Title

Owen Rouillé

Abstract. abstract

1 Introduction

Computer scientists are used to deal with discrete systems, i.e. systems whose evolution can be described by a sequence of discrete state changes. A widespread example is the program analysis: during the execution of a program, the system is abstracted into a machine which states changes every time an instruction is executed. Today, computer science is expected to be able to ensure the safety of very complex systems, such as planes or nuclear reactors, which state are described non only by a controller, which is discrete, but also by physical variables, which evolve continuously over the time. Such systems, with a continuous and a discrete component are called hybrid systems. These systems are omnipresent, whenever a computer interacts with continuous quantities there is an hybrid system, even for thing as simple as a thermostat with the temperature.

To study these systems, they are abstracted into hybrid automata. These automata are then model checked. As an example let's take a thermostat, which correspond to an hybrid system, and turn it into an hybrid automata.

An hybrid system is described by a set of possible discrete states and a set of continuous variables. The evolution of the variables is determined by the current discrete state. The current state of the system is defined by the discrete state it is in and by the current value of the continuous variables. A transition between two discrete states can be done under certain conditions (called a guard) over the variables, a well as staying in a given state (called the invariant of the state). A transition can reassign the variables.

Example 1. heat controller

Analyse such systems is very difficult, the table 1 gives the complexity of model checking different classes of automata. Several methods exist to deal with this problem, theorem proving (model for a solver), interval based methods(model it for an SMT solver) and flowpipe computation. This paper contributes to the flowpipe-construction-based reachability analysis methods. The idea is to discretize the time and, at each time step, determine set containing all the possible values for the continuous variables for that time step. The successive representations of the set define a flowpipe=¿ image.

There exists different sub-methods for the flowpipe construction, each corresponding to different methods to describe the set where of the possible values of the variables. Zonotopes, support functions and boxes are examples of representation. A lot of operations have to performed on these sets along the way,

Sub-classes	derivative	conditions	bounded reachability	unbounded reachability
TA	$\dot{x} = 1$	$x \stackrel{?}{=} c$	✓	✓
IRA	$\dot{x} \in [c_1; c_2]$	$x \stackrel{?}{\in} [c_1; c_2]$ jump must reset x when $\dot{x}$ changes	✓	✓
RA	$\dot{x} \in [c_1; c_2]$	$x \stackrel{?}{\in} [c_1; c_2]$	✓	X
LHA I	$\dot{x} = c$	$x \stackrel{?}{=} g_{linear}$	✓	X
LHA II	$\dot{x} = f_{linear}$	$x \stackrel{?}{=} g_{linear}$	X	X
HA	$\dot{x} = f$	$\dot{x} = g$	X	X

Table 1. Decidability results for subclasses of automata. c, c_1, c_2 are constants, f and g are function of other variables. TA = timed automata, IRA = initialised rectangular automata, RA = rectangular automata, LHA I = hybrid automata with constant derivatives, LHA II = hybrid automata with linear ODE's, HA = general hybrid automata.

for instance intersections and unions. This paper proposes a method to switch between two representations of polyhedra: the V representation and the H representation according to the theorem 1. First some notations and definitions for the rest of the paper:

Definition 1 (Some geometry). *The problem is studied in* $\mathbb{R}^d$.

- $conv(V)$ *defines the convex hull of the set of vertices* V*:* $\{x \in \mathbb{R}^d | x = \sum_{v \in V} \lambda_v v, \sum_{v \in V} \lambda_v = 1, \forall v \in V,\ 0 \leq \lambda_v \in \mathbb{R}\}$.
- $cone(C)$ *defines the conic hull of the vectors in* C*:* $\{x \in \mathbb{R}^d | x = \sum_{c \in C} \lambda_c c, \forall c \in C,\ 0 \leq \lambda_c \in \mathbb{R}\}$.
- $lineal(L)$ *is the linear space generated by* L*:* $\{x \in \mathbb{R}^d | x = \sum_{l \in L} \lambda_l l, \forall l \in L,\ \lambda_c \in \mathbb{R}\}$.
- *In the paper, a sum between two sets is the Minkowsky sum:* $S_1 + S_2 = \{s_1 + s_2 | s_1 \in S_1,\ s_2 \in S_2\}$.
- *An half-space* h *is an area of* $\mathbb{R}^d$ *defined by a normal vector* a *and a constant* b, $h = \{x \in \mathbb{R}^d | x.a \leq b\}$, *its border is the hyperplane* $\{x \in \mathbb{R}^d | x.a = b\}$. *A family of half-spaces are said independent is their normal vectors are independent, and independent hyperplanes are defined respectively , d independent hyperplanes intersect in a vertex.*
- *Let* A *be a matrix which rows are normal vectors of a finite set of half-spaces and* B *the column vector composed by the corresponding constants.* $P(A, B) = \{x \in \mathbb{R}^d | Ax \leq B\}$ *is the set of the points contained in all the half-spaces.*
- *An* H*-polyhedron* P *is the intersection of a given finite set of half-spaces. There exist* A *and* B *such that* $P = P(A, B)$.
- *A* V*-polyhedron* P *is the sum of the convex hull of a finite set of points and a conic hull of a finite set of points,* $P = conv(V) + cone(C)$ *for some finite* V *and* C.

– *A polyhedron is an H-polyhedron or a V-polyhedron.*
– *A polytope is a bounded polyhedron (and respectively with H and V polytopes).*

Theorem 1 (Polyhedra representation). *A Subset $P \subseteq \mathbb{R}^d$ is an H-polyhedron if and only if it is a V-polyhedron.*

These two representations have their own advantages and disadvantages, these are summed up in the following tableau:

V-Polyhedra				
H-Polyhedra				

Thus, being able to switch between these two representations is crucial.

The hosting research group develops in the context of the HyPro project an open-source stand-alone C++ library for the most relevant geometric state set representations Furthermore the library allows the combination of different representations and over-approximative conversion between them. So far it includes

=¿ state representation polyhedra theory advantage/inconveniants of both representations

Some operations might have to be applied to the polyhedron such like intersections, unions ect. The difficulty of these oprations greatly depends on the description used for the polyhedron.

Def: V poly Def: H poly Theorem: there is a V and a H descripion for each polyhedron.

Being able to swich between these descripions is an important issue to overcome in reachability analysis.

The following part describe an algorithm to convert a bounded H polyhedron (an H polytope) into its V description and extend it to the unbounded case.

2 State of the art

text

2.1 the Simplex Algorithm

Linear optimization. The simplex algorithm is an algorithm that solves *linear optimization problems*. A linear optimization problem is finding the positive optimal of a linear function (the objective) $(x_i)_{i=1}^n \mapsto \sum_{i=1}^n c_i x_i$ under a set of inequality constrains $\sum_{i=1}^n a_i x_i \leq b_i$. Geometrically, such a problem is to find the point which coordinates maximize the objective in an area defined by a set of half-spaces (each constrain is the equation of a half-spaces). ~~The example 2 is a two dimensional linear optimization problem.~~

Example Maximize $x + y$ under $x \leq 2$ and $-x + y \leq 3$

To simplify the problem, the origin is assumed to respect all the constrains (equivalently: all the b_i are positive). When a point is on a constrain, the constrain is said *saturated*. The set of the points respecting all the constrains is called the feasible area.

The first step of the algorithm is to turn all the inequalities into equalities by the addition of positive *slack* variables. These variables corresponds to a distance from the associated constrain. Note that every variable (slack or not) corresponds to a constrain and though to an half-space. The non-slack variables will be said original. ~~The example 2 becomes the example 3.~~

Example 3. Maximize $x+y$ with $2-x=s_1$ and $3+x-y=s_2$. All the variables are positive.

The dictionary. By adding to these equalities an other equality corresponding to the cost function, a tableau of coefficients is obtained. This tableau is called a dictionary. So far, every row corresponds to a slack variable plus one for the objective function and every column corresponds to an initial variable, plus one for the constants. The case in the constants' column and objective's row is the current value of the. The tableau corresponding to the previous example is given in the example 4. In the rest of this report, the coefficient in the row i and column j is named a_{ij}.

Example 4.

x ↓	y ↓	constants ↓	
-1	0	2	$= s_1$
1	-1	3	$= s_2$
1	1	0	← objective function

The dictionary allows to find the value of every variable knowing the value of those associated to the columns. The set of the variables associated to the columns is called the *cobasis*, the set of the variables associated to the row is called the *basis*. Setting all the variables of the cobasis to zero provides a valuation for all the original variables. Such valuation describes a point against d hyperplanes: a vertex of the hyperplane arrangement. The cobasis is a coordinate system composed by the normal vectors of the hyperplanes associated to the variables in the cobasis, with for origin point the intersection of these hyperplanes. In the following, the basis is named B, the cobasis N, the row corresponding to the objective function is f ($\in B$) and the column containing the constants is g ($\in N$).

The pivot. The idea of the simplex is to exchange the variables between the basis and the cobasis (and updating in consequence the coefficients of the dictionary) such that when the cobasis is set to zero, the objective function increases and no basic variable is set to a negative value. This operation is called a pivot. From a geometrical point of view this operations consist in keeping n-1 constrains

saturated and exchanging the last one for an other, increasing the objective and staying in the feasible area. The vertices of the feasible area are explored until a maximum is reached.

To maintain the fact that the cobasis expresses the vector of the basis, the following transformations are applied to the dictionary. The coefficients of the new dictionary are primed, the pivot occurs between $r \in B - f$ and $s \in N - f$ and the transformation is for all $(i, j) \in (B - r, N - s)$:

$$a'_{sr} = \frac{1}{a_{rs}}, \qquad a'_{ir} = \frac{a_{is}}{a_{rs}}, \qquad a'_{sj} = -\frac{a_{rj}}{a_{rs}}, \qquad a'_{ij} = a_{ij} - \frac{a_{is}a_{rj}}{a_{rs}}.$$

a_{rs} is refereed as the coefficient corresponding to the pivot. If it equal zero, the pivot is impossible and it means that the variable of the basis is independent from the variable from the cobasis.

The Bland's rule is a deterministic rule to find around which variable pivot. It ensures the termination of the algorithm and keeps the dictionary inside the feasible area. This rule applied to the dictionary of example 4 asks for pivoting around the first row and the first column. Which sends x to the basis and s_1 to the cobasis. Setting the cobasis to 0 indicates that the dictionary represents the point $(2, 0)$. The next pivoting is around the second row and the second column which leads to the point$(2, 7)$, the Bland's rule then state that the optimum is reached. The example 5 provides the transformations of the dictionary.

Example 5.

x ↓	y ↓	const ↓	
-1	0	2	$= s_1$
1	-1	3	$= s_2$
1	1	0	$\leftarrow$ obj

→

s_1 ↓	y ↓	const ↓	
-1	0	2	$= x$
-1	-1	5	$= s_2$
-1	1	2	$\leftarrow$ obj

→

s_1 ↓	s_2 ↓	const ↓	
-1	0	2	$= x$
-1	-1	5	$= y$
-2	-1	7	$\leftarrow$ obj

Some properties. A basic variable is primal feasible if the associated constant is non negative, a cobasic variable is said dual feasible is the associated coefficient in the objective function is non positive. A dictionary is primal (resp. dual) feasible if all the variables is the basis (resp. cobasis) are primal (resp. dual) feasible. A dictionary is optimal if it is both primal and dual feasible. A dictionary is primal feasible if and only if it describes a point inside the feasible area. A dictionary is dual feasible if and only if there is no pivot able to increase the objective. Note that the dictionary obtained in the example 5 is optimal.

Proposition 1. *The normal vectors of the hyperplanes associated to the variables of the cobasis describe an independent family: the cobasis is, in fact, a basis.*

This proposition is obtained by induction. At the beginning of the algorithm the cobasis is the canonic basis. The only operation applied to the dictionaries (the pivoting) maintain this property. If one tries to break this invariant, the new potential member of the cobasis is a linear combination of the others. This means the value of the corresponding variable is determined by the other member of cobasis which implies the pivoting provoked a division by zero.

Alternative form. Instead of setting the cobasis to zero, lower and/or upper bounds can be kept for each variables, alongside with a valuation (initialized at 0 for every variable). The original variables don't have bounds. Once again the valuation of the variables of the cobasis is sufficient to determine the valuation of the basis. The dictionary stays identical (only the constants' column becomes useless). Here the variables can be positive or negative.

The first phase will be to reach the feasible area, i.e. ensure that for all the variables, the valuation respects the bounds. To do so, let's pick a variable i in the basis which bounds are not satisfied, without loss of generality let's assume the upper bound is not satisfied. Then let's pivot it with a cobasic variable j smaller than its upper bound if $a_{ij} > 0$ or greater than its lower bound if $a_{ij} < 0$. Then the variable i is assigned to its upper bound and the new valuation of the basis is computed. This operation is to be repeated until all the variables are in their bounds, a feasible point is found, or until no suitable pivot can be found, the feasible area is empty. ~~The~~ example 6 gives a problem and executes this phase. Note that this does not ~~find~~ a vertex of the arrangement.

Example 6. Reach the feasible area of $-x \leq -1$ and $-x - y \leq 2$.
Creation of slack variables: $x = s_1$ and $x + y = s_2$.
Constrains: $x \in]-\infty;\infty[$, $y \in]-\infty;\infty[$, $s_1 \in]1;\infty[$, $s_2 \in]-2;\infty[$.
Let v be the valuation, mapping every variable to 0 so far.

x $\downarrow$	y $\downarrow$		
1	0	=	s_1
1	1	=	s_2

Step 1: s_1 is out of its bounds, x is suitable for the pivot. The dictionary becomes:

s_1 $\downarrow$	y $\downarrow$		
1	0	=	x
1	1	=	s_2

, and the valuation is $v(s_1) = v(x) = 1$ and $v(s_2) = v(y) = 0$, all the variables are in their bounds and $(1, 0)$ is feasible.

The second phase, the optimization is very similar to the classic algorithm, instead of pushing the cobasis to zero, the valuation is fixed and the variables are pushed to their upper or lower bound (depending on the objective) when they are sent in the cobasis. This second phase will not be used in this paper.

2.2 Fukuda's algorihm

Fukuda's algorithm allows to enumerate all the vertices of an H-polytope included in the positive orthant ($(\mathbb{R}_+)^d$, all its points have non-negative coordinates) for which the origin is a vertex. This algorithm uses the pivoting scheme of the simplex to explore the polytope.

It is based on a simple idea: starting from any vertex, the simplex algorithm allows to reach a vertex maximizing a linear function by a unique (thanks to Bland's rule) path among the vertices. The algorithm picks a function maximized

only by the origin on the positive orthant and walks backward on all the paths leading to it from all the vertices. Every time a vertex is encountered, it is output.

The first thing to do is to find the end of all these paths: the dictionary corresponding to the origin. It roughly consists in taking the canonical base as cobasis and the set constrains as the basis and setting $-\sum_{i=0}^{d} x_i$ as the cost function. All the possible pivots are tried from this dictionary. If a pivot leads to a primal feasible dictionary (defining a vertex inside the polytope) for which the Bland's method leads back to the previous point it is said to be a valid reverse Bland's pivot. It means that a step backward has been made on one of the paths, the new point is output, and the method has to be applied from this new point.

However, if a vertex is defined by more hyperplanes than required, several dictionaries will describe the same point, which is said to be degenerated. Fukuda overcomes this problem by defining a partial order on the dictionaries. This order corresponds to a lexicographic order on the basis for the dictionaries defining the same point. A point is only output if its dictionary is lexicographical minimal. Note that the algorithm continues on the dictionary, even if it is not lexicographical minimal.

If the origin is degenerated, then all the optimal dictionaries defining it have to be found. This is done by using a procedure very similar from the one above: by looking only at the hyperplanes redefining the origin, every pivot is tried. The difference is instead of using the Bland's rule, a variation of the dual Bland's rule is used. If the new dictionary is dual feasible (still an optimal for the cost function) and the dual Bland's rule brings it back to the previous one, then the new dictionary is valid with respect to the dual Bland's rule etc. This research presents the same uniqueness properties than the previous one. The previous research has to be launched from every dictionary obtained this way.

The following example illustrates Fukuda's algorithm on a polytope with no degenerated vertex:

Example 7. Let h be the polytope included in the positive orthant and the half-space $x + 2y \leq 4$ (plus the positve orthant, i.e. $-x \leq 0$ and $-y \leq 0$) in $\mathbb{R}^2$.

First step, the optimal dictionary:

x ↓	y ↓	const ↓	
-1	-2	4	$= s_1$
-1	-1	0	$\leftarrow$ obj

, (x, s_1) and (y, s_1) are both valid reverse Bland's pivot and gives respectively:

s_1 ↓	y ↓	const ↓	
-1	-2	4	$= x$
1	1	-4	$\leftarrow$ obj

for the point $(4, 0)$ and

x ↓	s_1 ↓	const ↓	
$-\frac{1}{2}$	$-\frac{1}{2}$	2	$= y$
$-\frac{1}{2}$	$-\frac{1}{2}$	-2	$\leftarrow$ obj

for the point $(0, 2)$. These two dictionaries have no valid reverse Bland's pivot, the enumeration is done.

3 Contribution

The purpose of this internship was to add two methods in HyPro: one to find the vertices, the cone and the linealty space of and H-polyhedron and one to find the convex hull of a V-polyhedron.
Fukuda's algorithm allows to enumerate the vertices of an H-polytope. The first thing to do was extending it into being able to handle non positive polytopes and then polyhedra. This method implemented, an other one had to be created to convert the convex hull problem such that h[illegible]rmer could handle the inverse problem.

3.1 Adaptation of Fukuda's algorithm for polyhedra

Converting to positive coordinates and detecting the linealy space Knowing a vertex of a polyhedron allows to find an affine transformation which sends it in the positive orthant. The process is to map the vertex to the origin and the normals of the hyperplanes defining it to the canonical base by respectively a translation and a change of basis.

Finding a vertex of the polyhedron is done using a dictionary like [illegible] the first phase of the general simplex seen in section 2.1. In fact, obtaining a cobasis composed only by slack variables pushed to their bounds with all the other variables in their bounds gives a dictionary that describes a vertex of the polyhedron. Here the row corresponding to the cost function and the column corresponding to the constants are not used.

To do so, the first thing is to b[illegible]ld a dictionary and to reach the feasible area, to check for emptiness. This is exactly the first phase of the simplex. Then, since the cobasis has to be composed only by slack variables, all the original variables in the cobasis are pivoted with a suitable slack variable (the corresponding coefficient has to be non null) but the assignments are not changed: it could bring the dictionary out of the polyhedron.

Detection of the linealty space: If an original variable has no suitable slack variable to be exchanged with, it means this variable has no influence on the distance to each constrains, there is a line included in the polyhedron: a linealty space has been found. To overcome this problem, two constrains are added with opposed normal vectors, their directions being the linealty space's one, and the constant is zero for both, which might implie[illegible] to reach again the feasible area once all the linealty direction are found. This provides a suitable slack variable to pivot around and the linealty space is saved, ~~the~~ example 8 illustrate this procedure. A cobasis full of slack variables means there exis[illegible]nough independent constrains to define a vertex and there is no linealty space left.

Example 8. Let's consider the following dictionary (the constrains d[illegible]not matter for the linealty space detection):

x $\downarrow$	y $\downarrow$	s_1 $\downarrow$	
2	1	0	$= z$
0	1	2	$= s_2$

The objective is to put x in the cobasis, which is not possible. In fact, for any value of x, there is a suitable z such that the current point is inside the feasible area: $z = 2x$ is the equation of a linealty direction.
The constrain to add has $(1, 0, 2)$ as normal vector, the problem is it has to be expressed in term of (x, y, s_1). The method is to add 2 times (the coefficient corresponding to z) the value of z, in the end the normal vector to add is $(5, 2, 0)$ and the dictionary becomes:

x ↓	y ↓	s_1 ↓	
2	1	0	$= z$
0	1	2	$= s_2$
5	2	0	$= s_+$
-5	-2	0	$= s_-$

The last thing is pushing one by one the variables of the cobasis to their bounds. If doing so would make other variables violate their bounds, the variable is exchanged with the one which bound would be violated first, the latter is then set to its bound (note that the original variables have no bounds). A vertex is found with the hyperplanes that define it.

Cone detection in Fukuda's algorithm Here the problem is reduced to the vertex enumeration of a potentially unbounded polyhedon in the positive orthant. Fukuda's algorithm allows to explore all the vertices. The unbounded side does not bother the algorithm: it just switches between the constrains, unboundedness is a lack of constrain.

Proposition 2. *Every non redundant vector of the cone lies in the intersection of $d-1$ linearly independent half-spaces which intersection with a d_{th} provides a vertex of the polyhedron.*

The proposition 2 states that to find all the vectors of the cone, it is sufficient to check every vertex for the existence of a conic direction, i.e. directions such that $n-1$ independent constrains are saturated. Fukuda's algorithm explores the polyhedron and finds all the possible arrangement of constrains that define the vertices, thus for every definition of a vertex, the dictionary is check for the existence of a cobasic variable that can be increased while not decreasing any other (not getting closer from an other constrain) and which decreases the objective function (not getting closer from an axis). Conversely, a vector found this way is a ray included in the polyhedron: it belongs to the cone. The rays can be output several times, a suppression of the duplicates is done. The example 9 detects the vertices and the cones according to this procedure.

Example 9. Sole constrain: $x - y \leq 2$. The initial dictionary:

x ↓	y ↓	const ↓	
-1	1	2	$= s_1$
-1	-1	0	← obj

This dictionary gives 0 vertex. Concerning the cones, as s_1 would be decreasing, x does not provides a cone direction, yet y does: the objective is decreased

and s_1 increases: $(0, 1)$ is a cone direction.
The next (and last) dictionary to explore is obtained by pivoting x and s_1:

s_1 ↓	y ↓	const ↓	
-1	1	2	$= x$
1	-2	-2	$\leftarrow$ obj

. The vertex obtained is $(2, 0)$ and a cone direction is found: $(1, 1)$ with the variable y.

The algorithm enumerates the linealty space when looking for a first vertex, and the cone and the vertices during Fukuda's algorithm: it allows to switch from an H-polyhedron to a V-polyhedron.

3.2 From vertex enumeration to convex hull

In geometry, vertices and half-spaces are close concepts: a vertex $(a_i)_{i=0}^d$ can be associated to an half-space $\sum_{i=0}^d x_i a_i \leq 1$ et vice-versa. This association is the *geometric duality* and it allows to use the vertex enumeration algorithm to find the convex hull of a polyhedron. The emptiness and the equality to a single point of a V-polyhedron is trivial to check, in the following, the polyhedron and polytopes are assumed to be at least two-dimensional.

The convex hull of a polytope The polytope studied is refereed as P, convex hull of the set V. Even it means to translate P, it is assumed that $\sum_{x \in V} x = 0$. Since the polytope is at least two-dimensional, the origin is not a vertex.

Definition 2 (The dual of a subset of $\mathbb{R}^d$). *The dual P^Δ of a subset P of $\mathbb{R}^d$ is defined by $\{c | \forall x \in P, cx \leq 1\}$*

The theorem 2 holds the idea of the transformation:

Theorem 2. *If P is a polytope and $0 \in P$, $P = (P^\Delta)^\Delta$.*

Starting with the set of the vertices of P, the algorithm begins by switching to P^Δ, which is described by a set of half-spaces. The vertices of P^Δ enumerated and the dual is taken, which leads to a set of half-spaces describing P. Indicing by the description mean, the algorithm does $P_V \to P_H^\Delta \to P_V^\Delta \to P_H^{\Delta\Delta} = P_H$.

The first step ($P_V \to P_H^\Delta$) does not hold any problem and is done with the theorem 3:

Theorem 3. *For P the convex hull of a given set V of vertices, $P^\Delta = \{c | \forall x \in V, cx \leq 1\}$.*

The second step ($P_H^\Delta \to P_V^\Delta$) is done by the vertex enumeration algorithm.

The third ($P_V^\Delta \to P_H^{\Delta\Delta}$) does not cause any problem if the origin is not on a facet of P (or a face of other dimension, seen as an intersection of facets): then P^Δ is bounded and the theorem 3 holds.

Otherwise, let $\{x|w.x \leq 0\}$ be a facet of P (since 0 is the iso-barycenter of P, $\{x| - w.x \leq 0\}$ is also a facet). Let $x \in P$, $y \in P^\Delta$ and $\alpha \in \mathbb{R}$, $(y + \alpha w).x = y.x + \alpha(w.x) = y.x$, this means that $vect(w)$ is a linealty space of P^Δ. $P^\Delta = conv(V') + lineal(L')$ thus $P^{\Delta\Delta} = conv(V')^\Delta \cap L'^\perp$ by the proposition 3. Which end the convex hull problem for a polytope.

Proposition 3. *For V a set of vertices and L a set of non null vectors $(conv(V) + lineal(L))^\Delta = conv(V')^\Delta \cap L'^\perp$.*

Proof. Let $y \in (conv(V) + lineal(L))^\Delta$, for all $v \in V$, $l \in L$ and $\alpha \in \mathbb{R}$, $y.(v + \alpha l) \leq 1$. The inequality holding for any α, $y.l = 0$ and $y.v \leq 1$ and $y \in conv(V')^\Delta \cap L'^\perp$. Conversely, let $y \in conv(V')^\Delta \cap L'^\perp$ which means $y.\alpha l = 0$ and $y.v \leq 1$ thus $y.(v + \alpha l) \leq 1$ for all $v \in V$, $l \in L$ and $\alpha \in \mathbb{R}$: $y \in (conv(V) + lineal(L))^\Delta$.

The convex hull of a cone To find the convex hull of a cone C, the convex hull algorithm for a set of points is used. The origin is added to the set of vectors describing the cone, as for all c in C, the segment between the origin and c must belong to the cone. Then the half-spaces for which the constant is not zero (i.e. the origin does not saturate the constrain) are deleted. This deletion is required to ensure the unboundedness of the cone. Note that this method allows to handle a linealty direction as the sum of two vectors in the cone.

The convex hull of a polyhedron Here the linealty space is injected in the cone (as the sum of two opposite vectors), as the treatment does not differ. Let $P = conv(V) + cone(C)$ be a polyhedron, the method is given by the following result:

Proposition 4. *Let $P = conv(V)+cone(C)$ be a polyhedron, C' the set obtained from C by adding a 0 as a 0^{th} coordinate to every vectors $(c \mapsto (0, c))$ and V' the set obtained by, by adding a 1 as a 0^{th} coordinate to every vectors $(v \mapsto (1, v))$, then $P = cone(V', C') \cap \{x|x = (1, y), y \in \mathbb{R}^d\}$*

Starting from P, C' and V' are easily obtained and $cone(V', C')$ is given in 3.2. The intersection is done as follows:

$$\{x \in \mathbb{R}^{d+1} | a_0 x_0 + \sum_{i=1}^{d} a_i x_i \leq b\} \cap \{x|x_0 = 1\} = \{x \in \mathbb{R}^d | \sum_{i=1}^{d} a_i x_i \leq b - a_0\}$$

This operation can generate half-spaces with an equation such as $0 \leq b$, such half-spaces are those which does not intersect with $\{x|x_0 = 1\}$ and they are eliminated.

3.3 Performances

4 Conclusion