



Complexity Estimates for Fourier-Motzkin Elimination

Rui-Juan Jing¹, Marc Moreno-Maza², and Delaram Talaashrafi²(✉)

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

² Western University, London, ON, Canada
moreno@csd.uwo.ca, dtalaash@uwo.ca

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 [3, 4, 6, 15, 16, 21, 38]. 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 (DD) method and Fourier-Motzkin Elimination (FME) have a lot in common, and, they are considered as the same algorithm in the paper [8] of Winfried Bruns and Bogdan Ichim, they are not totally similar. Quoting Komei Fukuda and Alain Prodon from [18]: “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 [14] of George Danzang and Section 12.2 of the book [35] of Alexander Schrijver, for a presentation 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 so-called *minimal representation* of the projection of a polyhedron. Leonid Khachiyan explained in [28] 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 [29]. As observed by Jean-Louis Imbert in [24], the method he proposed in that paper and that of Sergei N. Chernikov in [11] are equivalent. On the topic of finding extreme rays of a polyhedral set in H-representation, see Natałja V. Chernikova [12], Hervé Le Verge [30] and Komei Fukuda [18]. 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 the Fourier-Motzkin Elimination algorithm as described in Schrijver's book [35]. In fact, our goal is to find the 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, considering a non-empty, full-dimensional, and pointed polyhedron as the input. Our approach is based on an improved version of a method proposed by Egon Balas in [2]. 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 a non-empty, full-dimensional, and pointed polyhedron $Q \subseteq \mathbb{Q}^n$ as input, 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, as defined in the landmark book [19].

Therefore, we obtain a more favourable estimate than the one presented in [25, 26] for Fourier-Motzkin Elimination with a removal of the redundant inequalities via linear programming. Indeed, in those papers, the estimate is $O(n^2 m^{2n} \text{LP}(n, 2^n h n^2 m^n))$ bit operations, where $\text{LP}(d, H)$ is an upper bound for the number of bit operations required for solving a linear program in n variables and with total bit size H . For instance, in the case of Karmarkar's algorithm [27], we have $\text{LP}(d, H) \in O(d^{3.5} H^2 \cdot \log H \cdot \log \log H)$. 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 [25, 26].

Our algorithm is stated in Sect. 4 and follows the revisited version of Balas' algorithm presented in Sect. 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 Sect. 4 using the BPAS library [10] publicly available at www.bpaslib.org. We have compared our code against other implementations of Fourier-Motzkin Elimination including the CDD library [17]. Our experimental results, reported in Sect. 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. As mentioned above, Sect. 3 contains our revisited version of Balas' method and detailed proofs of its correctness. Based on this, Sect. 4 presents a new algorithm producing a *minimal projected representation* for a given full-dimensional pointed polyhedron. Complexity results are established in Sect. 5. In Sect. 6 we report on our experimentation and in Sect. 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. We conclude this section with the cost model that we shall use for complexity analysis, see Sect. 2.4. As we omit most proofs, for more details please refer to [18, 35, 37]. 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 \mathbb{R} of real numbers.

Throughout this paper, 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, as in A^t for the transposition of matrix A . The concatenation of two column vectors \mathbf{v} and \mathbf{w} is denoted (\mathbf{v}, \mathbf{w}) , thus using parentheses, while the concatenation of two row vector \mathbf{v}^t and \mathbf{w}^t is denoted $[\mathbf{v}^t, \mathbf{w}^t]$, thus using square brackets. For a matrix A and an integer k , we denote by A_k is the row of index k in A . More generally, if K is a set of integers, we denote by A_K the sub-matrix of A with row indices in K .

2.1 Polyhedral Cones and Polyhedra

Polyhedral Cone. A set 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 $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ be a set of vectors in \mathbb{Q}^n . The *cone generated* by $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$, denoted by $\text{Cone}(\mathbf{x}_1, \dots, \mathbf{x}_m)$, is the smallest convex cone containing those vectors. In other words, we have $\text{Cone}(\mathbf{x}_1, \dots, \mathbf{x}_m) = \{\lambda_1 \mathbf{x}_1 + \dots + \lambda_m \mathbf{x}_m \mid \lambda_1 \geq 0, \dots, \lambda_m \geq 0\}$. 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 $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ holds, for a matrix $A \in \mathbb{Q}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{Q}^m$, for some positive integer 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 this inequality 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 this lemma has different variants, we simply mention here the variant from [35] which we use in this paper.

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 $\mathbf{t} \in \mathbb{Q}^n$ with $\mathbf{t} \geq \mathbf{0}$ satisfying $A\mathbf{t} = \mathbf{b}$ if and if $\mathbf{y}^t \mathbf{b} \geq 0$ holds for every vector $\mathbf{y} \in \mathbb{Q}^m$ satisfying $\mathbf{y}^t A \geq \mathbf{0}$.*

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 $\mathbf{b} \in \mathbb{Q}^m$. Assume $A\mathbf{x} \leq \mathbf{b}$ is a consistent linear inequality system. Then, the inequality $\mathbf{c}^t \mathbf{x} \leq c_0$ is redundant w.r.t. $A\mathbf{x} \leq \mathbf{b}$ if and only if there exists a vector $\mathbf{t} \geq \mathbf{0}$ and a number $\lambda \geq 0$ satisfying $\mathbf{c}^t = \mathbf{t}^t A$ 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 $\text{CharCone}(P)$ and defined by $\text{CharCone}(P) = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{x} + \mathbf{y} \in P, \forall \mathbf{x} \in P\} = \{\mathbf{y} \mid A\mathbf{y} \leq \mathbf{0}\}$. The *linearity space* of the polyhedron P is the linear space denoted by $\text{LinearSpace}(P)$ and defined as $\text{CharCone}(P) \cap -\text{CharCone}(P) = \{\mathbf{y} \mid A\mathbf{y} = \mathbf{0}\}$, where $-\text{CharCone}(P)$ is the set of the $-\mathbf{y}$ for $\mathbf{y} \in \text{CharCone}(P)$. 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 $\text{dim}(P)$, is the maximum number of linearly independent vectors in P . We say that P is *full-dimensional* whenever $\text{dim}(P) = n$ 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 equation. 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}_{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 $\text{dim}(\text{LinearSpace}(C)) = t$. 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 $\text{CharCone}(P)$. 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 $\text{ExtremeRays}(C)$ the set of extreme rays of the cone C . Recall that all cones considered here are polyhedral. The following, see [32,37], 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 $\mathbf{t} \in C$ and a cone $C = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{0}\}$, 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 call the *algebraic test*, can be found in [18]: Let $\mathbf{r} \in C$. Then, the ray \mathbf{r} is an extreme ray of C if and only if we have $\text{rank}(A_{\zeta(\mathbf{r})}) = n - 1$. 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 [18], 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 defined as: $C^* = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{y}^t\mathbf{x} \leq 0, \forall \mathbf{x} \in C\}$. The proof of the following property can be found in [35]: 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 $\text{HomCone}(P)$ and defined by: $\text{HomCone}(P) = \{(\mathbf{x}, x_{\text{last}}) \in \mathbb{Q}^{n+1} \mid A\mathbf{x} - \mathbf{b}x_{\text{last}} \leq \mathbf{0}, x_{\text{last}} \geq 0\}$.

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 $A_i\mathbf{x} - b_ix_{\text{last}} \leq 0$ is redundant in $\text{HomCone}(P)$.*

Theorem 1 (Extreme rays of the homogenized cone). *Every extreme ray of the homogenized cone $\text{HomCone}(P)$ 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 Sect. 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 $\text{Cone}(R)$, where R is a matrix, the columns of which are the extreme rays of C (called *V-representation* of C). The pseudo-code of the *double description method*, as presented in [18], and implemented in the CDD library [17] is shown in Algorithm 1. This algorithm calls `partition` and `AdjacencyTest` functions. Given a set of vectors J and an inequality A_i , the `partition` function places each member j of J into one of the sets J^+ , J^0 , J^- , according to the sign (positive, null or negative) of A_{ij} . Moreover, the `AdjacencyTest` determines adjacency of the input extreme rays. This algorithm produces the V-representation of a pointed polyhedral cone given by its H-representation. Some of the results presented in our paper depend on algebraic complexity estimates for the double description method. In [18], 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 9.

Algorithm 1. DDmethod

```

1: Input: a matrix  $A \in \mathbb{Q}^{m \times n}$  defining the H-representation of a pointed cone  $C$ 
2: Output: a matrix  $R$  defining the V-representation of  $C$ 
3: let  $K$  be the set of the indices of  $A$ 's independent rows
4:  $R' := (A_K)^{-1}$ 
5: let  $J$  be the set of the columns of  $R'$ 
6: while  $K \neq \{1, \dots, m\}$  do
7:   select a  $A$ -row index  $i \notin K$ 
8:   set  $R'$  to be an empty matrix
9:    $J^+, J^0, J^- := \text{partition}(J, A_i)$ 
10:  add the vectors in  $J^+$  and  $J^0$  as columns to  $R'$ 
11:  for  $p \in J^+$  do
12:    for  $n \in J^-$  do
13:      if  $\text{AdjacencyTest}(A_K, \mathbf{r}_p, \mathbf{r}_n) = \text{true}$  then
14:         $\mathbf{r}_{\text{new}} := (A_i \mathbf{r}_p) \mathbf{r}_n - (A_i \mathbf{r}_n) \mathbf{r}_p$ 
15:        add  $\mathbf{r}_{\text{new}}$  as a column to  $R'$ 
16:      end if
17:    end for
18:  end for
19:  let  $J$  be the set of the columns of  $R'$ 
20:   $K := K \cup \{i\}$ 
21: end while
22: let  $R$  be the matrix created by the vectors in  $J$  as its columns
23: return ( $R$ )

```

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 $\text{proj}(P; \mathbf{x})$ the *projection of P on \mathbf{x}* , that is, the subset of \mathbb{Q}^q defined by $\text{proj}(P; \mathbf{x}) = \{\mathbf{x} \in \mathbb{Q}^q \mid \exists \mathbf{u} \in \mathbb{Q}^p, (\mathbf{u}, \mathbf{x}) \in P\}$.

Fourier-Motzkin elimination (FME for short) is an algorithm computing the projection $\text{proj}(P; \mathbf{x})$ 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 $\text{proj}(P; \mathbf{x})$ is also a polyhedron.

Let ℓ_1, ℓ_2 be two inequalities: $a_1x_1 + \dots + a_nx_n \leq c_1$ and $b_1x_1 + \dots + b_nx_n \leq c_2$. 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 $\text{Combine}(\ell_1, \ell_2, x_i)$, is:

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

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

Theorem 2 (Fourier-Motzkin theorem [29]). *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 :

$$\begin{aligned} S^+ &= \{\ell \in S \mid \text{coeff}(\ell, x_i) > 0\}, \\ S^- &= \{\ell \in S \mid \text{coeff}(\ell, x_i) < 0\}, \\ S^0 &= \{\ell \in S \mid \text{coeff}(\ell, x_i) = 0\}. \end{aligned}$$

We construct the following system of linear inequalities:

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

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

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 $O((\frac{m}{2})^{2p})$ inequalities. Thus, FME is double exponential in p .

On the other hand, from [33] and [26], 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 $\text{height}(\frac{a}{b})$, as $\log \max(|a|, |b|)$. For a given matrix $A \in \mathbb{Q}^{m \times n}$, let $\|A\|$ denote the infinity norm of A , that is, the maximum absolute value of a coefficient in A . We define the height of A , denoted by $\text{height}(A) := \text{height}(\|A\|)$, 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 [36]. 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 remainder) on input $a, b \in \mathbb{Z}$ with $|a|, |b| < 2^k$. Using the multiplication algorithm of Arnold Schönhage and Volker Strassen [34] 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 $\mathcal{MM}(n) \in O(n^\theta)$.

We now give the 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 $h_A = \text{height}(A)$ and $h_B = \text{height}(B)$. 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 Sect. 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 the minimal representation of a full-dimensional pointed polyhedron, as well as its projections. Throughout this section, we use Q to denote a full-dimensional pointed polyhedron in \mathbb{Q}^n , where

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

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 $\text{proj}(Q; \mathbf{x})$ given by $\text{proj}(Q; \mathbf{x}) := \{\mathbf{x} \mid \exists \mathbf{u}, s.t. (\mathbf{u}, \mathbf{x}) \in Q\}$. We call the cone $C := \{\mathbf{y} \in \mathbb{Q}^m \mid \mathbf{y}^t \mathbf{A} = 0 \text{ and } \mathbf{y} \geq \mathbf{0}\}$ 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 $\text{proj}(Q; \mathbf{x})$:

Lemma 5 ([11]). *The projection $\text{proj}(Q; \mathbf{x})$ of the polyhedron Q can be represented by*

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

where C is the projection cone of Q defined above.

Lemm 5 provides the main idea of the block elimination method. However, the representation produced in this way may have redundant inequalities. In [2], 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 Balas' algorithm¹ 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, $\mathbf{e}_{q+1}, \dots, \mathbf{e}_m$, 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:

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

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

Construction of Q^0 . We define:

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

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}, \dots, x_m) . Since the extra variables (x_{q+1}, \dots, x_m) are assigned to zero, we note that $\text{proj}(Q; \mathbf{x})$ and $\text{proj}(Q^0; \mathbf{x}')$ are "isomorphic" by means of the bijection Φ :

$$\Phi : \begin{array}{l} \text{proj}(Q; \mathbf{x}) \rightarrow \text{proj}(Q^0; \mathbf{x}') \\ (x_1, \dots, x_q) \mapsto (x_1, \dots, x_q, 0, \dots, 0) \end{array}$$

In the following, we will treat $\text{proj}(Q; \mathbf{x})$ and $\text{proj}(Q^0; \mathbf{x}')$ 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

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

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

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

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

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.

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

Lemma 6. *The operations “computing the characteristic cone” and “computing projections” commute. To be precise, we have:*

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

Proof. By the definition of the characteristic cone, we have $\text{CharCone}(Q) = \{(\mathbf{u}, \mathbf{x}) \mid A\mathbf{u} + B\mathbf{x} \leq \mathbf{0}\}$, 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 $\text{proj}(\text{CharCone}(Q); \mathbf{x})$ has the same left-hand side as $\text{proj}(Q; \mathbf{x})$. This is obvious with the Fourier-Motzkin Elimination procedure.

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

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

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

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

(\supseteq) For any $(\bar{\mathbf{v}}, -\bar{v}_0) \in \overline{\text{proj}(W^0; \{\mathbf{v}, v_0\})}$, we need to show that

$$[\bar{\mathbf{v}}^t, -\bar{v}_0][\mathbf{x}^t, x_{\text{last}}]^t \leq 0$$

holds when $(\mathbf{x}, x_{\text{last}}) \in \text{HomCone}(\text{proj}(Q; \mathbf{x}))$. Remember that Q is pointed. As a result, $\text{HomCone}(\text{proj}(Q; \mathbf{x}))$ is also pointed. Therefore, we only need to verify the desired property for the extreme rays of $\text{HomCone}(\text{proj}(Q; \mathbf{x}))$, which either have the form $(\mathbf{s}, 1)$ or $(\mathbf{s}, 0)$ (Theorem 1). Before continuing, we should notice that since $(\bar{\mathbf{v}}, \bar{v}_0) \in \text{proj}(W^0; \{\mathbf{v}, v_0\})$, there exists $\bar{\mathbf{w}}$ such that $[\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t, \bar{v}_0] \in W^0$. Cases 1 and 2 below conclude that $(\bar{\mathbf{v}}, -\bar{v}_0) \in \text{HomCone}(\text{proj}(Q; \mathbf{x}))^*$ holds.

Case 1: for the form $(\mathbf{s}, 1)$, we have $\mathbf{s} \in \text{proj}(Q; \mathbf{x})$. Indeed, \mathbf{s} is an extreme point of $\text{proj}(Q; \mathbf{x})$. Hence, there exists $\bar{\mathbf{u}} \in \mathbb{Q}^p$, such that we have $A\bar{\mathbf{u}} + B\mathbf{s} \leq \mathbf{c}$. By construction of Q^0 , we have $A_0\bar{\mathbf{u}} + B_0\mathbf{s}' \leq \mathbf{c}_0$, where $\mathbf{s}' = [\mathbf{s}^t, s_{q+1}, \dots, s_m]^t$ with $s_{q+1} = \dots = s_m = 0$. Therefore, we have:

$$[\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}A_0\bar{\mathbf{u}} + [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}B_0\mathbf{s}' \leq [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0.$$

This leads us to $\bar{\mathbf{v}}^t\mathbf{s} = [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]\mathbf{s}' \leq [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}\mathbf{c}_0 \leq \bar{v}_0$. Therefore, we have $[\bar{\mathbf{v}}^t, -\bar{v}_0][\mathbf{s}^t, x_{\text{last}}]^t \leq 0$, as desired.

Case 2: for the form $(\mathbf{s}, 0)$, we have

$$\mathbf{s} \in \text{CharCone}(\text{proj}(Q; \mathbf{x})) = \text{proj}(\text{CharCone}(Q); \mathbf{x}).$$

Thus, there exists $\bar{\mathbf{u}} \in \mathbb{Q}^p$ such that $A\bar{\mathbf{u}} + B\mathbf{s} \leq \mathbf{0}$. Similarly to Case 1, we have $[\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}A_0\bar{\mathbf{u}} + [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}B_0\mathbf{s}' \leq [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t]B_0^{-1}\mathbf{0}$. Therefore, we have

$\bar{\mathbf{v}}^t \mathbf{s} = [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t] \mathbf{s}' \leq [\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t] B_0^{-1} \mathbf{0} = 0$, and thus, we have $[\bar{\mathbf{v}}^t, -\bar{v}_0] [\mathbf{s}^t, x_{\text{last}}]^t \leq 0$, as desired.

(\subseteq) For any $(\bar{\mathbf{y}}, \bar{y}_0) \in \text{HomCone}(\text{proj}(Q; \mathbf{x}))^*$, we have $[\bar{\mathbf{y}}^t, \bar{y}_0] [\mathbf{x}^t, x_{\text{last}}]^t \leq 0$ for all $(\mathbf{x}, x_{\text{last}}) \in \text{HomCone}(\text{proj}(Q; \mathbf{x}))$. For any $\bar{\mathbf{x}} \in \text{proj}(Q; \mathbf{x})$, we have $\bar{\mathbf{y}}^t \bar{\mathbf{x}} \leq -\bar{y}_0$ since $(\bar{\mathbf{x}}, 1) \in \text{HomCone}(\text{proj}(Q; \mathbf{x}))$. Therefore, we have $\bar{\mathbf{y}}^t \mathbf{x} \leq -\bar{y}_0$, for all $\mathbf{x} \in \text{proj}(Q; \mathbf{x})$, which makes the inequality $\bar{\mathbf{y}}^t \mathbf{x} \leq -\bar{y}_0$ redundant in the system $\{\mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{x} \leq \mathbf{c}\}$. 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 $\mathbf{p}^t \mathbf{A} = \mathbf{0}$, $\bar{\mathbf{y}} = \mathbf{p}^t \mathbf{B}$, $\bar{y}_0 = \mathbf{p}^t \mathbf{c} + \lambda$. 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 $\mathbf{e}_{q+1}, \dots, \mathbf{e}_m$. Let $\bar{\mathbf{w}} = \mathbf{p}^t B'$. We then have

$$\{\mathbf{p}^t A_0 = \mathbf{0}, [\bar{\mathbf{y}}^t, \bar{\mathbf{w}}^t] = \mathbf{p}^t B_0, -\bar{y}_0 \geq \mathbf{p}^t \mathbf{c}_0, \mathbf{p} \geq \mathbf{0}\},$$

which is equivalent to

$$\{\mathbf{p}^t = [\bar{\mathbf{y}}^t, \bar{\mathbf{w}}^t] B_0^{-1}, [\bar{\mathbf{y}}^t, \bar{\mathbf{w}}^t] B_0^{-1} A_0 = \mathbf{0}, -\bar{y}_0 \geq [\bar{\mathbf{y}}^t, \bar{\mathbf{w}}^t] B_0^{-1} \mathbf{c}_0, [\bar{\mathbf{y}}^t, \bar{\mathbf{w}}^t] B_0^{-1} \geq \mathbf{0}\}.$$

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

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

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

Proof. By Theorem 3, the minimal representation of the homogenized cone $\text{HomCone}(\text{proj}(Q; \mathbf{x}))$ is given exactly by

$$\{\mathbf{v}\mathbf{x} - v_0 x_{\text{last}} \leq 0 \mid (\mathbf{v}, v_0) \in \text{ExtremeRays}(\text{proj}(W^0; (\mathbf{v}, v_0)))\}.$$

Using Lemma 4, any minimal representation of $\text{HomCone}(\text{proj}(Q; \mathbf{x}))$ has at most one more inequality than any minimal representation of $\text{proj}(Q; \mathbf{x})$. This extra inequality is $x_{\text{last}} \geq 0$ and, in this case, $\text{proj}(W^0; (\mathbf{v}, v_0))$ will have the extreme ray $(\mathbf{0}, 1)$, which can be detected easily. Therefore, the minimal representation of $\text{proj}(Q; \mathbf{x})$ is given by

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

For simplicity, we call the cone $\text{proj}(W^0; \{\mathbf{v}, v_0\})$ 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 4.

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 Sect. 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 7, we only need to compute the initial redundancy test cone, using Algorithm 2, 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 3, 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 a *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 notion of projected representation was introduced in [25,26] and will be reviewed in Definition 1.

For convenience, we rewrite the input polyhedron Q defined in Eq. (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 2. 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 $\mathbf{A}' = [\mathbf{e}_{n+1}, \dots, \mathbf{e}_m]$ 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: $\mathcal{P} = \text{proj}(W; \{\mathbf{v}, v_0\})$;
 - 4: **return** (\mathcal{P});
-

Remark 1. There are two important points about Algorithm 2. First, we only need a representation of the initial redundancy test cone. This representation does not need to be minimal. Therefore, calling Algorithm 2 in Algorithm 4 (which computes a minimal projected representation of a polyhedron) does not lead to a recursive call to Algorithm 4. Second, to compute the projection $\text{proj}(W; \{\mathbf{v}, v_0\})$, 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 out 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.

Lemma 7. *A 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. To distinguish from the construction of $\mathcal{P}(Q)$, we rename the variables $\mathbf{v}, \mathbf{w}, v_0$ as $\mathbf{v}_{\mathbf{u}}, \mathbf{w}_{\mathbf{u}}, v_{\mathbf{u}}$, when constructing W^0 and computing the test cone $\mathcal{P}_{\mathbf{u}}(Q)$.

That is, we have $\mathcal{P}_{\mathbf{u}}(Q) = \text{proj}(W^0; \{\mathbf{v}_{\mathbf{u}}, v_{\mathbf{u}}\})$, where W^0 is the set of all $(\mathbf{v}_{\mathbf{u}}, \mathbf{w}_{\mathbf{u}}, v_{\mathbf{u}}) \in \mathbb{Q}^q \times \mathbb{Q}^{m-q} \times \mathbb{Q}$ satisfying

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

while we have $\mathcal{P}(Q) = \text{proj}(W; \{\mathbf{v}, v_0\})$ 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 $\{(\mathbf{v}, \mathbf{w}, v_0) \mid -[\mathbf{v}^t, \mathbf{w}^t]A_0^{-1}\mathbf{c} + v_0 \geq 0, [\mathbf{v}^t, \mathbf{w}^t]A_0^{-1} \geq \mathbf{0}\}$.

By Step 1 of Algorithm 2, $[\mathbf{v}^t, \mathbf{w}^t]A_0^{-1}A = \mathbf{v}^t$ holds for all $(\mathbf{v}, \mathbf{w}, v_0) \in W$. We can 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} . Then, we have $[\mathbf{v}^t, \mathbf{w}^t]A_0^{-1}A = \mathbf{v}_1^t$ and $[\mathbf{v}^t, \mathbf{w}^t]A_0^{-1}B = \mathbf{v}_2^t$. Similarly, we have $[\mathbf{v}_{\mathbf{u}}^t, \mathbf{w}_{\mathbf{u}}^t]B_0^{-1}A = \mathbf{0}$ and $[\mathbf{v}_{\mathbf{u}}^t, \mathbf{w}_{\mathbf{u}}^t]B_0^{-1}B = \mathbf{v}_{\mathbf{u}}^t$ for all $(\mathbf{v}_{\mathbf{u}}, \mathbf{w}_{\mathbf{u}}, v_{\mathbf{u}}) \in W^0$. This lemma holds if we can show $\mathcal{P}_{\mathbf{u}} = \mathcal{P}|_{\mathbf{v}_1=0}$. We prove this in two steps:

(\subseteq) For any $(\bar{\mathbf{v}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) \in \mathcal{P}_{\mathbf{u}}(Q)$, there exists $\bar{\mathbf{w}}_{\mathbf{u}} \in \mathbb{Q}^{m-q}$, such that

$$(\bar{\mathbf{v}}_{\mathbf{u}}, \bar{\mathbf{w}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) \in W^0.$$

Let $[\bar{\mathbf{v}}^t, \bar{\mathbf{w}}^t] := [\bar{\mathbf{v}}_{\mathbf{u}}^t, \bar{\mathbf{w}}_{\mathbf{u}}^t]B_0^{-1}A_0$, where $\bar{\mathbf{v}}^t = [\bar{\mathbf{v}}_1^t, \bar{\mathbf{v}}_2^t]$ ($\bar{\mathbf{v}}_1 \in \mathbb{Q}^p, \bar{\mathbf{v}}_2 \in \mathbb{Q}^{n-p}$, and $\bar{\mathbf{w}} \in \mathbb{Q}^{m-n}$). Then, because $(\bar{\mathbf{v}}_{\mathbf{u}}, \bar{\mathbf{w}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) \in W^0$, we have $\bar{\mathbf{v}}_1^t = [\bar{\mathbf{v}}_{\mathbf{u}}^t, \bar{\mathbf{w}}_{\mathbf{u}}^t]B_0^{-1}A = \mathbf{0}$ and $\bar{\mathbf{v}}_2^t = [\bar{\mathbf{v}}_{\mathbf{u}}^t, \bar{\mathbf{w}}_{\mathbf{u}}^t]B_0^{-1}B = \bar{\mathbf{v}}_{\mathbf{u}}$. Let $\bar{v}_0 = \bar{v}_{\mathbf{u}}$, it is easy to verify that $(\bar{\mathbf{v}}, \bar{\mathbf{w}}, \bar{v}_0) \in W$. Therefore, $(\mathbf{0}, \bar{\mathbf{v}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) = (\bar{\mathbf{v}}, \bar{v}_0) \in \mathcal{P}(Q)$.

(\supseteq) For any $(\mathbf{0}, \bar{\mathbf{v}}_2, \bar{v}_0) \in \mathcal{P}(Q)$, there exists $\bar{\mathbf{w}} \in \mathbb{Q}^{m-n}$, such that

$$(\mathbf{0}, \bar{\mathbf{v}}_2, \bar{\mathbf{w}}, \bar{v}_0) \in W.$$

Let $[\bar{\mathbf{v}}_{\mathbf{u}}^t, \bar{\mathbf{w}}_{\mathbf{u}}^t] := [\mathbf{0}, \bar{\mathbf{v}}_2^t, \bar{\mathbf{w}}^t]A_0^{-1}B_0$. We have $\bar{\mathbf{v}}_{\mathbf{u}} = [\mathbf{0}, \bar{\mathbf{v}}_2^t, \bar{\mathbf{w}}^t]A_0^{-1}B = \bar{\mathbf{v}}_2$. Let $\bar{v}_{\mathbf{u}} = \bar{v}_0$, it is easy to verify that $(\bar{\mathbf{v}}_{\mathbf{u}}, \bar{\mathbf{w}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) \in W^0$. Therefore, $(\bar{\mathbf{v}}_2, \bar{v}_0) = (\bar{\mathbf{v}}_{\mathbf{u}}, \bar{v}_{\mathbf{u}}) \in \mathcal{P}_{\mathbf{u}}(Q)$.

Consider again the polyhedron $Q = \{\mathbf{y} \in \mathbb{Q}^n \mid \mathbf{A}\mathbf{y} \leq \mathbf{c}\}$, where $\mathbf{A} = [A, B] \in \mathbb{Q}^{m \times n}$, $n = p + q$ and $\mathbf{y} = [\mathbf{u}^t, \mathbf{x}^t]^t \in \mathbb{Q}^n$. Fix a variable ordering, say $y_1 > \dots > y_n$. For $1 \leq i \leq n$, we denote by $\mathbf{A}^{(y_i)}$ the inequalities in the representation $\mathbf{A}\mathbf{y} \leq \mathbf{c}$ of Q whose largest variable is y_i . We denote by $\text{ProjRep}(Q; y_1 > \dots > y_n)$ the linear system $\mathbf{A}^{(y_1)}$ if $n = 1$ and the conjunction of $\mathbf{A}^{(y_1)}$ and $\text{ProjRep}(\text{proj}(Q; \mathbf{y}_2); y_2 > \dots > y_n)$ otherwise, where $\mathbf{y}_2 = (y_2, \dots, y_n)$. Of course, $\text{ProjRep}(Q; y_1 > \dots > y_n)$ depends on the representation which is used of Q .

Definition 1 (Projected representation). *For the polyhedron $Q \subseteq \mathbb{Q}^n$, we call projected representation of Q w.r.t. the variable order $y_1 > \dots > y_n$ any linear system of the form $\text{ProjRep}(Q; y_1 > \dots > y_n)$. We say that such a linear system P is a minimal projected representation of Q 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, \dots, y_n .*

We can generate a *minimal projected representation* of a polyhedron, w.r.t. an specific variable ordering by Algorithm 4.

Algorithm 3. Extreme ray test

Input: (\mathcal{P}, ℓ) , 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** $\text{rank}(M_{\zeta(\mathbf{s})}) = n$ **then**
 - 4: return (true);
 - 5: **else**
 - 6: return (false);
 - 7: **end if**
-

5 Complexity Estimates

In this section, we analyze the computational complexity of Algorithm 4, which computes a 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 a minimal projected representation of a polyhedron is singly exponential in the dimension n of the ambient space. The most consuming procedure in Algorithm 4 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 begin this section by computing the bit complexity of the double description algorithm.

Lemma 8 (Coefficient bound of extreme rays). *Let*

$$S = \{\mathbf{x} \in \mathbb{Q}^n \mid A\mathbf{x} \leq \mathbf{0}\}$$

be a 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. From the properties of extreme rays, see Sect. 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 $A'\mathbf{r} = \mathbf{0}$. This means that \mathbf{r} is in the null-space of A' . Thus, the claim follows by proposition 6.6 of [36].

Algorithm 4. 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 ;

```

1: Generate the initial redundancy test cone  $\mathcal{P}$  by Algorithm 2;
2:  $S_0 := \{ \}$ ;
3: for  $i$  from 1 to  $m$  do
4:   Let  $f$  be the result of applying Algorithm 3 with the inputs  $\mathcal{P}$  and  $\mathbf{A}_i\mathbf{y} \leq \mathbf{c}_i$ ;
5:   if  $f = \text{true}$  then
6:      $S_0 := S_0 \cup \{\mathbf{A}_i\mathbf{y} \leq \mathbf{c}_i\}$ ;
7:   end if
8: end for
9:  $\mathcal{P} := \mathcal{P}|_{v_1=0}$ ;
10: for  $i$  from 0 to  $n - 1$  do
11:    $S_{i+1} := \{ \}$ ;
12:   for  $\ell_{\text{pos}} \in S_i$  with positive coefficient of  $y_{i+1}$  do
13:     for  $\ell_{\text{neg}} \in S_i$  with negative coefficient of  $y_{i+1}$  do
14:        $\ell_{\text{new}} := \text{Combine}(\ell_{\text{pos}}, \ell_{\text{neg}}, y_{i+1})$ ;
15:       Let  $f$  be the result of applying Algorithm 3 with the inputs  $\mathcal{P}$  and  $\ell_{\text{new}}$ ;
16:       if  $f = \text{true}$  then
17:          $S_{i+1} := S_{i+1} \cup \{\ell_{\text{new}}\}$ ;
18:       end if
19:     end for
20:   end for
21:   for  $\ell \in S_i$  with zero coefficient of  $y_{i+1}$  do
22:     Let  $f$  be the result of applying Algorithm 3 with the inputs  $\mathcal{P}$  and  $\ell$ ;
23:     if  $f = \text{true}$  then
24:        $S_{i+1} := S_{i+1} \cup \{\ell\}$ ;
25:     end if
26:   end for
27:    $\mathcal{P} := \mathcal{P}|_{v_{i+1}=0}$ ;
28: end for
29: return  $(S_0 \cup S_1 \cup \dots \cup S_n)$ ;

```

Lemma 9. Let $S = \{\mathbf{x} \in \mathbb{Q}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{0}\}$ be the minimal representation of a cone $C \subseteq \mathbb{Q}^n$, where $\mathbf{A} \in \mathbb{Q}^{m \times n}$. 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 \mathbf{A} .

Proof. The cost of Algorithm 1 during the processing of the first n inequalities (Line 4) is negligible (in comparison to the subsequent computations) since it is equivalent to find the inverse of an $n \times n$ matrix. Therefore, to analyze the complexity of the DD method, we focus on the while-loop located at Line 6 of the Algorithm 1. 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 (Line 9 of Algorithm 1). Note that we have at most $(t - 1)^{\lfloor \frac{n}{2} \rfloor}$ extreme rays (Eq. (1)) whose coefficients can be bounded over by $(n - 1)^n \|\mathbf{A}\|^{2(n-1)}$ (Lemma 8) at the $t - 1$ -iteration. Hence, this step needs at most $C_1 := (t - 1)^{\lfloor \frac{n}{2} \rfloor} \times n \times \mathcal{M}(\log((n - 1)^n \|\mathbf{A}\|^{2(n-1)})) \leq O(t^{\lfloor \frac{n}{2} \rfloor} n^{2+\epsilon} h^{1+\epsilon})$

bit operations. After partitioning the vectors, the next step is to check adjacency for each pair of vectors (Line 13 of of Algorithm 1). 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 (Line 14 of Algorithm 1). 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 8) 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 iteration t of the while loop is $C := C_1 + C_2 + C_3$. The claim follows after simplifying $m \times C$.

Lemma 10 (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 2) requires at most*

$$O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon})$$

bit operations. Moreover, $\text{proj}(W; \{\mathbf{v}, v_0\})$ 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. We analyze Algorithm 2 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 can 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

$$A_0^{-1} = \begin{pmatrix} A_1^{-1} & \mathbf{0} \\ -A_2 A_1^{-1} & I_{m-n} \end{pmatrix}.$$

Also, from Sect. 2.4 we have $\|A_1^{-1}\| \leq (\sqrt{n-1} \|A_1\|)^{n-1}$. Therefore, $\|A_0^{-1}\| \leq n^{\frac{n+1}{2}} \|A\|^n$, and $\|A_0^{-1} \mathbf{c}\| \leq n^{\frac{n+3}{2}} \|A\|^n \|\mathbf{c}\| + (m-n) \|\mathbf{c}\|$. That is, $\text{height}(A_0^{-1}) \in O(n^{1+\epsilon} h)$ and $\text{height}(A_0^{-1} \mathbf{c}) \in O(m^\epsilon + n^{1+\epsilon} h)$. 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

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

– 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

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

Step 3: Projecting W and Finding the Initial Redundancy Test Cone.

Following Lemma 5, we obtain a representation of $\text{proj}(W; \{\mathbf{v}, v_0\})$ 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}, \dots, 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 the double description algorithm 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 the double description algorithm with C' as input. This operation requires at most $C_2 := O(m^{n+3}(n+1)^{\theta+\epsilon} \max(\text{height}(E, \mathbf{g}^t))^{1+\epsilon}) \leq O(m^{n+3+\epsilon}(n+1)^{\theta+\epsilon}h^{1+\epsilon})$ 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 8 and Lemma 9, 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, $\text{proj}(W^0; \{\mathbf{v}, v_0\})$ can be represented by at most $O(m^{\lfloor \frac{n+1}{2} \rfloor})$ inequalities, each with a height bound of $O(m^\epsilon n^{2+\epsilon}h)$.

Lemma 11. *Algorithm 3 runs within $O(m^{\frac{n}{2}}n^{\theta+\epsilon}h^{1+\epsilon})$ bit operations.*

Proof. 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 $\text{proj}(W^0, \{\mathbf{v}, v_0\})$. Therefore, after eliminating p variables $c_M = q + 1$, where $q = n - p$ and $d_M \leq m^{\frac{n}{2}}$. Also, we have $\text{height}(M) \in O(m^\epsilon n^{2+\epsilon} h)$. 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. The claim follows after simplification.

Using Algorithms 2 and 3, 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 4 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. The correctness of the algorithm follows from Theorem 4 and Lemma 7.

By [24, 29], 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, the overall 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 report on serial and parallel implementation of the Minimal Projected Representation (MPR) algorithm. Comparing with the `Project` command of the `PolyhedralSets` package of Maple 2017 and the famous CDD library (version 2018), we have been able to solve our test cases 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 that are randomly generated. The systems S24 and S35 are 24-simplex and 35-simplex polytopes. The systems C56 and C510 are cyclic polytopes in dimension five with six and ten vertices, The system 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 [22]. 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: one iterative following closely Algorithm 4, and the other reorganizing that algorithm by means of a divide and conquer scheme. In both implementations, we use a dense

representation for the linear inequalities. In the first approach, 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 a linked list and we can improve data locality. However, we use simple linked lists in the divide and conquer version to save time on dividing and joining lists. Although both these approaches 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 scheme while using a varying threshold for recursion as the algorithm goes on.

Columns `MPR-itr` and `MPR-rec` of the Table 1 give the running time (in milliseconds) of these implementations on a configuration with an Intel-i7-7700T CPU (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.

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.

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

Using the divide and conquer scheme, we have been able to parallelize our program, with Cilk [5]. 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 2.6 GHz), using 1, 4, 8, and 12 Cilk workers, respectively. The numbers in brackets show the speedup we gain using multi-threading.

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.

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)

7 Related Works and Concluding Remarks

As we previously discussed, removing redundant inequalities during the execution of Fourier-Motzkin Elimination is the central issue towards efficiency. Different algorithms have been developed to solve this problem. They also have been implemented in the various software libraries, including but not limited to: CDD[17], VPL[7], PPL[1], Normaliz[9], PORTA[13], and Polymake[20] In this section, we briefly review *some* of these works.

In [11], Chernikov proposed a redundancy test with little added work, which greatly improves the practical efficiency of Fourier-Motzkin Elimination. Kohler proposed a method in [29] which only uses matrix arithmetic operations to test the redundancy of inequalities. As observed by Imbert in his work [24], the

method he proposed in his 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* [18] (also known as Chernikova's algorithm [12, 30]), this method can not remove all the redundant inequalities.

In [2], Balas shows that if certain *invertibility 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 the necessity of enumerating the extreme rays of the redundancy test cone (so as to produce a minimal representation of the projection $\text{proj}(Q; \mathbf{x})$) which is time consuming. Our algorithm tests the redundancy of the inequality $\mathbf{a}\mathbf{x} \leq c$ 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 concept of subsumption cone, as defined in [23]. Consider the polyhedron Q given in Eq. (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, and β is a variable. The *subsumption cone* of Q is obtained by eliminating λ in T , that is, $\text{proj}(T; \{\alpha, \beta\})$. 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.

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)* [1]. 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.

Recently, the verified polyhedron library (VPL) [7] 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 of removing redundancies, but this cannot remove

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 $\{(3, 4)^t, (4, 5)^t, (7, 9)^t\}$.

them all, thus Linear Programming (LP) is still needed. As pointed out in [31], 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 [9]. 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.

References

1. Bagnara, R., Hill, P.M., Zaffanella, E., Bagnara, A.: The parma polyhedra library user's manual (2002)
2. Balas, E.: Projection with a minimal system of inequalities. *Comput. Optim. Appl.* **10**(2), 189–193 (1998)
3. Bastoul, C.: 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 2004, pp. 7–16. IEEE Computer Society, Washington, DC (2004). <https://doi.org/10.1109/PACT.2004.11>
4. Benabderrahmane, M.-W., Pouchet, L.-N., Cohen, A., Bastoul, C.: The polyhedral model is more widely applicable than you think. In: Gupta, R. (ed.) CC 2010. LNCS, vol. 6011, pp. 283–303. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11970-5_16
5. Blumofe, R.D., Joerg, C.F., Kuszmaul, B.C., Leiserson, C.E., Randall, K.H., Zhou, Y.: Cilk: an efficient multithreaded runtime system. *SIGPLAN Not.* **30**(8), 207–216 (1995). <https://doi.org/10.1145/209937.209958>
6. Bondhugula, U., Hartono, A., Ramanujam, J., Sadayappan, P.: A practical automatic polyhedral parallelizer and locality optimizer. *SIGPLAN Not.* **43**(6), 101–113 (2008). <https://doi.org/10.1145/1379022.1375595>
7. Boulme, S., Marechaly, A., Monniaux, D., Perin, M., Yu, H.: The verified polyhedron library: an overview, pp. 9–17 (2018)
8. Bruns, W., Ichim, B.: Normaliz: algorithms for affine monoids and rational cones. *J. Algebra* **324**(5), 1098–1113 (2010)
9. Bruns, W., Römer, T., Sieg, R., Söger, C.: Normaliz 3.0 (2008)
10. Chen, C., Covanov, S., Mansouri, F., Maza, M.M., Xie, N., Xie, Y.: The basic polynomial algebra subprograms. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 669–676. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_100
11. Chernikov, S.N.: Contraction of systems of linear inequalities. *Dokl. Akad. Nauk SSSR* **131**(3), 518–521 (1960)
12. Chernikova, N.V.: Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities. *Zhurnal Vychislitel'noi Mat. Mat. Fiziki* **5**(2), 334–337 (1965)
13. Christof, T., Löbel, A.: PORTA: polyhedron representation transformation algorithm, version 1.3. 2. Konrad-Zuse-Zentrum für Informationstechnik Berlin (2000)
14. Dantzig, G.B.: Fourier-Motzkin elimination and its dual. Technical report (1972)

15. Feautrier, P.: Dataflow analysis of array and scalar references. *Int. J. Parallel Program.* **20**, 23–53 (1991). <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1342>
16. Feautrier, P.: Automatic parallelization in the polytope model. In: Perrin, G.-R., Darte, A. (eds.) *The Data Parallel Programming Model*. LNCS, vol. 1132, pp. 79–103. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61736-1_44
17. Fukuda, K.: The CDD and CDDplus homepage. https://www.inf.ethz.ch/personal/fukudak/cdd_home/
18. Fukuda, K., Prodon, A.: Double description method revisited. In: Deza, M., Euler, R., Manoussakis, I. (eds.) *CCS 1995*. LNCS, vol. 1120, pp. 91–111. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61576-8_77
19. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*, 2nd edn. Cambridge University Press, New York (2003)
20. Gawrilow, E., Joswig, M.: Polymake: a framework for analyzing convex polytopes. In: Kalai, G., Ziegler, G.M. (eds.) *Polytopes-Combinatorics and Computation*, pp. 43–73. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-0348-8438-9_2
21. Grosser, T., Zheng, H., Aloor, R., Simbürger, A., Größlinger, A., Pouchet, L.: Polly - polyhedral optimization in LLVM. In: *First International Workshop on Polyhedral Compilation Techniques (IMPACT 2011)*, Chamonix, France, April 2011
22. Henk, M., Richter-Gebert, J., Ziegler, G.M.: 16 basic properties of convex polytopes. *Handb. Discrete Comput. Geometry* 255–382 (2004)
23. Huynh, T., Lassez, C., Lassez, J.L.: Practical issues on the projection of polyhedral sets. *Ann. Math. Artif. Intell.* **6**(4), 295–315 (1992)
24. Imbert, J.L.: Fourier’s elimination: which to choose? In: *PPCP*, pp. 117–129 (1993)
25. Jing, R.-J., Moreno Maza, M.: Computing the integer points of a polyhedron, I: algorithm. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) *CASC 2017*. LNCS, vol. 10490, pp. 225–241. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66320-3_17
26. Jing, R.-J., Moreno Maza, M.: Computing the integer points of a polyhedron, II: complexity estimates. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) *CASC 2017*. LNCS, vol. 10490, pp. 242–256. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66320-3_18
27. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC 1984, pp. 302–311. ACM, New York (1984). <https://doi.org/10.1145/800057.808695>
28. Khachiyan, L.: Fourier-Motzkin elimination method. In: Floudas, C.A., Pardalos, P.M. (eds.) *Encyclopedia of Optimization*, 2nd edn., pp. 1074–1077. Springer, Heidelberg (2009). https://doi.org/10.1007/978-0-387-74759-0_187
29. Kohler, D.A.: *Projections of convex polyhedral sets*. Technical report, California University at Berkeley, Operations Research Center (1967)
30. Le Verge, H.: A note on Chernikova’s algorithm. Ph.D. thesis, INRIA (1992)
31. Maréchal, A., Périn, M.: Efficient elimination of redundancies in polyhedra by ray-tracing. In: Bouajjani, A., Monniaux, D. (eds.) *VMCAI 2017*. LNCS, vol. 10145, pp. 367–385. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52234-0_20
32. McMullen, P.: The maximum numbers of faces of a convex polytope. *Mathematika* **17**(2), 179–184 (1970)
33. Monniaux, D.: Quantifier elimination by lazy model enumeration. In: Touili, T., Cook, B., Jackson, P. (eds.) *CAV 2010*. LNCS, vol. 6174, pp. 585–599. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_51

34. Schönhage, A., Strassen, V.: Schnelle multiplikation großer zahlen. *Computing* **7**(3–4), 281–292 (1971). <https://doi.org/10.1007/BF02242355>
35. Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley, New York (1986)
36. Storjohann, A.: *Algorithms for matrix canonical forms*. Ph.D. thesis, Swiss Federal Institute of Technology Zurich (2000)
37. Terzer, M.: *Large scale methods to enumerate extreme rays and elementary modes*. Ph.D. thesis, ETH Zurich (2009)
38. Verdoolaege, S., Carlos Juega, J., Cohen, A., Ignacio Gómez, J., Tenllado, C., Cattloor, F.: Polyhedral parallel code generation for CUDA. *TACO* **9**(4), 54 (2013)