Polyhedral Analysis using Parametric Objectives

Jacob M. Howe[1] and Andy King[2]

[1] School of Informatics, City University London, EC1V 0HB, UK
[2] School of Computing, University of Kent, CT2 7NF, UK

Abstract. The abstract domain of polyhedra lies at the heart of many program analysis techniques. However, its operations can be expensive, precluding their application to polyhedra that involve many variables. This paper describes a new approach to computing polyhedral domain operations. The core of this approach is an algorithm to calculate variable elimination (projection) based on parametric linear programming. The algorithm enumerates only non-redundant inequalities of the projection space, hence permits anytime approximation of the output.

1 Introduction

Polyhedra [10] form the basis of a wide range of tools for static analysis and model checking. Their attraction is the expressivity of linear inequalities which capture not only the range of values that a variable can assume, but also dependencies between them. The drawback of polyhedra is the cost of the domain operations – this has motivated much recent work investigating the tradeoff of expressivity for efficiency. This paper introduces a new approach to polyhedral domain operations that sidesteps many of the problems associated with current approaches. At the heart of the work is a new approach to variable elimination (projection, or existential quantifier elimination).

Many polyhedral libraries are based on the double description method (DDM) [3, 5, 6, 24, 28, 35]. DDM maintains two representations: the constraint representation in which the polyhedron is described as the solutions of a system of linear inequalities, and the frame representation in which the polyhedron is generated from a finite set of points, rays and lines. The method has proved popular since the constraint representation is convenient when computing the meet of two polyhedra (intersection) and the frame representation is convenient when computing join (the frame representation of the join of two polyhedra is merely the union of their frame representations). The maintenance of the two representations requires an algorithm to convert between the two, and this is the core of the method. The drawback of working with a double description is that maintenance algorithms are expensive. For example, to apply join to two polyhedra in constraint representation they both have to be converted to the frame representation, a potentially exponential operation [21]. The dominating cost of maintaining the double description can be avoided by working solely with constraints and reformulating join in terms of variable elimination [32]. Variable elimination can be performed with Fourier-Motzkin elimination [27], but the

technique needs to be applied together with techniques for avoiding the generation of redundant inequalities [12, 17, 22] or methods for removing them after generation [23]. Even then, Fourier-Motzkin shares the problem with DDM that an intermediate result can be exponentially larger than the input, in addition to the problem of generation of redundant constraints that cannot always be removed until variable elimination is completed.

The algorithms deployed for manipulating unrestricted polyhedra [3, 24, 35] have barely changed since the inception of polyhedral analysis over 30 years ago [10]. This paper presents a radically different approach to the computation of projection and hence convex hull which can be reduced to projection [32]. The approach is based on the constraint representation, admits anytime approximation and invites parallelisation. The approach has two key steps. The first step describes the projection of the input constraint system as a bounded polyhedron (polytope) in a dual space. The vertices in this dual description of the projection correspond to non-redundant inequalities in the constraint representation of the projection. The second step is to enumerate these vertices. This is achieved by using parametric linear programming (PLP). The formulation of projection as PLP implementing vertex enumeration of a dual description of the projection space is not obvious, therefore this paper makes the following contributions:

- Lemma 3.2 of [19] builds on the projection lemma [36] to explain how projection can be reformulated as a vertex enumeration problem. Alas, this lemma is not correct to the level of generality that is required and the starting point of this paper is a reworking of this result.
- In a somewhat different way to [19] it is shown how PLP can be used to enumerate the vertices of a polytope. When this polytope is a description of the projection space, the output corresponds to irredundant inequalities of the constraint description of the projection.
- Together this gives an algorithm that projects arbitrary (possibly unbounded) polyhedra onto lower dimensions. (This fundamental algorithm may well find application outside static analysis, for example in control theory [19, 29].)
- It is shown that this formulation enables inequalities in the projection space to be enumerated one-by-one, without requiring any post-processing to remove redundant inequalities. A consequence of this is that projection is naturally anytime – it can be stopped prematurely to yield a safe over-approximation of the projection. This compares favourably with DDM which is monolithic in the sense that the inequalities need to be completely converted to a frame, and then the projected frame is completely converted to the constraints representation before a single inequality in the projection space is found. The force of the anytime property is that if the projection is found to be excessively large, then the projection need only be computed up to that point and no further whilst still yielding a useful result.

Given the novelty of the contributions to theory (and the length of their exposition) a description of the implementation is postponed to a later paper; to the best of the authors' knowledge the implementions techniques are themselves novel and require space in their own right.

2 Background

2.1 Matrices and vectors

Let $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle \in \mathbb{R}^n$ denote a column vector which is an $n \times 1$ matrix. If $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle$ then $a :: \boldsymbol{a} = \langle a, a_1, \ldots, a_n \rangle$. The dot product of two column vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ is defined $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T \boldsymbol{b}$ where A^T denotes transpose of a matrix A. For any matrix A, $(A)_j$ refers to the jth column of A and $(A)_J$ refers to the submatrix of columns indexed by J. Likewise $(A)^j$ refers to the jth row and $(A)^J$ refers to the submatrix of rows indexed by J. Similar notations are used for vectors, though bracketing is omitted when the column and row operator is clear. If $A = \{a_1, \ldots, a_n\} \subseteq \mathbb{R}$ and $a_1 < a_2 < \ldots < a_n$ then $(A)_i = (\boldsymbol{a})_i$ where $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle$. The lexicographical ordering relation on vectors is defined by $\langle\rangle \preceq \langle\rangle$ and $(a :: \boldsymbol{a}) \preceq (b :: \boldsymbol{b})$ iff $a \leq b$ or $(a = b$ and $\boldsymbol{a} \preceq \boldsymbol{b})$. If $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \subseteq \mathbb{R}^n$ then $\min(A) = \boldsymbol{a}_i$ such that $\boldsymbol{a}_i \preceq \boldsymbol{a}_j$ for all $1 \leq j \leq m$. A vector (row) is lex-positive iff its first non-zero element is positive.

2.2 Basic and non-basic variables

The simplex algorithm [7, 34] is formulated in terms of pivoting operations on bases. To introduce simplex consider maximising the cost function $\boldsymbol{c} \cdot \boldsymbol{\lambda}$, where $\boldsymbol{c} = \langle 1, 1, -3, -3, -1, -1, 0 \rangle$, subject to the constraints $A\boldsymbol{\lambda} = \boldsymbol{b}$ where $\boldsymbol{\lambda} \geq \boldsymbol{0}$ and

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 & 2 & -2 & 0 \\ 2 & & 2\,2 & & 2\,8 & & 8\,2 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Let $B = \{1, 6\}$ and $N = \{1, \ldots, 7\} \setminus B$ so that $\boldsymbol{\lambda}_B = \langle \lambda_1, \lambda_6 \rangle$ and $\boldsymbol{\lambda}_N = \langle \lambda_2, \ldots, \lambda_5, \lambda_7 \rangle$ and moreover

$$A_B = \begin{bmatrix} 1 & -2 \\ 2 & 8 \end{bmatrix} \quad A_N = \begin{bmatrix} -1 & 1 & -1 & 2 & 0 \\ & 2\,2 & & 2\,8 & 2 \end{bmatrix}$$

Then $A\boldsymbol{\lambda} = \boldsymbol{b}$ can be expressed as $A_B\boldsymbol{\lambda}_B + A_N\boldsymbol{\lambda}_N = \boldsymbol{b}$ hence $A_B\boldsymbol{\lambda}_B = \boldsymbol{b} - A_N\boldsymbol{\lambda}_N$. Since the square matrix A_B is nonsingular this is equivalent to

$$\boldsymbol{\lambda}_B = A_B^{-1}\boldsymbol{b} - A_B^{-1}A_N\boldsymbol{\lambda}_N \tag{1}$$

This suggests putting $\boldsymbol{\lambda}_N = \boldsymbol{0}$ to give

$$\boldsymbol{\lambda}_B = A_B^{-1}\boldsymbol{b} = \frac{1}{12}\begin{bmatrix} 8 & 2 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{12}\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Since $\boldsymbol{\lambda}_B \geq \boldsymbol{0}$ the point $\lambda = \langle \frac{1}{6}, 0, 0, 0, 0, \frac{1}{12}, 0 \rangle$ satisfies both the equality constraints $A\boldsymbol{\lambda} = \boldsymbol{b}$ and inequalities $\boldsymbol{\lambda} \geq \boldsymbol{0}$, five of which are saturated by $\boldsymbol{\lambda}$. Geometrically, $\boldsymbol{\lambda}$ is a vertex of the polyhedron $\{\boldsymbol{\lambda} \geq \boldsymbol{0} \mid A\boldsymbol{\lambda} = \boldsymbol{b}\}$ and for this point $\boldsymbol{c} \cdot \boldsymbol{\lambda} = \frac{1}{12}$. In fact a vertex is defined by any B for which A_B is invertible (though a vertex may be defined by different B). In this classical set up, B is called the basis, N the co-basis and $\boldsymbol{\lambda}_B$ and $\boldsymbol{\lambda}_N$ are, respectively, the basic and

non-basic variables. Moreover, the objective function $\boldsymbol{c} \cdot \boldsymbol{\lambda}$ too can be considered to be a function of the non-basic variables $\boldsymbol{\lambda}_N$. To see this, observe

$$\boldsymbol{c} \cdot \boldsymbol{\lambda} = \boldsymbol{c}_B \cdot \boldsymbol{\lambda}_B + \boldsymbol{c}_N \cdot \boldsymbol{\lambda}_N = \boldsymbol{c}_B \cdot A_B^{-1}\boldsymbol{b} + (\boldsymbol{c}_N^T - \boldsymbol{c}_B^T A_B^{-1} A_N) \cdot \boldsymbol{\lambda}_N \qquad (2)$$

The equalities of (1) and the objective given in (2) constitute the dictionary.

2.3 Pivoting

In the (revised) simplex method [11], a path is found between adjacent bases that terminates with a basis that maximises the objective. Adjacent bases differ by one index and pivoting is used to transition from one basis to another. In each pivoting step, the basis B is updated with $B' = (B \setminus \{i\}) \cup \{j\}$ where $i \in B$ is the index of a basic variable that leaves B and $j \in N$ is the index for a non-basic variable that enters B. The index $j \in N$ is chosen so that the corresponding element of $\boldsymbol{c}_N^T - \boldsymbol{c}_B^T A_B^{-1} A_N$ is positive. This is achieved by solving $\boldsymbol{y}^T A_B = \boldsymbol{c}_B^T$ since then $\boldsymbol{y}^T = \boldsymbol{c}_B^T A_B^{-1}$ hence $\boldsymbol{c}_N^T - \boldsymbol{c}_B^T A_B^{-1} A_N = \boldsymbol{c}_N^T - \boldsymbol{y}^T A_N$. To illustrate for $B = \{1, 6\}$, so that $\boldsymbol{c}_B = \langle 1, -1 \rangle$ and $\boldsymbol{c}_N = \langle 1, -3, -3, -1, 0 \rangle$. Then

$$\boldsymbol{y}^T = \boldsymbol{c}_B^T A_B^{-1} = \frac{1}{12} \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ -2 & 1 \end{bmatrix} = \frac{1}{12} \begin{bmatrix} 10 & 1 \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{c}_N^T - \boldsymbol{y}^T A_N &= \begin{bmatrix} 1 & -3 & -3 & -1 & 0 \end{bmatrix} - \tfrac{1}{12} \begin{bmatrix} 10 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 & 2 & 0 \\ 2 & 2 & 2 & 8 & 2 \end{bmatrix} \\ &= \tfrac{1}{6} \begin{bmatrix} 2 & -24 & -22 & -20 & -1 \end{bmatrix} \end{aligned}$$

The entering variable can only be λ_2. To find a leaving variable for $i = 2$, let

$$\boldsymbol{d} = A_B^{-1}(A)_2 = \frac{1}{12} \begin{bmatrix} 8 & 2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \frac{1}{12} \begin{bmatrix} -4 \\ 4 \end{bmatrix}$$

Then the largest $t \geq 0$ is found such that $\boldsymbol{\lambda}_B - t\boldsymbol{d} \geq 0$. This occurs when $t = \frac{1}{4}$ and then $\boldsymbol{\lambda}_B - t\boldsymbol{d} = \langle \frac{1}{4}, 0 \rangle$. The second element of this vector is 0 hence the second variable of B, namely λ_6, leaves the basis. This gives the new basis $B = \{1, 2\}$ for which $\boldsymbol{\lambda}_B = \langle \frac{1}{4}, \frac{1}{4} \rangle$ and $\boldsymbol{c} \cdot \boldsymbol{\lambda} = \frac{1}{2}$ which has increased as desired. Repeating the process with $B = \{1, 2\}$ gives $\boldsymbol{c}_B = \langle 1, 1 \rangle$, $\boldsymbol{c}_N = \langle -3, -3, -1, -1, 0 \rangle$ and

$$\boldsymbol{y}^T = \frac{1}{12} \begin{bmatrix} 6 & 3 \end{bmatrix} \qquad \boldsymbol{c}_N^T - \boldsymbol{y}^T A_N = \frac{1}{2} \begin{bmatrix} -8 & -6 & -8 & -4 & -1 \end{bmatrix}$$

Hence there is no variable to enter B and $\boldsymbol{c} \cdot \boldsymbol{\lambda}$ is maximal. Although revised simplex is usually introduced with this pivoting rule, alternative rules may be attractive in certain situations.

2.4 Avoiding cycling with lexicographic pivoting

Cycling, hence non-termination, can be resolved [2] by lexicographic pivot selection [7]. This pivoting rule is defined for a subset of the bases which are called

lex-positive; each vertex, including degenerate ones, has a unique lex-positive basis. The graph of lex-positive bases is a subgraph of the basis graph yet still covers all the vertices. In the rest of this paper the dictionary is embedded into $A\boldsymbol{\lambda} = \boldsymbol{b}$ by extending it to the system $C(\mu :: \boldsymbol{\lambda}) = (0 :: \boldsymbol{b})$ where

$$C = \left[\begin{array}{c|c} 1 & -\boldsymbol{c} \\ \hline 0 & A \end{array}\right]$$

The rows and columns of C (and only this matrix) are indexed from 0 to preserve the correspondance with $A\boldsymbol{\lambda} = \boldsymbol{b}$. To illustrate, if $B = \{0, 1, 6\}$ then

$$C = \left[\begin{array}{c|cccccc} 1 & -1 & -1 & 3 & 3 & 1 & 1 & 0 \\ \hline 0 & 1 & -1 & 1 & -1 & 2 & -2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 8 & 8 & 2 \end{array}\right] \quad C_B = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & -2 \\ 0 & 2 & 8 \end{bmatrix}$$

A basis B is said to be lex-positive if each row $(L)^j$ is lex-positive where $j > 0$ and $L = [C_B^{-1}(0 :: \boldsymbol{b}) \mid C_B^{-1}]$. For example, B is lex-positive since

$$C_B^{-1} = \frac{1}{12}\begin{bmatrix} 12 & 10 & 1 \\ 0 & 8 & 2 \\ 0 & -2 & 1 \end{bmatrix} \quad L = \frac{1}{12}\left[\begin{array}{c|cc} 1 & 12 & 10 & 1 \\ 2 & 0 & 8 & 2 \\ 1 & 0 & -2 & 1 \end{array}\right]$$

Lexicographic pivoting pivots whilst preserving the lex-positive basis property. In each pivoting step, row zero $r = (C_B^{-1}C)^0$ is inspected to find an index $j \in N$ such that $(r)_j < 0$. For

$$C_B^{-1}C = \frac{1}{12}\begin{bmatrix} 12 & 0 & -20 & 48 & 28 & 40 & 0 & 2 \\ 0 & 12 & -4 & 12 & -4 & 32 & 0 & 4 \\ 0 & 0 & 4 & 0 & 4 & 4 & 12 & 2 \end{bmatrix}$$

this would give $j = 2$. Then $i = \mathsf{lexminratio}(B, j) > 0$ is computed which prescribes which $i \in B$ should be replaced with j. The $\mathsf{lexminratio}$ operation is defined

$$\mathsf{lexminratio}(B, j) = \begin{cases} 0 & \text{if } S = \emptyset \\ (B)_{k+1} & \text{else if } (L)^k/(\boldsymbol{d})_k = \min(S) \end{cases}$$

where $\boldsymbol{d} = C_B^{-1}(C)_j$ and $S = \{(L)^k/(\boldsymbol{d})_k \mid 0 < k \wedge 0 < (\boldsymbol{d})_k\}$. Crucially if B is lex-positive then so is $B' = (B \setminus \{i\}) \cup \{j\}$. Continuing with $j = 2$, $\boldsymbol{d} = \langle -20, -4, 4\rangle$, $S = \{\frac{1}{48}[1, 0, -2, 1]\}$ and $i = (B)_2 = 6$. Hence $B' = \{0, 1, 2\}$. Observe that B' is lex-positive since

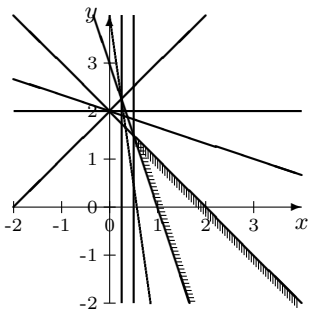
$$[C_{B'}^{-1}(0 :: \boldsymbol{b}) \mid C_{B'}^{-1}] = \frac{1}{4}\left[\begin{array}{c|cc} 2 & 4 & 0 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 0 & -2 & 1 \end{array}\right]$$

Then

$$C_{B'}^{-1}C = \frac{1}{4}\begin{bmatrix} 4 & 0 & 0 & 16 & 16 & 20 & 20 & 4 \\ 0 & 4 & 0 & 4 & 0 & 12 & 4 & 2 \\ 0 & 0 & 4 & 0 & 4 & 4 & 12 & 2 \end{bmatrix}$$

and since all elements of row $(C_{B'}^{-1}C)^0$ are positive, no new variable can enter B' and $\boldsymbol{c} \cdot \boldsymbol{\lambda}$ is maximal.

1: $x + y + z \leq 2$
2: $x + y - z \leq 2$
3: $-3x - y + z \leq -3$
4: $-3x - y - z \leq -3$
5: $-x + y + 2z \leq 2$
6: $-x + y - 2z \leq 2$



$1 + 2$: $x + y \leq 2$
$1 + 4$: $-2x \leq -1$
$1 + 6$: $y \leq 2$
$3 + 2$: $-2x \leq -1$
$3 + 4$: $-3x - y \leq -3$
$3 + 6$: $-4x \leq -1$
$5 + 2$: $x + 3y \leq 6$
$5 + 4$: $-7x - y \leq -4$
$5 + 6$: $-x + y \leq 2$

Fig. 1. (a) Inequalities (b) Projection (graphically) (c) Projection (Fourier-Motzkin)

3 Worked Example

This section centres on a worked example that illustrates the steps involved in setting up and applying the projection algorithm. The more formal aspects of the algorithm are detailed in the next section. The example involves eliminating the variable z from the following system of inequalities given in Fig 1(a). The system of inequalities is sufficiently simple for Fourier-Motzkin elimination to be applied. The projection is obtained by combining all the inequalities with a positive coefficient for z with those that have a negative coefficient for z. The resulting inequalities are given in Fig 1(c), where the left hand column indicates how the inequalities of Fig 1(a) are combined.

The system contains many redundant inequalities as is illustrated in the diagram given in Fig 1(b) – it can be seen that the projection is a cone that can be described by just two inequalities, namely $-3x - y \leq -3$ and $x + y \leq 2$. The challenge is to derive these constraints without generating the redundant inequalities, a problem that is magnified when eliminating many variables.

3.1 Overview

The algorithm presented in this section proceeds in three separate steps, each of which is detailed in its own section. Sect. 3.2 shows how to formulate the projection problem in a dual fashion so that any inequality entailed by the projection corresponds to points in a cone. Sect. 3.3 shows how to compute a slice through the cone yielding a polytope (a bounded polyhedra). The vertices of the polytope then represent the non-redundant inequalities in the projection. Sect. 3.5 explains how to use PLP to enumerate the vertices of this polytope. By enumerating each vertex exactly once, the inequalities are generated with no repetition and no redundancy other than the enumeration of the trivial constraint $0 \leq 1$.

3.2 Describing the output inequalities as a cone

To represent the inequalities in the projection as a cone, a formulation of [19] is adapted in which a set of points of the form $\langle \alpha_1, \alpha_2, \beta \rangle$ is used to represent

inequalities $\alpha_1 x + \alpha_2 y \leq \beta$ that are entailed by the system given in Fig 1(a). The inequalities are augmented with the trivial constraint $0 \leq 1$. These points are defined as solutions to the systems (3) and (4) below:

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta \end{bmatrix} = E^T \boldsymbol{\lambda} \qquad E^T = \begin{bmatrix} 1 & 1 & -3 & -3 & -1 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & 0 \\ 2 & 2 & -3 & -3 & 2 & 2 & 1 \end{bmatrix} \tag{3}$$

where $\boldsymbol{\lambda} = \langle \lambda_1, \ldots, \lambda_7 \rangle \geq \mathbf{0}$ and

$$D = \begin{bmatrix} 1 & -1 & 1 & -1 & 2 & -2 & 0 \end{bmatrix} \qquad D\boldsymbol{\lambda} = [0] \tag{4}$$

The matrix E^T represents the x, y coefficients and constants of the input inequalities. The λ variables are interpreted as weightings that prescribe positive linear combinations of the input inequalities that yield an entailed inequality. The equation given in (4) stipulates that the sum of the z coefficients is zero, in other words z is eliminated.

Let $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}^7 \mid D\boldsymbol{\lambda} = \mathbf{0} \wedge \boldsymbol{\lambda} \geq 0\}$ and $E^T\Lambda = \{E^T\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in \Lambda\}$. Observe that if $\langle \alpha_1, \alpha_2, \beta \rangle \in E^T\Lambda$, that is, the point $\langle \alpha_1, \alpha_2, \beta \rangle$ satisfies (3) and (4), then $\mu\langle \alpha_1, \alpha_2, \beta \rangle \in E^T\Lambda$ for any $\mu \geq 0$ hence $E^T\Lambda$ constitutes a cone. Importantly the final column of E^T permits the constant β of an inequality to be relaxed: if $\alpha_1 x + \alpha_2 y \leq \beta$ is entailed and $\beta \leq \beta'$, then $\alpha_1 x + \alpha_2 y \leq \beta'$ is also entailed.

3.3 Slicing the cone with a plane to obtain a polytope

In order to construct a polytope in the $\langle \alpha_1, \alpha_2, \beta \rangle$ space, a plane slicing through the cone $E^T\Lambda$ is required. To find such a plane, consider the inequalities that are entailed by the initial system given in Fig 1(a), again represented dually as a set of points, denoted G. Any inequality $\alpha_1 x + \alpha_2 y + \alpha_3 z \leq \beta$ entailed by the inequalities of Fig 1(a) is represented by a point $\langle \alpha_1, \alpha_2, \alpha_3, \beta \rangle \in G$ where $G = \{R\boldsymbol{\mu} \mid \boldsymbol{\mu} \geq \mathbf{0}\}$ and

$$R = \begin{bmatrix} 1 & 1 & -3 & -3 & -1 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & 0 \\ 1 & -1 & 1 & -1 & 2 & -2 & 0 \\ 2 & 2 & -3 & -3 & 2 & 2 & 1 \end{bmatrix}$$

Each column of R gives the coefficients α_i and the constant β of an inequality of Fig 1(a), again augmented with the additional inequality $0 \leq 1$. G is a cone incident to the origin where the columns of R are extremal rays of G (rays that cannot be obtained as positive linear combinations of others).

A plane that slices $E^T\Lambda$ can be derived from one that slices G. Let $a\alpha_1 + b\alpha_2 + c\alpha_3 + d\beta = 0$ be a plane that supports the cone G at the origin, hence all the rays of G are strictly above this plane. By setting $\boldsymbol{\mu} = \langle 1, \ldots, 1 \rangle$ in G, the ray $\{\mu\langle 1, 1, 1, 2 \rangle \mid \mu \geq 0\}$ gives the point $\langle 1, 1, 1, 2 \rangle$, similarly $\{\mu\langle 1, 1, -1, 2 \rangle \mid \mu \geq 0\}$ gives $\langle 1, 1, -1, 2 \rangle$, etc. Values for a, b, c, d can be found by setting up a linear program which asserts that each of these 7 points are strictly above the plane by at least some quantity ϵ:

$$
\begin{array}{lll}
\text{Maximise } \epsilon \text{ subject to} & \epsilon \leq a+b+c+2d & -1 \leq a \leq 1 \\
& \epsilon \leq a+b-c+2d & -1 \leq b \leq 1 \\
& \epsilon \leq -3a-b+c-3d & -1 \leq c \leq 1 \\
& \epsilon \leq -3a-b-c-3d & -1 \leq d \leq 1 \\
& \epsilon \leq -a+b+2c+2d & 0 \leq \epsilon \\
& \epsilon \leq -a+b-2c+2d & \epsilon \leq d
\end{array}
$$

The bounds $-1 \leq a, b, c, d \leq 1$ are included for normalisation since the plane $a\alpha_1 + b\alpha_2 + c\alpha_3 + d\beta = 0$ can also be described by $\mu a\alpha_1 + \mu b\alpha_2 + \mu c\alpha_3 + \mu d\beta = 0$ where $\mu \geq 0$ is any positive multiplier. Solving the linear program gives $a = -1$, $b = \frac{1}{3}$, $c = 0$, $d = \frac{2}{3}$ and $\epsilon = \frac{2}{3}$. The value of ϵ is discarded and the equation of the plane that supports G is $-3\alpha_1 + \alpha_2 + 0\alpha_3 + 2\beta = 0$.

Next observe that $E^T\Lambda = \{\langle \alpha_1, \alpha_2, \beta\rangle \mid \langle \alpha_1, \alpha_2, 0, \beta\rangle \in G\}$. As a consequence, a supporting plane for $E^T\Lambda$ can be found merely by removing the α_3 component (note that $c = 0$ is an oddity of this particular example). This gives $-3\alpha_1 + \alpha_2 + 2\beta = 0$ which indeed supports $E^T\Lambda$. Finally the constant for the plane is adjusted so that it slices through $E^T\Lambda$. Any positive value may be chosen, here the plane is set to have constant 1, that is, $-3\alpha_1 + \alpha_2 + 2\beta = 1$. Since $\langle \alpha_1, \alpha_2, \beta\rangle = E^T\boldsymbol{\lambda}$ the equation of the plane induces a further constraint on $\boldsymbol{\lambda}$:

$$
\begin{aligned}
1 = -3\alpha_1 + \alpha_2 + 2\beta = -3 \begin{bmatrix} 1 & 1 & -3 & -3 & -1 & -1 & 0 \end{bmatrix} \boldsymbol{\lambda} + \\
\begin{bmatrix} 1 & 1 & -1 & -1 & 1 & 1 & 0 \end{bmatrix} \boldsymbol{\lambda} + \\
2 \begin{bmatrix} 2 & 2 & -3 & -3 & 2 & 2 & 1 \end{bmatrix} \boldsymbol{\lambda} = \begin{bmatrix} 2 & 2 & 2 & 2 & 8 & 8 & 2 \end{bmatrix} \boldsymbol{\lambda}
\end{aligned}
$$

Augmenting equation (4) the system $A\boldsymbol{\lambda} = \boldsymbol{c}$ is obtained where:

$$
A = \begin{bmatrix} 1 & -1 & 1 & -1 & 2 & -2 & 0 \\ 2 & 2 & 2 & 2 & 8 & 8 & 2 \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5}
$$

Under this construction, the set $\Lambda' = \{\boldsymbol{\lambda} \in \mathbb{R}^7 \mid A\boldsymbol{\lambda} = \boldsymbol{c} \wedge \boldsymbol{\lambda} \geq 0\}$ is not a cone; it is a polytope. $E^T\Lambda'$ is a polytope as a consequence.

3.4 The vertices of the polytope as irredundant inequalities

For each non-redundant inequality $\alpha_1 x + \alpha_2 y \leq \beta$ in the projection there exists a unique vertex $\langle \alpha_1, \alpha_2, \beta\rangle \in E^T\Lambda'$. Moreover, if $\langle \alpha_1, \alpha_2, \beta\rangle$ is a vertex of $E^T\Lambda'$ there exists a vertex $\boldsymbol{\lambda}$ of Λ' such that $\langle \alpha_1, \alpha_2, \beta\rangle = E^T\boldsymbol{\lambda}$. However, the converse does not hold. If $\boldsymbol{\lambda}$ is a vertex of Λ' then $E^T\boldsymbol{\lambda}$ is not necessarily a vertex of $E^T\Lambda'$. To illustrate, the following table gives the vertices $\boldsymbol{\lambda}$ of Λ' for the system given in (5) and $\langle \alpha_1, \alpha_2, \beta\rangle = E^T\boldsymbol{\lambda}$:

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	α_1	α_2	β
$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1
$\frac{1}{4}$	0	0	$\frac{1}{4}$	0	0	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$
$\frac{1}{6}$	0	0	0	0	$\frac{1}{12}$	0	$\frac{1}{12}$	$\frac{1}{4}$	$\frac{1}{2}$
0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$
0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	$-\frac{3}{2}$

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	α_1	α_2	β
0	0	$\frac{1}{6}$	0	0	$\frac{1}{12}$	0	$-\frac{7}{12}$	$-\frac{1}{12}$	$-\frac{1}{3}$
0	$\frac{1}{6}$	0	0	$\frac{1}{12}$	0	0	$\frac{1}{12}$	$\frac{1}{4}$	$\frac{1}{2}$
0	0	0	$\frac{1}{6}$	$\frac{1}{12}$	0	0	$-\frac{7}{12}$	$-\frac{1}{12}$	$-\frac{1}{3}$
0	0	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$-\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$
0	0	0	0	0	0	$\frac{1}{2}$	0	0	$\frac{1}{2}$

First observe that $E^T\langle 0,0,\frac{1}{6},0,0,\frac{1}{12},0\rangle = \langle -\frac{7}{12},-\frac{1}{12},-\frac{1}{3}\rangle = E^T\langle 0,0,0,\frac{1}{6},\frac{1}{12},0,0\rangle$ and second that $-\frac{7}{12}x - \frac{1}{12}y \leq -\frac{1}{3}$ is a redundant inequality. In fact, only rows 1 and 5 give non-redundant inequalities; row 10 gives the trivial inequality $0 \leq 1$ and the remaining rows give inequalities that are redundant. (Note that this table is only given for the purposes of exposition and is not actually calculated as part of the projection algorithm.)

3.5 Enumerating inequalities using PLP

To enumerate the vertices of the polytope defined in section 3.3, hence the irredundant inequalities of the projection space, PLP is used. As the parameters vary the basis representing the optimum changes – a subset of these bases correspond to the vertices. Consider an objective function parameterised by variables δ_1 and δ_2:

$$\delta_1\alpha_1 + \delta_2\alpha_2 = \delta_1(E^T)^1 \cdot \boldsymbol{\lambda} + \delta_2(E^T)^2 \cdot \boldsymbol{\lambda} = \boldsymbol{c} \cdot \boldsymbol{\lambda}$$

where $\boldsymbol{c} = \langle \delta_1+\delta_2, \delta_1+\delta_2, -3\delta_1-\delta_2, -3\delta_1-\delta_2, -\delta_1+\delta_2, -\delta_1+\delta_2, 0\rangle$. The range of values taken by δ_1 and δ_2 can be constrained to $-1 \leq \delta_1, \delta_2 \leq 1$ without changing the set of possible objectives. This leads to tableau:

$$C = \left[\begin{array}{c|c|c} 1 & -\boldsymbol{c} & 0 \\ \hline 0 & A & \boldsymbol{b} \end{array}\right] = \left[\begin{array}{c|ccccccc|c} 1 & (-\delta_1-\delta_2) & (-\delta_1-\delta_2) & (3\delta_1+\delta_2) & (3\delta_1+\delta_2) & (\delta_1-\delta_2) & (\delta_1-\delta_2) & 0 & 0 \\ \hline 0 & 1 & -1 & 1 & -1 & 2 & -2 & 0 & 0 \\ 0 & 2 & 2 & 2 & 2 & 8 & 8 & 2 & 1 \end{array}\right]$$

An initial basis (hence vertex) is found by fixing δ_1 and δ_2 and optimising. Here, $\delta_1 = \delta_2 = 1$, hence $\alpha_1 + \alpha_2$ is maximised, to give $B = \{0,1,2\}$. The pivots involved in this optimisation lead to:

$$C_B = \begin{bmatrix} 1 & (-\delta_1-\delta_2) & (-\delta_1-\delta_2) \\ 0 & 1 & -1 \\ 0 & 2 & 2 \end{bmatrix} \qquad C_B^{-1} = \frac{1}{4}\begin{bmatrix} 4 & 0 & (2\delta_1+2\delta_2) \\ 0 & 2 & 1 \\ 0 & -2 & 1 \end{bmatrix}$$

$$T_1 = C_B^{-1}C = \begin{bmatrix} 1&0&0 & (4\delta_1+2\delta_2) & (4\delta_1+2\delta_2) & (5\delta_1+3\delta_2) & (5\delta_1+3\delta_2) & (\delta_1+\delta_2) & (2\delta_1+2\delta_2) \\ 0&1&0 & 1 & 0 & 3 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0&0&1 & 0 & 1 & 1 & 3 & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

Observe that with $\delta_1 = \delta_2 = 1$ this tableau represents an optimum since (in row T_1^0) the objective entry for each non-basic column is positive. However, decreasing the δ_1 parameter to $-\frac{1}{2}$ leads to the objective entries for columns 3 and 4 to be 0. Hence with the new parameters there are potential alternative bases that correspond to points optimal with respect to the objective (optimal bases). These possibilities are explored, that is, columns 3 and 4 are considered as candidates to enter the basis with objective $-\frac{1}{2}\alpha_1 + \alpha_2$. Note that this treatment of pivoting is slightly non-standard – when optimising with respect to an objective a column is considered as a candidate to enter the basis when its objective entry is strictly negative; here, it is optimal bases that are of interest and the condition is that objective entries that are zero. In the example column 3 is selected as the

candidate to enter the basis, $\mathsf{lexminratio}(B, 3) = 1$ so that column 1 leaves, with the result that B is now $\{0, 2, 3\}$. Pivoting gives:

$$T_2 = \begin{bmatrix} 1 & (-4\delta_1 - 2\delta_2) & 0 & 0 & (4\delta_1 + 2\delta_2) & (-7\delta_1 - 3\delta_2) & (\delta_1 + \delta_2) & -\delta_1 & -\frac{\delta_1}{2} \\ 0 & & 1 & 0 & 1 & 0 & 3 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & & 0 & 1 & 0 & 1 & 1 & 3 & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

Observe that with $\delta_1 = -\frac{1}{2}$ and $\delta_2 = 1$ this still represents an optimum. However, this tableau does not represent a vertex of the projection space. At a vertex, the parameters should have sufficient freedom that a perturbation in one parameter can be balanced by perturbations in the other parameters such that the perturbed objective is still optimal. In T_2, columns 1 and 4 can only be non-negative with the current parameter values – any perturbation cannot be balanced, leaving the objective non-optimal. Now column 4 enters the basis and column 2 leaves. The basis is now $\{0, 3, 4\}$ and pivoting gives:

$$T_3 = \begin{bmatrix} 1 & (-4\delta_1 - 2\delta_2) & (-4\delta_1 - 2\delta_2) & 0 & 0 & (-11\delta_1 - 5\delta_2) & (-11\delta_1 - 5\delta_2) & (-3\delta_1 - \delta_2) & (-\frac{3\delta_1}{2} - \frac{\delta_2}{2}) \\ 0 & 1 & 0 & 1 & 0 & 3 & 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & 1 & 0 & 1 & 1 & 3 & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

Observe that T_3 is a vertex – the columns with value zero (columns 1 and 2) can remain with non-negative entries when the values of δ_1 or δ_2 are perturbed. Next observe that no further pivots are available with $-1 \leq \delta_1 < -\frac{1}{2}$. Again, with $\delta_1 = -\frac{1}{2}$ fixed, there are no pivots available for any value $-1 \leq \delta_2 \leq 1$.

Returning to the original basis and tableau, and this time allowing the δ_2 parameter to vary, it can be observed that an alternative basis may be optimal when $\delta_2 = -1$, see column 7. When 7 enters the basis, 2 is chosen to leave the basis, giving basis $\{0, 1, 7\}$. Pivoting gives:

$$T_4 = \begin{bmatrix} 1 & 0 & (-2\delta_1 - 2\delta_2) & (4\delta_1 + 2\delta_2) & 2\delta_1 & (3\delta_1 + \delta_2) & (-\delta_1 - 3\delta_2) & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 2 & -2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 2 & 2 & 6 & 1 & \frac{1}{2} \end{bmatrix}$$

Again this represents a vertex. A further sequence of pivots is explored, with the basis becoming $\{0, 4, 7\}$ when $\delta_1 = \frac{1}{2}$ and $\delta_2 = -1$, then $\{0, 3, 4\}$ when $\delta_1 = \frac{1}{3}$, $\delta_2 = -1$. This leads again to the tableau T_3. No further vertices are generated, that is, the output is the three basis $\{0, 1, 2\}$, $\{0, 3, 4\}$, $\{0, 1, 7\}$ corresponding to the tableaux T_1, T_3 and T_4. The constant columns for these tableaux are:

$$\begin{bmatrix} -\frac{1}{2} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} \qquad \begin{bmatrix} -2 \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} \qquad \begin{bmatrix} 0 \\ \frac{1}{4} \\ 0 \end{bmatrix}$$

The basis and the weighting of the basis elements indicates how inequalities from the input are combined in order to give a non-redundant output inequality. In the $\{0, 1, 2\}$ basis the 1 and 2 inequalities are weighted equally giving $x + y \leq 2$, in the $\{0, 3, 4\}$ basis the 3 and 4 inequalities are weighted equally giving $-3x - y \leq -3$ and in the $\{0, 7, 2\}$ basis the 2 inequality is not weighted, giving the $0 \leq 1$ trivial constraint. That is, the output is, up to the trivial constraint, the non-redundant inequalities of the projection space.

4 Anytime Projection using Vertex Enumeration

This section explains how an anytime projection algorithm can be obtained through vertex enumeration, where each vertex is in one-to-one correspondence with an irredundant inequality in the projection (with the exception of a single vacuous inequality that is a by-product of the construction). To concisely formulate the projection problem consider the system $C\boldsymbol{x} + D\boldsymbol{y} \leq \boldsymbol{b}$ where C and D are matrices of coefficients of dimension $m \times d$ and $m \times d'$, $\boldsymbol{x}$ and $\boldsymbol{y}$ are d-ary and d'-ary vectors of (distinct) variables, and $\boldsymbol{b}$ is an m-ary vector of constants. The construction starts with the well-known projection lemma [36]. The lemma states that points in the projection satisfy linear combinations of the input inequalities:

Lemma 1. If $P = \{\boldsymbol{x} :: \boldsymbol{y} \in \mathbb{R}^{d+d'} \mid C\boldsymbol{x} + D\boldsymbol{y} \leq \boldsymbol{b}\}$ is a polyhedron, and $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}^m \mid D^T\boldsymbol{\lambda} = 0 \wedge \boldsymbol{\lambda} \geq 0\}$, then the projection of P onto $\boldsymbol{x}$ is given by

$$\pi_{\boldsymbol{x}}(P) = \{\boldsymbol{x} \in \mathbb{R}^d \mid \forall \boldsymbol{\lambda} \in \Lambda \,.\, \boldsymbol{\lambda}^T C\boldsymbol{x} \leq \boldsymbol{\lambda}^T \boldsymbol{b}\}$$

The next step in the construction is, on the face of it, rather odd. $C\boldsymbol{x} + D\boldsymbol{y} \leq \boldsymbol{b}$ is augmented with the vacuous inequality $0 \leq 1$. Thus let C' be the $m+1 \times d$ matrix where $C'_{m+1} = \boldsymbol{0}$, D' be the $m+1 \times d$ matrix where $D'_{m+1} = \boldsymbol{0}$, and $\boldsymbol{b'} = \boldsymbol{b} :: 1$ and $::$ denotes concatenation. To match against the previous section, define $E = [C' \mid \boldsymbol{b'}]$. The main result can now be stated:

Theorem 1. Suppose

$$P = \left\{\boldsymbol{x} :: \boldsymbol{y} \in \mathbb{R}^{d+d'} \,|C\boldsymbol{x} + D\boldsymbol{y} \leq \boldsymbol{b}\right\} \quad \Lambda' = \left\{\boldsymbol{\lambda'} \in \mathbb{R}^{m+1} \,\middle|\, D'^T\boldsymbol{\lambda'} = \boldsymbol{0} \;\wedge\; \boldsymbol{\lambda'} \geq \boldsymbol{0}\right\}$$

$$S = \left\{\boldsymbol{\alpha} :: \beta \in \mathbb{R}^{d+1} \;\middle|\; \exists \boldsymbol{\lambda'} \in \Lambda' \;\wedge\; \boldsymbol{\alpha} = C'^T\boldsymbol{\lambda'} \;\wedge\; \beta = \boldsymbol{b'}^T\boldsymbol{\lambda'}\right\}$$

and the plane $S' = \{(\boldsymbol{\alpha} :: \beta) \in \mathbb{R}^{d+1} \mid \boldsymbol{\alpha}^T\boldsymbol{c} + \beta = 1\}$ slices the cone S where $\boldsymbol{c} \in \mathbb{R}^d$. Then the following representation of $\pi_{\boldsymbol{x}}(P)$ is irredundant

$$\pi_{\boldsymbol{x}}(P) = \left\{\boldsymbol{x} \in \mathbb{R}^d \,\middle|\, \boldsymbol{\alpha}^T\boldsymbol{x} \leq \beta \;\wedge\; \boldsymbol{\alpha} :: \beta \in \mathit{vertex}(S \cap S') \;\wedge\; \boldsymbol{\alpha} :: \beta \neq \boldsymbol{0} :: 1\right\}$$

where $\mathit{vertex}(S \cap S')$ denotes the vertices of $S \cap S'$.

Proof.

- Let $\boldsymbol{\alpha} :: \beta \in S$ and $\boldsymbol{x} \in \pi_{\boldsymbol{x}}(P)$. Thus there exists $\boldsymbol{\lambda'} = \langle \lambda_1, \ldots, \lambda_{m+1} \rangle \in \Lambda'$ such that $\boldsymbol{\alpha} = C'^T\boldsymbol{\lambda'}$ and $\beta = \boldsymbol{b'}^T\boldsymbol{\lambda'}$. Let $\boldsymbol{\lambda} = \langle \lambda_1, \ldots, \lambda_m \rangle$. Since $D^T\boldsymbol{\lambda} = \boldsymbol{0}$ by lemma 1 it follows $\boldsymbol{\lambda}^T C\boldsymbol{x} \leq \boldsymbol{\lambda}^T\boldsymbol{b}$. But $\boldsymbol{\alpha} = C^T\boldsymbol{\lambda}$ hence $\boldsymbol{\alpha}^T = \boldsymbol{\lambda}^T C$ and $\beta = \boldsymbol{\lambda}^T\boldsymbol{b} + \lambda_{m+1}$ where $\lambda_{m+1} \geq 0$. Thus $\boldsymbol{\alpha}^T\boldsymbol{x} = \boldsymbol{\lambda}^T C\boldsymbol{x} \leq \boldsymbol{\lambda}^T\boldsymbol{b} \leq \beta$.
- Let $\boldsymbol{\alpha} :: \beta \in \mathit{vertex}(S \cap S')$ such that $\boldsymbol{\alpha} :: \beta \neq \boldsymbol{0} :: 1$. Suppose $\boldsymbol{\alpha} :: \beta = \mu_0 + \sum_{i=1}^{\ell} \mu_i(\boldsymbol{\alpha}_i :: \beta_i)$ for some $\boldsymbol{\alpha}_1 :: \beta_1, \ldots, \boldsymbol{\alpha}_\ell :: \beta_\ell \in S$ and $\mu_0 \geq 0$, $\mu_1 \geq 0$, $\ldots$, $\mu_\ell \geq 0$. Observe $\boldsymbol{0} :: 1 \in S \cap S'$ and put $\boldsymbol{\alpha}_0 :: \beta_0 = \boldsymbol{0} :: 1$. Thus $\boldsymbol{\alpha} :: \beta = \sum_{i=0}^{\ell} \mu_i(\boldsymbol{\alpha}_i :: \beta_i)$. But $1 = (\boldsymbol{\alpha} :: \beta)^T(\boldsymbol{c} :: 1) = \sum_{i=0}^{\ell} \mu_i(\boldsymbol{\alpha}_i :: \beta_i)^T(\boldsymbol{c} :: 1) = \sum_{i=0}^{\ell} \mu_i$ hence $1 = \sum_{i=0}^{\ell} \mu_i$. Since $\boldsymbol{\alpha} :: \beta \neq \boldsymbol{0} :: 1$ there exists $1 \leq i \leq \ell$ such that $\boldsymbol{\alpha} :: \beta = \boldsymbol{\alpha}_i :: \beta_i$ thus $\boldsymbol{\alpha}^T\boldsymbol{x} \leq \beta$ is irredundant. □

Note that the plane $\boldsymbol{\alpha}^T\boldsymbol{c} + \beta = 1$ used to define S' does not compromise generality. Indeed if it where $\boldsymbol{\alpha}^T\boldsymbol{c} + \beta.c_{n+1} = 1$ for some $c_{n+1} \in \mathbb{R}$ then it would follow that $c_{n+1} > 0$ since S' cuts the ray $\mathbf{0} :: 1$. Yet the translated plane $\boldsymbol{\alpha}^T\boldsymbol{c} + \beta.c_{n+1} = c_{n+1}$ also cuts the rays, hence the assumption $\boldsymbol{\alpha}^T\boldsymbol{c} + \beta = 1$. The force of the theorem is that it shows how inequalities in the projection can be found independently, one-by-one, except for the removal of the vacuous inequality. The sequel explains how vertex enumeration can be realised with PLP.

5 Vertex Enumeration using PLP

The algorithm to enumerate the vertices of the projection space using PLP is presented across Algorithms 1, 2 and 3, that are described separately below.

5.1 Vertex enumeration

Algorithm 1 takes as its argument the tableau of form C as described in section 3.5. The vector $\boldsymbol{\delta}$ represents the parameters of the objective. Each parameter δ_i assumes a value in range $[-1, 1]$ though initially $\boldsymbol{\delta} = \mathbf{1}$.

The algorithm uses a worklist WL of tableau/parameter pairs to drive the vertex enumeration. The output OP is a set of tableaux representing the vertices. The first step on line 3 of Algorithm 1 finds the tableau with the initial value of $\boldsymbol{\delta}$ optimised. The main loop removes a pair from the worklist, then for every parameter δ_i finds a tableau corresponding to an adjacent vertex and a corresponding value for that parameter such that the vertex is optimal.

These values are returned from the calls to nextVertex on lines 11 and 15, with first call searching for further vertices with the current parameters and the second call invoked when line 11 does not give a new tableau. Note that in some cases only δ_i changes its value and that *null* returns are possible for the tableau, indicating that pivoting is not possible. If no new tableau is found then δ_i is updated to -1 and this is added to the worklist (line 18). Otherwise, both the worklist and the output are updated.

5.2 Next parameter

Algorithm 2 returns for parameter δ_i the highest value less than its current value that induces a pivot. Again note that since it is optimal bases that are of interest, pivots occur when objective entries become zero, rather than when they are negative. Line 2 of the algorithm finds the set Δ' of values (less than δ_i) that the parameter can take in order that a non-basis objective entry can take value 0. Here $T_j^0(\boldsymbol{\delta})$ evaluates the objective entry in column j with parameters $\boldsymbol{\delta}$. If Δ' is non-empty the largest value less than the current value is returned (line 3). Otherwise the return is -1.

Algorithm 1 Vertex enumeration with PLP

```
1: function enumVertices(T_in)
2: WL = [], OP = [], δ = 1
3: T = maximise(T_in, δ)
4: WL.add(T, δ), OP.add(T)
5: while WL ≠ [] do
6:    (T, δ) = WL.remove()
7:    for i = 1 to |δ| do
8:       if δ_i ≠ −1 then
9:          T' = null
10:         if ∃j ∈ (T.cobasis).T_j^0(δ) = 0 then
11:            (T', δ') = nextVertex(T, δ, i)
12:         end if
13:         if T' = null then
14:            δ'' = nextDelta(T, δ, i)
15:            (T', δ') = nextVertex(T, δ'', i)
16:         end if
17:         if T' = null then
18:            WL.add(T, δ')
19:         else
20:            WL.add(T', δ'), OP.add(T')
21:         end if
22:      end if
23:   end for
24: end while
25: return OP
```

5.3 Next vertex

Algorithm 3 defines a recursive function that returns a tableau/parameter pair representing a vertex. That a tableau represents a vertex can be tested by solving a linear program describing that at a vertex the parameters should have sufficient freedom that a perturbation in one parameter can be balanced by perturbations in the others so that a pivot is not induced. Recall the example in section 3.5.

The algorithm performs a lexicographic pivot step with a candidate entering column at line 4. If the pivot leads to a basis that has already been generated, the resulting tableau is *null* and the loop moves on to the next j column. This avoids the cycling phenomena in which a vertex is visited repeatedly. Otherwise, if the new tableau/parameter pair does not represent a vertex, then the function calls itself, continuing its search for a vertex. The algorithm returns $(null, \boldsymbol{\delta})$ if there are no pivots available.

The combined effect of the three algorithms is to systematically explore the tableaux corresponding to optima with respect to the parameter space. By returning those tableaux corresponding to vertices the inequalities of the projection space can be found. In summary, projection is reduced to repeated pivoting.

Proposition 1. *Algorithm 1 is complete, that is, if* $\boldsymbol{\alpha} :: \beta \in vertex(S \cap S')$, $\boldsymbol{\alpha} :: \beta$ *is in its output.*

Algorithm 2 Finding the next δ value

```
1: function nextDelta(T, δ, i)
2: Δ' = {δ'_i | ∃j ∈ (T.cobasis).T^0_j(δ[i ↦ δ'_i]) = 0 ∧ δ'_i < δ_i}
3: δ*_i = max(Δ' ∪ {−1})
4: return δ[i ↦ δ*_i]
```

Algorithm 3 Finding the next vertex

```
1: function nextVertex(T, δ, i)
2: for j = 1 to |δ| do
3:    if j ∈ T.cobasis ∧ T^0_j(δ) = 0 then
4:       T' = T.pivot(j)
5:       if T' ≠ null then
6:          if T'.isVertex(δ) then
7:             return (T', δ)
8:          else
9:             return nextVertex(T', δ, i)
10:         end if
11:      end if
12:   end if
13: end for
14: return (null, δ)
```

Proof. (Outline) The algorithm is complete if it gives a series of pivots from the initial tableau to a tableau representing any vertex of the output space. Consider some vertex v, then there exists objective $\boldsymbol{\delta_v}$ such that a tableau for v is optimal with respect to $\boldsymbol{\delta_v}$. Now consider the initial tableau which is optimal with respect to $\mathbf{1}$. With objective $\boldsymbol{\delta_v}$ there must be a series of pivots from the initial tableau to that for vertex v. The parameter values $\boldsymbol{\delta}$ always suggest an entering column for pivoting. To see this consider a tableau representing vertex v' that is optimal with respect to $\boldsymbol{\delta}$: for some i a decrease in δ_i will suggest a pivot to a tableau that gives a higher value for $\boldsymbol{\delta_v}$ than that for v', hence this pivot must also be selectable when optimising $\boldsymbol{\delta_v}$ from v'. Therefore v is output by Algorithm 1. □

6 Related work

This paper can be considered to be a response to the agenda promoted by the weakly relational domains [14, 16, 25, 26, 33] which seek to curtail the expressiveness of the linear inequalities up front so as to recover tractability. These domains are classically formulated in terms of a closure operation which computes the planar shadows [1] of a higher-dimensional polyhedron defined over $x_1, \ldots, x_n$; one shadow for each x_i, x_j pair. Operations such as join are straightforward once the shadows are known. This hints at the centrality of projection in the design of a numeric domain, an idea that is taken to the limit in this paper. Other ingenious ways of realising weakly relational domains include representing inequalities with unary coefficients as binary decision diagrams [8], using an array

of size n to compactly represent a system of two variable equality constraints over n variables [13], and employing k-dimensional simplices as descriptions since they can be represented as $k+1$ linearly independent frame elements [31].

An interesting class of weakly relational domain are the template domains [30] in which the inequalities conform to patterns given prior to analysis, say, $ax_2 + bx_3 + cx_6 \leq d$. During analysis values for coefficients a, b and c (on the left) and constants d (on the right) are inferred using linear programming. This domain has recently been relaxed [9] so that the right-hand side can be generalised to any parametric two-variable linear expression. The advance in this work is that the domain operations can then be performed in an output sensitive way: the computational effort is governed not only by the size of the input but the output too, rather than that of any intermediate representation. Fractional linear programming [4] is used to simulate the Jarvis march [18] and thereby project a higher dimensional polyhedron onto a plane in an output sensitive fashion.

Further afield, finite-horizon optimal problems can be formulated as PLP [19]. PLPs allow the control action to be pre-computed off-line for every possible value of the parameter μ, simplifying an on-line implementation. In a study of how to solve PLPs, the link between PLP and projection has been explored [19], a connection that is hinted at in the seminal work on PLP [15]. Yet the foundation result of [19], lemma 3.2, overlooks the need to relax constants and only addresses the problem of the uniqueness of the representation of a vertex by making a general position assumption. This assumption is unrealistic in program analysis where polyhedra can be degenerate, hence the use of lexicographical pivoting in this work. As a separate work, there has been interest in realising PLP using reverse search [2, 20] though again making a general position assumption.

7 Conclusions

This paper has revisited the abstract domain of polyhedra, presenting a new algorithm to calculate projection. Apart from one trivial inequality that can be recognised syntactically the projection does not enumerate redundant inequalities, hence does not incur expensive post-processing. Moreover, if there are an excessively large number of inequalities in the projection then, since projection is computed incrementally, one inequality at a time, the calculation can be aborted prematurely yielding an over-approximation of the result without compromising soundness. The new algorithm is based on pivoting which is known to have fast implementations and even appears to be amenable to parallelisation. Since convex hull can be calculated using meet and projection [32] the presented algorithm can form the core of a polyhedra analysis.

Acknowledgements This work was funded by a Royal Society Industrial Fellowship number IF081178 and a Royal Society International Grant number JP101405. London Mathematical Society Scheme 7 enabled the authors to visit Freie Universität Berlin and they are indebted to Günter Rote for alerting them to the connection between projection and PLP. The authors also thank Darko Dimitrov, Axel Simon and Sriram Sankaranarayanan for interesting discussions.

References

1. N. Amenta and G. Ziegler. Shadows and Slices of Polytopes. In *Symposium on Computational Geometry*, pages 10–19. ACM Press, 1996.
2. D. Avis. lrs: A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm. In G. Kalai and G. M. Ziegler, editors, *Polytopes – Combinatorics and Computation*, pages 177–198. Brikhäuser Basel, 2000.
3. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. *Science of Computer Programming*, 72(1-2):3–21, 2008.
4. S. Boyd and S. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
5. E. Burger. Über Homogene Lineare Ungleichungssysteme. *Zeitschrift für Angewandte Mathematik und Mechanik*, 36:135–139, 1956.
6. N. V. Chernikova. Algorithm for Discovering the Set of All the Solutions of a Linear Programming Problem. *Computational Mathematics and Mathematical Physics*, 8(6):1387–1395, 1968.
7. V. Chvátal. *Linear Programming*. W. H. Freeman and Company, 1983.
8. R. Clarisó and J. Cortadella. The Octahedron Abstract Domain. *Science of Computer Programming*, 64(1):115–139, 2007.
9. M. Colón and S. Sankaranarayanan. Generalizing the Template Polyhedral Domain. In *European Symposium on Programming*, volume 6602 of *LNCS*, pages 176–195. Springer, 2011.
10. P. Cousot and N. Halbwachs. Automatic Discovery of Linear Restraints among Variables of a Program. In *Principles of Programming Languages*, pages 84–97. ACM Press, 1978.
11. G. B. Dantzig and W. Orchard-Hays. The Product Form for the Inverse in the Simplex Method. *Mathematical Tables and Other Aids to Computation*, 8(46):64–67, 1954.
12. R. J. Duffin. On Fourier's Analysis of Linear Inequality Systems. *Mathematical Programming Studies*, 1:71–95, 1974.
13. A. Flexeder, M. Müller-Olm, M. Petter, and H. Seidl. Fast Interprocedural Linear Two-Variable Equalities. *ACM Transactions on Programming Languages and Systems*, 33(6), 2011.
14. J. Fulara, K. Durnoga, K. Jakubczyk, and A. Shubert. Relational Abstract Domain of Weighted Hexagons. *Electronic Notes in Theoretical Computer Science*, 267(1):59–72, 2010.
15. S. Gass and T. Saaty. The Computational Algorithm for the Parametric Objective Function. *Naval Research Logistics Quarterly*, 2(1-2):39–45, 1955.
16. J. M. Howe and A King. Logahedra: a New Weakly Relational Domain. In *Automated Technology for Verification and Analysis*, volume 5799 of *LNCS*, pages 306–320. Springer, 2009.
17. J.-L. Imbert. Fourier's Elimination: Which to Choose? In *First Workshop on Principles and Practice of Constraint Programming*, pages 117–129, 1993.
18. R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973.
19. C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. On Polyhedral Projection and Parametric Programming. *Journal of Optimization Theory and Applications*, 138(2):207–220, 2008.

20. C. N. Jones and J. M. Maciejowski. Reverse Search for Parametric Linear Programming. In *IEEE Conference on Decision and Control*, pages 1504–1509, 2006.
21. L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating All Vertices of a Polyhedron is Hard. *Discrete and Computational Geometry*, 39:174–190, 2008.
22. D. A. Kohler. Projections of Convex Polyhedral Sets. Technical Report 67-29, Operations Research Centre, University of California, Berkeley, 1967.
23. J.-L. Lassez, T. Huynh, and K. McAloon. Simplification and Elimination of Redundant Linear Arithmetic Constraints. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming*, pages 73–87. MIT Press, 1993.
24. H. Le Verge. A Note on Chernikova's algorithm. Technical Report 1662, Institut de Recherche en Informatique, Campus Universitaire de Beaulieu, France, 1992.
25. A. Miné. A New Numeric Abstract Domain Based on Difference-Bound Matrices. In *Programs as Data Objects*, volume 2053 of *LNCS*, pages 155–172. Springer, 2001.
26. A. Miné. The Octagon Abstract Domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.
27. T. S. Motzkin. *Beiträge zur Theorie der Linearen Ungleichungen*. PhD thesis, Universität Zurich, 1936.
28. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The Double Description Method. In *Annals of Mathematics Studies*, volume 2, pages 51–73. Princeton University Press, 1953.
29. J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. On Computing Four-Finger Equilibrium and Force-Closure Grasps of Polyhedral Objects. *International Journal of Robotics Research*, 16(2):11–35, 1997.
30. S. Sankaranarayanan, M. Colón, H. Sipma, and Z. Manna. Efficient Strongly Relational Polyhedral Analysis. In *Verification, Model Checking and Abstract Interpretation*, volume 3855 of *LNCS*, pages 111–125. Springer, 2006.
31. H. Seidl, A. Flexeder, and M. Petter. Interprocedurally Analysing Linear Inequality Relations. In *European Symposium on Programming*, volume 4421, pages 284–299. Springer, 2007.
32. A. Simon and A. King. Exploiting Sparsity in Polyhedral Analysis. In *Static Analysis Symposium*, volume 3672 of *LNCS*, pages 336–351. Springer, 2005.
33. A. Simon, A. King, and J. M. Howe. Two Variables per Linear Inequality as an Abstract Domain. In M. Leuschel, editor, *Logic Based Program Development and Transformation*, volume 2664 of *LNCS*, pages 71–89. Springer, 2002.
34. M. J. Todd. The Many Facets of Linear Programming. *Mathematical Programming*, 91(3):417–436, 2002.
35. D. K. Wilde. A Library for Doing Polyhedral Operations. Technical Report 785, Institut de Recherche en Informatique, Campus Universitaire de Beaulieu, France, 1993.
36. G. M. Ziegler. *Lectures on Polytopes*. Springer, 2007.