $3x - 4y = 7$; its solutions, as computed by MAPLE, are of the form $x = 5 + 4\,\_Z1, y = 2 + 3\,\_Z1$, the description of which requires the use of the auxiliary variable $\_Z1$. In his Omega test [11, 12] William Pugh extended that idea for solving arbitrary systems of linear equations of $\mathbb{Z}$. For instance, for the system

$$\begin{cases} 7x + 12y + 31z = 17 \\ 3x + 5y + 14z = 7 \end{cases}$$

our implementation of the Omega test produces

$$\begin{cases} z = -t_0 - 1 \\ y = -5t_0 - 3 \\ x = 13t_0 + 12 \end{cases}$$

Of course, the introduction of the parameter t_0 can be avoided by re-writing x and z as a function of z, leading to:

$$\begin{cases} x = -1 - 13z \\ y = 2 + 5z \end{cases}$$

Consider now this other polyhedron P of $\mathbb{Q}^3$:

$$\begin{cases} x = 19 \\ y = 25 + (1/2)z \\ z \le 18 \\ z \le 0 \end{cases}$$

Because of the presence of the rational number 1/2, the above input system cannot be considered as a description of the set $P \cap \mathbb{Z}^3$. Using our algorithm [7, 8] inspired by the Omega test, we obtain the following:

$$\begin{cases} x = 19 \\ y = 25 + t_0 \\ z = 2\,t_0 \\ -t_0 \le 0 \\ t_0 \le 9 \end{cases}$$

Inspired by the work[14], we have substantially improved our algorithm in terms of efficiency and in terms of output conciseness. In particular, for the above example, we obtain in a Maple session, the result below: On the left-hand side of Figure 1, we retrieve our original polyhedron P and on the right-hand side, we have the lattice $\mathcal{L}$ of $\mathbb{Z}^n$ consisting of the points (x, y, z) where $z/2$ is integer.

$$\left[\left[\begin{cases} x = 19 \\ y = 25 + \dfrac{z}{2} \\ z \leq 18 \\ -z \leq 0 \end{cases}, \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right]\right]\right]$$

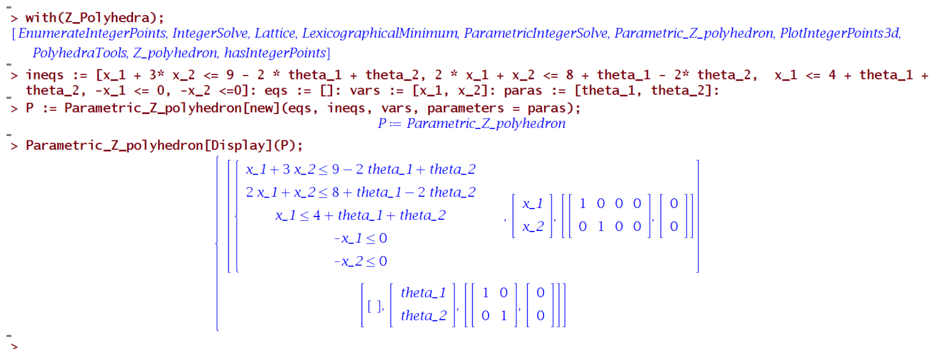
Fig. 1: A $\mathbb{Z}$-Polyhedron in PL format.

The intersection $P \cap \mathcal{L}$ is exactly $P \cap \mathbb{Z}^3$. More generally, encoding the integer points of a polyhedron using the above format, that we call the *PL format*, and thus using lattices, allows us to totally avoid the recourse to auxiliary variables. In addition, it is easy to convert any set of the form $P \cap \mathbb{Z}^n$ (where $P \subseteq \mathbb{Q}^n$ is a polyhedron) from PL format, say $P \cap \mathcal{L}(C, \mathbf{d})$, to the Omega test format, simply by substituting $\mathbf{x}$ with $C\mathbf{t} + \mathbf{d}$ into the representation of P, say $A\mathbf{x} \leq \mathbf{b}$.

2.4 Parametric $\mathbb{Z}$-Polyhedra

A *parametric $\mathbb{Z}$-Polyhedron* is a family of $\mathbb{Z}$-polyhedra
- given by the representation of a $\mathbb{Z}$-polyhedron where,
- the defining matrix or the defining vector depend linearly on parameters.

This notion is particularly useful in application problems, like parametric integer linear programming, where the feasible region, and thus optimal solutions, depend on the values of parameters.



```
> with(Z_Polyhedra);
[EnumerateIntegerPoints, IntegerSolve, Lattice, LexicographicalMinimum, ParametricIntegerSolve, Parametric_Z_polyhedron, PlotIntegerPoints3d,
    PolyhedraTools, Z_polyhedron, hasIntegerPoints]
> ineqs := [x_1 + 3* x_2 <= 9 - 2 * theta_1 + theta_2, 2 * x_1 + x_2 <= 8 + theta_1 - 2* theta_2,  x_1 <= 4 + theta_1 +
  theta_2, -x_1 <= 0, -x_2 <=0]: eqs := []: vars := [x_1, x_2]: paras := [theta_1, theta_2]:
> P := Parametric_Z_polyhedron[new](eqs, ineqs, vars, parameters = paras);
                                                P := Parametric_Z_polyhedron
> Parametric_Z_polyhedron[Display](P);
```

$$\left\{\begin{array}{l}\left[\left[\begin{cases} x\_1 + 3\,x\_2 \leq 9 - 2\,theta\_1 + theta\_2 \\ 2\,x\_1 + x\_2 \leq 8 + theta\_1 - 2\,theta\_2 \\ x\_1 \leq 4 + theta\_1 + theta\_2 \\ -x\_1 \leq 0 \\ -x\_2 \leq 0 \end{cases}, \begin{bmatrix} x\_1 \\ x\_2 \end{bmatrix}, \left[\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right]\right]\right] \\ \left[[\,], \begin{bmatrix} theta\_1 \\ theta\_2 \end{bmatrix}, \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right]\right]\end{array}\right.$$

```
>
```

Fig. 2: Making a new object of type `Parametric_Z_polyhedron`.

The MAPLE session on Figure 2 shows how to create (command `Parametric-_Z_polyhedron[new]`) and display (command `Parametric_Z_polyhedron[Display]`) a parametric $\mathbb{Z}$-Polyhedron from a list of equations, a list of inequalities, a list of the involved variables and a list of the involved parameters. In this example, the parameters are `theta_1` and `theta_2` while the unknowns are `x_1` and `x_2`. On can see that an object of type `Parametric_Z_polyhedron` is represented by a pair of $\mathbb{Z}$-polyhedra in PL format:

- one in the parameter space,
- one the whole ambient space.

.

3 Core algorithms and their implementation

The `Z_Polyhedra` library in Maple implements commands to manipulate $\mathbb{Z}$-polyhedra and in particular the integer points of polyhedra defined over $\mathbb{Q}$. To this end, the `Z_Polyhedra` library offers

1. 3 data-types in the form Maple modules: `Z_polyhedron`, `Lattice` and `Parametric_Z_polyhedron`,
2. a collection of solvers to compute the integer points of (parametric) $\mathbb{Z}$-polyhedra,
3. a fourth module gathering commands to operate on polyhedra and their rational points.

Data-Types and solvers are further discussed below.

3.1 Data-types

An object of the data-type `Z_polyhedron`, encodes the integer points of a polyhedron, using the PL format, specified in Section 2.3. An object of the data-type `Lattice` encodes a lattice as defined in Section 2.2. Finally, an object of the data-type `Parametric_Z_polyhedron` encodes a parametric $\mathbb{Z}$-polyhedron as defined in Section 2.4.

Each of these data-types is implemented in an “object-oriented” fashion using the Maple language construct of a module. Each of these three modules offers “get” methods to access the different attributes of an object, see Figure 3.

3.2 Solvers

The most commonly used solver is `IntegerSolve`. It takes as input a system of linear equations and inequalities, that is, a representation of some polyhedron $P \subseteq \mathbb{Q}^n$. It returns finitely many $\mathbb{Z}$-polyhedra

- either in PL format $P_1 \cap \mathcal{L}_1, \ldots, P_e \cap \mathcal{L}_e$ such that

$$P \cap \mathbb{Z}^n \quad = \quad (P_1 \cap \mathcal{L}_1) \cup \cdots \cup P_e \cap \mathcal{L}_e,$$

- or in Omega test format (thus using auxiliary variables) as on the example shown on Figure 4.

```
> with(Z_Polyhedra);
[EnumerateIntegerPoints, IntegerSolve, Lattice, LexicographicalMinimum,

    ParametricIntegerSolve, Parametric_Z_polyhedron, PlotIntegerPoints3d,

    PolyhedraTools, Z_polyhedron, hasIntegerPoints]

> with(Lattice);
           [DefiningMatrix, DefiningVector, Display, IsPointInLattice]

> with(Z_polyhedron);
            [Display, Equations, Inequalities, IsContained, Unknowns]

> with(Parametric_Z_polyhedron);
  [ConstraintsOnParameters, ConstraintsOnUnknowns, Display, Parameters, Unknowns]

> with(PolyhedraTools);
[IsNegative, IsNonNegative, IsNonPositive, IsPositive, IsRedundant, IsZero,

    MinimalProjectedRepresentation, MinimalRepresentation, hasRationalPoints]
```

Fig. 3: Maple session showing the commands and modules available to an end-user of the Z_Polyhedra library.

The core solver of the ℤ-Polyhedra library is `ParametricIntegerSolve`. Its specifications are similar to those of `IntegerSolve` but using parametric ℤ-polyhedra instead of ℤ-polyhedra. In fact, `IntegerSolve` is derived from `ParametricIntegerSolve` by letting the list of parameters be empty. Figure 5 shows what `ParametricIntegerSolve` does on the example of Section 2.4, that is, computing its minimal projected representation.

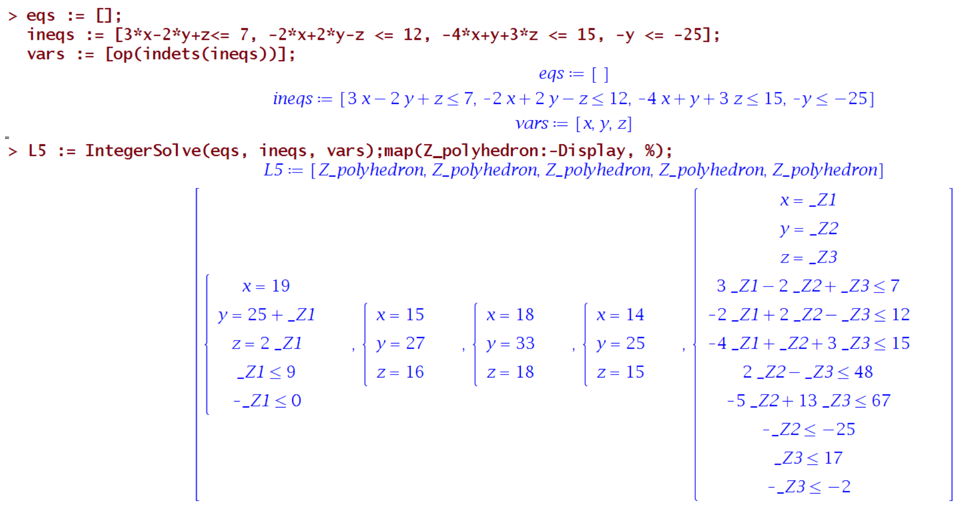
While solving a system of constraints does not mean enumerating its solutions, enumeration is sometimes what the user needs, in particular when plotting is involved Figure6 shows how the command `EnumerateIntegerPoints` enumerates the integer points of a polyhedron, after computing a minimal projected representation of that polyhedron. This is used by the command `PlotIntegerPoints3d` for plotting the same polyhedron.

4 Applications

4.1 Dependence analysis

Consider the following for-loop nest:

```
for $i = 1$ to 5 do
    for $j = i$ to 5 do
        $A[i,\ j+1] = A[5,\ j]$
```



```
> eqs := [];
  ineqs := [3*x-2*y+z<= 7, -2*x+2*y-z <= 12, -4*x+y+3*z <= 15, -y <= -25];
  vars := [op(indets(ineqs))];
```

$$eqs := [\ ]$$
$$ineqs := [3\,x-2\,y+z\le 7,\ -2\,x+2\,y-z\le 12,\ -4\,x+y+3\,z\le 15,\ -y\le -25]$$
$$vars := [x, y, z]$$

```
> L5 := IntegerSolve(eqs, ineqs, vars);map(Z_polyhedron:-Display, %);
```

$$L5 := [Z\_polyhedron, Z\_polyhedron, Z\_polyhedron, Z\_polyhedron, Z\_polyhedron]$$

$$\left[\begin{cases} x = 19 \\ y = 25 + \_Z1 \\ z = 2\,\_Z1 \\ \_Z1 \le 9 \\ -\_Z1 \le 0 \end{cases},\ \begin{cases} x = 15 \\ y = 27 \\ z = 16 \end{cases},\ \begin{cases} x = 18 \\ y = 33 \\ z = 18 \end{cases},\ \begin{cases} x = 14 \\ y = 25 \\ z = 15 \end{cases},\ \begin{cases} x = \_Z1 \\ y = \_Z2 \\ z = \_Z3 \\ 3\,\_Z1 - 2\,\_Z2 + \_Z3 \le 7 \\ -2\,\_Z1 + 2\,\_Z2 - \_Z3 \le 12 \\ -4\,\_Z1 + \_Z2 + 3\,\_Z3 \le 15 \\ 2\,\_Z2 - \_Z3 \le 48 \\ -5\,\_Z2 + 13\,\_Z3 \le 67 \\ -\_Z2 \le -25 \\ \_Z3 \le 17 \\ -\_Z3 \le -2 \end{cases}\right]$$

Fig. 4: Using `IntegerSolve` for decomposing the integer points of a polyhedron.

It is natural to ask whether two different iterations of the above for-loop nest can access the same coefficient in the array `A`, with at least one of those iterations writing that coefficient.

Consider first the case where one iteration (i, j) writes the same coefficient in `A` that another iteration (i', j') reads. If such a couple of iterations exists then the system below must have solutions for $i', j', i, j \in \mathbb{Z}$

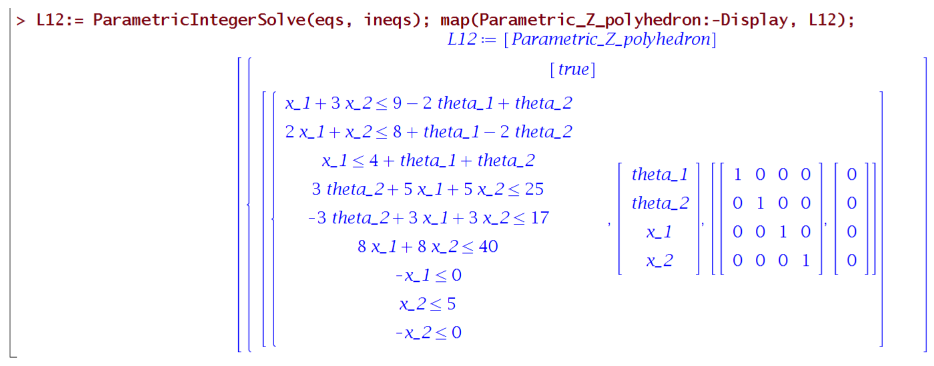
$$\begin{cases} 1 \le i \le j \le 5 \\ 1 \le i' \le j' \le 5 \\ i = 5 \\ j + 1 = j' \end{cases}$$

The Maple session on Figure 7 shows that the polyhedron defined by `eqs` and `ineqs` has no integer points. Similarly, it can be shown that no two iterations access the same coefficient in `A` both in writing. Consequently, each for-loop in the above for-loop nest can be executed in a parallel fashion without any race conditions.

We consider now a more difficult example used by Paul Feautrier in his lectures, see Figure 8. Here again, the `IntegerSolve` command can be used to prove that none of these 3 statements yield dependence, see Figure 9.

4.2 Cache lines accessed by a for-loop nest

The question considered here is counting the total number of cache lines accessed by a for-loop nest (a 5-point stencil computation code) see Example 5 in [12].



```
> L12:= ParametricIntegerSolve(eqs, ineqs); map(Parametric_Z_polyhedron:-Display, L12);
                              L12 := [Parametric_Z_polyhedron]
```

$$\left[\left[\begin{array}{c}[true]\\ \left[\left\{\begin{array}{c} x\_1 + 3\,x\_2 \le 9 - 2\,theta\_1 + theta\_2 \\ 2\,x\_1 + x\_2 \le 8 + theta\_1 - 2\,theta\_2 \\ x\_1 \le 4 + theta\_1 + theta\_2 \\ 3\,theta\_2 + 5\,x\_1 + 5\,x\_2 \le 25 \\ -3\,theta\_2 + 3\,x\_1 + 3\,x\_2 \le 17 \\ 8\,x\_1 + 8\,x\_2 \le 40 \\ -x\_1 \le 0 \\ x\_2 \le 5 \\ -x\_2 \le 0 \end{array}\right., \begin{bmatrix} theta\_1 \\ theta\_2 \\ x\_1 \\ x\_2 \end{bmatrix}, \left[\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right]\right]\end{array}\right]\right]$$

Fig. 5: Using `ParametricIntegerSolve`.

The polyhedron studied in the Maple session on Figure 10 is more general than the one of William Pugh. Indeed, we have replaced the loop bound 500 by a parameter `N`. What `IntegerSolve` computes in this case is a minimal projected representation of that polyhedron. This essentially provides an enumeration of its integer points. Focusing on the variables `i` and `j` leads to the desired answer.

4.3 Parametric linear programming

This last application presents work in progress. Consider the following for-loop nest:

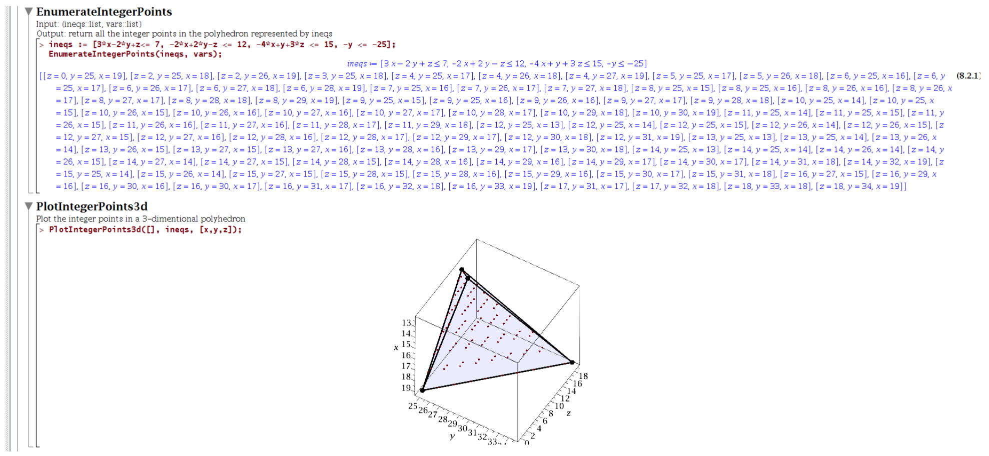
```
for $i = 0$ to $m$ do
    for $j = 0$ to $n$ do
        $A[2 * i + j] = i + j$
```

A natural question of concurrency is to determine, for a given k, what is the last iteration (i, j) at which $A[2 * i + j]$ receives the value k, see the driving problem in [5] by Paul Feautrier.

The command `LexicographicalMinimum` addresses a similar question with *maximum* replaced by *minimum*. Hence, to answer the original question, we need a natural change of coordinates where (i, j) is mapped to $(m - i, n - j)$. We are now looking at the following parametric $\mathbb{Z}$-polyhedron $P(k, m, n)$:

$$\left\{\begin{array}{r} 0 \le i \\ i \le m \\ 0 \le j \\ j \le n \\ 2i + j - k \le -k + 2m + n \\ -2i - j + k \le -k + 2m + n \end{array}\right.$$

In its current version, `LexicographicalMinimum` finds the lexicographical minimum of (i, j) within $P(k, m, n)$, viewing i, j, k, m, n as rational numbers



EnumerateIntegerPoints

Input: (ineqs::list, vars::list)
Output: return all the integer points in the polyhedron represented by ineqs

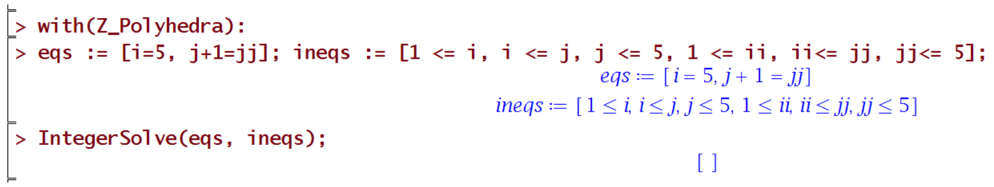
```
> ineqs := [3*x-2*y+z<= 7, -2*x+2*y-z <= 12, -4*x+y+3*z <= 15, -y <= -25];
  EnumerateIntegerPoints(ineqs, vars);
                         ineqs := [3 x − 2 y + z ≤ 7, -2 x + 2 y − z ≤ 12, -4 x + y + 3 z ≤ 15, -y ≤ −25]
[[z = 0, y = 25, x = 19], [z = 2, y = 25, x = 18], [z = 2, y = 26, x = 19], [z = 3, y = 25, x = 18], [z = 4, y = 25, x = 17], [z = 4, y = 26, x = 18], [z = 4, y = 27, x = 19], [z = 5, y = 25, x = 17], [z = 5, y = 26, x = 18], [z = 6, y = 25, x = 16], [z = 6, y     (8.2.1)
    = 25, x = 17], [z = 6, y = 26, x = 17], [z = 6, y = 27, x = 18], [z = 6, y = 28, x = 19], [z = 7, y = 25, x = 16], [z = 7, y = 26, x = 17], [z = 7, y = 27, x = 18], [z = 8, y = 25, x = 15], [z = 8, y = 25, x = 16], [z = 8, y = 26, x = 16], [z = 8, y = 26, x
    = 17], [z = 8, y = 27, x = 17], [z = 8, y = 28, x = 18], [z = 8, y = 29, x = 19], [z = 9, y = 25, x = 15], [z = 9, y = 25, x = 16], [z = 9, y = 26, x = 16], [z = 9, y = 27, x = 17], [z = 9, y = 28, x = 18], [z = 10, y = 25, x = 14], [z = 10, y = 25, x
    = 15], [z = 10, y = 26, x = 15], [z = 10, y = 26, x = 16], [z = 10, y = 27, x = 16], [z = 10, y = 27, x = 17], [z = 10, y = 28, x = 17], [z = 10, y = 29, x = 18], [z = 10, y = 30, x = 19], [z = 11, y = 25, x = 14], [z = 11, y = 25, x = 15], [z = 11, y
    = 26, x = 15], [z = 11, y = 26, x = 16], [z = 11, y = 27, x = 16], [z = 11, y = 28, x = 17], [z = 11, y = 29, x = 18], [z = 12, y = 25, x = 13], [z = 12, y = 25, x = 14], [z = 12, y = 25, x = 15], [z = 12, y = 26, x = 14], [z = 12, y = 26, x = 15], [z
    = 12, y = 27, x = 15], [z = 12, y = 27, x = 16], [z = 12, y = 28, x = 16], [z = 12, y = 28, x = 17], [z = 12, y = 29, x = 17], [z = 12, y = 30, x = 18], [z = 12, y = 31, x = 19], [z = 13, y = 25, x = 13], [z = 13, y = 25, x = 14], [z = 13, y = 26, x
    = 14], [z = 13, y = 26, x = 15], [z = 13, y = 27, x = 15], [z = 13, y = 27, x = 16], [z = 13, y = 28, x = 16], [z = 13, y = 29, x = 17], [z = 13, y = 30, x = 18], [z = 14, y = 25, x = 13], [z = 14, y = 25, x = 14], [z = 14, y = 26, x = 14], [z = 14, y
    = 26, x = 15], [z = 14, y = 27, x = 14], [z = 14, y = 27, x = 15], [z = 14, y = 28, x = 15], [z = 14, y = 28, x = 16], [z = 14, y = 29, x = 16], [z = 14, y = 29, x = 17], [z = 14, y = 30, x = 17], [z = 14, y = 31, x = 18], [z = 14, y = 32, x = 19], [z
    = 15, y = 25, x = 14], [z = 15, y = 26, x = 14], [z = 15, y = 27, x = 15], [z = 15, y = 28, x = 15], [z = 15, y = 28, x = 16], [z = 15, y = 29, x = 16], [z = 15, y = 30, x = 17], [z = 15, y = 31, x = 18], [z = 16, y = 27, x = 15], [z = 16, y = 29, x
    = 16], [z = 16, y = 30, x = 16], [z = 16, y = 30, x = 17], [z = 16, y = 31, x = 17], [z = 16, y = 32, x = 18], [z = 16, y = 33, x = 19], [z = 17, y = 31, x = 17], [z = 17, y = 32, x = 18], [z = 18, y = 33, x = 18], [z = 18, y = 34, x = 19]]
```

PlotIntegerPoints3d

Plot the integer points in a 3-dimentional polyhedron

```
> PlotIntegerPoints3d([], ineqs, [x,y,z]);
```

Fig. 6: Using `EnumerateIntegerPoints`.



```
> with(Z_Polyhedra):
> eqs := [i=5, j+1=jj]; ineqs := [1 <= i, i <= j, j <= 5, 1 <= ii, ii<= jj, jj<= 5];
                                            eqs := [i = 5, j + 1 = jj]
                              ineqs := [1 ≤ i, i ≤ j, j ≤ 5, 1 ≤ ii, ii ≤ jj, jj ≤ 5]
> IntegerSolve(eqs, ineqs);
                                                  [ ]
```

Fig. 7: Using `IntegerSolve` for dependence analysis.

(instead of integers) which yields the solution shown on Figure 11. The output consists of 4 pairs; each pair gives a lexicographical minimum together with the corresponding conditions on k, m, n under which this minimum is reached.

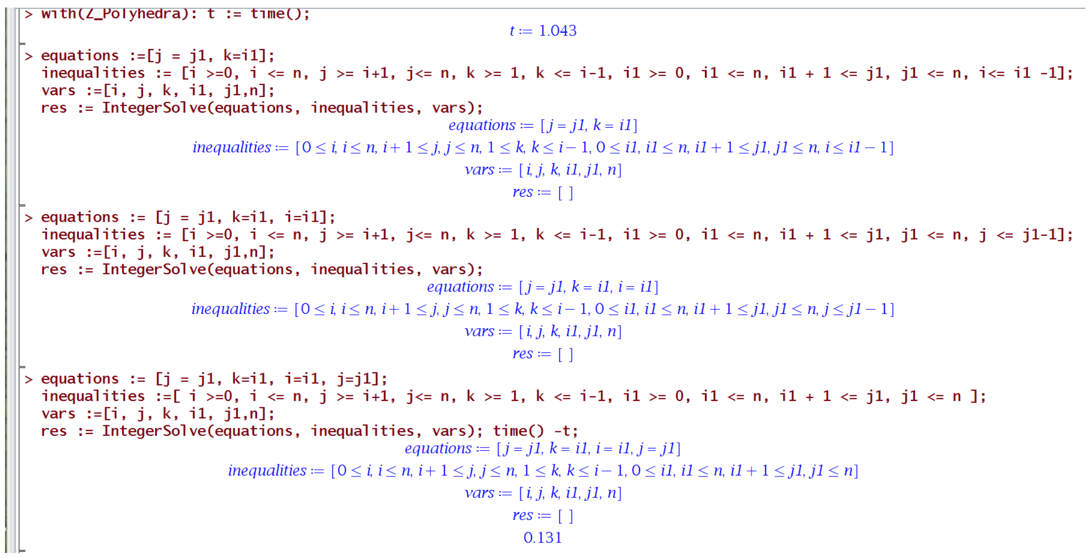
Acknowledgements

The authors would like to thank IBM Canada Ltd (CAS project 880) and NSERC of Canada (CRD grant CRDPJ500717-16), as well as the Fields Institute of Canada for supporting their work.

```
    for(i=1; i<=n; i++){
1:      x = a[i][i];
        for(k=1; k<i; k++)
2:          x = x - a[i][k]*a[i][k];
3:      p[i] = 1.0/sqrt(x);
        for(j=i+1; j<=n; j++){
4:          x = a[i][j];
            for(k=1; k<i; k++)
5:              x = x - a[j][k]*a[i][k];
6:          a[j][i] = x * p[i];
            }
```

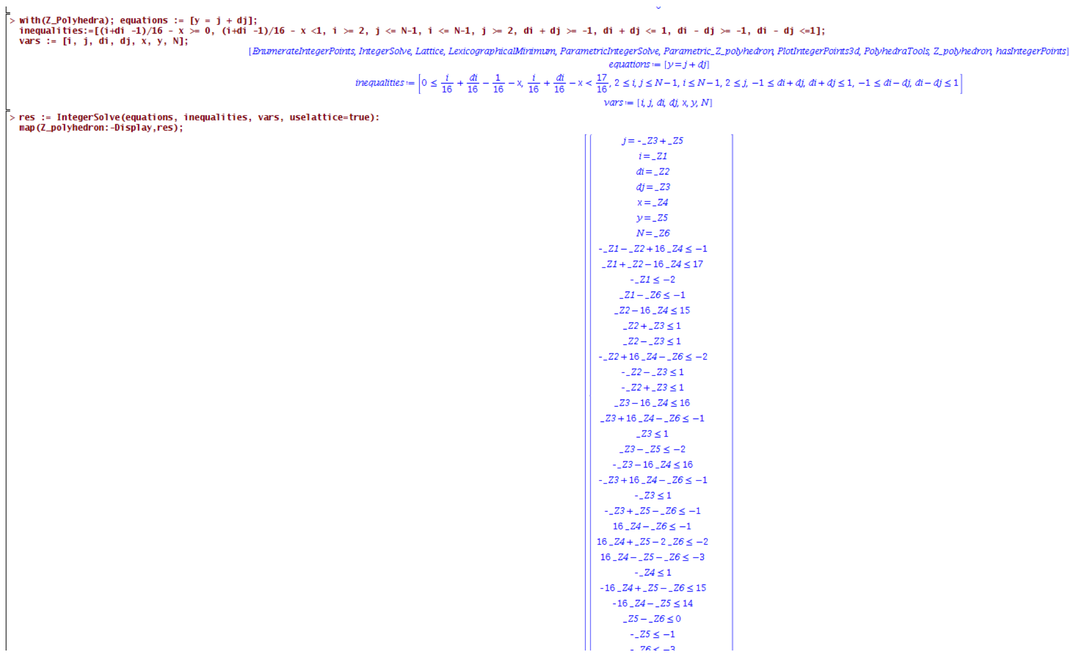
$(5, i, j, k) : A[j][k]]$		$(6, i', j') : A[j'][i']$
$0 \leq i \leq n$ $i+1 \leq j \leq n$ $1 \leq k \leq i-1$		$0 \leq i' \leq n$ $i'+1 \leq j' \leq n$
	$j = j', k = i'$	
$i < i'$	$i = i', j < j'$	$i = i', j = j'$

Fig. 8: On the right: pseudo-code for Choleski LU. On the Left: dependence analysis of 3 statements of this pseudo-code.



```
> with(Z_Polyhedra): t := time();
                                        t ≔ 1.043
> equations :=[j = j1, k=i1];
  inequalities := [i >=0, i <= n, j >= i+1, j<= n, k >= 1, k <= i-1, i1 >= 0, i1 <= n, i1 + 1 <= j1, j1 <= n, i<= i1 -1];
  vars :=[i, j, k, i1, j1,n];
  res := IntegerSolve(equations, inequalities, vars);
                                   equations ≔ [j = j1, k = i1]
           inequalities ≔ [0 ≤ i, i ≤ n, i + 1 ≤ j, j ≤ n, 1 ≤ k, k ≤ i − 1, 0 ≤ i1, i1 ≤ n, i1 + 1 ≤ j1, j1 ≤ n, i ≤ i1 − 1]
                                     vars ≔ [i, j, k, i1, j1, n]
                                           res ≔ [ ]
> equations := [j = j1, k=i1, i=i1];
  inequalities := [i >=0, i <= n, j >= i+1, j<= n, k >= 1, k <= i-1, i1 >= 0, i1 <= n, i1 + 1 <= j1, j1 <= n, j <= j1-1];
  vars :=[i, j, k, i1, j1,n];
  res := IntegerSolve(equations, inequalities, vars);
                                equations ≔ [j = j1, k = i1, i = i1]
           inequalities ≔ [0 ≤ i, i ≤ n, i + 1 ≤ j, j ≤ n, 1 ≤ k, k ≤ i − 1, 0 ≤ i1, i1 ≤ n, i1 + 1 ≤ j1, j1 ≤ n, j ≤ j1 − 1]
                                     vars ≔ [i, j, k, i1, j1, n]
                                           res ≔ [ ]
> equations := [j = j1, k=i1, i=i1, j=j1];
  inequalities :=[ i >=0, i <= n, j >= i+1, j<= n, k >= 1, k <= i-1, i1 >= 0, i1 <= n, i1 + 1 <= j1, j1 <= n ];
  vars :=[i, j, k, i1, j1,n];
  res := IntegerSolve(equations, inequalities, vars); time() -t;
                             equations ≔ [j = j1, k = i1, i = i1, j = j1]
               inequalities ≔ [0 ≤ i, i ≤ n, i + 1 ≤ j, j ≤ n, 1 ≤ k, k ≤ i − 1, 0 ≤ i1, i1 ≤ n, i1 + 1 ≤ j1, j1 ≤ n]
                                     vars ≔ [i, j, k, i1, j1, n]
                                           res ≔ [ ]
                                             0.131
```

Fig. 9: Using `IntegerSolve` for dependence analysis on Choleski LU.



```
> with(Z_Polyhedra); equations := [y = j + dj];
  inequalities:=[(i+di -1)/16 - x >= 0, (i+di -1)/16 - x <1, i >= 2, j <= N-1, i <= N-1, j >= 2, di + dj >= -1, di + dj <= 1, di - dj >= -1, di - dj <=1];
  vars := [i, j, di, dj, x, y, N];
```

[EnumerateIntegerPoints, IntegerSolve, Lattice, LexicographicalMinimum, ParametricIntegerSolve, Parametric_Z_polyhedron, PlotIntegerPoints3d, PolyhedraTools, Z_polyhedron, hasIntegerPoints]

$$equations := [y = j + dj]$$

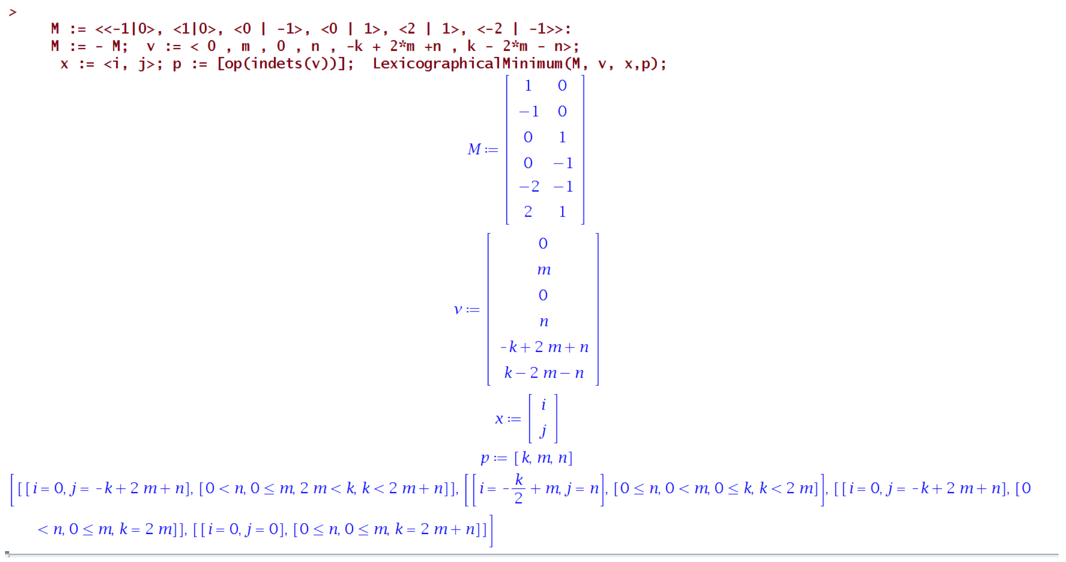
$$inequalities := \left[0 \le \frac{i}{16} + \frac{di}{16} - \frac{1}{16} - x,\ \frac{i}{16} + \frac{di}{16} - x < \frac{17}{16},\ 2 \le i,\ j \le N-1,\ i \le N-1,\ 2 \le j,\ -1 \le di + dj,\ di + dj \le 1,\ -1 \le di - dj,\ di - dj \le 1\right]$$

$$vars := [i, j, di, dj, x, y, N]$$

```
> res := IntegerSolve(equations, inequalities, vars, uselattice=true):
  map(Z_polyhedron:-Display,res);
```

```
j = -_Z3 + _Z5
i = _Z1
di = _Z2
dj = _Z3
x = _Z4
y = _Z5
N = _Z6
-_Z1 - _Z2 + 16 _Z4 ≤ -1
_Z1 + _Z2 - 16 _Z4 ≤ 17
-_Z1 ≤ -2
_Z1 - _Z6 ≤ -1
_Z2 - 16 _Z4 ≤ 15
_Z2 + _Z3 ≤ 1
_Z2 - _Z3 ≤ 1
-_Z2 + 16 _Z4 - _Z6 ≤ -2
-_Z2 - _Z3 ≤ 1
-_Z2 + _Z3 ≤ 1
_Z3 - 16 _Z4 ≤ 16
_Z3 + 16 _Z4 - _Z6 ≤ -1
_Z3 ≤ 1
_Z3 - _Z5 ≤ -2
-_Z3 - 16 _Z4 ≤ 16
-_Z3 + 16 _Z4 - _Z6 ≤ -1
-_Z3 ≤ 1
-_Z3 + _Z5 - _Z6 ≤ -1
16 _Z4 - _Z6 ≤ -1
16 _Z4 + _Z5 - 2 _Z6 ≤ -2
16 _Z4 - _Z5 - _Z6 ≤ -3
-_Z4 ≤ 1
-16 _Z4 + _Z5 - _Z6 ≤ 15
-16 _Z4 - _Z5 ≤ 14
_Z5 - _Z6 ≤ 0
-_Z5 ≤ -1
-_Z6 ≤ -3
```

Fig. 10: Using `IntegerSolve` for cache line accesses.



```
>
    M := <<-1|0>, <1|0>, <0 | -1>, <0 | 1>, <2 | 1>, <-2 | -1>>:
    M := - M;  v := < 0 , m , 0 , n , -k + 2*m +n , k - 2*m - n>;
     x := <i, j>; p := [op(indets(v))];  LexicographicalMinimum(M, v, x,p);
```

$$M := \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -2 & -1 \\ 2 & 1 \end{bmatrix}$$

$$v := \begin{bmatrix} 0 \\ m \\ 0 \\ n \\ -k + 2\,m + n \\ k - 2\,m - n \end{bmatrix}$$

$$x := \begin{bmatrix} i \\ j \end{bmatrix}$$

$$p := [k, m, n]$$

$$\Big[[\,[i = 0, j = -k + 2\,m + n], [0 < n, 0 \le m, 2\,m < k, k < 2\,m + n]\,], \left[\left[i = -\frac{k}{2} + m, j = n\right], [0 \le n, 0 < m, 0 \le k, k < 2\,m]\right], [\,[i = 0, j = -k + 2\,m + n], [0 < n, 0 \le m, k = 2\,m]\,], [\,[i = 0, j = 0], [0 \le n, 0 \le m, k = 2\,m + n]\,]\Big]$$

Fig. 11: Using `LexicographicalMinimum` for parametric linear programming.

Bibliography

[1] Egon Balas. Projection with a minimal system of inequalities. *Computational Optimization and Applications*, 10(2):189–193, 1998.

[2] Alexander I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Math. Oper. Res.*, 19(4):769–779, 1994.

[3] Alexander I. Barvinok. *Integer Points in Polyhedra*. Contemporary mathematics. European Mathematical Society, 2008.

[4] Matthias Beck. *Integer Points in Polyhedra–Geometry, Number Theory, Representation Theory, Algebra, Optimization, Statistics: AMS-IMS-SIAM Joint Summer Research Conference, June 11-15, 2006, Snowbird, Utah*. Contemporary mathematics - American Mathematical Society. American Mathematical Society, 2008.

[5] Paul Feautrier. Parametric integer programming. *RAIRO Recherche Opérationnelle*, 22, 1988. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.9957&rep=rep1&type=pdf`.

[6] Paul Feautrier. Automatic parallelization in the polytope model. In *The Data Parallel Programming Model: Foundations, HPF Realization, and Scientific Applications*, pages 79–103, London, UK, UK, 1996. Springer-Verlag. `http://dl.acm.org/citation.cfm?id=647429.723579`.

[7] Rui-Juan Jing and Marc Moreno Maza. Computing the integer points of a polyhedron, I: algorithm. In *Computer Algebra in Scientific Computing - 19th International Workshop, CASC 2017, Beijing, China, September 18-22, 2017, Proceedings*, pages 225–241, 2017.

[8] Rui-Juan Jing and Marc Moreno Maza. Computing the integer points of a polyhedron, II: complexity estimates. In *Computer Algebra in Scientific Computing - 19th International Workshop, CASC 2017, Beijing, China, September 18-22, 2017, Proceedings*, pages 242–256, 2017.

[9] Rui-Juan Jing, Marc Moreno Maza, and Delaram Talaashrafi. Complexity estimates for fourier-motzkin elimination. *CoRR*, abs/1811.01510, 2018.

[10] Matthias Köppe and Sven Verdoolaege. Computing parametric rational generating functions with a primal barvinok algorithm. *Electr. J. Comb.*, 15(1), 2008.

[11] William Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. In Joanne L. Martin, editor, *Proceedings Supercomputing '91, Albuquerque, NM, USA, November 18-22, 1991*, pages 4–13. ACM, 1991.

[12] William Pugh. Counting solutions to presburger formulas: How and why. In Vivek Sarkar, Barbara G. Ryder, and Mary Lou Soffa, editors, *Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI), Orlando, Florida, USA, June 20-24, 1994*, pages 121–134. ACM, 1994.

[13] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[14] Rachid Seghir, Vincent Loechner, and Benoît Meister. Integer affine transformations of parametric $\mathbb{Z}$-polytopes and applications to loop nest optimization. *TACO*, 9(2):8:1–8:27, 2012.