CddInterface

Gap interface to Cdd package

0.1

18/11/2015

Kamal Saleh

Kamal Saleh
Email: kamal.saleh@rwth-aachen.de
Homepage:
Address: Templergraben

Contents

Chapter 1

Introduction

1.1 Why CddInterface

We know that every convex polyhedron has two representations, one as the intersection of finite half-spaces and the other as Minkowski sum of the convex hull of finite points and the nonnegative hull of finite directions. These are called H-representation and V-representation, respectively.
CddInterface is basicly written to translate between these two representations.

1.2 H-representation and V-representation of polyhedra

Let us start by introducing the H-representation. Let A be $m \times d$ matrix and let b be a column m-vector. The H-representation of the polyhedron defined by the system $b + Ax \geq 0$ of m inequalities and d variables $x = (x_1, \ldots, x_d)$ is as follows:

H-representation
linearity $t, \;\; [i_1, i_2, \ldots, i_t]$
begin
$m \times (d+1)$ numbertype
$b \qquad A$
end

The linearity line is added when we want to specify that some rows of the system $b + Ax$ are equalities. That is, $k \in \{i_1, i_2, \ldots, i_t\}$ means that the row k of the system $b + Ax$ is specified to be equality. For example, the H-representation of the polyhedron defined by the following system:

$4 - 3x_1 + 6x_2 - 5x_4 = 0$
$1 + 2x_1 - 2x_2 - 7x_3 \geq 0$
$-3x_2 + 5x_4 = 0$

is as follows:

H-representation
linearity $2, \;\; [1, 3]$
begin
3×5 rational

4	−3	6	0	−5
1	2	−2	−7	0
0	0	−3	0	5

end

Next we define Polyhedra V-format. Let P be represented by n gerating points and s generating directions (rays) as

$$P = conv(v_1,\dots,v_n) + nonneg(r_{n+1},\dots,r_{n+s}).$$

Then the Polyhedra V-format is for P is:

V-representation
linearity $t,\ [i_1,i_2,\dots,i_t]$
begin
$(n+s)\times(d+1)$ numbertype

1	v_1
$\vdots$	$\vdots$
1	v_n
0	r_{n+1}
$\vdots$	$\vdots$
0	r_{n+s}

end
In the above format the generating points and generating rays may appear mixed in arbitrary order. Linearity for V-representation specifies a subset of generators whose coefficients are relaxed to be free. That is, $k \in \{i_1,i_2,...,i_t\}$ specifies that the k-th generator is specified to be free. This means for each such a ray r_k , the line generated by r_k is in the polyhedron, and for each such a vertex v_k , its coefficient is no longer nonnegative but still the coefficients for all v_i's must sum up to one.
For example the V-representation of the polyhedron defined as

$$P := conv((2,3),(-2,-3),(3,5)) + nonneg(\ (1,2),(-1,-2),(2,11)\ )$$

V-representation
linearity $2,\ [1,3]$
begin
4×3 numbertype

1	2	3
1	3	5
0	1	2
0	2	11

end

Chapter 2

Creating polyhedras and their Operations

2.1 Creating a polyhedra

2.1.1 Cdd_PolyhedraByInequalities

▷ `Cdd_PolyhedraByInequalities(`*arg*`)` (function)

Returns: a `CddPolyhedra` Object

The function takes a list in which every entry represents an inequality(or equality). In case we want some entries to represent equalities we should refer in a second list to their indices.

Example

```
gap> A:= Cdd_PolyhedraByInequalities( [ [ 0, 1, 0 ], [ 0, 1, -1 ] ] );
< Polyhedra given by its H-representation >
gap> Display( A );
H-representation
begin
   2 X 3  rational

   0   1   0
   0   1  -1
end
gap> B:= Cdd_PolyhedraByInequalities( [ [ 0, 1, 0 ], [ 0, 1, -1 ] ], [ 2 ] );
< Polyhedra given by its H-representation >
gap> Display( B );
H-representation
Linearity 1, [ 2 ]
begin
   2 X 3  rational

   0   1   0
   0   1  -1
end
```

2.1.2 Cdd_PolyhedraByGenerators

▷ `Cdd_PolyhedraByGenerators(arg)` (function)

Returns: a `CddPolyhedra` Object

The function takes a list in which every entry represents a vertex in the ambient vector space. In case we want some vertices to be free(the vertex and its negative belong to the polyhedra) we should refer in a second list to their indices .

Example

```
gap> A:= Cdd_PolyhedraByGenerators( [ [ 0, 1, 3 ], [ 1, 4, 5 ] ] );
< Polyhedra given by its V-representation >
gap> Display( A );
V-representation
begin
   2 X 3  rational

   0  1  3
   1  4  5
end
gap> B:= Cdd_PolyhedraByGenerators( [ [ 0, 1, 3 ] ], [ 1 ] );
< Polyhedra given by its V-representation >
gap> Display( B );
V-representation
Linearity 1, [ 1 ]
begin
   1 X 3  rational

   0  1  3
end
```

2.2 Some operations on polyhedras

2.2.1 Cdd_Canonicalize (for IsCddPolyhedra)

▷ `Cdd_Canonicalize(poly)` (operation)

Returns: a `CddPolyhedra` Object

The function takes a polyhedra and reduces its defining inequalities (generators set) by deleting all redundant inequalities (generators).

Example

```
gap> A:= Cdd_PolyhedraByInequalities( [ [ 0, 2, 6 ], [ 0, 1, 3 ], [1, 4, 10 ] ] );
< Polyhedra given by its H-representation >
gap> B:= Cdd_Canonicalize( A );
< Polyhedra given by its H-representation >
gap> Display( B );
H-representation
begin
   2 X 3  rational

   0   1   3
   1   4  10
end
```

2.2.2 Cdd_V_Rep (for IsCddPolyhedra)

▷ `Cdd_V_Rep(poly)` (operation)

Returns: a `CddPolyhedra` Object

The function takes a polyhedra and returns its reduced V-representation.

2.2.3 Cdd_H_Rep (for IsCddPolyhedra)

▷ `Cdd_H_Rep(poly)` (operation)

Returns: a `CddPolyhedra` Object

The function takes a polyhedra and returns its reduced H-representation.

Example

```
gap> A:= Cdd_PolyhedraByInequalities( [ [ 0, 1, 1 ], [0, 5, 5 ] ] );
 Polyhedra given by its H-representation >
gap> B:= Cdd_V_Rep( A );
< Polyhedra given by its V-representation >
gap> Display( B );
V-representation
Linearity 1, [ 2 ]
begin
   2 X 3  rational

   0   1   0
   0  -1   1
end
gap> C:= Cdd_H_Rep( B );
< Polyhedra given by its H-representation >
gap> Display( C );
H-representation
begin
   1 X 3  rational

   0  1  1
end
gap> D:= Cdd_PolyhedraByInequalities( [ [ 0, 1, 1, 34, 22, 43 ],
> [ 11, 2, 2, 54, 53, 221 ], [33, 23, 45, 2, 40, 11 ] ] );
< Polyhedra given by its H-representation >
gap> Cdd_V_Rep( D );
< Polyhedra given by its V-representation >
gap> Display( last );
V-representation
Linearity 2, [ 5, 6 ]
begin
   6 X 6  rational

   1  -743/14   369/14   11/14        0        0
   0    -1213      619      22        0        0
   0       -1        1       0        0        0
   0      764     -390     -11        0        0
   0   -13526     6772      99      154        0
   0  -116608    59496    1485        0      154
end
```

Chapter 3

Linear Programs

3.1 Creating a linear program

3.1.1 Cdd_LinearProgram (for IsCddPolyhedra, IsString, IsList)

▷ `Cdd_LinearProgram(poly, str, obj)` (operation)

Returns: a `CddLinearProgram` Object

The function takes three variables. The first is a polyhedra `poly`, the second `str` should be max or min and the third `obj` is the objective.

3.1.2 Cdd_SolveLinearProgram (for IsCddLinearProgram)

▷ `Cdd_SolveLinearProgram(lp)` (operation)

Returns: a list if the program is optimal, otherwise returns the value 0

The function takes a linear program. If the program is optimal, the function returns a list of two entries: the solution vector and the optimal value of the objective, otherwise it returns the value 0.

To illustrate the using of these functions, let us solve the linear program given by: **Maximize** $P(x,y) = 1 - 2x + 5y$, with
$100 \leq x \leq 200$
$80 \leq y \leq 170$
$y \geq -x + 200$

We bring the inequalities to the form $b + AX \geq 0$, we get:
$-100 + x \geq 0$
$200 - x \geq 0$
$-80 + y \geq 0$
$170 - y \geq 0$
$-200 + x + y \geq 0$

Example

```
gap> A:= Cdd_PolyhedraByInequalities( [ [ -100, 1, 0 ], [ 200, -1, 0 ],
> [ -80, 0, 1 ], [ 170, 0, -1 ], [ -200, 1, 1 ] ] );
< Polyhedra given by its H-representation >
gap> Lp:= Cdd_LinearProgram( A, "max", [1, -2, 5 ] );
< Linear program >
```

```
gap> S:= Cdd_SolveLinearProgram( Lp );
[ [ 100, 170 ], 651 ]
gap> B:= Cdd_V_Rep( A );
< Polyhedra given by its V-representation >
gap> Display( Lp );
Linear program given by:
H-representation
begin
   5 X 3  rational

   -100     1     0
    200    -1     0
    -80     0     1
    170     0    -1
   -200     1     1
end
max  [ 1, -2, 5 ]
gap> Display( B );
V-representation
begin
   5 X 3  rational

   1  100  170
   1  100  100
   1  120   80
   1  200   80
   1  200  170
end
```

So the optimal solution is $(x = 100, y = 170)$ with optimal value $p = 1 - 2(100) + 5(170) = 651$.

Index