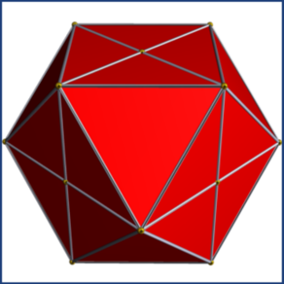
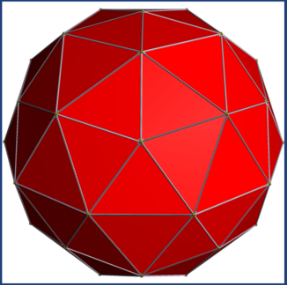
ETH zürich





Convex Hull
Representation Conversion: cddlib and lrslib

Niklas Pfister

Contents

1 Introduction
- Polytopes
- Minkowski-Weyl Theorem
- V- and H-Representation

2 Theoretical Algorithms
- Representation Conversion Problem
- Incremental Algorithm
- Pivoting Algorithm

3 Implementations cddlib and lrslib
- Input
- Implementation Details (cddlib)
- Examples
- GNU GMP
- Additional Functionality

4 References

Polytopes

Definition (convex polytope/polyhedron)

Let $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$, then a *convex polyhedron* P corresponding to the pair (A, b) is defined as

$$P := \{x \in \mathbb{R}^d : Ax \leq b\}.$$

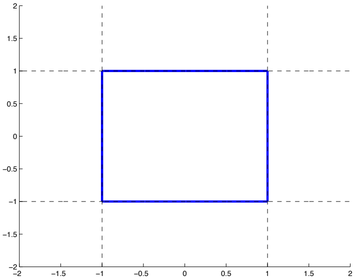
We call P a *convex polytope* if P is bounded.

Polytopes

Example

The square S given by $S = \{x \in \mathbb{R}^2 : -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}$ can be represented using the following (A, b) pair:

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}, \; b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$



Minkowski-Weyl Theorem

Theorem (Minkowski-Weyl)

For a subset P of $\mathbb{R}^d$, the following statements are equivalent:

(i) P is a Polyhedron, i.e., there exist $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ such that $P := \{x \in \mathbb{R}^d : Ax \leq b\}$.

(ii) P is finitely generated, i.e., there exist finitely many vectors v_i's and r_j's such that $P = conv(\{v_1, \ldots, v_s\}) + cone(\{r_1, \ldots, r_t\})$.

Minkowski-Weyl Theorem

Remarks

- *conv* denotes the convex hull, i.e.

$$conv(\{v_1, \ldots, v_s\}) := \{x : x = \sum_i a_i v_i,\ \|a\|_1 = 1 \text{ and } a \geq \mathbf{0}\}$$

- *cone* denotes the nonnegative hull, i.e.

$$cone(\{r_1, \ldots, r_t\}) := \{x : x = \sum_i \lambda_i r_i,\ \lambda_i \geq 0\}$$

- (ii) can be equivalently written in matrix form:
  P is finitely generated, i.e., there exists $V \in \mathbb{R}^{d \times s}$ and $R \in \mathbb{R}^{d \times t}$ for some s and t such that
  $P := \{x : x = V\mu + R\lambda,\ \mu \geq \mathbf{0}, \|\mu\|_1 = 1,\ \lambda \geq \mathbf{0}\}$.

V- and H-Representation

The Minkowski-Weyl theorem suggests the following definition:

Definition (H- and V-Representation)

- A polyhedron has H-Representation if it is given in the form (i) (Halfspace-representation)
- A polyhedron has V-Representation if it is given in the form (ii) (Vertex-representation)

V- and H-Representation

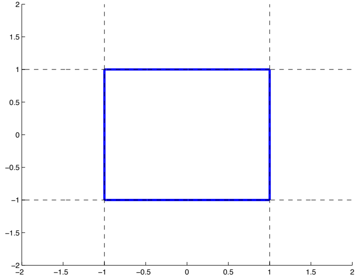
Example (continued)

- S is given in H-representation by

$$S = \{x \in \mathbb{R}^2 : Ax \leq b\}.$$

- S is given in V-representation by

$$S = conv(\{(1,1),(1,-1),(-1,-1),(1,-1)\}).$$



V- and H-Representation

- **Question** How can we convert between the two representations of a Polyhedron?
  - This problem of computing a (minimal) V-representation from an H-representation or vice versa is known as the *representation conversion problem* for polyhedra.
  - It can be shown using the concept of dual polytopes that switching from the V-representation to the H-representation of a Polyhedron is essentially the same as switching from the V-representation to the H-representation.
  - Therefore an algorithm describing one direction of the conversion will also give the other direction.

Contents

1 Introduction
- Polytopes
- Minkowski-Weyl Theorem
- V- and H-Representation

2 Theoretical Algorithms
- Representation Conversion Problem
- Incremental Algorithm
- Pivoting Algorithm

3 Implementations cddlib and lrslib
- Input
- Implementation Details (cddlib)
- Examples
- GNU GMP
- Additional Functionality

4 References

Representation Conversion Problem

An important property of the representation conversion problem is that the size of the output is not easy to measure given the size of the input. For example the d-dimensional cube has $2d$ facets and 2^d vertices. Consequently a good algorithm should

- be sensitive to the *output size*. Ideally bounded by a polynomial function of both output and input size.
- be light on the *memory usage*. Ideally the required memory is bounded by a polynomial of the input size.

For the general representation problem no such algorithm exist. However restricted to special classes of polyhedra this is possible.

Incremental Algorithm

Double Description Method

Input: Matrix $A \in \mathbb{R}^{m \times d}$.
Output: Matrix $R \in \mathbb{R}^{d \times n}$ such that (A, R) are a DD pair, i.e. the columns of R generate $C(A) := \{x : Ax \leq 0\}$.

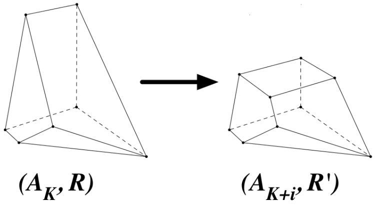
General Step

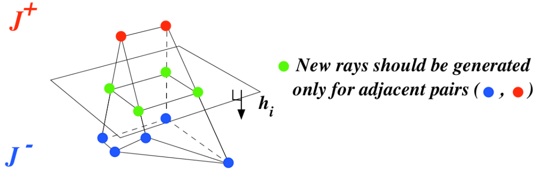
Let K subset of row indices $\{1, 2, \ldots, m\}$ of A and let A_k the matrix of the rows of A indexed by K. Assume (A_K, R) is a DD pair.

- If $A = A_K$ we are done.
- Else take i not in K and construct the DD pair (A_{K+i}, R').

Incremental Algorithm

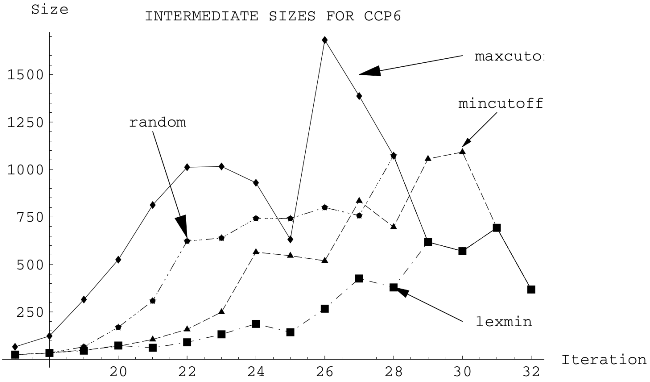
How can R' be constructed?



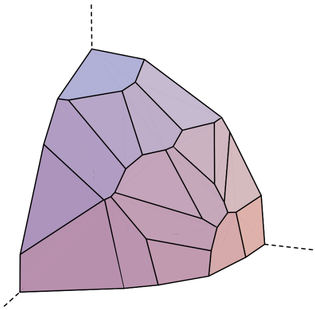


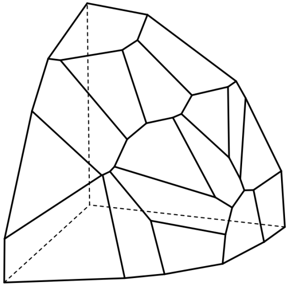
Incremental Algorithm

In what order should the rows be chosen?



Pivoting Algorithm





A polytope P and its graph (1-skeleton)

Contents

1 Introduction
- Polytopes
- Minkowski-Weyl Theorem
- V- and H-Representation

2 Theoretical Algorithms
- Representation Conversion Problem
- Incremental Algorithm
- Pivoting Algorithm

3 Implementations cddlib and lrslib
- Input
- Implementation Details (cddlib)
- Examples
- GNU GMP
- Additional Functionality

4 References

Input

Inputs for cddlib and lrslib are given in the following form for H-representations and V-representations respectively.

various comments
H-representation
(**linearity** t i_1 i_2 $\ldots$ i_t)
begin
m $d+1$ numbertype
b $-A$
end
various options

various comments
V-representation
(**linearity** t i_1 i_2 $\ldots$ i_t)
begin
$n+s$ $d+1$ numbertype
1 v_1
$\vdots$ $\vdots$
1 v_n
0 r_{n+1}
$\vdots$ $\vdots$
0 r_{n+s}
end
various options

Implementation Details

```
typedef struct dd_matrixdata *dd_MatrixPtr;
typedef struct dd_matrixdata {
  dd_rowrange rowsize;
  dd_rowset linset;
    /* a subset of rows of linearity (ie, generators of
       linearity space for V-representation, and equations
       for H-representation. */
  dd_colrange colsize;
  dd_RepresentationType representation;
  dd_NumberType numbtype;
  dd_Amatrix matrix;
  dd_LPObjectiveType objective;
  dd_Arow rowvec;
} dd_MatrixType;
```

Listing 1: cdd_MatrixPtr in cddtypes.h

Implementation Details

```
void dd_set_global_constants(void)
```

initializes global constants such as dd_zero and dd_purezero

```
void dd_free_global_constants(void)
```

frees the global constants again

```
dd_MatrixPtr dd_PolyFile2Matrix(f, dd_ErrorType* err)
```

read polyhedra data from stream f into matrixdata and return a pointer to it

```
dd_PolyhedraPtr dd_DDMatrix2Poly(dd_MatrixPtr matrix,
   dd_ErrorType* err)
```

store the representation given by matrix in a polyhedra data and generate the second representation of *poly

Implementation Details

```
dd_PolyhedraPtr dd_DDMatrix2Poly2(dd_MatrixPtr matrix,
    dd_RowOrderType roworder, dd_ErrorType* err)
```

same as above with the extra argument roworder specifying the insertion order (eg. dd_MaxCutoff, dd_LexMin or dd_RandomRow)

```
dd_MatrixPtr dd_CopyInequalities(dd_PolyhedraPtr poly)
```

copy inequality representation pointed to by poly to matrixdata and return a pointer

```
dd_MatrixPtr dd_CopyGenerators(dd_PolyhedraPtr poly)
```

copy generator representation pointed to by poly to matrixdata and return a pointer

Examples

- example showing basic functionality (2-d cube)
- example demonstrating the importance of the ordering (8-d, 10-d cross polytope)
- cddlib vs lrslib example, i.e. very degenerate polytope (cddlib good) non degenerate polytope (lrslib good)

GMP: Why is this useful?

cddlib and lrslib can both be compiled with the GNU GMP library. This allows the conversion of polytopes with rational representation to be converted exact without the use of approximation by floating points.

Additional Functionality

Both libraries also contain an LP-solver to solve linear programming problems such as

$$\begin{aligned} \text{maximize} \quad & c^\top x \\ \text{subject} \quad & Ax \leq b \end{aligned}$$

References

- Komei Fukuda, *cddlib Reference Manual*. 2008.
- Komei Fukuda, *Frequently Asked Questions in Polyhedral Computation*. 2004.
- Komei Fukuda, *Lecture: Polyhedral Computation, Spring 2014*. 2014.
- David Avis, *User's Guide for lrs - Version 4.2b*. 2012.