On canonical representations of convex polyhedra

David Avis Komei Fukuda Stefano Picozzi

February 14, 2002

Abstract

Every convex polyhedron in the Euclidean space $\mathbb{R}^d$ admits both H-representation and V-representation. When working with convex polyhedra, in particular large-scale ones in high dimensions, it is useful to have a canonical representation that is minimal and unique up to some elementary operations. Such a representation allows one to compare two H-polyhedra or two V-polyhedra efficiently. In this paper, we define such representations that are simple and can be computed in polynomial time. The key ingredients are redundancy removal for linear inequality systems and affine transformations of polyhedra.

1 Introduction

A *convex polyhedron* or simply *polyhedron* in $\mathbb{R}^d$ is the set of solutions to a finite system of inequalities with real coefficients in d real variables. For a matrix $A \in \mathbb{R}^{m\times d}$ and a vector $b \in \mathbb{R}^m$, a pair (b, A) is said to be an *H-representation* of a convex polyhedron P if $P = \{x \in \mathbb{R}^d \mid b + Ax \geq 0\}$. Motzkin's decomposition theorem (see, e.g. [3, 4]) states that every polyhedron has another representation called a V-representation. For matrices $V \in \mathbb{R}^{p\times d}$ and $R \in \mathbb{R}^{q\times d}$, a pair (V, R) is said to be a *V-representation* of a polyhedron P if $P = conv(V) + cone(R)$, where $conv(M)$ ($cone(M)$, respectively) denotes the convex hull (the nonnegative hull) of the row vectors of the matrix M. In each of representations, there are trivial transformations that preserve the represented polyhedron. For an H-representation, each inequality can be multiplied with any positive numbers. For a V-representation, each row of R can be scaled by any positive number. Also, any permutation of inequalities and the row vectors within V and R does not modify the polyhedron. Two representations are considered *equal* if they are the same up to these transformations.

One can easily see that neither V- nor H-representations are unique for a given polyhedron. First of all both V- and H-representations can have an unlimited number of redundant vectors or inequalities. Furthermore, even without redundancy, representations are not unique. Namely there are infinite many nonredundant V-representations of polyhedra without extreme points, and infinite many nonredundant H-representation of polyhedra that are not full-dimensional.

The main objective of the present paper is to define a unique V-representation (H-representation, respectively) of a polyhedron that can be computed in polynomial time from any V-representation (H-representation). We shall review the unique polyhedral representations discussed in the book of Schrijver [3], and propose a variation that is computationally simpler and guaranteed to have certain sparsity property. The key ingredients are efficient redundancy removal by linear programming and certain affine mappings of convex polyhedra to an appropriate space. As a consequence, it is possible to check in polynomial time whether two H-representations (or two V-representations) define the same polyhedron or not. We also show that redundancy removal is polynomially equivalent to the linear programming. This justifies the use of linear programming algorithms for redundancy removal.

Our main motivation arises from the continuing effort by the first two authors to develop polyhedral computation codes, such as cddlib [2] and lrslib [1], that perform basic transformations of polyhedral representations. While the main transformation of these codes is the conversion between H- and V-representations, it is increasing important to make an efficient implementation of algorithms to compute canonical representations. This would allow users to compare two H-polyhedra (or two V-polyhedra) quickly without applying the (often too) expensive conversion. In addition, the computation of a canonical representation can be considered as a preprocessing step for the main transformation, which in some cases results in a drastic speed up.

Due to space limitations, many of the proofs are omitted, and will be included in the full version of the paper.

2 Representations of Convex Polyhedra

In this section, we present basic results on representations of convex polyhedra. Most of the results are classical and not very hard to prove. However there is no thorough treatment of this subject for both V- and H-representations and in particular no study with computational considerations. Throughout this section, P is a non-empty polyhedron in R^d.

The notation for H-representation and V-representation given in the last section, although standard, hides the underlying duality of polyhedral cones. A natural way to unify the two representations is through homogenization. We encode a V-representation (V, R) as a pair of a vector and a matrix $\left(\begin{bmatrix} e \\ 0 \end{bmatrix}, \begin{bmatrix} V \\ R \end{bmatrix}\right)$, where e is the column vector of all 1's. In this representation written as (b, A), each row index i with $b_i = 1$ indicates that the corresponding ith row vector A_i of A is a convex hull generator, and each $b_i = 0$ indicates A_i is a cone generator. This way, both representations are of the same form. Actually we must go even further to define a unique representation of any given polyhedron, in both V and H formats.

Let A be an $m \times d$ matrix, b an m-dimensional vector and $\{I, L\}$ be a partition of $[m] := \{1, \ldots, m\}$. We define a quadruple (b, A, I, L) to be an H-representation of the polyhedron

$$P_H = \{x \in \mathbb{R}^d \mid b_I + A_I x \geq \mathbf{0},\ b_L + A_L x = \mathbf{0}\}. \tag{2.1}$$

We define a quadruple (b, A, I, L) to be a V-representation of the polyhedron

$$P_V = \{x \in \mathbb{R}^d \mid x = A^T y,\ y_I \geq \mathbf{0}\}, \tag{2.2}$$

if $b = \mathbf{0}$, and a V-representation of the polyhedron

$$P_V = \{x \in \mathbb{R}^d \mid x = A^T y,\ y_I \geq \mathbf{0},\ b^T y = 1\}, \tag{2.3}$$

otherwise.

The set L is called the *linearity* of the representation. Two H-representations (or two V-representations) (b, A, I, L) and (b', A', I', L') are said to be *equivalent* if the represented polyhedra are equal. Each row index $i \in [m]$ is called *redundant in a representation* (b, A, I, L) if the represented polyhedron stays unchanged after deleting the i^{th} data, namely removing b_i, A_i, i from I or L whichever contains i, and shifting the rest of the data accordingly. A representation is *nonredundant* if it has no redundant row index. For technical reasons, if the zero vector appears in a V-representation it will be considered non redundant.

A V-representation (b, A, I, L) is called *standard* if b is binary (i.e. $b_i \in \{0, 1\}$) and $b_L = \mathbf{0}$. A V-representation can be transformed to an equivalent standard V-representation in quadratic time. For this reason, we assume for the rest of the paper that any V-representation is standard. Each

vector A_i in a representation is called a *generator*. It is a *point generator* if $b_i = 1$, a *ray generator* if $b_i = 0$ and $i \notin L$, and a *line generator* if $b_i = 0$ and $i \in L$.

We say two representations (b, A, I, L) and (b', A', I', L') *equal* if $b_L + A_L x = 0 \Leftrightarrow b_{L'} + A_{L'} x = 0$ and if there is a permutation π of I such that $\pi(I) = I'$ and each (b'_i, A'_i) is a positive multiple of $(b_{\pi(i)}, A_{\pi(i)})$ for any $i \in I$. Clearly, if two representations of the same type are equal, they are equivalent and thus define the same polyhedron. In particular for V-representation, $b_L = 0$ and then the first equivalence coincides with the statement $\{x \in \mathbb{R}^d \mid x = (A_L)^T y\} = \{x \in \mathbb{R}^d \mid x = (A_{L'})^T y\}$.

While the equivalence of two representations is nontrivial to check, the equality is easy to check using Gaussian elimination. Our main goal is to discuss an efficient procedure to reduce any representation to a uniquely defined equivalent representation so that the equivalence of two representations of the same type can be checked quickly.

In particular, we want to extend the definition of the canonical representation discussed in the book of Schrijver [3] in order to get a canonical representation guaranteeing a certain sparsity property. Extending this definition uses the notion of compatible and complementary linear subspaces of $\mathbb{R}^d$. Definitions and basic properties of these subspaces are described in the next section.

2.1 Compatible and Complementary linear subspaces

We say two linear subspaces S_1 and S_2 of $\mathbb{R}^d$ having same dimension are *compatible* if the orthogonal projection of S_1 onto S_2 is S_2 itself, or, equivalently, if the orthogonal projection of S_2 onto S_1 is S_1 itself. We say two linear subspaces S_1 and S_2 of $\mathbb{R}^d$ are *complementary* if any basis of S_1 and any basis of S_2 form, together, a basis of $\mathbb{R}^d$. It follows from elementary linear algebra that S_1 and S_2 are complementary if and only if $\dim(S_1) + \dim(S_2) = d$ and $S_1 \cap S_2 = \{\mathbf{0}\}$. Similarly any $r \in \mathbb{R}^d$ can be written uniquely as $r = r_1 + r_2$, where $r_1 \in S_1$ and $r_2 \in S_2$ For any two complementary linear subspaces S_1 and S_2 of $\mathbb{R}^d$, we let $\mathcal{P}_{S_1 S_2}$ denote the *projector onto* S_1 *along* S_2 defined as

$$\begin{aligned} \mathcal{P}_{S_1 S_2} : \quad \mathbb{R}^d &\longrightarrow S_1 \\ r &\longmapsto \mathcal{P}_{S_1 S_2}(r) = r_1, \end{aligned}$$

and r_1 is called the *projection of* r *onto* S_1 *along* S_2. It is easy to see that $\mathcal{P}_{S_1 S_2}$ is a projector in the sense of the linear algebra. By linearity, this definition can be extended to any subset of $\mathbb{R}^d$. For any set $W \subseteq \mathbb{R}^d$, and for any $i = 1, 2$ and $j = 1, 2$, $i \neq j$, we call the set $W_i = \mathcal{P}_{S_i S_j}(W)$ the *projection of* W *onto* S_i *along* S_j. We would point out that the relation $W = W_1 + W_2$ does not hold in general.

2.2 Canonical V-Representation

To define a unique V-representation of a general (non-pointed) polyhedron we need a few more definitions. The *linearity space* $lin.space(P)$ of P is defined as

$$lin.space(P) = \{z \in \mathbb{R}^d \mid x + \alpha z \in P \text{ for all } x \in P \text{ and } \alpha \in \mathbb{R}\}. \tag{2.4}$$

The *characteristic cone* $char.cone(P)$ of P is

$$char.cone(P) = \{z \in \mathbb{R}^d \mid x + \alpha z \in P \text{ for all } x \in P \text{ and } \alpha \geq 0\}. \tag{2.5}$$

Using these definitions, the V-representation discussed in the book of Shrijver may be summarized as the following lemma :

Lemma 2.1. *A nonredundant V-representation* (b, A, I, L) *of polyhedron* P *satisfying the following exists and is unique:*

(1) $lin.space(P) = \{z \in \mathbb{R}^d \mid z = (A_L)^T y_L\}$, *and*

(2) all row vectors A_i $(i \in I)$ *are orthogonal to* $lin.space(P)$.

This lemma gives a theoretically satisfactory definition of the canonical representation. The weakness of this definition is that asking the generators to be orthogonal to the linearity space tends to make the resulting representation dense. The next theorem gives more general criteria which allows one to look for unique V-representations satisfying certain sparsity properties.

Theorem 2.2. *A nonredundant V-representation* (b, A, I, L) *of polyhedron* P *satisfying the following exists and is unique:*

(1) $lin.space(P) = \{z \in \mathbb{R}^d \mid z = (A_L)^T y_L\}$, *and*

(2) all row vectors A_i $(i \in I)$ *are orthogonal to* S,

where S *is any fixed subspace of* $\mathbb{R}^d$ *compatible with* $lin.space(P)$.

The V-representation discussed in the book of Shrijver is the special case of Theorem 2.2 when S is chosen to be $lin.space(P)$ itself.

2.3 Canonical H-Representation

In this section, we use the notion of compatible spaces to find a general canonical H-representation. We denote by $aff(P)$ the *affine hull* of P.

Here is the H-representation discussed in the book of Shrijver:

Lemma 2.3. *A nonredundant H-representation* (b, A, I, L) *of a nonempty polyhedron satisfying the following exists and is unique:*

(1) $aff(P) = \{x \in \mathbb{R}^d \mid b_L + A_L x = \mathbf{0}\}$, *and*

(2) each row of A_I *is the linear space parallel to* $aff(P)$.

Again this representation might be computationally inconvenient. The next theorem gives more flexible notion of canonical H-representations which allows one to look for those satisfying additional favorable properties such as sparsity.

Theorem 2.4. *A nonredundant H-representation* (b, A, I, L) *of polyhedron* P *satisfying the following exists and is unique:*

(1) $aff(P) = \{x \in \mathbb{R}^d \mid b_L + A_L x = \mathbf{0}\}$, *and*

(2) each row of A_I *is in* S,

where S *is any linear subspace compatible with the linear space parallel to* $aff(P)$.

3 Computing Canonical Representations

Given any V- (resp. H-) representation (b, A, I, L) of a polyhedron P in $\mathbb{R}^d$, our goal is to compute efficiently the unique equivalent representation (b', A', I', L') satisfying the following :

(1) it is nonredundant;

(2) $lin.space(P) = \{x \in \mathbb{R}^d \mid x = (A'_{L'})^T y\}$ (resp. of $aff(P) = \{x \in \mathbb{R}^d \mid x = b'_{L'} + A'_{L'} x = 0\}$);

(3) the rows of $A'_{I'}$ are orthogonal to a subspace S compatible with *lin.space*(P) (resp. are contained in a subspace S compatible with $\mathcal{P}_0$).

One has the freedom to choose S as any linear subspace compatible with *lin.space*(P) or with *aff*(P), accordingly to the considered format. Condition (3) can be expressed in the same form for both V- and H- formats letting $\bar{P}_0$ denote *lin.space*(P) in V-format (and *aff*(P) in H-format), and letting $\bar{S}$ denote $S^\perp$ in V-format (and S in H-format). Condition (3) is equivalent to

(3') the rows of $A'_{I'}$ are the projections onto $\bar{S}$ along $\bar{P}_0$ of any equivalent set of generators of P.

We now discuss two special choices for the linear subspace S, which lead to computationally interesting representations of polyhedron P.

Any linear subspace is compatible with itself. Choosing S as *lin.space*(P) itself (resp. as *aff*(P) itself) leads to the *orthogonal representation* discussed in the book of Shrijver [3]. In this case, the projector $\mathcal{P}_{\bar{S}\bar{P}_0}$ onto $\bar{S}$ along $\bar{P}_0$ coincides with the orthogonal projector $\mathcal{P}\bar{S}$ onto $\bar{S}$. This representation is perhaps mathematically the most natural but it tends to lead to a very dense representation, as you will see in Section 6

Let us consider another natural choice for S. A *coordinate subspace* is any vector subspace of R^d generated by some unit vectors e^j $(j = 1, \ldots, d)$. Selecting $\bar{S}$ as a coordinate subspace guarantees a certain sparsity of the projected vectors. Moreover, selecting $\bar{S}$ as the lexicographically smallest subspace complementary with with $(\bar{P}_0)$ guarantees that the nonzero coordinates are indexed by indices as small as possible. The resulting representation will be called the *lexicographically smallest coordinate subspace* or *lexico-smallest* representation.

Computationally, the orthogonal representation requires the computation of orthogonal basis of a linear subspace, which amounts to doing the Gram-Schmidt orthogonalization procedure. The lexico-smallest representation needs only to check whether a unit e^j is linearly independent of a given set of chosen vectors, which amounts to applying Gaussian eliminations.

4 Redundancy Removal

In this section, we show how redundancies may be removed efficiently from any given representation (b, A, I, L) of a nonempty polyhedron P. By definition, any row index $k \in [m] = I \cup L$ is redundant in (b, A, I, L) if and only if the represented polyhedron stays unchanged after deleting the k^{th} row.

4.1 Redundancy Removal in V-representation

For any given V-representation (b, A, I, L) of a polyhedron P, we let $I_0 = \{i \in I \mid b_i = 0\}$ and $I_1 = \{i \in I \mid b_i = 1\}$. It follows from the definitions, that $P = \mathit{conv}(A_{I_1}) + \mathit{cone}(R)$, where $R := \begin{bmatrix} A_{I_0} \\ A_L^{\pm} \end{bmatrix}$ and $A_L^{\pm} := \begin{bmatrix} +A_L \\ -A_L \end{bmatrix}$. And we denote by I_R the set of the row indices of matrix R

For any $k \in I_1$, we say that *the row vector A_k is redundant in* $\mathit{conv}(A_{I_1})$ if $\mathit{conv}(A_{I_1}) = \mathit{conv}(A_{I_1} \setminus \{A_k\})$. Similarly, for any $k \in I_0 \cup L$, we say that *the row vector R_k is redundant in* $\mathit{cone}(R)$ if $\mathit{cone}(R) = \mathit{cone}(R \setminus \{R_k\})$.

Let (b, A, I, L) be any V-representation of polyhedron P. The following characterization of the redundancies (b, A, I, L) directly follows from the decomposition of P as $P = \mathit{conv}(A_{I_1}) + \mathit{cone}(R)$, and from the definition of point, ray and line generators of P.

(1) $k \in I_0$ is redundant in (b, A, I, L) if and only if A_k is redundant in $\mathit{cone}(R)$;

(2) $k \in L$ is redundant in (b, A, I, L) if and only if A_k and $-A_k$ are redundant in $\mathit{cone}(R)$.

(3) $k \in I_1$ is redundant in (b, A, I, L) if and only if A_k is redundant in $conv(V) + cone(R)$.

And, for any row index $k \in I \cup L$, we define the linear program R.LPV(b_k, A_k) as

$$\begin{array}{lllll} \max & 0 & & & \\ \text{s.t.} & (A_k)^T & = & b_k \sum_{i \neq k} (A_i)^T x_i + \sum_{l \neq k} (R_l)^T y_l & i \in I_1, l \in I_R \\ & b_k & = & b_k \sum_{i \neq k} x_i, & i \in I_1 \\ & x_i \geq 0, & & y_l \geq 0 & \forall i \in I,\ \forall l \in I_R \end{array}$$

where I_R is the index set or the rows of matrix R. This linear program is either infeasible or it has optimal solution zero.

Theorem 4.1. *Let (b, A, I, L) be any V-representation of polyhedron P. Then,*

(1) any row index $k \in I$ is redundant in (b, A, I, L) if and only if R.LPV(b_k, A_k) has optimal value zero, and

(2) any row index $k \in L$ is redundant in (b, A, I, L) if and only if both R.LPV$(0, A_k)$ and R.LPV$(0, -A_k)$ have optimal value zero.

Removing sequentially the redundancies from any representation (b, A, I, L) of a polyhedron P using this theorem leads to a nonredundant representation (b', A', I', L') in $\mathcal{O}(m \times \text{LP}(m, d))$ time, where $m = |I \cup L|$.

4.2 Redundancy Removal in H-representation

By definition, any row index $k \in I$ (resp. $k \in L$) is redundant in an H-representation (b, A, I, L) of polyhedron $P = \{x \in \mathbb{R}^d \mid b_I + A_I x \geq \mathbf{0},\ A_L x + b_L = \mathbf{0}\}$ if and only if $P = \{x \in \mathbb{R}^d \mid b_{\bar{I}} + A_{\bar{I}} x \geq \mathbf{0},\ A_{\bar{L}} x + b_{\bar{L}} = \mathbf{0}\}$, where $\bar{I} = I \setminus \{h\}$ and $\bar{L} = L$ (resp. $\bar{I} = I$ and $\bar{L} = L \setminus \{h\}$). In other words, $k \in I$ (resp. $k \in L$) is redundant in (b, A, I, L) if and only if the inequality $\{b_k + A_k x \geq 0\}$ (resp. $\{b_k + A_k x = 0\}$) is redundant in the system of inequalities defining P. For any row A_k of A and for any component b_k of b, we let R.LPH(b_k, A_k) denote the linear program defined as

$$\begin{array}{lll} \min & b_k + A_k x & \\ \text{s.t.} & b_i + A_i x \geq 0 & \forall i \in I,\ i \neq k \\ & b_l + A_l x = 0 & \forall l \in I,\ i \neq k \\ & 0 \geq b_k + A_k x \geq -1 & \end{array}$$

Theorem 4.2. *Let (b, A, I, L) be any H-representation of a polyhedron P. Then,*

(1) any row index $k \in I$ is redundant in (b, A, I, L) if and only if R.LPH(b_k, A_k) has optimal value zero, and

(2) any row index $k \in L$ is redundant in (b, A, I, L) if and only both R.LPH(b_k, A_k) and R.LPH$(-b_k, -A_k)$ have optimal value zero.

4.3 Computing *lin.space*(P) and *aff*(P)

The canonical V- (resp. H-) representation of polyhedron P is defined so that the rows of the matrix A_L form a basis of $S(A_L)$. In this section, we present how one can decide efficiently whether any given row index $k \in I \cup L$ from any nonredundant representation (b, A, I, L) of P belongs to the linearity of the canonical representation of P. Solving this decision problem for every $k \in I \cup L$ leads to a set of indices, say L'', such as $(A_{L''})^T y_L \in lin.space(P)$ (resp. such as $b_{L''} + A_{L''} x = \mathbf{0}$). The linearity L' of the canonical representation of P then arises from L'' by removing the redundancies.

4.3.1 Computing *lin.space*(P) in V-representation

Given any V-representation (b, A, I, L) of polyhedron P, we let $I_0 := \{i \in I \mid b_i = 0\}$. For any row index $k \in L \cup I$, we let L.LPV(A_k) denote the linear program defined as

$$\begin{array}{lllll} \max & 0 & & & \\ \text{s.t.} & (A_k)^T & = & -\sum_{l \neq k} (R_l)^T y_l, & l \in I_R \\ & x_i \geq 0, & & y_l \geq 0 & \forall i \in I,\ \forall l \in I_R \end{array}$$

where the matrix R is the matrix defined in Section 4.1, and I_R is the index set of the rows of R.

Theorem 4.3. *Let (b, A, I, L) be any V-representation of polyhedron P. Then, any row A_k of matrix A belongs to lin.space(P) if and only if $k \in L \cup L_I$, where $L_I = \{i \in I_0 \mid$ L.LPV(A_i) has optimal value $0\}$.*

We would point out that, in general, using this theorem to repartition the indices of any given representation tends to introduce new redundancies in the linearity of the representation, which one has to remove by applying redundancy removal. Nevertheless, it is possible to directly compute a nonredundant set using the following instead of Theorem 4.3.

Theorem 4.4. *Let (b, A, I, L) be any V-representation of polyhedron $P = conv(A_{I_1}) + cone(R)$, where I_0 and R are defined as in Section 4.2. Then, any row A_k of matrix A belongs to lin.space(P) if and only if $k \in L \cup L_I$, where $L_I = \{i \in I_0 \mid A_k$ is redundant in $cone(-R)\}$.*

4.3.2 Computing *aff*(P) in H-representation

Any canonical H-representation (b, A, I, L) of a nonempty polyhedron P has to satisfy $aff(P) = \{x \in \mathbb{R}^d \mid b_L + A_L x = \mathbf{0}\}$, and the rows of $b_L + A_L x \geq 0$ form a nonredundant set of inequalities. Let (b, A, I, L) be any V-representation of P.

For any row index $k \in I \cup L$, I and L from any fixed V-representation of P, we let L.LPH(b_k, A_k) denote the linear program defined as

$$\begin{array}{ll} \max\limits_{x \in P} & b_i + A_i x \\ \text{s.t.} & b_i + A_i x \leq 1. \end{array}$$

Theorem 4.5. *Let (b, A, I, L) be any H-representation of polyhedron P. Then, any row A_k of matrix A is such as $\{x \in \mathbb{R}^d \mid b_k + A_k x = 0\} \supseteq aff(P)$ if and only if $k \in L \cup L_I$, where $L_I = \{i \in I \mid$ L.LPH(b_k, A_k) has optimal value $0\}$.*

As was the case for V-representations, the set L' obtained using this theorem may contain redundancies. A nonredundant set may be obtained by applying redundancy removal on L' or one can directly compute a nonredundant set L' using the following theorem.

Theorem 4.6. *Let (b, A, I, L) be any H-representation of polyhedron P. Then, any row A_k of matrix A is such as $\{x \in \mathbb{R}^d \mid b_k + A_k x = 0\} \supseteq aff(P)$ if and only if $k \in L \cup L_I$, where $L_I = \{i \in I \mid -b_i - A_i x \geq 0$ is redundant with $\{b_I + A_I x \geq 0\} \cup \{b_L + A_L x = 0\}\}$.*

5 Computational Equivalence of Problems

Given any representation (b, A, I, L) of a nonempty polyhedron P, the results of last Section show how a canonical representation (b', A', I', L') of P may be computed efficiently. The three basic operations to perform are :1) redundancy removal, 2) computation of the linearity L' of the representation 3) computation of matrix $A'_{I'}$ using the compatible space S.

Theorem 5.1. *Redundancy removal and the computation of a canonical representation are polynomially equivalent problems.*

Proof. It is clear that one may remove the redundancies from any given representation of a polyhedron P by computing the canonical representation of P. On the other hand, as stated in Section 3, *lin.space*(P) (resp. *aff*(P)) may be solved by removing redundancies. It follows that, when computing a canonical representation, the unique operation that does not consist in removing redundancies is the computation of the rows of A_I, which we prove to be a polynomial time operation. Thus, one may compute efficiently a canonical representation just by removing redundancies. □

As we saw in Section 4, redundancy removal has complexity $\mathcal{O}(m \times \text{LP}(m, d))$, where $m = |I \cup L|$. Solving these linear programs is expensive in time and the natural question that arises from this result concerns the necessity of solving linear programs to remove redundancies. In other words, are linear programming and redundancy removal (and therefore, the computation of a canonical representation) polynomially equivalent problems? In the following we prove that this is the case;in particular, we prove that the redundancy removal problem is as hard as solving an LP, which justifies the method used in Section 4 for redundancy removal.

It is well known that checking feasibility of a linear system is polynomially equivalent to linear programming. In order to prove that redundancy removal is as hard as the linear programming, we may prove that it is as hard as checking the feasibility of a given system of inequalities, say $b + Ax \geq \mathbf{0}$. This may be easily proved considering the homogenized (feasible) system $bx_0 + Ax \geq \mathbf{0}$, $x_0 \in \mathbb{R}$.

Lemma 5.2. *Let $Ax + b \geq \mathbf{0}$, where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^d$, be any linear system of inequalities. Then, $Ax+b \geq \mathbf{0}$ is feasible if and only if the inequality $\{x_0 \leq 1\}$ is not redundant with $Ax+bx_0 \geq \mathbf{0}$.*

Proof. Consider the homogenized system $Ax+bx_0 \geq \mathbf{0}$. Clearly, $Ax+b \geq \mathbf{0}$ is feasible if and only if it exist a vector $\bar{x} \in \mathbb{R}^d$ such that the pair $(x_0 = 1; x = \bar{x})$ forms a feasible solution of $bx_0 + Ax \geq \mathbf{0}$. And this last condition is satisfied if and only if $(x = \alpha\bar{x}; x_0 = \alpha)$ forms a feasible solution of $bx_0 + Ax \geq \mathbf{0}$ for any fixed nonnegative real α. In particular, $b + Ax \geq \mathbf{0}$ is feasible if and only if it exist a vector $x \in \mathbb{R}^d$ such that the pair $(x; x_0 > 1)$ is a feasible solution of $bx_0 + Ax \geq \mathbf{0}$. On the other hand, the system $bx_0 + Ax \geq \mathbf{0}$, $x_0 \leq 1$ is feasible. Then, one may apply the method of Section 4.2 to check whether the inequality $\{x_0 \leq 1\}$ is redundant with $bx_0 + Ax \geq \mathbf{0}$ or not. In particular regarding to the linear program R.LPH$((0, A_k))$ (see Section 4.2), it is not hard to see that the system $\begin{cases} bx_0 + Ax \geq 0 \\ x_0 > 1 \end{cases}$ is feasible if and only if $\{x_0 \leq 1\}$ is redundant with $bx_0 + Ax \geq \mathbf{0}$. □

It follows from the above results that we may check the feasibility of any linear system $Ax \leq b$ in polynomial time by performing redundancy removal. As a direct consequence we has the argued result that,

Theorem 5.3. *Redundancy removal, the computation of a canonical representation and linear programming are polynomially equivalent problems.*

6 Some Examples

We start by illustrating the difference between the two canonical representations described in Section 6. Consider the following (b, A, I, L) V-representation:

$$b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}, I = \{1, 2, 3\}, L = \emptyset.$$

(This corresponds to the three hamiltonian circuits for the complete graph on four vertices.) An orthogonal H-representation in this case is given by:

$$b = \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \\ 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ -1/3 & 1/6 & 1/6 & 1/6 & 1/6 & -1/3 \\ 1/6 & -1/3 & 1/6 & 1/6 & -1/3 & 1/6 \\ 1/6 & 1/6 & -1/3 & -1/3 & 1/6 & 1/6 \end{bmatrix}, I = \{5, 6, 7\}, L = \{1, 2, 3, 4\}.$$

Note that apart from the linearity space, the matrix is completely dense. An H-representation with lexicographically smallest coordinate subspace is given by:

$$b = \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \\ 1 \\ 1 \\ -1 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, I = \{5, 6, 7\}, L = \{1, 2, 3, 4\}.$$

This polytope is two dimensional, and so the second representation is quite sparse and contains a square 3 by 4 submatrix of zeroes in the rows indexed by I. (These inequalities have the natural interpretation $x_{12} \leq 1, x_{13} \leq 1, x_{12} + x_{13} \geq 1$ for the travelling salesman problem).

Redundancy removal can have can have a significant effect in speeding up the running time of vertex enumeration algorithms. This is particularly pronounced on algorithms which enumerate bases, such as pivot based algorithms, and other algorithms that use symbolic perturbation to resolve degeneracy. However improvement is also noticable for double description algorithms. Here we give a few examples, using lrs [1] as an example of a pivot based method, and cdd [2] as an example of a double description based method.

Consider the metric cone which for any integer $n \geq 3$ is a polyhedron in $R^{n(n-1)/2}$ defined by the triangle inequalities $x_{ij} - x_{ik} - x_{jk} \geq 0$ for all distinct $1 \leq i, j, k \leq n$, where $x = (x_{ij})$ $1 \leq i < j \leq n$. The metric cone is extremely degenerate. Vectors x satsfying these inequalities are known as semimetrics: they are nonnegative and satisfy all triangle inequalities. Often the nonnegativity condition $x_{ij} \geq 0$ is explicitly specified although these inequalities are in fact redundant. The program lrs has a nonnegative option for specifying these additional inequalities, but using it here causes a big increase in the running time. For example, with $n = 6$ and without the redundant constraints, lrs generates 203,956 bases to find the 296 extreme rays. With the nonnegative option, it generates 1,960,411 bases and so the running time is nearly 10 times longer. The effect on cdd is not so pronounced but still noticeable: including the redundant inequalities nearly doubles the computation time.

Another way redundancy occurs is when the programs are used to generate rays lying on lower dimensional faces. This is easily performed in both lrs and cdd by including a linearity option which specifies that certain inequalities should be treated as equations. This normally results in some of the original inequalities becoming redundant. Taking one such linearity causes all rays on a facet to be generated. For the metric cone with $n = 6$, a facet contains 113 extreme rays. Without removing redundant inequalities, lrs generates 121,215 bases. There are 14 redundant inequalities, and after there removal lrs generates 38,119 bases, a speed up of a factor of about 4. Choosing two linearities causes ray enumeration on a ridge. Here the effect of redundancy removal is much more pronounced, as 30 of the original constraints are redundant. lrs runs more than 50 times faster

after redundancy removal. The effect of redundancy removal on cdd is also noticeable, with speed ups of the order of about 3 1/2 and 6 times respectively.

References

[1] D. Avis. *lrs Homepage*, 2001. School of Computer Science, McGill University, Canada. http://cgm.cs.mcgill.ca/~avis/C/lrs.html.

[2] K. Fukuda. *cddlib reference manual, cddlib Version 092a.* Swiss Federal Institute of Technology, Switzerland, 2001. http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html.

[3] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, 1986.

[4] G.M. Ziegler. *Lectures on polytopes.* Graduate Texts in Mathematics 152. Springer-Verlag, 1994.