Reduce the Complexity of the Polyhedron Minimization Using the Max Plus Pruning Method

Yassamine Seladji(✉)

STIC Laboratory, University of Tlemcen, Tlemcen, Algeria
yassamine.seladji@gmail.com

Abstract. The polyhedral analysis is widely used for the static analysis of programs, thanks to its expressiveness but it is also time consuming. To deal with that, a sub-polyhedral analysis has been developed which offers a good tread off between expressiveness and sufficiency. This analysis is based on a set of directions which is defined statically at the beginning of the analysis. More the cardinality of Δ is big, more the precision of the result is high. Even if the set Δ is big, the sub-polyhedral analysis can be done in a linear time. The bottleneck is that to construct the resulting polyhedron with a large number of constraints (one constraints per direction) is time consuming. In this article, we present a minimization method that allows to deal with that, using the max plus pruning method. We demonstrate the efficiency of our method on some benchmarks. The first results are very encouraging.

1 Introduction

In the abstract interpretation [2,3], the key component is represented by the abstract domain. A lot of them have been developed to deal with the multiple challenges of the program analysis. The most expressiveness one is the polyhedra abstract domain [4], but its analysis is time consuming. To deal with that, a lot of effort have been done by the researchers in the field and that to find a good trade-off between expressiveness and efficiency. A lot of domains have been developed, known as the sub-polyhedra or weakly relational abstract domains [6,8,9,11,12]. The authors in [10] present an abstract domain based on support functions, noted $\mathbb{P}^\sharp_\Delta$. This domain proposes a good balance between expressiveness and computational time. The lattice of $\mathbb{P}^\sharp_\Delta$ is closed to the lattice of the template abstract domain [9], but its result is more accurate than the one obtained using the template analysis. Because the precision of the polyhedral analysis is preserved using the $\mathbb{P}^\sharp_\Delta$ analysis and that based on the choice of a finite set of direction Δ. The larger the cardinality of Δ, the higher the precision of the result.

The execution time of the $\mathbb{P}^\sharp_\Delta$ analysis is linear in the cardinality of Δ. So, we can get a precise post fixed point using a large number of random directions. The problem is that taking a large number of directions means that the obtained

S. Bogomolov et al. (Eds.): NSV 2016, LNCS 10152, pp. 96–104, 2017.
DOI: 10.1007/978-3-319-54292-8_9

polyhedron contains the same number of constraints. So, the minimisation of this polyhedron is very time consuming. Because the minimization method, firstly, deletes the redundant constraints then computes the intersection of the other constraints. In this article, we present a new version of the polyhedral minimization method, called the k-minimization. This methods is based on the max plus pruning method [5].

2 Background

2.1 The Sub-polyhedral Abstract Domain Based Support Functions

The sub-polyhedral abstract domain presented in [10] is based on support function [7]. This domain is an abstraction of convex polyhedra over $\mathbb{R}^n$, where n is the space dimension. We denote by $\mathbb{P}^\sharp_\Delta$ the abstract domain using support functions. The lattice definition is closed to the lattice of the Template abstract domain [9]. $\mathbb{P}^\sharp_\Delta$ is parametrize by a finite set of directions $\Delta = \{d_1, \ldots, d_l\}$. The directions in Δ are uniformly distributed on the unit sphere, noted B^n. The definition of $\mathbb{P}^\sharp_\Delta$ is given as follow:

Let $\Delta \subseteq B^n$ be the set of directions. We define $\mathbb{P}^\sharp_\Delta$ as the set of all functions from Δ to $\mathbb{R}_\infty$, i.e. $\mathbb{P}^\sharp_\Delta = \Delta \to \mathbb{R}_\infty$. We denote $\bot_\Delta$ (resp. $\top_\Delta$) the function such that $\forall d \in \Delta,\ \bot_\Delta(d) = -\infty$ (resp. $\top_\Delta(d) = +\infty$).

For each $\Omega \in \mathbb{P}^\sharp_\Delta$, we write $\Omega(d)$ the value of Ω in direction $d \in \Delta$. Intuitively, Ω is a support function with finite domain.

The abstraction and concretization functions of $\mathbb{P}^\sharp_\Delta$ are defined as follows:

Let $\Delta \subseteq B^n$ be the set of directions.

We define the concretization function $\gamma_\Delta : \mathbb{P}^\sharp_\Delta \to \mathbb{P}$ by:

$$\forall \Omega \in \mathbb{P}^\sharp_\Delta,\ \gamma_\Delta(\Omega) = \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n, < x, d > \leq \Omega(d)\} .$$

where, $< x, d >$ is the scalar product of x by the direction d.

The abstraction function $\alpha_\Delta : \mathbb{P} \to \mathbb{P}^\sharp_\Delta$ is defined by:

$$\forall \mathsf{P} \in \mathbb{P}, \alpha_\Delta(\mathsf{P}) = \begin{cases} \bot & \text{if } \mathsf{P} = \emptyset \\ \top & \text{if } \mathsf{P} = \mathbb{R}^n \\ \lambda d.\ \delta_\mathsf{P}(d) & \text{otherwise} \end{cases} .$$

where, $\delta_\mathsf{P}(d)$ is the support function of the polyhedron P in the direction d, and $\mathbb{P}$ represents the polyhedra abstract domain.

Note that, the concretization of an abstract element of $\mathbb{P}^\sharp_\Delta$ is a polyhedron defined by the intersection of half-spaces, where each one is characterized by its normal vector $d \in \Delta$ and the coefficient $\Omega(d)$. The abstraction function on the other side is the restriction of the support function of the polyhedra on the set of directions Δ. The order structure of $\mathbb{P}^\sharp_\Delta$ is defined using properties of support functions [7]. The static analysis of a program consists in computing the least fixed point of a monotone map. To do so, the most used method is the Kleene

Algorithm. By combining the Kleene algorithm and the $\mathbb{P}^{\sharp}_{\Delta}$ abstract domain, the obtained algorithm has a polynomial complexity in the number of iterations and linear in the number of directions in Δ. In addition, its result is as accurate as possible: at each iterate, we have that $\Omega_i = \alpha_{\Delta}(\mathsf{P}_i)$, such that Ω_i (resp. P_i) is the result of the i^{th} Kleene iteration using the $\mathbb{P}^{\sharp}_{\Delta}$ (respectively the polyhedra abstract domain). So $\Omega_{\infty} = \alpha_{\Delta}(\mathsf{P}_{\infty})$, with Ω_{∞} is the fixed point obtained in the $\mathbb{P}^{\sharp}_{\Delta}$ analysis and P_{∞} is the one obtained using polyhedra domain. That why the $\mathbb{P}^{\sharp}_{\Delta}$ analysis is more precise than the Template analysis [9]. In other terms, the $\mathbb{P}^{\sharp}_{\Delta}$ analysis is done with the precision of the polyhedra domain and the over-approximation is done only, at the end, in the concretization function. When with Template domain, all the analysis is done in a less expressive domain.

2.2 The Max Plus Pruning Method

In [5], the authors present a method to reduce the curse of dimensionality in solving an optimal control problem. In this method, the value function is over-approximated using a set of Max-Plus basis functions. The general formulation for the pruning problem appearing in max-plus basis methods is the following: Let $F = \{1, 2, \ldots, m\}$ be a set of integer, and let $g(x) : \mathbb{R}^n \mapsto \mathbb{R}$ be a function defined as follow:

$$g(x) = sup_{i \in F} g_i(x)$$

where $\forall i \in [1, m], g_i(x)$ is a basis function. Let $B = \{g_1, \ldots, g_m\}$ be the set of these m basis functions. Note that, when solving an optimal control problem the cardinality of B can be very large. The idea is to approximate $g(x)$ by keeping only $0 \leq k \leq m$ basis functions from the set B. For that, the authors in [5] need to compute the set $S \subset F$ with cardinality k and then approximate the function g by:

$$g(x) \simeq sup_{i \in S} g_i(x)$$

The obtained set S should minimize the approximation error. This is known as a pruning problem. To solve it, a set of witness points W is used to measure the approximation error, such that: $W = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$, where n is the space dimension. This set is constructed using a random points in the space. Afterwards, $\forall x_i \in W, \forall g_j \in B$ the importance metric is computed, which represents the distance between $g(x_i)$ and $g_j(x_i)$. This is denoted by c_{ij}, such that:

$$c_{ij} = g(x_i) - g_j(x_i)$$

The obtained results represents the cost matrix $C \in \mathbb{R}^m \times \mathbb{R}^m$ (dessiner la matrix). For a better comprehension, let us take the example of Fig. 1. In this example, we want to approximate the smooth convex function g (the red graph) using the basis functions g_0, g_1, g_2 (the green lines). The point x_0 is one witness point. It is used to compute the distance between the function g and all basis functions. The distance between $g(x_0)$ and $g_0(x_0)$ is represented by the red dashed line. In this example, if we want to keep only two of the three basis function. The couple (g_0,g_2) is the best choice.

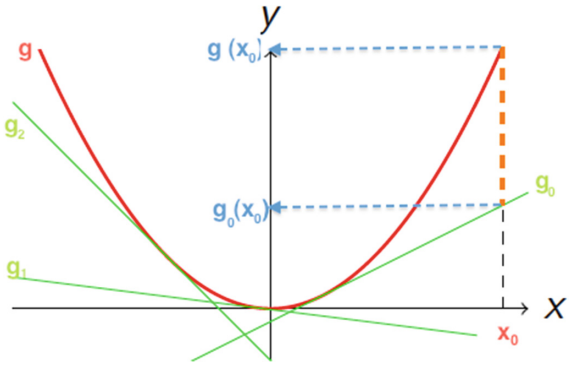

Fig. 1. In this figure, we represent the function g (red graph) and some basis functions (green lines). The dashed red line is the distance between g and g_0 using the point x_0. (Color figure online)

Afterwards, the cost matrix C is used to compute the set $|S| = k$. And that by applying one of the two methods:

– the K-median problem [1]: to minimize the sum of the lost, such that:

$$\min_{S \subset I} \sum_{i=1}^{m} \min_{j \in S} c_{ij}$$

– the K-center problem: to maximize the lost, such that:

$$\min_{S \subset I} \max_{i \in [1,m]} \min_{j \in S} c_{ij}$$

The obtained set S is used to approximate the function g as follow:

$$g(x) \simeq sup_{i \in S} g_i(x)$$

with $g_i \in B$.

3 The k-Minimization Method

In this section, we develop our main contribution which is a novel approach to reduce the complexity of the polyhedron construction and that by reducing the number of constraints, this method is called the *K-minimization*, which is inspired from the Max-Plus pruning method [5].

Let P^m be a polyhedron represented by the intersection of $m \in \mathbb{N}$ half-spaces *i.e.* $\mathsf{P}^m = \bigcap_{i=1}^{m} H_i$ where $H_i = \{x \in \mathbb{R}^n : \langle x, a_i \rangle \leq b_i\}$ with $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. For an m very large, the computation of P^m is time consuming. To improve this computation, we over-approximate P^m by keeping only $k < m$ of their half-spaces, let $\mathsf{P}^k = \bigcap_{i=1}^{k} H_i$ be the resulted polyhedron, such that:

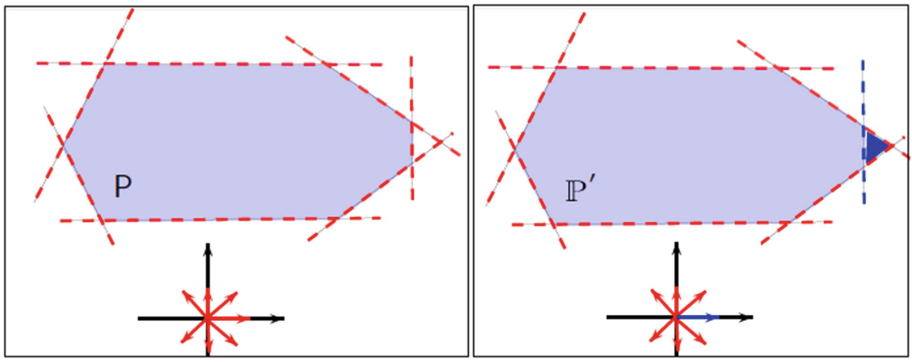

Fig. 2. (The left part) The intersection of the red dashed half spaces defines the polyhedron P. The red directions in bottom represent the normal vectors of the lines those support these half-spaces. (The right part) the polyhedron P′ is an over-approximation of P, that by deleting the dashed blue half-space (Color figure online)

- $\mathsf{P}^k \supseteq \mathsf{P}^m$.
- P^k is the best approximation of P^m using k half-spaces. Note that, the definition of the best approximation strongly depend on the definition of the Hausdorff distance between two polyhedra.

For a better comprehension, let me explain the motivation using the example of Fig. 2. In this figure, the left polyhedron P is defined using the intersection of seven half-spaces. Each half space, is represented by one direction (given in red in the bottom of the left figure). We want to over-approximate P by taking only 6 half-spaces from the 7 one. The resulted polyhedron P′ is given in the right figure. Where, the half-space to delete is the blue dashed one. Noted that, $\mathsf{P} \subseteq \mathsf{P}'$, with P′ is the best approximation of P using only 6 half-spaces from the initial one. This approximation is known as the pruning problem, to be able to solve its automatically, we propose a method called the *K-minimization* method.

The *K-minimization* method is based essentially on three steps:

1 The computation of the witness points.
2 The computation of the cost matrix.
3 The application of the K-median algorithm.

Let us detailed these steps:

The witness points computation: Let $w \subseteq \mathbb{R}^n$ be a set of points, where each point of w belongs to one face of P^m. So, the cardinality of w is equal to m, *i.e.* $|w| = m$. In the following, each half-space H_i will be characterized by its corresponding point x_i in the set w, these points are called *witness points*. Note that, the computation of these points is known as a convex optimization problem and to solve it, we may solve m LP problems. For all $i \in \{1, \ldots, m\}$:

– Solve the following LP problem using the interior point algorithm:

$$\min\langle x, a_i\rangle - b_i$$

$$s.t. : \forall j \in \{1, \ldots, m\} \setminus \{i\}, \langle x, a_j\rangle \leq b_j$$

– Add the obtained point to w the witness point set.

The cost matrix computation. Let $C \in R^{m\times m}$ be a square matrix. We have that $\forall i, j \in [1, m]$ C_{ij} represents the euclidean distance between x_i the i^{th} point in the set w and $proj(x_i, L_j)$ the orthogonal projection of x_i on the plane L_j. This distance is obtained as follows:

$$\begin{aligned} C_{ij} &= \|x_i - proj(x_i, L_j)\| \\ &= \frac{|\langle x_i, a_j\rangle - b_j|}{\|a_j\|}. \end{aligned}$$

So, each line i of the matrix C contains distances between x_i and all the lines that support the faces of P^m. The matrix C is called *the cost matrix*.

The application of the k-median algorithm. To tackle the fact that we want to choose k half-spaces from the m ones and that by minimizing the approximation error, we propose the use of *the k-median algorithm*i [1]. The k-median problem is one of the most studied clustering problem, such that for n points given in a metric space the aim is to identify the $k < n$ ones that minimize the sum of the distance to their nearest points.

In our problem, we want to define the k witness points such that the sum of the distance between these k points and their projections is minimized. For that, we use the cost matrix C to formalize the k-median problem as follows:

$$\min_{S\subset F, |S|=k} \sum_{i=1}^{m} \min_{j\in S} C_{ij}.$$

with $F = \{1, 2, \ldots, m\}$ the set of the witness point indices in w.

Several algorithms are known to solve this problem, and that returns the set S of indices of the witness points in w that minimize the sum of distance. We know that each witness point in w represents one half-space that is used to define P^m. So, the resulted polyhedron P^k is defined as follows:

$$\mathsf{P}^k = \bigcap_{i\in S} H_i.$$

Thus, we have that $\mathsf{P}^m \subseteq \mathsf{P}^k$.

To summarize, the k-minimization algorithm is given in Algorithm 1.

Algorithm 1. The K-minimization algorithm

Require: P^m, $k \in \mathbb{N}$
  $w = witnessPoint(\mathsf{P}^n)$
  for $i = 0$ to $m - 1$ **do**
    for $j = 0$ to $m - 1$ **do**
      $C[i][j]$= Distance$(w[i], proj(w[i], L[j]))$
    end for
  end for
  $S = KmedianAlgo(C, k)$
  return P^k

In Algorithm 1, the set of witness points, noted w, is computed using the function *witnessPoint*. This function uses m LP solver. Then, the euclidean distance is computed between all the points of w and their orthogonal projection on L, where L are the set of planes that support the faces of the polyhedron P^m. These distances are computed using the function *Distance* and the results are putted in the matrix C. Afterwards, the k-median algorithm is applied using the matrix C.

In the case of the $\mathbb{P}^\sharp_\Delta$ analysis, let Ω be the obtained fixed point. Then $\gamma_\Delta(\Omega) = \bigcap_{d\in\Delta}\{x \in \mathbb{R}^n, < x, d > \leq \Omega(d)\}$. This is the concretisation of Ω in the polyhedra abstract domain. Note that, the cardinality of Δ the set of directions can be very large, So the concretisation of Ω can be time consuming. That why, the *K-minimization* method is applied at the end of the $\mathbb{P}^\sharp_\Delta$ analysis to over-approximate the result of $\gamma_\Delta(\Omega)$. The concretisation of Ω is very useful, it allows us to compare our result with the one obtained using the polyhedral analysis. It, also, can be used as the input of another analysis. Recall that the $\mathbb{P}^\sharp_\Delta$ analysis uses a polyhedron as initial input set. The preliminary results are given in the next section.

4 Benchmarks

We implemented the k-minimization algorithm on the top level of the Parma Polyhedra Library (PPL: http://bugseng.com/products/ppl/). We apply it on some programs, which represent digital filters. The obtained results are given in Fig. 3.

At the end of the $\mathbb{P}^\sharp_\Delta$ analysis, we concretise the result in the polyhedra abstract domain. In the table of Fig. 3, we compare the results obtained using the combination of the k-minimization and the $\mathbb{P}^\sharp_\Delta$ analysis [10], with the one obtained using only the $\mathbb{P}^\sharp_\Delta$ analysis. Note that, in the first result we apply the k-minimization method before the application of the concretisation function. Where in the second one, we apply the concretisation function directly on the result of the $\mathbb{P}^\sharp_\Delta$ analysis, and that using all the directions in Δ. The first results are very encouraging, with our method the analysis terminates in some minutes. Where with the standard minimization function the analysis did not terminates,

Program			The $\mathbb{P}^\sharp_\Delta$ analysis + the K-minimization		The $\mathbb{P}^\sharp_\Delta$ analysis
Name	$\|V\|$	$\|\Delta\|$	t(s)	K	t(s)
lead_leg_controller	5	350	$1m53.811s$	116	TO
lp_iir_9600_2	6	372	$3m0.165s$	124	TO
lp_iir_9600_4	10	500	$6m21.984s$	166	TO
lp_iir_9600_4_elliptic	10	500	$6m20.659s$	166	TO
lp_iir_9600_6_elliptic	14	692	$21m56.076s$	230	TO
bs_iir_9600_12000_10_chebyshev	22	1268	TO	422	TO

Fig. 3. The execution time obtained using the k-minimization method and the standard polyhedral minimization

and that after 10 h of executions. Even, with our method, the analysis of the last program did not terminate in a reasonable execution time. The lack of the k-minimization algorithm is the computation of the witness points which is time consuming, because we apply one LP solver per constraints. The improvement of this point is the subject of our ongoing work.

5 Conclusion

In the $\mathbb{P}^\sharp_\Delta$ analysis, The execution time is linear in the cardinality of Δ. So to improve the precision of the analysis, we can take Δ with a large cardinality. The result of this analysis can be concretised in the polyhedra abstract domain, where the number of the constraints of the obtained polyhedron is equal to the cardinality of Δ. So, if we take Δ very large, the computation of the resulted polyhedron can be time consuming. For that, we present in this paper a method called k-minimization. This method can be applied before the concretisation method to reduce its complexity. The k-minimization method is inspired from the Max-Plus Pruning method. This method over-approximate the obtained polyhedron by keeping only k from their half-spaces, where k is chosen statically smaller then the cardinality of Δ. The obtained over-approximation is the one that minimise the loss of precision.

References

1. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM (1999)
2. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference Record of the Fourth ACM Symposium on Principles of Programming Languages (POPL 1977), pp. 238–252. ACM Press (1977)
3. Cousot, P., Cousot, R.: Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In: Bruynooghe, M., Wirsing, M. (eds.) PLILP 1992. LNCS, vol. 631, pp. 269–295. Springer, Heidelberg (1992). doi:10.1007/3-540-55844-6_142

4. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: POPL, pp. 84–97. ACM Press (1978)
5. Gaubert, S., McEneaney, W.M., Qu, Z.: Curse of dimensionality reduction in max-plus based approximation methods: theoretical estimates and improved pruning algorithms. In: CDC-ECE, pp. 1054–1061. IEEE (2011). http://dblp.uni-trier.de/db/conf/cdc/cdc2011.html#GaubertMQ11
6. Goubault, E., Putot, S., Védrine, F.: Modular static analysis with zonotopes. In: Miné, A., Schmidt, D. (eds.) SAS 2012. LNCS, vol. 7460, pp. 24–40. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33125-1_5
7. Hiriart-Urrut, J.B., Lemaréchal, C.: Fundamentals of Convex Analysis. Springer, Heidelberg (2004)
8. Miné, A.: The octagon abstract domain. High. Order Symbolic Comput. **19**(1), 31–100 (2006)
9. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Scalable analysis of linear systems using mathematical programming. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 25–41. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30579-8_2
10. Seladji, Y., Bouissou, O.: Numerical abstract domain using support functions. In: Brat, G., Rungta, N., Venet, A. (eds.) NFM 2013. LNCS, vol. 7871, pp. 155–169. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38088-4_11
11. Simon, A., King, A., Howe, J.M.: Two variables per linear inequality as an abstract domain. In: Leuschel, M. (ed.) LOPSTR 2002. LNCS, vol. 2664, pp. 71–89. Springer, Heidelberg (2003). doi:10.1007/3-540-45013-0_7
12. Venet, A.J.: The gauge domain: scalable analysis of linear inequality invariants. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 139–154. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31424-7_15