Complexity Estimates for Fourier-Motzkin Elimination

Rui-Juan Jing[1] Marc Moreno-Maza[2] Delaram Talaashrafi[3]

[1] Jiangsu University, Zhenjiang, China rjing@ujs.edu.cn,

[2] Western University, London, Canada moreno@csd.uwo.ca,

[3] Western University, London, Canada dtalaash@uwo.ca

June 16, 2020

Abstract

In this paper, we propose an efficient method for removing all redundant inequalities generated by Fourier-Motzkin Elimination. This method is based on an improved version of Balas' work and can also be used to remove all redundant inequalities in the input system. Moreover, our method only uses arithmetic operations on matrices and avoids resorting to linear programming techniques. Algebraic complexity estimates and experimental results show that our method outperforms alternative approaches, in particular those based on linear programming and the simplex algorithm.

Keywords: Polyhedral set, Fourier-Motzkin Elimination, Algebraic complexity, Efficient implementation

1 Introduction

Polyhedral sets play an important role in computational sciences. For instance, they are used to model, analyze, transform and schedule for-loops of computer programs; we refer to the articles [13, 14, 17, 3, 5, 2, 34]. Of prime importance are the following operations on polyhedral sets: (i) conversion between H-representation and V-representation (performed, for instance, by the double description method); and (ii) projection, as performed by Fourier-Motzkin Elimination.

Although the double description method and Fourier-Motzkin Elimination have a lot in common, and, they are considered as the same algorithm in the paper [7] of Winfried Bruns and Bogdan Ichim, they are not totally similar. Quoting Komei Fukuda and Alian Prodon from [16]: "the FME algorithm is more general than the DD method, but often considered as the same method partly because it can be used to solve the extreme ray enumeration problem".

Fourier-Motzkin Elimination is an algorithmic tool for projecting a polyhedral set onto a linear subspace. It was proposed independently by Joseph Fourier and Theodore Motzkin, respectively in 1827 and 1936. See the paper [12] of George Danzing and Section 12.2 of in the book [31] of Alexander Schrijver, for a presentaion of Fourier-Motzkin Elimination. The original version of this algorithm produces

large amounts of redundant inequalities and has a double exponential algebraic complexity. Removing all these redundancies is equivalent to giving the minimal representation of the projection of a polyhedron. Leonid Khachiyan explained in [24] how linear programming (LP) could be used to remove all redundant inequalities, thereby reducing the cost of Fourier-Motzkin Elimination to a number of machine word operations singly exponential in the dimension of the ambient space. However, Khachiyan did not state a more precise running time estimate taking into account the characteristics of the polyhedron being projected, such as the number of its facets.

As we shall prove in this paper, rather than using linear programming one may use only matrix arithmetic, increasing the theoretical and practical efficiency of Fourier-Motzkin Elimination while still producing an irredundant representation of the projected polyhedron.

Other algorithms for projecting polyhedral sets remove some (but not all) redundant inequalities with the help of extreme rays: see the work of David A. Kohler [25]. As observed by Jean-Louis Imbert in [20], the method he proposed in that paper and that of Sergei N. Chernikov in [10] are equivalent. On the topic of finding extreme rays of a polyhedral set in H-representation, see Nataĺja V. Chernikova [11], Hervé Le Verge [26] and Komei Fukuda [16]. These methods are very effective in practice, but none of them can remove all redundant inequalities generated by Fourier-Motzkin Elimination.

Fourier-Motzkin Elimination is well suited for projecting a polyhedron, described by its facets (given by linear inequalities), onto different sub-spaces. And our paper is about projecting polyhedral sets to lower dimensions, eliminating one variable after another, thanks to Fourier-Motzkin Elimination algorithm as described in Schrijver's book [31]. In fact, our goal is to find the so-called *minimal representations* of all of the successive projections of a given polyhedron (in H-representation, thus given by linear inequalities), by eliminating variables one after another, using the Fourier-Motzkin Elimination algorithm. Computing these successive projections has applications in the analysis, scheduling and transformation of for loop nests of computer programs. For instance, after applying a uni-modular transformation to the loop counters of a for loop nest, the loop bounds of the new for loop nest are derived from the successive projections of a well-chosen polyhedron.

In this paper, we show how to remove all the redundant inequalities generated by Fourier-Motzkin Elimination. Our approach is based on an improved version of Balas' work [1] . To be more specific, we first compute a so-called initial redundancy test cone, from which we can derive the so-called redundancy test cone, which is used to detect the redundant inequalities generated after each elimination of a variable.

Consider as input a full-dimensional pointed polyhedron $Q \subseteq \mathbb{Q}^n$, given by a system of m linear inequalities of height h. We show, see Theorem 5, that eliminating the variables from that system, one after another (thus performing Fourier-Motzkin Elimination) can be done within $O(m^{\frac{5n}{2}} n^{\theta+1+\epsilon} h^{1+\epsilon})$ bit operations, for any $\epsilon > 0$, where θ is the exponent of linear algebra.

Therefore, we obtain a more favourable estimate than the one presented in [21, 22] for Fourier-Motzkin Elimination with a removal of the redundant inequalities via linear programming. Indeed, in those papers, the estimate is $O(n^2\, m^{2n}\, \mathsf{LP}(n, 2^n h n^2 m^n))$ bit operations, where $\mathsf{LP}(d, H)$ is an upper bound for the number of bit operations required for solving a linear program with total bit size H and with n vari-

ables. For instance, in the case of Karmarkar's algorithm [23], we have $\mathsf{LP}(d, H) \in O(d^{3.5}H^2 \cdot \log H \cdot \log\log H)$. Then, comparing the exponents of m, n and h, we have $\frac{5n}{2}$, $\theta + 1 + \epsilon$, $1 + \epsilon$ respectively with the method proposed in the present paper and $4n + \epsilon$, $6 + \epsilon$, $2 + \epsilon$ respectively with the estimate of [21, 22].

Our algorithm is stated in Section 4 and follows a revisited version of Balas' algorithm presented in Section 3. Since the maximum number of facets of any standard projection of Q is $O(m^{\lfloor n/2 \rfloor})$, our running time for Fourier-Motzkin Elimination is satisfactory; the other factors in our estimate come from the cost of linear algebra operations for testing redundancy.

We have implemented the algorithms proposed in Section 4 using the BPAS library [9] publicly available at `www.bpaslib.org`. We have compared our code against other implementations of Fourier-Motzkin Elimination including the CDD library [15]. Our experimental results, reported in Section 6, show that our proposed method can solve more test-cases (actually all) that we used while the counterpart software failed to solve some of them.

Section 2 provides background materials about polyhedral sets and polyhedral cones together with the original version of Fourier-Motzkin Elimination. Section 3 contains a revisited version of Balas' method and detailed proofs of its correctness. Based on this, Section 4 presents a new algorithm producing a *minimal projected representation* for a given full-dimensional pointed polyhedron. Complexity results are established in Section 5. In Section 6 we report on our experimentation and in Section 7 we discuss related works.

To summarize, our contributions are: (i) making Balas' algorithm to be practical, by devising a method for finding the initial redundancy test cone efficiently and using it in the Fourier-Motzkin Elimination, (ii) exhibiting the theoretical efficiency of the proposed algorithm by analyzing its bit complexity, and, (iii) demonstrating its practical effectiveness (implemented as part of the BPAS library) compared to other available related software

2 Background

In this section, we review the basics of polyhedral geometry. Section 2.1 is dedicated to the notions of polyhedral sets and polyhedral cones. Sections 2.2 and 2.3 review the double description method and Fourier-Motzkin elimination, which are two of the most important algorithms for operating on polyhedral sets. We conclude this section with the cost model that we shall use for complexity analysis, see Section 2.4. As we omit most proofs, for more details please refer to [31, 33, 16]. For the sake of simplicity in the complexity analysis of the presented algorithms, we constraint our coefficient field to the field of rational numbers $\mathbb{Q}$. However, all of the algorithms presented in this paper apply to polyhedral sets with coefficients in the field real numbers $\mathbb{R}$.

2.1 Polyhedral cones and polyhedra

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 matrix A. For

a matrix A and an integer k, A_k is the row of index k in A. Also, if K is a set of integers, A_K denotes the sub-matrix of A with row indices in K.

Polyhedral cone. A subset of points $C \subseteq \mathbb{Q}^n$ is called a *cone* if for each $\mathbf{x} \in C$ and each real number $\lambda \geq 0$ we have $\lambda\mathbf{x} \in C$. A cone $C \subseteq \mathbb{Q}^n$ is called *convex* if for all $\mathbf{x}, \mathbf{y} \in C$, we have $\mathbf{x} + \mathbf{y} \in C$. If $C \subseteq \mathbb{Q}^n$ is a convex cone, then its elements are called the *rays* of C. For two rays $\mathbf{r}$ and $\mathbf{r}'$ of C, we write $\mathbf{r}' \simeq \mathbf{r}$ whenever there exists $\lambda \geq 0$ such that we have $\mathbf{r}' = \lambda\mathbf{r}$. A cone $C \subseteq \mathbb{Q}^n$ is a *polyhedral cone* if it is the intersection of finitely many half-spaces, that is, $C = \{x \in \mathbb{Q}^n \mid Ax \leq \mathbf{0}\}$ for some matrix $A \in \mathbb{Q}^{m\times n}$. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a set of vectors in $\mathbb{Q}^n$. The *cone generated* by $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, denoted by $\mathsf{Cone}(\mathbf{x}_1, \cdots, \mathbf{x}_m)$, is the smallest convex cone containing those vectors. In other words, we have $\mathsf{Cone}(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \{\lambda_1\mathbf{x}_1 + \cdots + \lambda_m\mathbf{x}_m \mid \lambda_1 \geq 0, \ldots, \lambda_m \geq 0\}$. A cone obtained in this way is called a *finitely generated cone*.

Polyhedron. A set of vectors $P \subseteq \mathbb{Q}^n$ is called a *convex polyhedron* if $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, for a matrix $A \in \mathbb{Q}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{Q}^m$. Moreover, the polyhedron P is called a *polytope* if P is bounded. From now on, we always use the notation $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ to represent a polyhedron in $\mathbb{Q}^n$. The system of linear inequalities $\{A\mathbf{x} \leq \mathbf{b}\}$ is called a *representation* of P. We say an inequality $\mathbf{c}^t\mathbf{x} \leq c_0$ is *redundant* w.r.t. a polyhedron representation $A\mathbf{x} \leq \mathbf{b}$ if it is implied by $A\mathbf{x} \leq \mathbf{b}$. A representation of a polyhedron is *minimal* if no inequality of that representation is implied by the other inequalities of that representation. To obtain a minimal representation for the polyhedron P, we need to remove all the redundant inequalities in its representation. This requires the famous Farkas' lemma. Since it has many different variants, here we only mention a variant from [31], which is applicable in the next algorithms.

Lemma 1 (Farkas' lemma) *Let $A \in \mathbb{Q}^{m\times n}$ be a matrix and $\mathbf{b} \in \mathbb{Q}^m$ be a vector. Then, there exists a vector $\mathbf{t} \in \mathbb{Q}^n$, $\mathbf{t} \geq \mathbf{0}$ satisfying $A\mathbf{t} = \mathbf{b}$ if and if $\mathbf{y}^t\mathbf{b} \geq 0$ holds for each vector $\mathbf{y} \in \mathbb{Q}^m$ such that we have $\mathbf{y}^tA \geq 0$.*

A consequence of Farkas' lemma is the following criterion for testing whether an inequality $\mathbf{c}^t\mathbf{x} \leq c_0$ is redundant w.r.t. a polyhedron representation $A\mathbf{x} \leq \mathbf{b}$.

Lemma 2 (Redundancy test criterion) *Let $\mathbf{c} \in \mathbb{Q}^n$, $c_0 \in \mathbb{Q}$, $A \in \mathbb{Q}^{m\times n}$ and $\mathbf{b} \in \mathbb{Q}^m$. Then, the inequality $\mathbf{c}^t\mathbf{x} \leq c_0$ is redundant w.r.t. the system of inequalities $A\mathbf{x} \leq \mathbf{b}$ if and only if there exists a vector $\mathbf{t} \geq \mathbf{0}$ and a number $\lambda \geq 0$ satisfying $\mathbf{c}^t = \mathbf{t}^tA$ and $c_0 = \mathbf{t}^t\mathbf{b} + \lambda$.*

Characteristic Cone and Pointed Polyhedron. The *characteristic cone* of P is the polyhedral cone denoted by $\mathsf{CharCone}(P)$ and defined by $\mathsf{CharCone}(P) = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{x} + \mathbf{y} \in P,\ \forall \mathbf{x} \in P\} = \{\mathbf{y} \mid A\mathbf{y} \leq \mathbf{0}\}$. The *linearity space* of the polyhedron P is the linear space denoted by $\mathsf{LinearSpace}(P)$ and defined as $\mathsf{CharCone}(P) \cap -\mathsf{CharCone}(P) = \{\mathbf{y} \mid A\mathbf{y} = \mathbf{0}\}$, where $-\mathsf{CharCone}(P)$ is the set of the $-\mathbf{y}$ for $\mathbf{y} \in \mathsf{CharCone}(P)$. The polyhedron P is *pointed* if its linearity space is $\{\mathbf{0}\}$.

Lemma 3 (Pointed polyhedron criterion) *The polyhedron P is pointed if and only if the matrix A is full column rank.*

Extreme point and extreme ray. The *dimension* of the polyhedron P, denoted by $\dim(P)$, is the maximum number of linearly independent vectors in P. We say

that P is *full-dimensional* whenever $\dim(P) = n$ holds. 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 $\mathbf{x} \in P$. Then, P is full-dimensional if and only if it does not have any implicit equations. A subset F of the polyhedron P is called a *face* of P if F equals $\{\mathbf{x} \in P \mid A_{\text{sub}}\mathbf{x} = \mathbf{b}_{\text{sub}}\}$ for a sub-matrix A_{sub} of A and a sub-vector $\mathbf{b}_{\text{sub}}$ of $\mathbf{b}$. A face of P, distinct from P and of maximum dimension is called a *facet* of P. A non-empty face that does not contain any other face of a polyhedron is called a *minimal face* of that polyhedron. Specifically, if the polyhedron P is pointed, each minimal face of P is just a point and is called an *extreme point* or *vertex* of P. Let C be a cone such that $\dim(\mathsf{LinearSpace}(C)) = t$. Then, a face of C of dimension $t + 1$ is called a minimal proper face of C. In the special case of a pointed cone, that is, whenever $t = 0$ holds, the dimension of a minimal proper face is 1 and such a face is called an *extreme ray*. We call an *extreme ray* of the polyhedron P any extreme ray of its characteristic cone $\mathsf{CharCone}(P)$. We say that two extreme rays $\mathbf{r}$ and $\mathbf{r}'$ of the polyhedron P are *equivalent*, and denote it by $\mathbf{r} \simeq \mathbf{r}'$, if one is a positive multiple of the other. When we consider the set of all extreme rays of the polyhedron P (or the polyhedral cone C) we will only consider one ray from each equivalence class. A pointed cone C can be generated by its extreme rays, that is, we have $C = \{\mathbf{x} \in \mathbb{Q}^n \mid (\exists \mathbf{c} \geq \mathbf{0})\ \mathbf{x} = R\mathbf{c}\}$, where the columns of R are the extreme rays of C. We denote by $\mathsf{ExtremeRays}(C)$ the set of extreme rays of the cone C. Recall that all cones considered here are polyhedral. The following, see [28, 33], is helpful in the analysis of algorithms manipulating extreme rays of cones and polyhedra. Let $E(C)$ be the number of extreme rays of a polyhedral cone $C \in \mathbb{Q}^n$ with m facets. Then, we have:

$$E(C) \leq \binom{m - \lfloor \frac{n+1}{2} \rfloor}{m-1} + \binom{m - \lfloor \frac{n+2}{2} \rfloor}{m-n} \leq m^{\lfloor \frac{n}{2} \rfloor}. \tag{1}$$

Algebraic test of (adjacent) extreme rays. Given a cone $C = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{0}\}$ and $\mathbf{t} \in C$, we define the *zero set* $\zeta_A(\mathbf{t})$ as the set of row indices i such that $A_i\mathbf{t} = 0$, where A_i is the i-th row of A. For simplicity, we use $\zeta(\mathbf{t})$ instead of $\zeta_A(\mathbf{t})$ when there is no ambiguity. The proof of the following, which we have so-called the *algebraic test*, can be found in [16]: Let $\mathbf{r} \in C$. Then, the ray $\mathbf{r}$ is an extreme ray of C if and only if we have $\text{rank}(A_{\zeta(r)}) = n - 1$. Two distinct extreme rays $\mathbf{r}$ and $\mathbf{r}'$ of the polyhedral cone C are called *adjacent* if they span a 2-dimensional face of C. From [16], we have: Two distinct extreme rays, $\mathbf{r}$ and $\mathbf{r}'$, of C are adjacent if and only if $\text{rank}(A_{\zeta(\mathbf{r}) \cap \zeta(\mathbf{r}')}) = n - 2$ holds.

Polar cone. Given a polyhedral cone $C \subseteq \mathbb{Q}^n$, the *polar cone* induced by C, denoted by C^*, is defines as: $C^* = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{y}^t\mathbf{x} \leq \mathbf{0}, \forall \mathbf{x} \in C\}$. The proof of the following property can be found in [31]: For a given cone $C \in \mathbb{Q}^n$, there is a one-to-one correspondence between the faces of C of dimension k and the faces of C^* of dimension $n - k$. In particular, there is a one-to-one correspondence between the facets of C and the extreme rays of C^*.

Homogenized cone. The *homogenized cone* of the polyhedron $P = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{b}\}$ is denoted by $\mathsf{HomCone}(P)$ and defined by: $\mathsf{HomCone}(P) = \{(\mathbf{x}, x_{\text{last}}) \in \mathbb{Q}^{n+1} \mid A\mathbf{x} - \mathbf{b}x_{\text{last}} \leq \mathbf{0}, x_{\text{last}} \geq 0\}$.

Lemma 4 (H-representation correspondence) *An inequality $A_i\mathbf{x} \leq b_i$ is redundant in P if and only if the corresponding inequality $A_i\mathbf{x} - b_i x_{\text{last}} \leq 0$ is redundant in $\mathsf{HomCone}(P)$.*

Theorem 1 (Extreme rays of the homogenized cone) *Every extreme ray of the homogenized cone* $\mathsf{HomCone}(P)$ *associated with the polyhedron* P *is either of the form* $(\mathbf{x}, 0)$ *where* $\mathbf{x}$ *is an extreme ray of* P*, or* $(\mathbf{x}, 1)$ *where* $\mathbf{x}$ *is an extreme point of* P*.*

2.2 The double description method

It follows from Section 2.1 that any pointed polyhedral cone C can be represented either as the intersection of finitely many half-spaces (given as a system of linear inequalities $A\mathbf{x} \leq \mathbf{0}$ and called *H-representation* of C) or as $\mathsf{Cone}(R)$, where R is a matrix, the columns of which are the extreme rays of C (called *V-representation* of C). The *double description method*, proposed by Komei Fukuda in [16] and implemented in the CDD library, produces the V-representation of a pointed polyhedral cone given in the H-representation. Some of the results presented in our paper depend on an algebraic complexity estimate for the double description method. In [16], one can find an estimate in terms of arithmetic operations on the coefficients of the input H-representation. Since we need a bit complexity estimate, we provide one as Lemma 8. A proof for it can be found in Section 8 and a brief review of the DD method is given in Appendix page 23.

2.3 Fourier-Motzkin elimination

Let $A \in \mathbb{Q}^{m\times p}$ and $B \in \mathbb{Q}^{m\times q}$ be matrices. Let $\mathbf{c} \in \mathbb{Q}^m$ be a vector. Consider the polyhedron $P = \{(\mathbf{u}, \mathbf{x}) \in \mathbb{Q}^{p+q} \mid A\mathbf{u} + B\mathbf{x} \leq \mathbf{c}\}$. We denote by $\mathsf{proj}(P; \mathbf{x})$ the *projection of* P *on* $\mathbf{x}$, that is, the subset of $\mathbb{Q}^q$ defined by $\mathsf{proj}(P; \mathbf{x}) = \{\mathbf{x} \in \mathbb{Q}^q \mid \exists\ \mathbf{u} \in \mathbb{Q}^p,\ (\mathbf{u}, \mathbf{x}) \in P\}$.

Fourier-Motzkin elimination (FME for short) is an algorithm computing the projection $\mathsf{proj}(P; \mathbf{x})$ of the polyhedron of P by successively eliminating the $\mathbf{u}$-variables from the inequality system $A\mathbf{u} + B\mathbf{x} \leq \mathbf{c}$. This process shows that $\mathsf{proj}(P; \mathbf{x})$ is also a polyhedron.

Let ℓ_1, ℓ_2 be two inequalities: $a_1x_1 + \cdots + a_nx_n \leq c_1$ and $b_1x_1 + \cdots + b_nx_n \leq c_2$. Let $1 \leq i \leq n$ such that the coefficients a_i and b_i of x_i in ℓ_1 and ℓ_2 are positive and negative, respectively. The *combination* of ℓ_1 and ℓ_2 w.r.t. x_i, denoted by $\mathsf{Combine}(\ell_1, \ell_2, x_i)$, is:

$$-b_i(a_1x_1 + \cdots + a_nx_n) + a_i(b_1x_1 + \cdots + b_nx_n) \leq -b_ic_1 + a_ic_2.$$

Theorem 2 shows how to compute $\mathsf{proj}(P; \mathbf{x})$ when $\mathbf{u}$ consists of a single variable x_i. When $\mathbf{u}$ consists of several variables, FME obtains the projection $\mathsf{proj}(P; \mathbf{x})$ by repeated applications of Theorem 2.

Theorem 2 (Fourier-Motzkin theorem [25]) *Let* $A \in \mathbb{Q}^{m\times n}$ *be a matrix and let* $\mathbf{c} \in \mathbb{Q}^m$ *be a vector. Consider the polyhedron* $P = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{c}\}$*. Let* S *be the set of inequalities defined by* $A\mathbf{x} \leq \mathbf{c}$*. Also, let* $1 \leq i \leq n$*. We partition* S *according to the sign of the coefficient of* x_i*:* $S^+ = \{\ell \in S \mid \mathrm{coeff}(\ell, x_i) > 0\}$*,* $S^- = \{\ell \in S \mid \mathrm{coeff}(\ell, x_i) < 0\}$ *and* $S^0 = \{\ell \in S \mid \mathrm{coeff}(\ell, x_i) = 0\}$*. We construct the following system of linear inequalities:*

$$S' = \{\mathsf{Combine}(s_p, s_n, x_i) \mid (s_p, s_n) \in S^+ \times S^-\} \ \cup\ S^0.$$

Then, S' is a representation of $\mathsf{proj}(P; \{\mathbf{x} \setminus \{x_i\}\})$.

With the notations of Theorem 2, assume that each of S^+ and S^- counts $\frac{m}{2}$ inequalities. Then, the set S' counts $(\frac{m}{2})^2$ inequalities. After eliminating p variables, the projection would be given by $O((\frac{m}{2})^{2^p})$ inequalities. Thus, FME is *double exponential* in p.

On the other hand, from [29] and [22], we know that the maximum number of facets of the projection on $\mathbb{Q}^{n-p}$ of a polyhedron in $\mathbb{Q}^n$ with m facets is $O(m^{\lfloor n/2 \rfloor})$. Hence, it can be concluded that most of the generated inequalities by FME are redundant. Eliminating these redundancies is the main subject of the subsequent sections.

2.4 Cost model

for any rational number $\frac{a}{b}$, thus with $b \neq 0$, we define the *height* of $\frac{a}{b}$, denoted as $\mathsf{height}(\frac{a}{b})$, as $\log \max(|a|, |b|)$. For a given matrix $A \in \mathbb{Q}^{m \times n}$, let $\|A\|$ denote the infinite norm of A, that is, the maximum absolute value of a coefficient in A. We define the height of A, denoted by $\mathsf{height}(A) := \mathsf{height}(\|A\|)$, as the maximal height of a coefficient in A. For the rest of this section, our main reference is the PhD thesis of Arne Storjohann [32]. Let k be a non-negative integer. We denote by $\mathcal{M}(k)$ an upper bound for the number of bit operations required for performing any of the basic operations (addition, multiplication, and division with reminder) on input $a, b \in \mathbb{Z}$ with $|a|, |b| < 2^k$. Using the multiplication algorithm of Arnold Schönhage and Volker Strassen [30] one can choose $\mathcal{M}(k) \in O(k \log k \log \log k)$.

We also need complexity estimates for some matrix operations. For positive integers a, b, c, let us denote by $\mathcal{MM}(a, b, c)$ an upper bound for the number of arithmetic operations (on the coefficients) required for multiplying an $(a \times b)$-matrix by an $(b \times c)$-matrix. In the case of square matrices of order n, we simply write $\mathcal{MM}(n)$ instead of $\mathcal{MM}(n, n, n)$. We denote by θ the exponent of linear algebra, that is, the smallest real positive number such that $\mathcal{MM}(n) \in O(n^\theta)$.

In the following, we give complexity estimates in terms of $\mathcal{M}(k) \in O(k \log k \log \log k)$ and $\mathcal{B}(k) = \mathcal{M}(k) \log k \in O(k(\log k)^2 \log \log k)$. We replace every term of the form $(\log k)^p (\log \log k)^q (\log \log \log k)^r$, (where p, q, r are positive real numbers) with $O(k^\epsilon)$ where ϵ is a (positive) infinitesimal. Furthermore, in the complexity estimates of algorithms operating on matrices and vectors over $\mathbb{Z}$, we use a parameter β, which is a bound on the magnitude of the integers occurring during the algorithm. Our complexity estimates are measured in terms of *machine word operations.* Let $A \in \mathbb{Z}^{m \times n}$ and $B \in \mathbb{Z}^{n \times p}$. Then, the product of A by B can be computed within $O(\mathcal{MM}(m, n, p)(\log \beta) + (mn + np + mp)\mathcal{B}(\log \beta))$ word operations, where $\beta = n \|A\| \|B\|$ and $\|A\|$ (resp. $\|B\|$) denotes the maximum absolute value of a coefficient in A (resp. B). Neglecting logarithmic factors, this estimate becomes $O(\max(m, n, p)^\theta \max(h_A, h_b))$ where $h_A = \mathsf{height}(A)$ and $h_B = \mathsf{height}(B)$. For a matrix $A \in \mathbb{Z}^{m \times n}$, a cost estimate of Gauss-Jordan transform is $O(nmr^{\theta-2}(\log \beta) + nm(\log r)\mathcal{B}(\log \beta))$ word operations, where r is the rank of the input matrix A and $\beta = (\sqrt{r}\|A\|)^r$. Let h be the height of A, for a matrix $A \in \mathbb{Z}^{m \times n}$, with height h, the rank of A is computed within $O(mn^{\theta+\epsilon}h^{1+\epsilon})$ word operations, and the inverse of A (when this matrix is invertible over $\mathbb{Q}$ and $m = n$) is computed within $O(m^{\theta+1+\epsilon}h^{1+\epsilon})$ word operations. Let $A \in \mathbb{Z}^{n \times n}$ be an integer

matrix, which is invertible over $\mathbb{Q}$. Then, the absolute value of any coefficient in A^{-1} (inverse of A) can be bounded up to $(\sqrt{n-1}\|A\|^{(n-1)})$.

3 Revisiting Balas' method

As recalled in Section 2, FME produces a representation of the projection of a polyhedron by eliminating one variable at a time. However, this procedure generates lots of redundant inequalities limiting its use in practice to polyhedral sets with a handful of variables only. In this section, we propose an efficient algorithm which generates a minimal representation of a full-dimensional pointed polyhedron, as well as its projections. Through this section, we use Q to denote a full-dimensional pointed polyhedron in $\mathbb{Q}^n$, where

$$Q = \{(\mathbf{u}, \mathbf{x}) \in \mathbb{Q}^p \times \mathbb{Q}^q \mid A\mathbf{u} + B\mathbf{x} \leq \mathbf{c}\}, \tag{2}$$

with $A \in \mathbb{Q}^{m\times p}$, $B \in \mathbb{Q}^{m\times q}$ and $\mathbf{c} \in \mathbb{Q}^m$. Thus, Q has no implicit equations in its representation and the coefficient matrix $[A, B]$ has full column rank. Our goal in this section is to compute the minimal representation of the projection $\mathsf{proj}(Q; \mathbf{x})$ given by $\mathsf{proj}(Q; \mathbf{x}) := \{\mathbf{x} \mid \exists \mathbf{u}, s.t.(\mathbf{u}, \mathbf{x}) \in Q\}$. We call the cone $C := \{\mathbf{y} \in \mathbb{Q}^m \mid \mathbf{y}^t A = 0 \text{ and } \mathbf{y} \geq \mathbf{0}\}$ the *projection cone* of Q w.r.t.$\mathbf{u}$. When there is no ambiguity, we simply call C the projection cone of Q. Using the following so-called *projection lemma*, we can compute a representation for the projection $\mathsf{proj}(Q; \mathbf{x})$:

Lemma 5 ([10]) *The projection* $\mathsf{proj}(Q; \mathbf{x})$ *of the polyhedron* Q *can be represented by*

$$S := \{\mathbf{y}^t B\mathbf{x} \leq \mathbf{y}^t\mathbf{c}, \forall \mathbf{y} \in \mathsf{ExtremeRays(C)}\},$$

where C *is the projection cone of* Q *defined above.*

Lemma 5 provides the main idea of the block elimination method. However, the represention produced in this way may have redundant inequalities. In [1], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron (the proof of this fact is similar to the proof of our Theorem 3). Using this fact, Balas developed an algorithm to find all redundant inequalities for all cases, including the cases where B is singular.

In this section, we will explain the Balas' algorithm [1] in detail. To achieve this, we lift the polyhedron Q to a space in higher dimension by constructing the following objects.

Construction of B_0. Assume that the first q rows of B, denoted as B_1, are independent. Denote the last $m-q$ rows of B as B_2. Add $m-q$ columns, $\mathbf{e}_{q+1}, \ldots, \mathbf{e}_m$, to B, where $\mathbf{e}_i$ is the i-th vector in the canonical basis of $\mathbb{Q}^m$, thus with 1 in the i-th position and 0's anywhere else. The matrix B_0 has the following form:

$$B_0 = \begin{bmatrix} B_1 & \mathbf{0} \\ B_2 & I_{m-q} \end{bmatrix}.$$

[1] It should be noted that, although we are using his idea, we have found a flaw in Balas' paper. In fact, the last inequality in representation of W^0 is written as equality that paper.

To maintain consistency in the notation, let $A_0 = A$ and $\mathbf{c}_0 = \mathbf{c}$.

Construction of Q^0. We define:

$$Q^0 := \{(\mathbf{u}, \mathbf{x}') \in \mathbb{Q}^p \times \mathbb{Q}^m \mid A_0\mathbf{u} + B_0\mathbf{x}' \leq \mathbf{c}_0 \ , \ x_{q+1} = \cdots = x_m = 0\}.$$

From now on, we use $\mathbf{x}'$ to represent the vector $\mathbf{x} \in \mathbb{Q}^q$, augmented with $m - q$ variables $(x_{q+1}, \ldots, x_m)$. Since the extra variables $(x_{q+1}, \ldots, x_m)$ are assigned to zero, we note that $\mathsf{proj}(Q; \mathbf{x})$ and $\mathsf{proj}(Q^0; \mathbf{x}')$ are "isomorphic" by means of the bijection Φ:

$$\Phi : \begin{array}{c} \mathsf{proj}(Q; \mathbf{x}) \to \mathsf{proj}(Q^0; \mathbf{x}') \\ (x_1, \ldots, x_q) \mapsto (x_1, \ldots, x_q, 0, \ldots, 0) \end{array}$$

In the following, we will treat $\mathsf{proj}(Q; \mathbf{x})$ and $\mathsf{proj}(Q^0; \mathbf{x}')$ as the same polyhedron when there is no ambiguity.

Construction of W^0. Define W^0 to be the set of all $(\mathbf{v}, \mathbf{w}, v_0) \in \mathbb{Q}^q \times \mathbb{Q}^{m-q} \times \mathbb{Q}$ satisfying

$$\begin{aligned} W^0 = \{(\mathbf{v}, \mathbf{w}, v_0) \mid [\mathbf{v}^t, \mathbf{w}^t]B_0^{-1}A_0 = 0, [\mathbf{v}^t, \mathbf{w}^t]B_0^{-1} \geq 0, \\ - [\mathbf{v}^t, \mathbf{w}^t]B_0^{-1}\mathbf{c}_0 + v_0 \geq 0\}. \end{aligned} \tag{3}$$

Similar to the discussion in Balas' work, the extreme rays of the cone $\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})$ are used to construct the minimal representation of the projection $\mathsf{proj}(Q; \mathbf{x})$.

Theorem 3 shows that extreme rays of the cone $\overline{\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})}$, which is defined as

$$\overline{\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})} := \{(\mathbf{v}, -v_0) \mid (\mathbf{v}, v_0) \in \mathsf{proj}(W^0; \{\mathbf{v}, v_0\})\},$$

are in one-to-one correspondence with the facets of the homogenized cone of $\mathsf{proj}(Q; \mathbf{x})$. As a result its extreme rays can be used to find the minimal representation of $\mathsf{HomCone}(\mathsf{proj}(Q; \mathbf{x}))$.

Theorem 3 *The polar cone of* $\mathsf{HomCone}(\mathsf{proj}(Q; \mathbf{x}))$ *is equal to* $\overline{\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})}$.

Proof ▷ Please refer to Section 8.1 ◁

Theorem 4 *The minimal representation of* $\mathsf{proj}(Q; \mathbf{x})$ *is given exactly by*

$$\{\mathbf{v}^t\mathbf{x} \leq v_0 \mid (\mathbf{v}, v_0) \in \mathsf{ExtremeRays}(\mathsf{proj}(W^0; (\mathbf{v}, v_0))) \setminus \{(\mathbf{0}, 1)\}\}.$$

Proof ▷ By Theorem 3, a minimal representation of the homogenized cone $\mathsf{HomCone}(\mathsf{proj}(Q; \mathbf{x}))$ is given exactly by $\{\mathbf{v}\mathbf{x} - v_0 x_{\text{last}} \leq 0 \mid (\mathbf{v}, v_0) \in \mathsf{ExtremeRays}(\mathsf{proj}(W^0; (\mathbf{v}, v_0)))\}$. Using Lemma 4, any minimal representation of $\mathsf{HomCone}(\mathsf{proj}(Q; \mathbf{x}))$ has at most one more inequality than any minimal representation of $\mathsf{proj}(Q; \mathbf{x})$. This extra inequality would be $x_{\text{last}} \geq 0$ and, in this case, $\mathsf{proj}(W^0; (\mathbf{v}, v_0))$ would have the extreme ray $(\mathbf{0}, 1)$, which can be detected easily. Therefore, a minimal representation of $\mathsf{proj}(Q; \mathbf{x})$ is given by $\{\mathbf{v}^t\mathbf{x} \leq v_0 \mid (\mathbf{v}, v_0) \in \mathsf{ExtremeRays}(\mathsf{proj}(W^0; (\mathbf{v}, v_0))) \setminus \{(\mathbf{0}, 1)\}\}$. ◁

For simplicity, we call the cone $\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})$ the *redundancy test cone* of Q w.r.t. $\mathbf{u}$ and denote it by $\mathcal{P}_{\mathbf{u}}(Q)$. When $\mathbf{u}$ is empty, we define $\mathcal{P}(Q) := \mathcal{P}_{\mathbf{u}}(Q)$ and we call it the *initial redundancy test cone*. It should be noted that $\mathcal{P}(Q)$ can be used to detect redundant inequalities in the input system, as it is shown in Steps 3 to 8 of Algorithm 3.

4 Minimal representation of the projected polyhedron

In this section, we present our algorithm for removing all the redundant inequalities generated during Fourier-Motzkin elimination. Our algorithm detects and eliminates redundant inequalities, right after their generation, using the redundancy test cone introduced in Section 3. Intuitively, we need to construct the cone W^0 and obtain a representation of the redundancy test cone, $\mathcal{P}_{\mathbf{u}}(Q)$, where $\mathbf{u}$ is the vector of eliminated variables, each time we eliminate a variable during FME. This method is time consuming because it requires to compute the projection of W^0 onto $\{\mathbf{v}, v_0\}$ space at each step. However, as we prove in Lemma 6, we only need to compute the initial redundancy test cone, using Algorithm 1, and the redundancy test cones, used in the subsequent variable eliminations, can be found incrementally without any extra cost. After generating the redundancy test cone, the algorithm, using Algorithm 2, keeps the newly generated inequality only if it is an extreme ray of the redundancy test cone.

Note that a byproduct of this process is the *minimal projected representation* of the input system, according to the specified variable ordering. This representation is useful for finding solutions of linear inequality systems. The projected representation was introduced in [21, 22] and will be reviewed in Definition 1.

For convenience, we rewrite the input polyhedron Q defined in Equation (2) as: $Q = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{A}\mathbf{y} \leq \mathbf{c}\}$, where $\mathbf{A} = [A, B] \in \mathbb{Q}^{m \times n}$, $n = p + q$ and $\mathbf{y} = [\mathbf{u}^t, \mathbf{x}^t]^t \in \mathbb{Q}^n$. We assume the first n rows of $\mathbf{A}$ are linearly independent.

Algorithm 1 Generate initial redundancy test cone

Input: $S = \{\mathbf{A}\mathbf{y} \leq \mathbf{c}\}$, a representation of the input polyhedron Q;
Output: $\mathcal{P}$, a representation of the initial redundancy test cone;

1: Construct $\mathbf{A}_0$ in the same way we constructed B_0, that is, $\mathbf{A}_0 := [\mathbf{A}, \mathbf{A}']$, where $\mathbf{A}' = [\mathbf{e}_{n+1}, \ldots, \mathbf{e}_m]$ with $\mathbf{e}_i$ being the i-th vector of the canonical basis of $\mathbb{Q}^m$;
2: Let $W := \{(\mathbf{v}, \mathbf{w}, v_0) \in \mathbb{Q}^n \times \mathbb{Q}^{m-n} \times \mathbb{Q} \mid -[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}\mathbf{c} + v_0 \geq 0, [\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1} \geq \mathbf{0}\}$;
3: $\mathcal{P} = \mathsf{proj}(W; \{\mathbf{v}, v_0\})$;
4: **return** $(\mathcal{P})$;

Remark 1 *There are two important points about Algorithm 1. First, we only need a representation of the initial redundancy test cone. This representation does not need to be minimal. Therefore, calling Algorithm 1 in Algorithm 3 (which computes a minimal projected representation of a polyhedron) does not lead to a recursive call to Algorithm 3. Second, to compute the projection* $\mathsf{proj}(W; \{\mathbf{v}, v_0\})$*, we need to eliminate* $m - n$ *variables from* $m + 1$ *inequalities. The block elimination method is applied to achieve this. As it is shown in Lemma 5, the block elimination method will require to compute the extreme rays of the projection cone (denoted by* C*), which contains* $m + 1$ *inequalities and* $m + 1$ *variables. However, considering the structural properties of the coefficient matrix of the representation of* C*, we found that computing the extreme rays of* C *is equivalent to computing the extreme rays*

of another simpler cone, which still has $m+1$ inequalities but only $n+1$ variables. For more details, please refer to Step 3 of Section 8.5.

Lemma 6 *Representation of the redundancy test cone $\mathcal{P}_{\mathbf{u}}(Q)$ can be obtained from $\mathcal{P}(Q)$ by setting coefficients of the corresponding p eliminated variables to 0 in the representation of $\mathcal{P}(Q)$.*

Proof ▷ Please refer to Section 8.2 ◁

For the polyhedron Q, given a variable order $y_1 > \cdots > y_n$, for $1 \leq i \leq n$, we denote by $Q^{(y_i)}$ the inequalities in the representation of Q whose largest variable is y_i.

Definition 1 (Projected representation) Projected representation *of Q w.r.t. the variable order $y_1 > \cdots > y_n$, denoted $\mathsf{ProjRep}(Q; y_1 > \cdots > y_n)$, is a linear system given by $Q^{(y_1)}$ if $n = 1$, and is the conjunction of $Q^{(y_1)}$ and $\mathsf{ProjRep}(\mathsf{proj}(Q; \mathbf{y}_2); y_2 > \cdots > y_n)$ otherwise. We say that $P := \mathsf{ProjRep}(Q; y_1 > \cdots > y_n)$ is the* minimal projected representation *if, for all $1 \leq k \leq n$, every inequality of P with y_k as largest variable is not redundant among all the inequalities of P with variables among $y_k, \ldots, y_n$.*

We can generate the *minimal projected representation* of a polyhedron by Algorithm 3.

Algorithm 2 Extreme ray test

Input: $(\mathcal{P}, \ell)$, where (i) $\mathcal{P} := \{(\mathbf{v}, v_0) \in \mathbb{Q}^n \times \mathbb{Q} \mid M[\mathbf{v}^t, v_0]^t \leq \mathbf{0}\}$ with $M \in \mathbb{Q}^{m \times (n+1)}$, (ii) $\ell : \mathbf{a}^t\mathbf{y} \leq c$ with $\mathbf{a} \in \mathbb{Q}^n$ and $c \in \mathbb{Q}$;
Output: true if $[\mathbf{a}^t, c]^t$ is an extreme ray of $\mathcal{P}$, false otherwise;
1: Let $\mathbf{s} := M[\mathbf{a}^t, c]^t$;
2: Let $\zeta(\mathbf{s})$ be the index set of the zero coefficients of $\mathbf{s}$;
3: **if** $\mathrm{rank}(M_{\zeta(\mathbf{s})}) = n$ **then**
4: return (true);
5: **else**
6: return (false);
7: **end if**

5 Complexity estimates

We analyze the computational complexity of Algorithm 3, which computes the minimal projected representation of a given polyhedron. This computation is equivalent to eliminating all variables, one after another, in Fourier-Motzkin elimination. We prove that using our algorithm, finding the minimal projected representation of a polyhedron is singly exponential in the dimension n of the ambient space. The most consuming procedure in Algorithm 3 is finding the initial redundancy test cone. This operation requires another polyhedron projection in higher dimension. As it is shown in Remark 1, we can use block elimination method to perform this task efficiently. This requires the computation of the extreme rays of the projection cone. The double description method is an efficient way to solve this problem. We

Algorithm 3 Minimal Projected Representation of Q

Input: $S = \{\mathbf{A}\mathbf{y} \leq \mathbf{c}\}$: a representation of the input polyhedron Q;
Output: A minimal projected representation of Q;

Generate the initial redundancy test cone $\mathcal{P}$ by Algorithm 1;
$S_0 := \{\ \}$;
for i from 1 to m **do**
  Let f be the result of applying Algorithm 2 with the inputs $\mathcal{P}$ and $\mathbf{A}_i\mathbf{y} \leq \mathbf{c}_i$;
  if f = true **then**
    $S_0 := S_0 \cup \{\mathbf{A}_i\mathbf{y} \leq \mathbf{c}_i\}$;
  end if
end for
$\mathcal{P} := \mathcal{P}|_{v_1=0}$;
for i from 0 to $n-1$ **do**
  $S_{i+1} := \{\ \}$;
  for $\ell_{\text{pos}} \in S_i$ with positive coefficient of y_{i+1} **do**
    for $\ell_{\text{neg}} \in S_i$ with negative coefficient of y_{i+1} **do**
      $\ell_{\text{new}} := \mathsf{Combine}(\ell_{\text{pos}}, \ell_{\text{neg}}, y_{i+1})$;
      Let f be the result of applying Algorithm 2 with the inputs $\mathcal{P}$ and ℓ_{new};
      if f = true **then**
        $S_{i+1} := S_{i+1} \cup \{\ell_{\text{new}}\}$;
      end if
    end for
  end for
  for $\ell \in S_i$ with zero coefficient of y_{i+1} **do**
    Let f be the result of applying Algorithm 2 with the inputs $\mathcal{P}$ and ℓ ;
    if f = true **then**
      $S_{i+1} := S_{i+1} \cup \{\ell\}$;
    end if
  end for
  $\mathcal{P} := \mathcal{P}|_{v_{i+1}=0}$;
end for
return $(S_0 \cup S_1 \cup \cdots \cup S_n)$;

begin this section by computing the bit complexity of the double description algorithm. For detailed explanation of the double description method, please refer to Section 8.7.

Lemma 7 (Coefficient bound of extreme rays) *Let $S = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{0}\}$ be the minimal representation of a cone $C \subseteq \mathbb{Q}^n$, where $A \in \mathbb{Q}^{m\times n}$. Then, the absolute value of a coefficient in any extreme ray of C is bounded over by $(n-1)^n \|A\|^{2(n-1)}$.*

Proof ▷ Please refer to Section 8.3 ◁

Lemma 8 *Let $S = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{0}\}$ be the minimal representation of a cone $C \subseteq \mathbb{Q}^n$, where $A \in \mathbb{Q}^{m\times n}$. The double description method requires $O(m^{n+2}n^{\theta+\epsilon}h^{1+\epsilon})$ bit operations, where h is the height of the matrix A.*

Proof ▷ Please refer to Section 8.4 ◁

Lemma 9 (Complexity of constructing the initial redundancy test cone) *Let h be the maximum height of A and $\mathbf{c}$ in the input system, then generating the initial redundancy test cone (Algorithm 1) requires at most $O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon})$ bit operations. Moreover, $\mathsf{proj}(W;\{\mathbf{v},v_0\})$ can be represented by $O(m^{\lfloor\frac{n+1}{2}\rfloor})$ inequalities, each with a height bound of $O(m^{\epsilon}n^{2+\epsilon}h)$.*

Proof ▷ Please refer to Section 8.5 ◁

Lemma 10 *Algorithm 2 runs within $O(m^{\frac{n}{2}}n^{\theta+\epsilon}h^{1+\epsilon})$ bit operations.*

Proof ▷ Please refer to Section 8.6 ◁

Using Algorithms 1 and 2, we can find the minimal projected representation of a polyhedron in singly exponential time w.r.t. the number of variables n.

Theorem 5 *Algorithm 3 is correct. Moreover, a minimal projected representation of Q can be produced within $O(m^{\frac{5n}{2}}n^{\theta+1+\epsilon}h^{1+\epsilon})$ bit operations.*

Proof ▷ Correctness of the algorithm follows from Theorem 4, Lemma 6.

By [20, 25], we know that after eliminating p variables, the projection of the polyhedron has at most m^{p+1} facets. For eliminating the next variable, there will be at most $(\frac{m^{p+1}}{2})^2$ pairs of inequalities to be considered and each of the pairs generate a new inequality which should be checked for redundancy. Therefore, overall the complexity of the algorithm is:

$$O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon}) + \sum_{p=0}^{n} m^{2p+2}O(m^{\frac{n}{2}}n^{\theta+\epsilon}h^{1+\epsilon}) = O(m^{\frac{5n}{2}}n^{\theta+1+\epsilon}h^{1+\epsilon}).$$

◁

6 Experimentation

In this section we report on our software implementation of the algorithms presented in the previous sections. Our implementation as well as our test cases are part of the BPAS library, available at http://www.bpaslib.org/.

We present the serial and the parallel running time for the Minimal Projected Representation (MPR) algorithm. Comparing with the `Project` method of the `PolyhedralSets` package of Maple 2017 and the famous CDD library (version 2018), we have been able to solve problems more efficiently. We believe that this is the result of using a more effective algorithm and an efficient implementation in `C`.

As test cases we use 16 consistent linear inequality systems. The first 9 test cases, (t1 to t9) are linear inequality systems with random coefficients. The systems S24 and S35 are 24-simplex and 35-simplex polytopes, C56 and C510 are cyclic polytopes in dimension five with six and ten vertices, C68 is a cyclic polytope in dimension six with eight vertices, C1011 is cyclic polytope in dimension ten with eleven vertices, and, Cro6 is the cross polytope in 6 dimension, [18]. The `test` column of Table 1 shows these systems along with the number of variables and the number of inequalities for each of them. We implemented the MPR algorithm with two different approaches: iterative approach and divide and conquer approach. In

Test (var,ineq)	MPR-itr	MPR-rec	CDD	Maple	Maple-MPR
S24 (24,25)	46	41	411	6485	3040
S35 (35,36)	205	177	2169	57992	9840
Cro6 (6,64)	28	29	329	246750	8610
C56 (5,6)	1	1	13	825	140
C68 (6,16)	4	4	866	20154	650
C1011 (10,11)	95	92	>1h	>1h	>1h
C510 (5,42)	23	22	7674	6173	6070
T1 (5,10)	7	7	142	7974	1400
T2 (10,12)	109	112	122245	3321217	13330
T3 (7,10)	26	26	8207	117021	2900
T4 (10,12)	368	370	1177807	>1h	26650
T5 (5,11)	7	7	75	8229	1650
T6 (10,20)	26591	26156	>1h	>1h	>1h
T7 (9,19)	162628	158569	>1h	>1h	>1h
T8 (8,19)	21411	20915	>1h	>1h	>1h
T9 (6,18)	1281	1263	77372	>1h	267920

Table 1: Running time (in milliseconds) table for a set of examples, varying in the number of variables and inequalities, collected on a system with Intel-i7-7700T 4-core processor, clocking at 3.8 GHz.

both implementations, we use dense representation for storing linear inequalities. In the first method, we use *unrolled linked lists* to encode linear inequality systems. Indeed, using this data structure, we are able to store an array of inequalities in each node of linked list and we can improve data locality. However, we use simple linked list in the divide and conquer version to save time on dividing and joining lists. Although both these methods have shown quite similar and promising results in terms of running time, we anticipate to get better results if we combine unrolled linked lists with the divide and conquer method while using a varying threshold for recursion as the algorithm goes on. Columns `MPR-itr` and `MPR-rec` of the Table 1 illustrates the running time (in milliseconds) of these implementations on a configuration with Intel-i7-7700T (4 cores, 8 threads, clocking at 3.8 GHz). Also, columns `CDD`, `Maple`, and `Maple-MPR` are corresponding to running times of the Fourier algorithm in the CDD library, which uses LP for redundancy elimination, the function `PolyhedralSets:-Project` of Maple, and, an implementation of our algorithm in the Maple programming language, on the same system, respectively.

Using divide and conquer method, we have been able to parallelize our program, with Cilk [4]. We call this algorithm Parallel Minimal Projected Representation (PMPR). Table 2 presents the running time (in milliseconds) and speedup of the multi-core version of the algorithm. The columns `PMPR-1`, `PMPR-4`, `PMPR-8`, and, `PMPR-12` demonstrate the running time of the multi-core program on a system with Intel-Xeon-X5650 (12 cores, 24 threads, clocking at 2.6GHz), using 1, 4, 8, and 12 Cilk workers, respectively. The numbers in brackets show the speedup we gain using multi-threading.

Test	PMPR-1	PMPR-4		PMPR-8		PMPR-12	
S24	67	71	(0.9 x)	73	(0.9 x)	83	(0.8 x)
S35	291	308	(0.9 x)	310	(0.9 x)	375	(0.7 x)
Cro6	54	45	(1.2 x)	36	(1.5 x)	34	(1.5 x)
C56	2	3	(0.6 x)	3	(0.6 x)	12	(0.1 x)
C68	8	7	(1.1 x)	7	(1.1 x)	19	(0.4 x)
C1011	176	62	(2.8 x)	47	(3.7 x)	53	(3.3 x)
C510	38	33	(1.1 x)	34	(1.1 x)	40	(0.9 x)
T1	13	8	(1.6 x)	9	(1.4 x)	17	(0.7 x)
T2	205	67	(3.0 x)	55	(3.7 x)	57	(3.5 x)
T3	48	20	(2.4 x)	18	(2.6 x)	20	(2.4 x)
T4	685	207	(3.3 x)	141	(4.8 x)	126	(5.4 x)
T5	14	9	(1.5 x)	10	(1.3 x)	11	(1.2 x)
T6	44262	12995	(3.4 x)	6785	(6.5 x)	5163	(8.5 x)
T7	282721	78176	(3.6 x)	48048	(5.8 x)	35901	(7.8 x)
T8	41067	10669	(3.8 x)	5689	(7.2 x)	4471	(9.1 x)
T9	2407	742	(3.2 x)	491	(4.8 x)	448	(5.3 x)

Table 2: Running time (in milliseconds) table for our set of examples, with different number of Cilk workers, collected on a system Intel-Xeon-X5650 and 12 CPU cores, clocking at 2.6GHz.

7 Conclusion and Related works

As we previously discussed, removing redundant inequalities during the execution of Fourier-Motzkin Elimination is the central issue towards efficiency. To our knowledge, all available implementations of Fourier-Motzkin Elimination rely on linear programming for removing all the redundant inequalities, an idea suggested in [24]. However, there are alternative algorithmic approaches relying on linear algebra. In [10], Chernikov proposed a redundancy test with little added work, which greatly improves the practical efficiency of Fourier-Motzkin Elimination. Kohler proposed a method in [25] which only uses matrix arithmetic operations to test the redundancy of inequalities. As observed by Imbert in his work [20], the method he proposed in this paper as well as those of Chernikov and Kohler are essentially equivalent. Even though these works are effective in practice, none of them can remove all redundant inequalities generated by Fourier-Motzkin Elimination.

Besides Fourier-Motzkin Elimination, block elimination is another algorithmic tool to project polyhedra on a lower dimensional subspace. This method relies on the extreme rays of the so-called projection cone. Although there exist efficient methods to enumerate the extreme rays of this projection cone, like the *double description method* [16] (also known as Chernikova's algorithm [11, 26]), this method can not remove all the redundant inequalities.

In [1], Balas shows that if certain *inconvertibility conditions* are satisfied, then the extreme rays of the redundancy test cone exactly defines a minimal representation of the projection of a polyhedron. As Balas mentioned in his paper, this method can be extended to any polyhedron.

A drawback of Balas' work is that the necessity of enumerating the extreme rays of the redundancy test cone in order to produce a minimal representation of the projection $\mathsf{proj}(Q; \mathbf{x})$, which is time consuming. Our algorithm tests the

redundancy of the inequality $\mathbf{ax} \leq c$ by checking whether $(\mathbf{a}, c)$ is an extreme ray of the redundancy test cone or not.

Another related topic to our work is the subsumption cone [19]. Consider the polyhedron Q given in Equation (2), define $T := \{(\lambda, \alpha, \beta) \mid \lambda^t \mathbf{A} = \alpha^t, \lambda^t \mathbf{c} \leq \beta, \lambda \geq \mathbf{0}\}$, where λ and α are vectors of dimension m and n respectively, β is a variable. The *subsumption cone* of Q is obtained by eliminating λ in T, that is, $\mathsf{proj}(T; \{\alpha, \beta\})$. We proved that considering a full-dimensional, pointed polyhedron, where the first n rows of the coefficient matrix are linearly independent, the initial redundancy test cone and the subsumption cone are equivalent. For more details, please refer to Section 8.7.

Given a V-representation of a polyhedron P, one can obtain the V-representation of any projection of P[2]. The double description method turns the V-representation of the projection to its H-representation. Most existing software libraries dealing with polyhedral sets store a polyhedron with these two representations, like the *Parma Polyhedra Library (PPL)* and Polylib. In this case, it is convenient to compute the projection using the block elimination method. When we are only given the H-representation, the first thing is to compute the V-representation, which is equivalent to the procedure of computing the initial test cone in our method. When we need to perform successive projections, it is well-known that Fourier-Motzkin Elimination performs better than repeated applications of the double description method (as PPL does).

Recently, the verified polyhedron library (VPL) [6] takes advantage of parametric linear programming to project a polyhedron. Like PPL, VPL may not beat Fourier-Motzkin Elimination when we need to perform successive projections. In VPL, the authors rely on *raytracing* to remove redundant inequalities. This is an efficient way for removing redundancies, but this cannot remove them all, thus Linear Programming (LP) is still needed. As pointed out in [27], raytracing is effective when there are not many redundancies; unfortunately, Fourier-Motzkin Elimination typically generates lots of redundancies.

Another modern library dealing with polyhedral sets computation is the Normaliz library [8]. In this library, Fourier-Motzkin Elimination is used for conversion between different descriptions of polyhedral sets. This is a different strategy than the one of our paper. As discussed in the introduction, we are motivated here by performing successive projections as required in the analysis, scheduling and transformation of for loop nests of computer programs.

8 Proofs

8.1 Proof of Theorem 3

To prove the Theorem, we need a preliminary observation.

Lemma 11 *The operations "computing the characteristic cone" and "computing projections" commute. To be precise, we have:*

$$\mathsf{CharCone}(\mathsf{proj}(Q; \mathbf{x})) = \mathsf{proj}(\mathsf{CharCone}(Q); \mathbf{x}).$$

[2]for example, P is generated by $\{(1,2,3,4)^t, (2,3,4,5)^t, (2,3,7,9)^t\}$, the projection of P onto the last two coordinates is generated by $\{(3,4)^t, (4,5)^t, (7,9)^t\}$

Proof $\triangleright$ By the definition of the characteristic cone, we have $\mathsf{CharCone}(Q) = \{(\mathbf{u},\mathbf{x}) \mid A\mathbf{u} + B\mathbf{x} \leq \mathbf{0}\}$, whose representation has the same left-hand side as the one of Q. The lemma is valid if we can show that the representation of $\mathsf{proj}(\mathsf{CharCone}(Q);\mathbf{x})$ has the same left-hand side as $\mathsf{proj}(Q;\mathbf{x})$. This is obvious with the Fourier-Motzkin Elimination procedure. $\triangleleft$

Proof $\triangleright$ By definition, the polar cone $(\mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))^*$ is equal to

$$\{(\mathbf{y},y_0) \mid [\mathbf{y}^t,y_0][\mathbf{x}^t,x_{\text{last}}]^t \leq 0, \forall\ (\mathbf{x},x_{\text{last}}) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))\}.$$

This claim follows immediately from: $(\mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))^* = \overline{\mathsf{proj}(W^0;\{\mathbf{v},v_0\})}$. We shall prove this latter equality in two steps.

($\supseteq$) For any $(\overline{\mathbf{v}},-\overline{v}_0) \in \overline{\mathsf{proj}(W^0;\{\mathbf{v},v_0\})}$, we need to show that $[\overline{\mathbf{v}}^t,-\overline{v}_0][\mathbf{x}^t,x_{\text{last}}]^t \leq 0$ holds when $(\mathbf{x},x_{\text{last}}) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))$. Remember that we assume that Q is pointed. Observe that $\mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))$ is also pointed. Therefore, we only need to verify the desired property for the extreme rays of $\mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))$, which either have the form $(\mathbf{s},1)$ or are equal to $(\mathbf{s},0)$ (Theorem 1). Before continuing, we should notice that since $(\overline{\mathbf{v}},\overline{v}_0) \in \mathsf{proj}(W^0;\{\mathbf{v},v_0\})$, there exists $\overline{\mathbf{w}}$ such that $\{[\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}A_0 = 0, -[\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0 + \overline{v}_0 \geq 0, [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1} \geq 0\}$. Cases 1 and 2 below conclude that $(\overline{\mathbf{v}},-\overline{v}_0) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))^*$ holds.

Case 1: For the form $(\mathbf{s},1)$, we have $\mathbf{s} \in \mathsf{proj}(Q;\mathbf{x})$. Indeed, $\mathbf{s}$ is an extreme point of $\mathsf{proj}(Q;\mathbf{x})$. Hence, there exists $\overline{\mathbf{u}} \in \mathbb{Q}^p$, such that we have $A\overline{\mathbf{u}} + B\mathbf{s} \leq \mathbf{c}$. By construction of Q^0, we have $A_0\overline{\mathbf{u}} + B_0\mathbf{s}' \leq \mathbf{c}_0$, where $\mathbf{s}' = [\mathbf{s}^t, s_{q+1}, \ldots, s_m]^t$ with $s_{q+1} = \cdots = s_m = 0$. Therefore, we have: $[\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}A_0\overline{\mathbf{u}} + [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}B_0\mathbf{s}' \leq [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0$. This leads us to $\overline{\mathbf{v}}^t\mathbf{s} = [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]\mathbf{s}' \leq [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0 \leq \overline{v}_0$. Therefore, we have $[\overline{\mathbf{v}}^t,-\overline{v}_0][\mathbf{s}^t,x_{\text{last}}]^t \leq 0$, as desired.

Case 2: For the form $(\mathbf{s},0)$, we have $\mathbf{s} \in \mathsf{CharCone}(\mathsf{proj}(Q;\mathbf{x})) = \mathsf{proj}(\mathsf{CharCone}(Q);\mathbf{x})$. Thus, there exists $\overline{\mathbf{u}} \in \mathbb{Q}^p$ such that $A\overline{\mathbf{u}} + B\mathbf{s} \leq \mathbf{0}$. Similarly to Case 1, we have $[\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}A_0\overline{\mathbf{u}} + [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}B_0\mathbf{s}' \leq [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{0}$. Therefore, we have $\overline{\mathbf{v}}^t\mathbf{s} = [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]\mathbf{s}' \leq [\overline{\mathbf{v}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{0} = 0$, and thus, we have $[\overline{\mathbf{v}}^t,-\overline{v}_0][\mathbf{s}^t,x_{\text{last}}]^t \leq 0$, as desired.

($\subseteq$) For any $(\overline{\mathbf{y}},\overline{y}_0) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))^*$, we have $[\overline{\mathbf{y}}^t,\overline{y}_0][\mathbf{x}^t,x_{\text{last}}]^t \leq 0$ whenever we have $(\mathbf{x},x_{\text{last}}) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))$. For any $\overline{\mathbf{x}} \in \mathsf{proj}(Q;\mathbf{x})$, we have $\overline{\mathbf{y}}^t\overline{\mathbf{x}} \leq -\overline{y}_0$ since $(\overline{\mathbf{x}},1) \in \mathsf{HomCone}(\mathsf{proj}(Q;\mathbf{x}))$. Therefore, we have $\overline{\mathbf{y}}^t\mathbf{x} \leq -\overline{y}_0$, for all $\mathbf{x} \in \mathsf{proj}(Q;\mathbf{x})$, which makes the inequality $\overline{\mathbf{y}}^t\mathbf{x} \leq -\overline{y}_0$ redundant in the system $\{A\mathbf{u} + B\mathbf{x} \leq \mathbf{c}\}$. By Farkas' Lemma (see Lemma 2), there exists $\mathbf{p} \geq \mathbf{0}, \mathbf{p} \in \mathbb{Q}^m$ and $\lambda \geq 0$ such that $\mathbf{p}^tA = \mathbf{0}$, $\overline{\mathbf{y}} = \mathbf{p}^tB$, $\overline{y}_0 = \mathbf{p}^t\mathbf{c} + \lambda$. Remember that $A_0 = A$, $B_0 = [B,B']$, $\mathbf{c}_0 = \mathbf{c}$. Here B' is the last $m-q$ columns of B_0 consisting of $\mathbf{e}_{q+1},\ldots,\mathbf{e}_m$. Let $\overline{\mathbf{w}} = \mathbf{p}^tB'$. We then have

$$\{\mathbf{p}^tA_0 = \mathbf{0},\ [\overline{\mathbf{y}}^t,\overline{\mathbf{w}}^t] = \mathbf{p}^tB_0, -\overline{y}_0 \geq \mathbf{p}^t\mathbf{c}_0, \mathbf{p} \geq \mathbf{0}\},$$

which is equivalent to

$$\begin{aligned}&\{\mathbf{p}^t = [\overline{\mathbf{y}}^t,\overline{\mathbf{w}}^t]B_0^{-1}, [\overline{\mathbf{y}}^t,\overline{\mathbf{w}}^t]B_0^{-1}A_0 = \mathbf{0},\\ &-\overline{y}_0 \geq [\overline{\mathbf{y}}^t,\overline{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0, [\overline{\mathbf{y}}^t,\overline{\mathbf{w}}^t]B_0^{-1} \geq \mathbf{0}\}\end{aligned}$$

Therefore, $(\overline{\mathbf{y}},\overline{\mathbf{w}},-\overline{y}_0) \in W^0$, and $(\overline{\mathbf{y}},-\overline{y}_0) \in \mathsf{proj}(W^0;\{\mathbf{v},v_0\})$. From this, we deduce that $(\overline{\mathbf{y}},\overline{y}_0) \in \overline{\mathsf{proj}(W^0;\{\mathbf{v},v_0\})}$ holds. $\triangleleft$

8.2 Proof of Lemma 6

To distinguish from the construction of $\mathcal{P}(Q)$, we rename the variables $\mathbf{v}, \mathbf{w}, v_0$ as $\mathbf{v_u}, \mathbf{w_u}, v_\mathbf{u}$, when constructing W^0 and computing the test cone $\mathcal{P}_\mathbf{u}(Q)$. That is, we have $\mathcal{P}_\mathbf{u}(Q) = \mathsf{proj}(W^0; \{\mathbf{v_u}, v_\mathbf{u}\})$, where W^0 is the set of all $(\mathbf{v_u}, \mathbf{w_u}, v_\mathbf{u}) \in \mathbb{Q}^q \times \mathbb{Q}^{m-q} \times \mathbb{Q}$ satisfying

$$\{(\mathbf{v_u}, \mathbf{w_u}, v_\mathbf{u}) \mid [\mathbf{v_u^t}, \mathbf{w_u^t}]B_0^{-1}A = \mathbf{0}, -[\mathbf{v_u^t}, \mathbf{w_u^t}]B_0^{-1}\mathbf{c} + v_\mathbf{u} \geq 0, \\ [\mathbf{v_u^t}, \mathbf{w_u^t}]B_0^{-1} \geq \mathbf{0}\},$$

while we have $\mathcal{P}(Q) = \mathsf{proj}(W; \{\mathbf{v}, v_0\})$ where W is the set of all $(\mathbf{v}, \mathbf{w}, v_0) \in \mathbb{Q}^n \times \mathbb{Q}^{m-n} \times \mathbb{Q}$ satisfying $\{(\mathbf{v}, \mathbf{w}, v_0) \mid -[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}\mathbf{c} + v_0 \geq 0, [\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1} \geq \mathbf{0}\}$. Proof $\triangleright$ By Step 1 of Algorithm 1, $[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}\mathbf{A} = \mathbf{v}^t$ holds whenever $(\mathbf{v}, \mathbf{w}, v_0) \in W$. Rewrite $\mathbf{v}$ as $\mathbf{v}^t = [\mathbf{v}_1^t, \mathbf{v}_2^t]$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the first p and last $n-p$ variables of $\mathbf{v}$. We have $[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}A = \mathbf{v}_1^t$ and $[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}B = \mathbf{v}_2^t$. Similarly, we have $[\mathbf{v_u^t}, \mathbf{w_u^t}]B_0^{-1}A = \mathbf{0}$ and $[\mathbf{v_u^t}, \mathbf{w_u^t}]B_0^{-1}B = \mathbf{v_u^t}$ whenever $(\mathbf{v_u}, \mathbf{w_u}, v_\mathbf{u}) \in W^0$. This lemma holds if we can show $\mathcal{P}_\mathbf{u} = \mathcal{P}|_{\mathbf{v}_1 = \mathbf{0}}$. We prove this in two steps.

($\subseteq$) For any $(\overline{\mathbf{v}}_\mathbf{u}, \overline{v}_\mathbf{u}) \in \mathcal{P}_\mathbf{u}(Q)$, there exists $\overline{\mathbf{w}}_\mathbf{u} \in \mathbb{Q}^{m-q}$ satisfying $(\overline{\mathbf{v}}_\mathbf{u}, \overline{\mathbf{w}}_\mathbf{u}, \overline{v}_\mathbf{u}) \in W^0$. Let $[\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t] := [\overline{\mathbf{v}}_\mathbf{u}^t, \overline{\mathbf{w}}_\mathbf{u}^t]B_0^{-1}\mathbf{A}_0$, where $\overline{\mathbf{v}}^t = [\overline{\mathbf{v}}_1^t, \overline{\mathbf{v}}_2^t]$ with $\overline{\mathbf{v}}_1 \in \mathbb{Q}^p, \overline{\mathbf{v}}_2 \in \mathbb{Q}^{n-p}$ and $\overline{\mathbf{w}} \in \mathbb{Q}^{m-n}$. Then, $\overline{\mathbf{v}}_1^t = [\overline{\mathbf{v}}_\mathbf{u}^t, \overline{\mathbf{w}}_\mathbf{u}^t]B_0^{-1}A = \mathbf{0}$ and $\overline{\mathbf{v}}_2^t = [\overline{\mathbf{v}}_\mathbf{u}^t, \overline{\mathbf{w}}_\mathbf{u}^t]B_0^{-1}B = \overline{\mathbf{v}}_\mathbf{u}$ due to $(\overline{\mathbf{v}}_\mathbf{u}, \overline{\mathbf{w}}_\mathbf{u}, \overline{v}_\mathbf{u}) \in W^0$. Let $\overline{v}_0 = \overline{v}_\mathbf{u}$, it is easy to verify that $(\overline{\mathbf{v}}, \overline{\mathbf{w}}, \overline{v}_0) \in W$. Therefore, $(\mathbf{0}, \overline{\mathbf{v}}_\mathbf{u}, \overline{v}_\mathbf{u}) = (\overline{\mathbf{v}}, \overline{v}_0) \in \mathcal{P}(Q)$.

($\supseteq$) For any $(\mathbf{0}, \overline{\mathbf{v}}_2, \overline{v}_0) \in \mathcal{P}(Q)$, there exists $\overline{\mathbf{w}} \in \mathbb{Q}^{m-n}$ satisfying $(\mathbf{0}, \overline{\mathbf{v}}_2, \overline{\mathbf{w}}, \overline{v}_0) \in W$. Let $(\overline{\mathbf{v}}_\mathbf{u}, \overline{\mathbf{w}}_\mathbf{u}) := (\mathbf{0}, \overline{\mathbf{v}}_2, \overline{\mathbf{w}})\mathbf{A}_0^{-1}B_0$. We have $\overline{\mathbf{v}}_\mathbf{u} = (\mathbf{0}, \overline{\mathbf{v}}_2, \overline{\mathbf{w}})\mathbf{A}_0^{-1}B = \overline{\mathbf{v}}_2$. Let $\overline{v}_\mathbf{u} = \overline{v}_0$, it is easy to verify $(\overline{\mathbf{v}}_\mathbf{u}, \overline{\mathbf{w}}_\mathbf{u}, \overline{v}_\mathbf{u}) \in W^0$. Therefore, $(\overline{\mathbf{v}}_2, \overline{v}_0) = (\overline{\mathbf{v}}_\mathbf{u}, \overline{v}_\mathbf{u}) \in \mathcal{P}_\mathbf{u}(Q)$. $\triangleleft$

8.3 Proof of Lemma 7

Proof $\triangleright$ From the properties of extreme rays, see Section 2.1, we know that when $\mathbf{r}$ is an extreme ray, there exists a sub-matrix $A' \in \mathbb{Q}^{(n-1)\times n}$ of A, such that $A'\mathbf{r} = 0$. This means that $\mathbf{r}$ is in the null-space of A'. Thus, the claim follows by proposition 6.6 of [32]. $\triangleleft$

8.4 Proof of Lemma 8

Proof $\triangleright$ To analyze the complexity of the DD method after adding t inequalities, with $n \leq t \leq m$, the first step is to partition the extreme rays at the $t-1$-iteration, with respect to the newly added inequality. Note that we have at most $(t-1)^{\lfloor \frac{n}{2} \rfloor}$ extreme rays (Equation (1)) whose coefficients can be bounded over by $(n-1)^n \|A\|^{2(n-1)}$ (Lemma 7) at the $t-1$-iteration. Hence, this step needs at most $C_1 := (t-1)^{\lfloor \frac{n}{2} \rfloor} \times n \times \mathcal{M}(\log((n-1)^n \|A\|^{2(n-1)})) \leq O(t^{\lfloor \frac{n}{2} \rfloor} n^{2+\epsilon} h^{1+\epsilon})$ bit operations. After partitioning the vectors, the next step is to check adjacency for each pair of vectors. The cost of this step is equivalent to computing the rank of a sub-matrix $A' \in \mathbb{Q}^{(t-1)\times n}$ of A. This should be done for $\frac{t^n}{4}$ pairs of vectors. This step needs at most $C_2 := \frac{t^n}{4} \times O((t-1)n^{\theta+\epsilon}h^{1+\epsilon}) \leq O(t^{n+1}n^{\theta+\epsilon}h^{1+\epsilon})$ bit operations. We know there are at most $t^{\lfloor \frac{n}{2} \rfloor}$ pairs of adjacent extreme rays. The

next step is to combine every pair of adjacent vectors in order to obtain a new extreme ray. This step consists of n multiplications in $\mathbb{Q}$ of coefficients with absolute value bounded over by $(n-1)^n\|A\|^{2(n-1)}$ (Lemma 7) and this should be done for at most $t^{\lfloor\frac{n}{2}\rfloor}$ vectors. Therefore, the bit complexity of this step, is no more than $C_3 := t^{\lfloor\frac{n}{2}\rfloor} \times n \times \mathcal{M}(\log((n-1)^n\|A\|^{2(n-1)})) \leq O(t^{\lfloor\frac{n}{2}\rfloor}n^{2+\epsilon}h^{1+\epsilon})$. Finally, the complexity of step t of the algorithm is $C := C_1 + C_2 + C_3$. The claim follows after simplifying $m \cdot C$. $\lhd$

8.5 Proof of Lemma 9

Proof $\rhd$ We analyze Algorithm 1 step by step.

Step 1: construction of A_0 from A. The cost of this step can be neglected. However, it should be noticed that the matrix A_0 has a special structure. Without loss of generality, we cam assume that the first n rows of A are linearly independent. The matrix A_0 has the following structure $A_0 = \begin{pmatrix} A_1 & \mathbf{0} \\ A_2 & I_{m-n} \end{pmatrix}$, where A_1 is a full rank matrix in $\mathbb{Q}^{n\times n}$ and $A_2 \in \mathbb{Q}^{(m-n)\times n}$.

Step 2: construction of the cone W. Using the structure of the matrix A_0, its inverse can be expressed as $A_0^{-1} = \begin{pmatrix} A_1^{-1} & \mathbf{0} \\ -A_2A_1^{-1} & I_{m-n} \end{pmatrix}$. Also, from Section 2.4 we have $\|A_1^{-1}\| \leq (\sqrt{n-1}\|A_1\|)^{n-1}$. Therefore, $\|A_0^{-1}\| \leq n^{\frac{n+1}{2}}\|A\|^q$, and $\|A_0^{-1}\mathbf{c}\| \leq n^{\frac{n+3}{2}}\|A\|^n\|\mathbf{c}\| + (m-n)\|\mathbf{c}\|$. That is, $\mathsf{height}(A_0^{-1}) \in O(n^{1+\epsilon}h)$ and $\mathsf{height}(A_0^{-1}\mathbf{c}) \in O(m^\epsilon + n^{1+\epsilon}h)$. As a result, height of coefficients of W can be bounded over by $O(m^\epsilon + n^{1+\epsilon}h)$.

To estimate the bit complexity, we need the following consecutive steps:

- Computing A_0^{-1}, which requires

$$\begin{aligned} &O(n^{\theta+1+\epsilon}h^{1+\epsilon}) + O((m-n)n^2\mathcal{M}(\max(\mathsf{height}(A_2), \mathsf{height}(A_1^{-1})))) \\ \leq &O(mn^{\theta+1+\epsilon}h^{1+\epsilon}) \text{ bit operations;} \end{aligned}$$

- Constructing $W := \{(\mathbf{v}, \mathbf{w}, v_0) \mid -[\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1}\mathbf{c} + v_0 \geq 0, [\mathbf{v}^t, \mathbf{w}^t]\mathbf{A}_0^{-1} \geq \mathbf{0}\}$ requires at most

$$\begin{aligned} C_1 :=&O(m^{1+\epsilon}n^{\theta+1+\epsilon}h^{1+\epsilon}) + O(mn\mathcal{M}(\mathsf{height}(A_0^{-1}, \mathbf{c}))) \\ &+O((m-n)h) \leq O(m^{1+\epsilon}n^{\theta+\epsilon+1}h^{1+\epsilon}) \text{ bit operations.} \end{aligned}$$

Step 3: projecting W and finding the initial redundancy test cone. Following Lemma 5, we obtain a representation of $\mathsf{proj}(W; \{\mathbf{v}, v_0\})$ through finding extreme rays of the corresponding projection cone.
Let $E = (-A_2A_1^{-1})^t \in \mathbb{Q}^{n\times(m-n)}$ and $\mathbf{g}^t$ be the last $m-n$ elements of $(A_0^{-1}\mathbf{c})^t$. Then, the projection cone can be represented by:

$$C = \{\mathbf{y} \in \mathbb{Q}^{m+1} \mid \mathbf{y}^t \begin{pmatrix} E \\ \mathbf{g}^t \\ I_{m-n} \end{pmatrix} = \mathbf{0}, \mathbf{y} \geq \mathbf{0}\}.$$

Note that $y_{n+2}, \ldots, y_{m+1}$ can be solved from the system of equations in the representation of C. We substitute them in the inequalities and obtain a representation of the cone C', given by:

$$C' = \{\mathbf{y}' \in \mathbb{Q}^{n+1} \mid \mathbf{y}'^t \begin{pmatrix} E \\ \mathbf{g}^t \end{pmatrix} \leq \mathbf{0}, \mathbf{y}' \geq \mathbf{0}\}$$

In order to find the extreme rays of the cone C, we can find the extreme rays of the cone C' and then back-substitute them into the equations to find the extreme rays of C. Applying Algorithm 5 to C', we can obtain all extreme rays of C', and subsequently, the extreme rays of C. The cost estimate of this step is bounded over by the complexity of Algorithm 5 with C' as input. This operation requires at most $C_2 := O(m^{n+3}(n+1)^{\theta+\epsilon} \max(\mathsf{height}(E, \mathbf{g}^t))^{1+\epsilon}) \leq O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon})$ bit operations. The overall complexity of the algorithm can be bounded over by: $C_1 + C_2 \leq O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon})$. Also, by Lemma 7 and Lemma 8, we know that the cone C has at most $O(m^{\lfloor \frac{n+1}{2} \rfloor})$ distinct extreme rays, each with height no more than $O(m^{\epsilon}n^{2+\epsilon}h)$. That is, $\mathsf{proj}(W^0; \{\mathbf{v}, v_0\})$ can be represented by at most $O(m^{\lfloor \frac{n+1}{2} \rfloor})$ inequalities, each with a height bound of $O(m^{\epsilon}n^{2+\epsilon}h)$. $\lhd$

8.6 Proof of Lemma 10

Proof $\rhd$ The first step is to multiply the matrix M and the vector $(\mathbf{t}, t_0)$. Let d_M and c_M be the number of rows and columns of M, respectively, thus $M \in \mathbb{Q}^{d_M \times c_M}$. We know that M is the coefficient matrix of $\mathsf{proj}(W^0, \{\mathbf{v}, v_0\})$. Therefore, after eliminating p variables $c_M = q + 1$, where $q = n - p$ and $d_M \leq m^{\frac{n}{2}}$. Also, we have $\mathsf{height}(M) \in O(m^{\epsilon}n^{2+\epsilon}h)$. With these specifications, the multiplication step and the rank computation step need $O(m^{\frac{n}{2}}n^{2+\epsilon}h^{1+\epsilon})$ and $O(m^{\frac{n}{2}}(q+1)^{\theta+\epsilon}h^{1+\epsilon})$ bit operations, respectively, and the claim follows after simplification. $\lhd$

8.7 Subsumption cone

Lemma 12 *The subsumption cone of Q equals to its initial redundancy test cone $\mathcal{P}$.*

Proof $\rhd$ Let $\lambda^t := [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1}$ and $\beta = \overline{v}_0$, we prove the lemma in two steps.

($\subseteq$) For any (α, β) in the subsumption cone $\mathsf{proj}(T; \{\alpha, \beta\})$, there exists $\lambda \in \mathbb{Q}^m$ satisfying $(\lambda, \alpha, \beta) \in T$. Remember that $\mathbf{A}_0 = [\mathbf{A}, \mathbf{A}']$, where $\mathbf{A}' = [\mathbf{e}_{n+1}, \ldots, \mathbf{e}_m]$ with $\mathbf{e}_i$ being the i-th canonical basis of $\mathbb{Q}^n$ for $i : n+1 \leq i \leq m$, we have $\mathbf{A}_0^{-1}\mathbf{A} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]$ with $\mathbf{e}_i$ being the i-th canonical basis of $\mathbb{Q}^n$ for $i : 1 \leq i \leq n$. Hence, $\alpha^t = \lambda^t\mathbf{A} = [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1}\mathbf{A} = \overline{\mathbf{v}}^t$. Also, we have $[\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1}\mathbf{c} \leq \beta = \overline{v}_0, [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1} \geq \mathbf{0}$. Therefore, $(\alpha, \beta) = (\overline{\mathbf{v}}, \overline{v}_0) \in \mathsf{proj}(W; \{\mathbf{v}, v_0\})$.

($\supseteq$) For any $(\overline{\mathbf{v}}, \overline{v}_0)$ in the initial redundancy test cone $\mathsf{proj}(W; \{\mathbf{v}, v_0\})$, there exists $\overline{\mathbf{w}} \in \mathbb{Q}^{m-n}$ satisfying $(\overline{\mathbf{v}}, \overline{\mathbf{w}}, \overline{v}_0) \in \mathsf{proj}(W; \{\mathbf{v}, v_0\})$. Let $\alpha = \overline{\mathbf{v}}$. Then, $\alpha^t = \overline{\mathbf{v}}^t = [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1}\mathbf{A} = \lambda^t\mathbf{A}$, $\lambda^t\mathbf{c} = [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1}\mathbf{c} \leq \overline{v}_0 = \beta$ and $\alpha^t = [\overline{\mathbf{v}}^t, \overline{\mathbf{w}}^t]\mathbf{A}_0^{-1} \geq \mathbf{0}$. Therefore, $(\overline{\mathbf{v}}, \overline{v}_0) = (\alpha, \beta) \in \mathsf{proj}(T; \{\alpha, \beta\})$. $\lhd$

References

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

[2] C. Bastoul. Code generation in the polyhedral model is easier than you think. In *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, PACT '04, pages 7–16, Washington, DC, USA, 2004. IEEE Computer Society.

[3] M. Benabderrahmane, L. Pouchet, A. Cohen, and C. Bastoul. The polyhedral model is more widely applicable than you think. In *Proceedings of the 19th joint European conference on Theory and Practice of Software, international conference on Compiler Construction*, CC'10/ETAPS'10, pages 283–303, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou. Cilk: An efficient multithreaded runtime system. *SIGPLAN Not.*, 30(8):207–216, August 1995.

[5] U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. *SIGPLAN Not.*, 43(6):101–113, June 2008.

[6] Sylvain Boulme, Alexandre Marechaly, David Monniaux, Michael Perin, and Hang Yu. The verified polyhedron library: an overview. pages 9–17, 2018.

[7] Winfried Bruns and Bogdan Ichim. Normaliz: algorithms for affine monoids and rational cones. *Journal of Algebra*, 324(5):1098–1113, 2010.

[8] Winfried Bruns, Tim Römer, Richard Sieg, and Christof Söger. Normaliz 3.0. 2008.

[9] Changbo Chen, Svyatoslav Covanov, Farnam Mansouri, Marc Moreno Maza, Ning Xie, and Yuzhen Xie. The basic polynomial algebra subprograms. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, pages 669–676, 2014.

[10] Sergei Nikolaevich Chernikov. Contraction of systems of linear inequalities. *Doklady Akademii Nauk SSSR*, 131(3):518–521, 1960.

[11] Natal'ja V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 5(2):334–337, 1965.

[12] George B Dantzig. Fourier-motzkin elimination and its dual. Technical report, 1972.

[13] P. Feautrier. Dataflow analysis of array and scalar references. *International Journal of Parallel Programming*, 20, 1991.

[14] 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`.

[15] Komei Fukuda. The CDD and CDDplus homepage. `https://www.inf.ethz.ch/personal/fukudak/cdd_home/`.

[16] Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and computer science*, pages 91–111. Springer, 1996.

[17] T. Grosser, H. Zheng, R. Aloor, A. Simbürger, A. Größlinger, and L. Pouchet. Polly - polyhedral optimization in llvm. In *First International Workshop on Polyhedral Compilation Techniques (IMPACT'11)*, Chamonix, France, April 2011.

[18] Martin Henk, Jürgen Richter-Gebert, and Günter M Ziegler. 16 basic properties of convex polytopes. *Handbook of discrete and computational geometry*, pages 255–382, 2004.

[19] Tien Huynh, Catherine Lassez, and Jean-Louis Lassez. Practical issues on the projection of polyhedral sets. *Annals of mathematics and artificial intelligence*, 6(4):295–315, 1992.

[20] Jean-Louis Imbert. Fourier's elimination: Which to choose? In *PPCP*, pages 117–129, 1993.

[21] 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.

[22] 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.

[23] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[24] Leonid Khachiyan. Fourier-motzkin elimination method. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization, Second Edition*, pages 1074–1077. Springer, 2009.

[25] David A. Kohler. Projections of convex polyhedral sets. Technical report, California Univ. at Berkeley, Operations Research Center, 1967.

[26] Hervé Le Verge. *A note on Chernikova's algorithm*. PhD thesis, INRIA, 1992.

[27] Alexandre Maréchal and Michaël Périn. Efficient elimination of redundancies in polyhedra by raytracing. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 367–385. Springer, 2017.

[28] Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184, 1970.

[29] David Monniaux. Quantifier elimination by lazy model enumeration. In *International Conference on Computer Aided Verification*, pages 585–599. Springer, 2010.

[30] Arnold Schönhage and Volker Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7(3-4):281–292, 1971.

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

[32] Arne Storjohann. *Algorithms for matrix canonical forms*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2000.

[33] Marco Terzer. *Large scale methods to enumerate extreme rays and elementary modes*. PhD thesis, ETH Zurich, 2009.

[34] S. Verdoolaege, J. Carlos Juega, A. Cohen, J. Ignacio Gómez, C. Tenllado, and F. Catthoor. Polyhedral parallel code generation for CUDA. *TACO*, 9(4):54, 2013.

Appendix: Double description method

The *double description method* works in an incremental manner. Denoting by $H_1, \ldots, H_m$ the half-spaces corresponding to the inequalities of the H-representation of C, we have $C = H_1 \cap \cdots \cap H_m$. Let $1 < i \leq m$ and assume that we have computed the extreme rays of the cone $C^{i-1} := H_1 \cap \cdots \cap H_{i-1}$. Then the i-th iteration of the DD method deduces the extreme rays of C^i from those of C^{i-1} and H_i.

Assume that the half-spaces $H_1, \ldots, H_m$ are numbered such that H_i is given by $A_i\mathbf{x} \leq 0$, where A_i is the i-th row of the representing matrix A. We consider the following partition of $\mathbb{Q}^n$:

$H_i^+ = \{\mathbf{x} \in \mathbb{Q}^n \mid A_i\mathbf{x} > 0\}$, $H_i^0 = \{\mathbf{x} \in \mathbb{Q}^n \mid A_i\mathbf{x} = 0\}$ and $H_i^- = \{\mathbf{x} \in \mathbb{Q}^n \mid A_i\mathbf{x} < 0\}$.

Assume that we have found the DD-pair (A^{i-1}, R^{i-1}) of C^{i-1}. Let J be the set of the column indices of R^{i-1}. We use the above partition $\{H_i^+, H_i^0, H_i^-\}$ to partition J as follows:

$J_i^+ = \{j \in J \mid \mathbf{r}_j \in H^+\}$, $J_i^0 = \{j \in J \mid \mathbf{r}_j \in H^0\}$ and $J_i^- = \{j \in J \mid \mathbf{r}_j \in H^-\}$, where $\{\mathbf{r}_j \mid j \in J\}$ is the set of the columns of R^{i-1}, hence the set of the extreme rays of C^{i-1}.

For future reference, let us denote by $\mathsf{partition}(\mathsf{J}, \mathsf{A_i})$ the function which returns J^+, J^0, J^- as defined above. The proof can be found in [16].

Lemma 13 (Double description method) *Let $J' := J^+ \cup J^0 \cup (J^+ \times J^-)$. Let R^i be the $(n \times |J'|)$-matrix consisting of*

- *the columns of R^{i-1} with index in $J^+ \cup J^0$, followed by*
- *the vectors $\mathbf{r'}_{(j,j')}$ for $(j, j') \in (J^+ \times J^-)$, where*

$$\mathbf{r'}_{(j,j')} = (A_i \mathbf{r}_j)\mathbf{r}_{j'} - (A_i \mathbf{r}_{j'})\mathbf{r}_j,$$

Then, the pair (A^i, R^i) *is a DD pair of* C^i.

The most efficient way to start the incremental process is to choose the largest sub-matrix of A with linearly independent rows; we call this matrix A^0. Indeed, denoting by C^0 the cone with A^0 as representation matrix, the matrix A^0 is invertible and its inverse gives the extreme rays of C^0, that is:

$$\mathsf{ExtremeRays}(C^0) = (A^0)^{-1}.$$

Therefore, the first DD-pair that the above incremental step should take as input is $(A^0, (A^0)^{-1})$.

The next key point towards a practically efficient DD method is to observe that most of the vectors $\mathbf{r'}_{(j,j')}$ in Lemma 13 are redundant. Indeed, Lemma 13 leads to a construction of a generating matrix of C (in fact, this would be Algorithm 5 where Lines 13 and 16 are suppressed) producing a double exponential number of rays (w.r.t. the ambient dimension n) whereas Equation (1) guarantees that the number of extreme rays of a polyhedral cone is singly exponential in its ambient dimension. To deal with this issue of redundancy, we need the notion of *adjacent* extreme rays.

Definition 2 (Adjacent extreme rays) *Two distinct extreme rays* $\mathbf{r}$ *and* $\mathbf{r}'$ *of the polyhedral cone* C *are called adjacent if they span a* 2*-dimensional face of* C. [3]

The following lemma shows how we can test whether two extreme rays are adjacent or not. The proof can be found in [16].

Proposition 1 (Adjacency test) *Let* $\mathbf{r}$ *and* $\mathbf{r}'$ *be two distinct rays of* C. *Then, the following statements are equivalent:*

1. $\mathbf{r}$ *and* $\mathbf{r}'$ *are adjacent extreme rays,*
2. $\mathbf{r}$ *and* $\mathbf{r}'$ *are extreme rays and* $\mathrm{rank}(A_{\zeta(\mathbf{r})\cap\zeta(\mathbf{r}')}) = n - 2$,
3. *if* $\mathbf{r}''$ *is a ray of* C *with* $\zeta(\mathbf{r}) \cap \zeta(\mathbf{r}') \subseteq \zeta(\mathbf{r}'')$, *then* $\mathbf{r}''$ *is a positive multiple of either* $\mathbf{r}$ *or* $\mathbf{r}'$.

It should be noted that the second statement is related to algebraic test for extreme rays while the third one is related to the combinatorial test.

Based on Proposition 1, we have Algorithm 4 for testing whether two extreme rays are adjacent or not.

The following lemma explains how to obtain (A^i, R^i) from (A^{i-1}, R^{i-1}), where A^{i-1} (resp. A^i) is the sub-matrix of A consisting of its first $i-1$ (resp. i) rows. The double description method is a direct application of this lemma, see [16] for details.

Lemma 14 *As above, let* (A^{i-1}, R^{i-1}) *be a DD-pair and denote by* J *be the set of indices of the columns of* R^{i-1}. *Assume that* $\mathrm{rank}(A^{i-1}) = n$ *holds. Let* $J' := J^- \cup J^0 \cup \mathrm{Adj}$, *where* Adj *is the set of the pairs* $(j, j') \in J^+ \times J^-$ *such that* $\mathbf{r}_j$, *and* $\mathbf{r}_{j'}$ *are adjacent as extreme rays of* C^{i-1}, *the cone with* A^{i-1} *as representing matrix. Let* R^i *be the* $(n \times |J'|)$*-matrix consisting of*

[3] We do not use the minimal face, as it used in the main reference because it makes confusion.

Algorithm 4 AdjacencyTest

1: **Input:** $(A, \mathbf{r}, \mathbf{r}')$, where $A \in \mathbb{Q}^{m \times n}$ is the representation matrix of cone C, $\mathbf{r}$ and $\mathbf{r}'$ are two extreme rays of C
2: **Output:** true if $\mathbf{r}$ and $\mathbf{r}'$ are adjacent, false otherwise
3: $\mathbf{s} := A\mathbf{r},\ \mathbf{s}' := A\mathbf{r}'$
4: let $\zeta(\mathbf{r})$ and $\zeta(\mathbf{r}')$ be set of indices of zeros in $\mathbf{s}$ and $\mathbf{s}'$ respectively
5: $\zeta := \zeta(\mathbf{r}) \cap \zeta(\mathbf{r}')$
6: **if** $\text{rank}(A_\zeta) = n - 2$ **then**
7: **return** true
8: **else**
9: **return** false
10: **end if**

- *the columns of* R^{i-1} *with index in* $J^- \cup J^0$*, followed by*
- *the vectors* $\mathbf{r}'_{(j,j')}$ *for* $(j, j') \in (J^+ \times J^-)$*, where*

$$\mathbf{r}'_{(j,j')} = (A_i \mathbf{r}_j)\mathbf{r}_{j'} - (A_i \mathbf{r}'_j)\mathbf{r}_j,$$

Then, the pair (A^i, R^i) *is a DD pair of* C^i*. Furthermore, if* R^{i-1} *is a minimal generating matrix for the representation matrix* A^{i-1}*, then* R^i *is also a minimal generating matrix for the representation matrix* A^i*.*

Using Proposition 1 and Lemma 14 we can obtain Algorithm 5 [4] for computing the extreme rays of a cone.

[4] In this algorithm, A^i shows the representation matrix in step i

Algorithm 5 DDmethod

Input: a matrix $A \in \mathbb{Q}^{m \times n}$, a representation matrix of a pointed cone C
Output: R, the minimal generating matrix of C
let K be the set of indices of A's independent rows
$A^0 := A_K$
$R^0 := (A^0)^{-1}$
let J be set of column indices of R^0
while $K \neq \{1, \cdots, m\}$ **do**
  select a A-row index $i \notin K$
  $J^+,\ J^0,\ J^- := \mathsf{partition}(J, A_i)$
  add vectors with indices in J^+ and J^0 as columns to R^i
  for $p \in J^+$ **do**
    for $n \in J^-$ **do**
      if AdjacencyTest$(A^{i-1}, \mathbf{r}_p, \mathbf{r}_n) =$ true **then**
        $\mathbf{r}_{\text{new}} := (A_i \mathbf{r}_p)\mathbf{r}_n - (A_i \mathbf{r}_n)\mathbf{r}_p$
        add $\mathbf{r}_{\text{new}}$ as columns to R^i
      end if
    end for
  end for
  let J be set of indices in R^i
end while