Approximating the convex hull of polytopes using projection,

Romain Testylier (VERIMAG), Thao Dang (VERIMAG),
Bertrand Jeannet (INRIA)

March 30, 2011

1 Introduction

Analysis of continuous systems like reachability computation is a strongly studied field [2], however its tracktability over dimension lead to some limitations concerning set representation . Analysis system require to have a robust set representation over operation such like set intersection, minkowsky sum or convex hull between two set.

Set representation class wich can be used in reachability analysis are multiple (boxes, ellipsoids, zonotope...) [3]. Polytopes class is a very efficient set representation in reachability analysis, our recent work show that we can execute reachable computation on high dimensional discrete system using convex polytopes [1]. Typically these polytopes can be represented either by the convex hull of a list of vertice or by the intersection of a finite set of half-spaces given by linear inequalities.

It is sometime usefull to maintain this two representations together for set operations, by example it is easy to check the enclosure of a point using half space and sets of points is traditionally used to compute the exact convex hull of two polytopes. However, in high dimension the representation of polytope like hypercube lead to an exponential growth of the number of vertice (2^n in n dimension), only half-space based representation can be used. To extend our method of analysis to continuous dynamic we need a method to compute convex hull operation between two polytope represented by half-spaces,

Recent studies in polyhedral analysis [6] show that we can compute an approximation of the convex hull of two polytope without frame representation by projecting variables in a set of inequalities given by the mathematic definition of convex hull of two sets. We have investigate the work of Axel Simon and Andy King concerning the computation of a possibly approximated convex hull using projection and exploiting sparsity of inequalities.

In a first part we will give a mathematic definition of the convex hull, then we present two methods of projection of variable in a set of inequality

and presents their interrest in our problem, finally we will present some experimental results.

2 Convex hull

The convex hull operation takes as input two inequality sets representing two polytopes P_1 and P_2 in a n-dimensional space. Each one of these set are represented by the intersection of their half-spaces. The set of inequalities can be represented by a matrix with m rows and n columns and and a column vector with m elements (m is the number of inequality) $A \cdot x \leq c\}$. The two polytope can be expressed as:

$$P_1 = \{x \mid A_1 \cdot x \leq c_1\}$$

$$P_2 = \{x \mid A_2 \cdot x \leq c_2\}$$

This operation produce a polytope P_{hull} that is the smallest convex set including all points x included in P_1 or P_2.

$$P_{hull} = \left\{ x \;\middle|\; \begin{array}{l} x = \sigma_1 x_1 + \sigma_2 x_2 \wedge \\ \sigma_1 + \sigma_2 = 1 \wedge \\ \sigma_1 \geq 0 \ \wedge \ \sigma_2 \geq 0 \wedge \\ A_1 \cdot x_1 \leq c_1 \ \wedge \ A_2 \cdot x_2 \leq c_2 \end{array} \right\}$$

to avoid non-linearity $x = \sigma_1 x_1 + \sigma_2 x_2$ we can relax the system by introducing new variables $y_1 = \sigma_1 x_1$ and $y_2 = \sigma_2 x_2$ so that $x = y_1 + y_2$ and $A_1 \cdot y_1 \leq \sigma_1 c_1$ and $A_2 \cdot y_2 \leq \sigma_2 c_2$. We obtain:

$$P'_{hull} = \left\{ x \;\middle|\; \begin{array}{l} x = y_1 + y_2 \wedge \\ \sigma_1 + \sigma_2 = 1 \wedge \\ \sigma_1 \geq 0 \ \wedge \ \sigma_2 \geq 0 \wedge \\ A_1 \cdot y_1 \leq \sigma_1 c_1 \ \wedge \ A_2 \cdot y_2 \leq \sigma_2 c_2 \end{array} \right\}$$

If the two polytopes P_1 and P_2 are in n-variable space, their convex hull P'_{hull} will is described in a $(3n + 2)$-variable space. To find the convex hull in the original space we have to project the variables σ_1, σ_2, y_1 and y_2. Note that the following projection methods are designed to work with strict inequalities only, the best way to obtain a set of strict inequalities is to use the gauss pivot by posing $y_1 = y_2 - x$ and $\sigma_1 = 1 - \sigma_2$. This let us project y_1 and σ_1 and remove the equalities of the convex hull expression. We obtain the following inequality set:

$$P'_{hull} = \left\{ x \;\middle|\; \begin{array}{l} \sigma_1 \geq 0 \ \wedge \ \sigma_2 \geq 0 \wedge \\ A_1 \cdot (y_2 - x) \leq (1 - \sigma_2) c_1 \ \wedge \ A_2 \cdot y_2 \leq \sigma_2 c_2 \end{array} \right\}$$

The remaining variables to project are y_2 and σ_2. The projection space will be composed only by the x variables.

3 Projection techniques

To compute the convex hull we have to project for two polytopes in a n-dimensional space $(n+2)$ variables from a set of stric inequalities. We have implemented two projection method, one computing the exact convex hull powerfull with sparse inequality and the other computing an approximation.

3.1 Fourier-Motzkin

The Fourier-Motzkin projection method use the basic observation that inequalities may only be scaled by non negative numbers which implies that the set of inequality include either positive and negative coefficient for a projected variable. This method take in parameters a variable x_i to project and a set of inequalities E and return, in case of success, a set E' in the projection space .

The first step of this method is to divide E in three subsets E^+, E^-, E^r of inequalities with positive, negative and nul coefficient for x_i. If E^+ or E^- is empty the projection cannot be done. Then a set of new inequalities in the projection space is computed by combining inequalities in E^+ and E^-. 1 describe this method, we use the notation $v[i]$ to access

Algorithm 1 Fourier-Motzkin

for $a^+ \cdot x \leq c^+ \in E^+$ **do**
  for $a^- \cdot x \leq c^+ \in E^-$ **do**
    $a \cdot x \leq c \leftarrow (a^+[i] \cdot a^- + a^-[i] \cdot a^+) \cdot x \leq a^+[i] \cdot c^- + a^-[i] \cdot c^+$
    if $a \neq 0$ **then**
      $E^r \leftarrow E^r \cup a \cdot x \leq c$
    end if
  end for
end for
RETURN E^r

The number of new inequalities is equal to $|E^+||E^-| - (|E^+| + |E^-|)$, this can lead to a fast explosion of the number of inequalities in the case of successive variable elimination.

Also this method generate sytematically redundant inequalities, it is possible then possible to clean the result. A first cleaning process can be done by looking for quasy syntactaly inequalities, $a_1 \cdot x \leq c_1$ and $a_2 \cdot x \leq c_2$ are quasi syntactaly redundant if $a_1 = a_2$, we keep in this case the inequality with the lower c_i. Another cleaning method more expensive is to solve an optimisation problem to check if an inequality is redundant. For an inequality $a_i \cdot x \leq c_i$ we solve $\max(a_i \dot{x})$ submit to $E \backslash \{a_i \cdot x \leq c\}$ and we check if the result is greater than c_i to know if the contraint is redundant.

This method is particularly efficient in the case of sparse system leading

to a limited occurence of a variable in a set of inequality and if $|E^+|$ or $|E^-|$ are equal to 1 the system shrink. this method can also be fitted in the case of polytopes with templates like hypercube or octahedrons.

However this method can be limited in the case of convex hull of polytopes representing reachable sets especially with faces dynamics leading generally to dense systems. This method can still be use in the case of use of template polytopes.

3.2 Extreme points

3.2.1 Initial algorithm

Extreme point projection is a method proposed by Huynh and al. [4] to work with non sparse system. This method find inequalities incrementally in the projection space by enabling the projection to be approximated with a limited number of inequalities. The projection method take as input $E = \{a_1 \cdot x \leq c_1, \ldots, a_m \cdot x \leq c_m\}$ a set of inequalities and $Y \in X$ a list of variables to project. It returns a set of inequalities such that the polytope represented by these half-spaces is an overapproximation of the convex hull or in some cases the exact convex hull.

In a first step, this algorithm cut each vectors in two matrices A and B such that A possess all the inequalities coefficients concerning variables to project and B contains the other coefficients.

$$\begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} = (A|B) \quad \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y \\ \hline z \end{pmatrix} y \in Y \quad c = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix}$$

Then we can write our set of inequalities as $Ay + Bz \leq c$. Now we search a linear combination λ of all inequality scaling each one such that $\lambda \cdot A = 0$ and $\lambda[i] \geq 0, i \in [1..m]$. Each λ computed correspond to a new inequality in the projection space given by $\lambda \cdot (Ay + Bz) \leq \lambda \cdot c$ equivalent to $(\lambda \cdot B)z \leq \lambda \cdot c$.

Note that $(\lambda \cdot B) = 0$ will not produce a valid inequality in the projection space, we have to check that $\lambda \cdot B$ give a non nul vector before adding the new inequality to the output.

A trivial solution is $\lambda = 0$, we also have to check that $\lambda \cdot B \neq 0$, to avoid this tautology we enforce the contraint with the equality $\lambda \cdot B = 1$. we define the polytope $\Lambda \leftarrow \{\lambda \cdot A = 0\} \cup \{\lambda_i \geq 0 \mid \lambda_i \in \lambda\} \cup \{\sum \lambda_i = 1\}$ and solve linear optimisation problem using goal functions that we generate to find its vertices, each computed vertex will be used for computing a new inequality.

2 summarise this method, the number of goal functions will control the number of resulting inequalities.

Note that two different λ can lead to the same inequality in the projection space because the resulting inequality consider only the matrix B, the

Algorithm 2 Extreme Point

$(A|B) \leftarrow \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \wedge A \cdot y + B \cdot z \leq \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \wedge y \in Y$
$\Lambda \leftarrow \{\lambda \cdot A = 0\} \cup \{\lambda_i \geq 0 \mid \lambda_i \in \lambda\} \cup \{\sum \lambda_i = 1\}$
$E^r \leftarrow \oslash$
$F \leftarrow$ generate_goals
for $f \in F$ **do**
  $m \leftarrow \text{simplex}(f, \lambda, \Lambda)$ {maximise $f \cdot \lambda$ submit to Λ}
  $E^r \leftarrow E^r \cup (m \cdot B \leq m \cdot < c_1, \cdots, c_n >)$
end for
RETURN E^r

resulting set of inequalities can contains redundant constraints. To deal with it we use the same cleaning techniques described in the previous chapter.

generate_goals in the original algorithm is set with unitary vectors $F \leftarrow \{< 1, 0, \ldots, 0 >, < 0, 1, \ldots, 0 >, \ldots, < 0, 0, \ldots, 1 >\}$ F contains m inequalitie that induce no growth of the result even shrink in the case of redundant constraints.

However using this technique we compute an overapproximation of the convex hull.

3.2.2 Precision improvement

The exact convex hull can be obtained by computing all vertices of Λ, finding more vertices may induce adding non redundant constraint to the convex hull inequalities set result and gain accuracy.

While the previous methods are based on algebraic manipulation, we adopt a more geometric approach to improve the previous technique accuracy. However enumerating all vertices of Λ will lead to the initial issue concerning frame representation when we work on high dimensionals system.

An intuitive way to compute some new vertices can be to straightfowardly add new goals function in generate_goals. We choose in a first time this approach to deal with simple 2D systems. However each new goals function does not lead necessarly to a new vertex, this method require to clean the result of all redundant constraint.

The geometric method proposed in [5] can be used to compute efficiently new vertices directly in the projection space.

4 Experimentation

We have implemented The two projection methods and execute some preliminarie example. The program was written in C++ and we use the simplex

routine provided by the open source library LPsolve.

On simple test using boxes, the Fourier-Motzkin projection gave us the exact convex hull, however the number of equalities generated growth exponentially wen we start to use different type of polyhedron, it limits clearly the dimension increase. We focused our effort on extreme point projection. We successfully compute an overapproximation of the convex hull in high dimension.

The figure 1 shows the convex set computed by extreme point projection using unitary goals functions, it illustrate the overapproximation issue occuring with this technique.

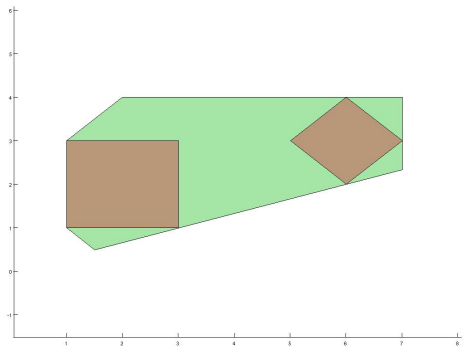

Figure 1: Convex hull using extreme point projection with unitary goal function

Figure 2 shows the interest of using additional goal function, in this case we obtain the exact convex hull by adding goals functions with 2 non nul variables.

$$F \leftarrow \left\{ \begin{array}{c} < 1, 0, \ldots, 0 >, < 0, 1, \ldots, 0 >, \ldots, < 0, 0, \ldots, 1 > \\ < 1, 1, 0, \ldots, 0 >, < 0, 1, 1, \ldots, 0 >, \ldots, < 1, 0, \ldots, 0, 1 > \\ < -1, -1, 0, \ldots, 0 >, < 0, -1, -1, \ldots, 0 >, \ldots, < -1, 0, \ldots, 0, -1 > \end{array} \right\}$$

However using these addition goal functions does not guarantee the exact convex hull in other example but we observed each time a gain of precision.

Figure 3 show the result of this projection method in a 3-dimensionals problem. We generate constraint in the same scheme than the previous example.

Finally we test the execution time of the convex hull of two randomly created bounded polytopes with each time 3*n faces in dimension n. The table 3 show that the extreme point projection is scalable and can be use to compute an overapproximation of the convex hull of two polytope in high dimension in a polynomial time.

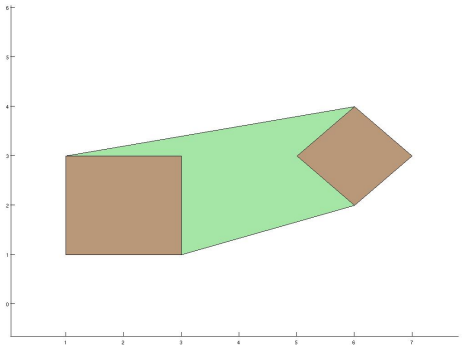

Figure 2: Convex hull using extreme point projection with unitary goal and 2 non nul variables vector function

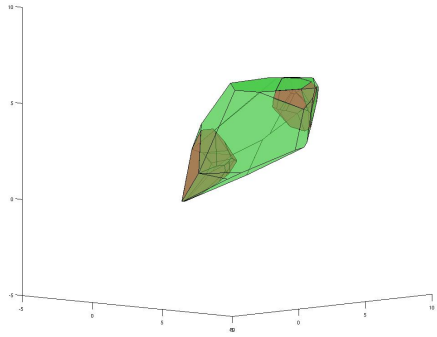

Figure 3: Convex hull of 3d randomly generated set using extreme point projection with unitary goal and 2 non nul variable vector function

Dimension	Total time (s)	number of resulting faces
2	0.02s	12
5	0.05s	32
10	0.28s	62
15	0.82s	92
20	2.11s	122
30	8.37s	182
40	23.06s	241
50	53.00s	302
75	225.54s	447

Figure 4: Computation times for convex hull operation using extreme point projection.

5 Conclusion

In this report we have implemented and tested an algorithm to compute an approximation of the convex hull operation between two polytope defined by half-spaces, that can be in some case the exact convex hull. We have shown that we can improve the extreme point projection accuracy by adding additional goal functions to find more vertices in the polytope Λ. The main interest of this methods is to be scalable over dimension. The accuracy control of the extreme point projection result remains an issue and a research investigation field.

References

[1] Eugene Asarin, Thao Dang, Oded Maler, and Romain Testylier. Using redundant constraints for refinement. In *ATVA*, pages 37–51, 2010.

[2] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. In *CMSB*, pages 126–141, 2009.

[3] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *CAV*, pages 540–554, 2009.

[4] Tien Huynh, Catherine Lassez, and Jean-Louis Lassez. Practical issues on the projection of polyhedral sets. *Ann. Math. Artif. Intell.*, 6(4):295–315, 1992.

[5] Catherine Lassez and Jean-Louis Lassez. *Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm.* Kapur and Mundy (Academic Press), 1991.

[6] Axel Simon and Andy King. Exploiting sparsity in polyhedral analysis. In *SAS*, pages 336–351, 2005.