Flowpipe-Guard Intersection for Reachability Computations with Support Functions [⋆]

Goran Frehse [∗] **Rajarshi Ray** [∗]

[∗] *Université Grenoble 1, Verimag,*
Centre Equation - 2, Avenue de Vignate, 38610 Gieres, France
(e-mail: goran.frehse,rajarshi.ray@imag.fr).

Abstract: Recently, efficient reachability algorithms for hybrid systems with piecewise affine dynamics have been developed. They achieve good scalability and precision by using support functions to represent continuous sets. In this paper, we propose an improvement of these algorithms that reduces the overapproximation error of the image computation of discrete transitions (jumps). The critical operation of this image computation is the intersection of the flowpipe with the guard sets of the transitions, since intersection is in general a difficult operation when using support functions. We propose an approach for computing the intersection of the flowpipe with polyhedral guards up to arbitrary accuracy. We reduce computing the support function of the intersection of a single convex set with a guard to a convex minimization problem. To solve it, we present a custom-tailored sandwich algorithm. The intersection of a flowpipe (a sequence of convex sets) with a guard reduces to a set of such minimization problems. Where possible, we use branch-and-bound techniques and solve these minimization problems simultaneously to avoid redundant computations. Experimental results illustrate the gain in accuracy and the performance of the algorithms.

Keywords: Hybrid automata, support functions, reachability, convex piecewise linear functions, minimization.

1. INTRODUCTION

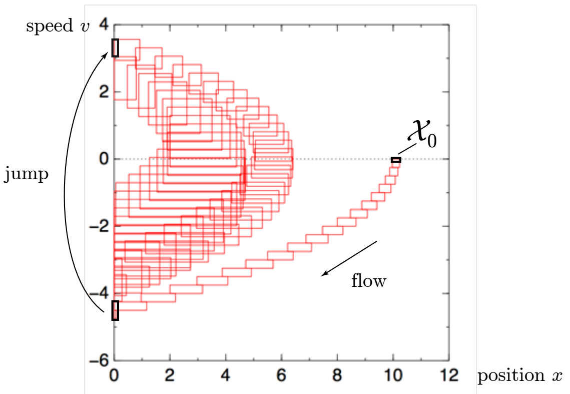
A scalable technique for computing the reachable states for hybrid systems with piecewise affine dynamics was proposed by Le Guernic and Girard [2009] and later improved by Frehse et al. [2011]. Simply put, its efficiency hinges on computing an overapproximation of the reachable states in the form of template polyhedra. Here, template polyhedra are polyhedra whose facet normals consist of user-defined template directions. One of the strong points of the algorithm is the precision of time elapse operator. At each application of the operator, the approximation error can be fixed to an arbitrarily small value and does not accumulate with time if the ODEs are deterministic. While restricting the approximation to few template directions makes this approach scalable, it can result in a large approximation error when computing the image of a discrete transition.

To illustrate this phenomenon, consider the example of a bouncing ball, for whom we compute an overapproximation of the reachable states as shown in Fig. 1(a). Starting from an initial set of states $\mathcal{X}_0$, our algorithm computes a cover of the flowpipe in the form of boxes. As the ball hits the ground at $x = 0$, its state jumps from negative velocity v to positive velocity. For reasons of efficiency, we are bound to compute this jump using the box approximation, which incurs a significant error. This can be seen from the fact that the next flowpipe is significantly larger after the first jump, and even larger after the second jump. Indeed, the states computed after further jumps eventually diverge to infinity. By adding the right template directions, we obtain a much more precise result. Figure 1(b) shows the approximation with automatically synthesized template directions for the first two flowpipes. By adding only a few critical directions, the approximation error induced by the jumps has become negligible.

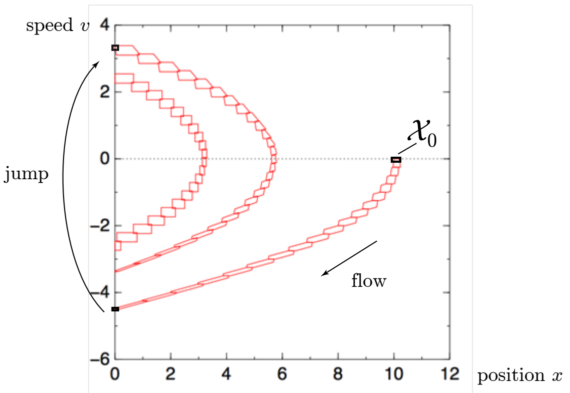
The accuracy of the flowpipe-guard intersection has therefore considerable impact on the quality of the computed reachable set. An approach to more accurately compute the flowpipe-guard intersection for hyperplanar guards was proposed by Le Guernic and Girard [2009], see also Le Guernic [2009, p.114–122]. Their algorithm computes the intersection of a convex set, represented by its support function, with a single hyperplane. This reduces the intersection to the minimization of a unimodal function, for which they propose a dichotomic search and a golden section search algorithm, for more details see Sect. 2.3.

In this paper, we revisit this approach, generalizing it from hyperplanes to halfspaces and polyhedra. Using a different formulation of the problem, we reduce the intersection to minimizing a *convex* function, which allows us to compute the optimality gap and thus obtain a result of guaranteed accuracy. In the setting of our reachability algorithm, the minimization function is the support function of a polyhedral set and therefore piecewise linear. For this class, our minimization algorithm terminates in finite

[⋆] This work was supported in part by the EU project MULTIFORM under grant INFSO-ICT-224249.



(a) Using outer approximations for the intersection with the guard set ($x = 0$), the approximation error increases with each jump, to the point where the computed set diverges when more jumps are added



(b) Our accurate intersection algorithm adds template directions when required by subsequent jump computations. Here, the flowpipe after the second jump has no additional template directions since the third jump has not yet been computed

Figure 1. Reachable state computation for two jumps (discrete transitions) of a bouncing ball, without and with accurate intersection, all other parameters being equal. The user-defined template directions are the axis directions, resulting in a bounding-box overapproximation

number of steps for any desired error bound including zero. Similar to Le Guernic [2009], we use branch-and-bound techniques and solve these minimization problems simultaneously to avoid redundant computations. In the case of polyhedra, the exact solution leads to a multi-dimensional optimization problem. As an alternative with lower complexity, we propose a per-constraint intersection to which branch-and-bound techniques are readily applied.

Our minimization algorithm is similar to sandwich algorithms used in literature mainly for approximation, see Burkard et al. [1991] and references therein. While the litterature on minimizing piecewise linear functions is mostly concerned with convergence properties, our focus is on the result for a very small number of function evaluations, since in our application each function evaluation is rather costly.

In the next section we discuss the problem of intersecting a hyperplane, halfspace or polyhedron with a convex set represented by its support function. We present our minimization algorithm and compare its performance for hyperplane intersection to the golden section search proposed by Le Guernic and Girard [2009]. In Sect. 3, we apply this technique to the flowpipe-guard intersection problem. Since our flowpipe approximation consists generally of a large number of convex sets, we discuss efficiency improvements such as branch-and-bound techniques. The performance of our algorithm is illustrated by experimental results in Sect. 4.

2. INTERSECTING A CONVEX SET WITH A POLYHEDRON USING SUPPORT FUNCTIONS

2.1 Representing Sets with Support Functions

A convex set can be represented by its support function, which attributes to each direction in $\mathbb{R}^n$ the signed distance of the farthest point of the set to the origin in that direction. Computing the value of the support function for a given set of directions, one obtains a polyhedron that overapproximates the set. A *halfspace* $\mathcal{H} \subseteq \mathbb{R}^n$ is the set of points satisfying a linear constraint

$$\mathcal{H} = \{x \mid a^{\mathsf{T}}x \leq b\},$$

where $a = (a_1 \cdots a_n) \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A *polyhedron* $\mathcal{P}$ is the intersection of a finite number of halfspaces

$$\mathcal{P} = \Big\{x \;\Big|\; \bigwedge_{i=1}^{m} a_i^{\mathsf{T}}x \leq b_i\Big\},$$

where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. A *polytope* is a bounded polyhedron. The *convex hull* $\mathrm{CH}(\mathcal{X})$ of a set $\mathcal{X} \subseteq \mathbb{R}^n$ is

$$\mathrm{CH}(\mathcal{X}) = \Big\{\sum_{i=1}^{m} \lambda_i v_i \;\Big|\; v_i \in \mathcal{X}, \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1\Big\}.$$

Given a matrix $M \in \mathbb{R}^{n\times n}$, $M\mathcal{X} = \{Mx \mid x \in \mathcal{X}\}$. The *Minkowski sum* of sets $\mathcal{X}_1$ and $\mathcal{X}_2$ is $\mathcal{X}_1 \oplus \mathcal{X}_2 = \{x_1 + x_2 \mid x_1{\in}\mathcal{X}_1, x_2{\in}\mathcal{X}_2\}$.

The *support function* of a closed and bounded convex set $\mathcal{X} \subseteq \mathbb{R}^n$ attributes to a direction vector $\ell \in \mathbb{R}^n$ the real

$$\rho_{\mathcal{X}}(\ell) = \max\{\ell^{\mathsf{T}}x \mid x \in \mathcal{X}\}.$$

We use the following operations with support functions:

$$\rho_{\mathrm{CH}(\mathcal{X}_1\cup\mathcal{X}_2)}(\ell) = \max\left(\rho_{\mathcal{X}_1}(\ell), \rho_{\mathcal{X}_2}(\ell)\right), \tag{1}$$

$$\rho_{M\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(M^{\mathsf{T}}\ell), \tag{2}$$

$$\rho_{\mathcal{X}_1\oplus\mathcal{X}_2}(\ell) = \rho_{\mathcal{X}_1}(\ell) + \rho_{\mathcal{X}_2}(\ell). \tag{3}$$

Given a set of directions $L = \{\ell_1, \ldots, \ell_K\}$, we have the polyhedral *outer approximation*

$$\lceil\mathcal{X}\rceil_L = \Big\{x \;\Big|\; \bigwedge_{k=1,\ldots,K} \ell_k^{\mathsf{T}}x \leq \rho_{\mathcal{X}}(\ell_k)\Big\}. \tag{4}$$

Lemma 1. For all L, $\mathcal{X} \subseteq \lceil X\rceil_L$. Given a polyhedron $\mathcal{P}$ and L such that $\{a_1, \ldots, a_m\} \subseteq L$, $\mathcal{P} = \lceil\mathcal{P}\rceil_L$.

When we consider the set L as fixed and given, we speak of *template directions*, and call the outer approximation in those directions the *template hull*. The template hull of a set of convex sets is the template hull of their convex hull. It is easy to compute by applying (1).

2.2 Intersection with a Halfspace or Hyperplane

We now consider the intersection of a closed and bounded convex set $\mathcal{X}$ with a halfspace or a hyperplane. Later we

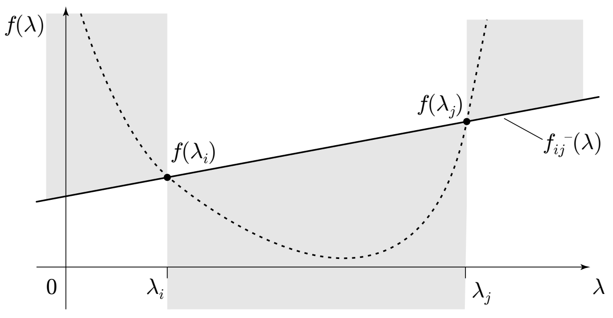


Figure 2. The straight line through two points on a convex function $f(\lambda)$ is a lower bound on $f(\lambda)$ to the left and to the right of those two points

extend these results to polyhedra. As noted by Le Guernic [2009], the support function of the intersection of compact convex sets $\mathcal{X}$ and $\mathcal{Y}$ can be reduced to the minimization problem

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) = \inf_{v\in\mathbb{R}^n} (\rho_{\mathcal{X}}(\ell - v) + \rho_{\mathcal{Y}}(v)). \tag{5}$$

If $\mathcal{Y}$ is a halfspace or a hyperplane, we show that (5) simplifies to a univariate minimization problem as follows.

Lemma 2. Consider the halfspace $\mathcal{H} = \{x \mid a^\intercal x \le b\}$ and the hyperplane $\mathcal{H}' = \{x \mid a^\intercal x = b\}$, and let

$$f(\lambda) = \rho_{\mathcal{X}}(\ell - \lambda a) + \lambda b. \tag{6}$$

Then we have

$$\rho_{\mathcal{X}\cap\mathcal{H}}(\ell) = \inf_{\lambda\in\mathbb{R}^{\ge 0}} f(\lambda), \quad \rho_{\mathcal{X}\cap\mathcal{H}'}(\ell) = \inf_{\lambda\in\mathbb{R}} f(\lambda). \tag{7}$$

Note that $f(\lambda)$ is convex, since every support function is convex and the sum of two convex functions is convex. If $\mathcal{X}$ is a polyhedron, (7) is a parametric linear program (LP), with λ as parameter. Consequently, $f(\lambda)$ is continuous, convex, piecewise linear function, see Dantzig and Thapa [2003].

Lemma 3. We note the following facts about the intersection with a halfspace $\mathcal{H}$:

(1) $\mathcal{X} \cap \mathcal{H} = \emptyset$ iff $-\rho_X(-a) > b$.
(2) If $\rho_{\mathcal{X}}(a) \le b$, then $\rho_{\mathcal{X}\cap\mathcal{H}}(\ell) = \rho_{\mathcal{X}}(\ell)$.
(3) $f(\lambda) \to -\infty$ as $\lambda \to \infty$ iff $\mathcal{X} \cap \mathcal{H} = \emptyset$.
(4) If $\mathcal{X} \cap \mathcal{H} \ne \emptyset$, then $f(\lambda) \ge -\rho_{\mathcal{X}}(-\ell)$.

2.3 A Sandwich Algorithm for the Direct Minimization of Convex Functions

We have the following sandwich algorithm to find a sequence of λ_i that converges towards the minimum of $f(\lambda)$. For each λ_i, we compute the corresponding value $f(\lambda_i)$, and we call $(\lambda_i, f(\lambda_i))$ a *sample* of $f(\lambda)$. We iteratively compute a function $f^-(\lambda)$ that is a lower bound on $f(\lambda)$, updating it with each newly computed sample.

The source of our lower bound function is the following property of convex functions, which is illustrated by Fig. 2. Given two samples $(\lambda_i, f(\lambda_i))$ and $(\lambda_j, f(\lambda_j))$, $\lambda_i < \lambda_j$, the convexity of $f(\lambda)$ implies that the straight line through them, described by

$$f_{ij}^-(\lambda) = \frac{f(\lambda_j) - f(\lambda_i)}{\lambda_j - \lambda_i}(\lambda - \lambda_i) + f(\lambda_i), \tag{8}$$

is a lower bound on $f(\lambda)$ to the left and right of the two points, i.e., for all $\lambda \le \lambda_i$ and $\lambda \ge \lambda_j$, and an upper bound between them, i.e., for $\lambda_i \le \lambda \le \lambda_j$. We combine (8) for all known samples $(\lambda_i, f(\lambda_i))$ to the following lower bound function, which is defined pointwise over λ:

$$f^-(\lambda) = \max\Big(-\infty, \max_{\lambda\le\lambda_i<\lambda_j} f_{ij}^-(\lambda), \max_{\lambda_i<\lambda_j\le\lambda} f_{ij}^-(\lambda)\Big). \tag{9}$$

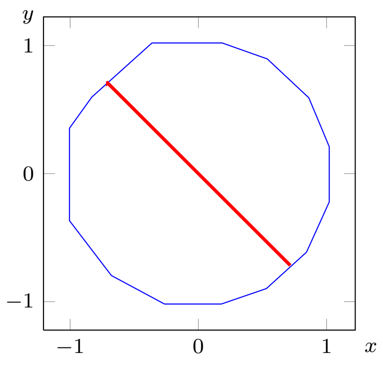
Given an error threshold $\varepsilon \ge 0$, our algorithm computes an interval $[r_-, r_+]$ such that

$$r_- \le \min_{\lambda\in\mathbb{R}^{\ge 0}} f(\lambda) \le r_+ \quad \text{and} \quad r_+ - r_- \le \varepsilon.$$

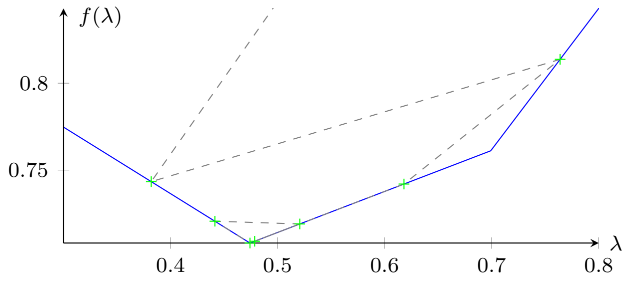
Since the algorithm chooses the next sample based on the current lower bound, we call it *Lower Bound search.* It proceeds as follows, see also Fig. 3:

(1) Let $i = 0$, $\lambda_i = 0$, $r_- = -\infty$, $r_+ = +\infty$.
(2) Bracket the minimum by adding samples until a turning point is found:
    (a) Until $f(\lambda_{i-1}) \le f(\lambda_{i-2})$ and $f(\lambda_{i-1}) \le f(\lambda_i)$, increase the distance between λ_i exponentially.
(3) Compute $f(\lambda_i)$ and tighten the interval bounds $r_- \leftarrow \inf_{\lambda\in\mathbb{R}^{\ge 0}} f^-(\lambda)$, $r_+ \leftarrow \min(r_+, f(\lambda_i))$
(4) Choose the next sample at the lowest point of $f^-(\lambda)$ unless already visited:
    (a) Let $\lambda_{i+1} \leftarrow \operatorname{arginf}_{\lambda\in\mathbb{R}^{\ge 0}}(f^-(\lambda))$.
    (b) If $\lambda_{i+1} \in \{\lambda_0, \ldots, \lambda_i\}$, let $\lambda_{i+1} \leftarrow (\lambda_{i+1} + \lambda_j)/2$, where λ_j is an appropriate neighboring sample.
(5) If $r_+ - r_- > \varepsilon$, let $i \leftarrow i + 1$ and go to (3). Otherwise terminate and return the interval $[r_-, r_+]$.

If $f(\lambda)$ is piecewise linear with a finite number of pieces, the above algorithm terminates with a finite number of steps for every $\varepsilon \ge 0$. This is the case if $\mathcal{X}$ is a polytope.



(a) The polytope $\mathcal{P}$ and its intersection with the hyperplane $\mathcal{H}'$



(b) To compute $\rho_{\mathcal{X}\cap\mathcal{H}'}(\ell)$, we minimize $f(\lambda)$. The function and the samples chosen by the Lower Bound search are shown for $\ell = (x = 0, y = 1)$

Figure 3. Intersection of the hyperplane $\mathcal{H}' = \{x + y = 0\}$ with a polytope $\mathcal{P}$ with 15 facets

Table 1. Average performance of Lower Bound search (exact solution) vs GSPD (fixed to 14 function evaluations), intersecting a hyperplane with a polytope

	Lower Bound			GSPD		
facets	samples	err	time(ms)	samples	err$\times 10^{-4}$	time(ms)
4	6.741	0	0.15	14	8.197	0.71
8	8.523	0	0.34	14	3.200	0.82
16	9.611	0	0.50	14	1.612	1.27
24	10.222	0	0.74	14	1.111	1.80

Table 2. Average performance of Lower Bound search vs GSPD, intersecting a hyperplane with a polytope for a fixed number of function evaluations (6 samples)

	Lower Bound			GSPD	
facets	opt. gap	err	time(ms)	err	time(ms)
4	0.0338614	0.0285933	0.16	0.0351516	0.25
8	0.0274455	0.0107857	0.10	0.0228235	0.32
16	0.0298651	0.0063944	0.28	0.0156476	0.51
24	0.0302147	0.0044049	0.47	0.0131288	0.98

Our implementation contains further steps to reduce the number of samples and to account for floating point errors. We refer the reader for further details to Ray and Frehse [2011].

Related Work: To the best of our knowledge, this is the first published solution of the intersection with a halfspace or polyhedron. It is derived from previous work on the intersection with a hyper*plane* by Le Guernic and Girard [2009]. There, computing the support function of the intersection is reduced to a univariate minimization problem that is derived geometrically. Its parameter $\theta \in (0,\pi)$ describes the angle between the sample direction and the normal vector of the hyperplane $\mathcal{H}' = \{x \mid a^{\intercal}x = b\}$:

$$\rho_{\mathcal{X}\cap\mathcal{H}'}(\ell) = \inf_{\theta\in(0,\pi)} \frac{\rho_{\mathcal{X}}(\ell\sin\theta + a\cos\theta) - b\cos\theta}{\sin\theta}. \qquad (10)$$

While (10) has the advantage over (7) that its argument ranges over a finite interval, its cost function is only known to be unimodal instead of convex. We refer to the approximate solution of (10) by golden section search as *Golden Section Search in the Polar Domain* (GSPD).

Experiments: The following experiments shall illustrate the performance of Lower Bound search in comparison with GSPD. The results in this paper were obtained on a standard x86 machine with 32bit operating system. To measure the computation error

$$err = r^{+} - \inf_{\lambda\in\mathbb{R}} f(\lambda),$$

we compare the different output values to the result of the Lower Bound search for $\varepsilon = 0$.

Table 1 compares the support function computation of the intersection between a regular n-polyhedron in two dimensions with the line $x\cos\theta + y\sin\theta = 0$ in the direction $[x = 0, y = 1]$. It shows the averaged results for 1000 uniformly distributed $\theta \in [0,\pi]$ by GSPD and our Lower Bound search. Note that the error of GSPD decreases as the number of facets increases. The reason is that the function becomes flatter near the minimum for a larger number of facets. Hence for a fixed interval in the function domain bracketing the minimum, the difference between the minimum and the upper bound decreases.

Table 2 compares the intersection between a regular n-polyhedron with the line $x\cos\theta + y\sin\theta = 0$ in the direction $[x = 0, y = 1]$. It shows the averaged results for 1000 uniformly distributed $\theta \in [0,\pi]$, where the number of samples (function evaluations) has been fixed to 6.

Remark 1. In Table 2 the computation times differ even though the same number of samples is computed for both LB search and GSPD. Indeed the computation time of a sample is data as well as state dependent. In particular, the LP solver computing the support function keeps its state between calls. A sample can therefore be computed faster if its optimal solution for the corresponding direction is close to the one computed in the previous call.

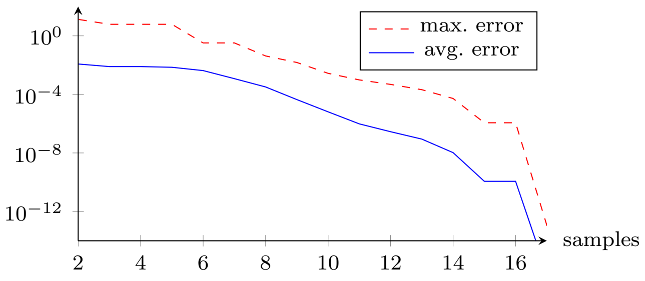


Figure 4. Approximation error over the number of samples for the intersection of random halfspaces with random polytopes with 16 facets

Figure 4 shows the approximation error of the intersection with a halfspace as a function of the number of samples taken. We measure the absolute error over 10000 random instances of a polytope with 16 facets intersected with a halfspace. The polytope and the intersection are by construction nonempty and the halfspace is nonredundant. After 17 samples, both maximum and average error are below 10^{-13}, which is about as close as we expect given machine precision.

2.4 Intersection with a Polyhedron

Since a polyhedron is an intersection of halfspaces, we can apply (7) repeatedly to obtain the support function of the intersection of a set $\mathcal{X}$ with a polyhedron $\mathcal{P}$:

Lemma 4.

$$\rho_{\mathcal{X}\cap\mathcal{P}}(\ell) = \inf_{\lambda\in\mathbb{R}^m,\lambda\geq 0} \rho_{\mathcal{X}}(\ell - \sum_i \lambda_i a_i) + \sum_i \lambda_i b_i. \qquad (11)$$

This is a convex minimization problem over m variables, where m is the number of constraints in $\mathcal{P}$.

In our implementation, we compute the intersection with each halfspace separately, which allows us to apply the results from the previous section. We intersect $\mathcal{X}$ with each halfspace of $\mathcal{P}$ separately, and combine the results in the approximation

$$\rho_{\mathcal{X}\cap\mathcal{P}}(\ell)^{+} = \min_{i=1,\ldots,m} \rho_{\mathcal{X}\cap\{a_i^{\intercal}x\leq b_i\}}(\ell). \qquad (12)$$

Since $\mathcal{X}\cap\mathcal{P}$ is contained in all of the sets $\mathcal{X}\cap\{a_i^{\intercal}x \leq b_i\}$,

$$\rho_{\mathcal{X}\cap\mathcal{P}}(\ell) \le \rho_{\mathcal{X}\cap\{a_i^{\mathsf{T}}x\le b_i\}}(\ell).$$

Consequently, $\rho_{\mathcal{X}\cap\mathcal{P}}(\ell) \le \rho_{\mathcal{X}\cap\mathcal{P}}(\ell)^+$, so we are sure to obtain an overapproximation. An outer approximation computed with $\rho_{\mathcal{X}\cap\mathcal{P}}(\ell)^+$ may be non-empty even though $\mathcal{X}\cap\mathcal{P}$ is empty.

3. FLOWPIPE-GUARD INTERSECTION

We now apply the results from the previous section in the reachability computation of a hybrid system. Since the details of the reachability algorithm have been reported elsewhere and are not essential for the results of this paper, we provide a brief summary and refer the reader to Frehse and Ray [2009], Frehse et al. [2011].

3.1 Hybrid Automata and Reachability

We consider hybrid systems modeled according to Alur et al. [1995] by a *hybrid automaton*

$$H = (Loc, Inv, Flow, Trans, Init).$$

It has a set of discrete states *Loc* called *locations*. Each $l \in Loc$ is associated with a set of differential equations (or inclusions) $Flow(l)$ that defines the time-driven evolution of the continuous variables. A *state* $s \in Loc \times \mathbb{R}^n$ consists of a location and values for the n continuous variables. A set of *discrete transitions Trans* defines how the state can jump between locations and instantaneously modify the values of continuous variables. A jump can take place when the state is inside the transition's *guard* set, and the target states are given by the transition's *assignment*. The system can remain in a location l while the state is inside the *invariant* set $Inv(l)$. All behavior originates from the set of *initial states Init*.

In this paper, we consider $Flow(l)$ to be continuous dynamics of the form

$$\dot{x}(t) = Ax(t) + u(t), \qquad u(t) \in \mathcal{U}, \tag{13}$$

where $x(t) \in \mathbb{R}^n$ is an n-dimensional vector, $A \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^n$ is a closed and bounded convex set. Transition assignments are of the form

$$x' = Rx + w, \qquad w \in \mathcal{W}, \tag{14}$$

where $x' \in \mathbb{R}^m$ the values after the transition, $R \in \mathbb{R}^m \times \mathbb{R}^n$ is the assignment map, and $\mathcal{W} \subseteq \mathbb{R}^n$ is a closed and bounded convex set of non-deterministic inputs.

We compute the reachable states by recursively computing the image of the initial states with respect to time elapse and discrete transitions until a fixpoint is reached. In the next two sections, we discuss how we compute both images.

3.2 Image Computation of Time Elapse

In a given location, the states reachable from an initial set $\mathcal{X}_0$ by time elapse are referred to as the *flowpipe* of $\mathcal{X}_0$. We use the approximation reported by Frehse et al. [2011] and compute a sequence of closed and bounded convex sets $\Omega_0, \ldots, \Omega_N$ that covers the flowpipe starting from $\mathcal{X}$, with convex inputs $\mathcal{U}$. We denote this operation by

$$(\Omega_0, \ldots, \Omega_N) = post_c\,(\mathcal{X},\mathcal{U})\,. \tag{15}$$

Each Ω_i is the result of convex hull and Minkowski sum operations on polytopes. So Ω_i is by construction a polytope, but such that computing its constraint representation would be prohibitively expensive. Its support function can, however, be computed efficiently for any given direction. The computation of the sequence Ω_i amounts to a symbolic integration of the ODE (13), so the values of Ω_i depend on the values of Ω_{i-1}, etc. This gives us the following limitation, which will become an important when we consider intersections:

Assumption 1. To compute $\rho_{\Omega_i}(\ell)$, we also need to compute $\rho_{\Omega_j}(\ell)$ for $j = 0, \ldots, i-1$.

There is a partial remedy to this problem. Consider the case where we are interested in computing a subsequence of the flowpipe approximation Ω_i, say for $i \in [c,d]$. This could be, e.g., sets that may take a transition. Under Assumption 1 this requires us to compute the $d+1$ sets with $i \in [0,d]$. We can reduce this computational burden as follows. The sequence Ω_i is constructed such that each set covers the flowpipe over a known time interval $[t_i, t_{i+1}]$. We decompose the system into its autonomous dynamics ($\mathcal{U} = \emptyset$) and its input dynamics ($\mathcal{X} = \emptyset$). Recall that for autonomous dynamics, the set of states reached at exactly time t_c is $\mathcal{X}_{t_c} = e^{At_c}\mathcal{X}$. Starting the flowpipe computation for the autonomous dynamics from $t = t_c$ instead of $t = 0$, we end up with fewer sets to compute. Let

$$(\Omega^x_c, \ldots, \Omega^x_d) = post_c\left(e^{At_c}\mathcal{X}, \emptyset\right), \tag{16}$$

$$(\Omega^u_0, \ldots, \Omega^u_c, \ldots, \Omega^u_d) = post_c\,(\emptyset, \mathcal{U})\,, \tag{17}$$

such that Ω^x_i and Ω^u_i cover the respective flowpipe on the same time interval $[t_i, t_{i+1}]$. Then using the superposition principle we have that $\Omega^x_i \oplus \Omega^u_i$ covers the flowpipe of $\mathcal{X}$ and $\mathcal{U}$ on the time interval $[t_i, t_{i+1}]$. This means we only need to compute the $d-c+1$ sets of (16). While (17) still requires the computation of $d+1$ sets, the set $\mathcal{U}$ is in practice often simple, e.g., a hyperbox, so that its support function can be computed much quicker than that of $\mathcal{X}$.

3.3 Image Computation of Discrete Transitions

The main concern of this paper is to efficiently and accurately compute the image of the discrete transitions. Let $\mathcal{G}$ be the guard set of the transition, $\mathcal{I}^-$ the invariant of the source location, $\mathcal{I}^+$ the invariant of the target location, and let the transition assignment be (14). The image of a set $\mathcal{X}$ with respect to the transition is

$$post_d\,(\mathcal{X}) = \big(R\big(\mathcal{X}\cap\mathcal{G}\cap\mathcal{I}^-\big)\oplus\mathcal{W}\big)\cap\mathcal{I}^+. \tag{18}$$

We assume $\mathcal{G}$,$\mathcal{I}^-$,$\mathcal{I}^+$ to be polyhedra and assume that the set of template directions L contains the normal vectors of the constraints of these polyhedra. To make the intersection of the support function object $\mathcal{X}$ and $\mathcal{G}$,$\mathcal{I}^-$,$\mathcal{I}^+$ scalable, one can compute the outer approximation before the intersection operation, as in Frehse et al. [2011]. For lack of a better term, we call this the *standard* discrete image operator in this paper:

$$post^{std}_d\,(\mathcal{X}) = \big\lceil R\big(\lceil\mathcal{X}\rceil_L\cap\mathcal{G}\cap\mathcal{I}^-\big)\oplus\mathcal{W}\big\rceil_L\cap\mathcal{I}^+. \tag{19}$$

Since all operators that make up $post_d\,(\mathcal{X})$ are monotone,

$$post_d\,(\mathcal{X}) \subseteq post^{std}_d\,(\mathcal{X})\,. \tag{20}$$

Note that if R is invertible and $\mathcal{W}$ is deterministic (a point), the outermost outer approximation is not necessary since the resulting polyhedron can be computed efficiently with exact methods. With the intersection operator pro-

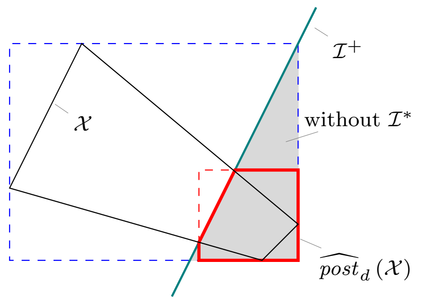


Figure 5. The image of $\mathcal{X}$ using the approximation operator (23), with the axis directions as template directions. Here, $\mathcal{I}^- = \mathcal{G} = \mathbb{R}^n, R = I, \mathcal{W} = 0$, so $\mathcal{I}^+ = \mathcal{I}^*$. Due to the intersection with the pre-image of the target invariant, $\mathcal{I}^*$, the result of (23) (shown in thick red) is considerably more accurate than the same approximation without $\mathcal{I}^*$ (shown shaded gray)

posed in this paper, we aim at increasing the precision by computing instead of $post_d^{std}(\mathcal{X})$

$$\lceil R(\mathcal{X} \cap \mathcal{G} \cap \mathcal{I}^-) \oplus \mathcal{W} \rceil_L \cap \mathcal{I}^+. \tag{21}$$

To further improve the accuracy of this approximation, we include the pre-image of the target invariant as follows. This can lead to substantial improvements, as shown in Fig. 5. Let the target invariant be

$$\mathcal{I}^+ = \Big\{ x \;\Big|\; \bigwedge_{i=1}^{m} \bar{a}_i^\intercal x \le \bar{b}_i \Big\}.$$

An overapproximation of the pre-image of $\mathcal{I}^+$ with respect to (14) is given by

$$\mathcal{I}^* = \Big\{ x \;\Big|\; \bigwedge_{i=1}^{m} \bar{a}_i^\intercal R x \le \bar{b}_i + \rho_{\mathcal{W}}(-\bar{a}_i) \Big\}. \tag{22}$$

Lemma 5. $(R\mathcal{X} \oplus \mathcal{W}) \cap \mathcal{I}^+ \subseteq R(\mathcal{X} \cap \mathcal{I}^*) \oplus \mathcal{W}$.
Equality holds if $\mathcal{W} = \{w\}$.

We obtain our image operator

$$\widehat{post}_d(\mathcal{X}) = \lceil R(\mathcal{X} \cap \mathcal{G} \cap \mathcal{I}^- \cap \mathcal{I}^*) \oplus \mathcal{W} \rceil_L \cap \mathcal{I}^+. \tag{23}$$

With Lemmas 1 and 5, it is straightforward to show that this is a tight overapproximation in the following sense:

Lemma 6. $post_d(\mathcal{X}) \subseteq \widehat{post}_d(\mathcal{X})$.
If $\mathcal{W} = \{w\}$, then $\widehat{post}_d(\mathcal{X}) = \lceil post_d(\mathcal{X}) \rceil_L \cap \mathcal{I}^+$.

Note that $\mathcal{G}, I^-, \mathcal{I}^*$ frequently contain redundant constraints and have matching inequalities that can be simplified to equality constraints. Let $\mathcal{G}^* = \mathcal{G} \cap \mathcal{I}^- \cap \mathcal{I}^*$ be simplified this way. The result of the operator (23) is a polyhedral outer approximation. Recalling its definition from (4), it involves computing for each $\ell \in L$ the support

$$\rho_{R(\mathcal{X} \cap \mathcal{G}^*) \oplus \mathcal{W}}(\ell) = \rho_{\mathcal{X} \cap \mathcal{G}^*}(R^\intercal \ell) + \rho_{\mathcal{W}}(\ell), \tag{24}$$

which we obtain exactly or approximately through minimization as in Sect 2.4. In the next section, we discuss how to do this efficiently for flowpipes.

3.4 Intersecting a Flowpipe with a Halfspace/Hyperplane

In our reachability algorithm, we need to apply the discrete image operator from the previous section to the flowpipe approximation $\Omega_0, \ldots, \Omega_N$, where N is possibly very large. For now we assume that the invariants and the guard are such that $\mathcal{G}^* = \mathcal{G} \cap \mathcal{I}^- \cap \mathcal{I}^*$ is the halfspace $\{\bar{a}^\intercal x \le \bar{b}\}$. The extension to hyperplanes is straightforward, as only the domain of the parameter λ changes in (7).

According to (23), the image is nonempty only for Ω_i where $\Omega_i \cap \mathcal{G}^*$ is nonempty. So with Lemma 3, we can limit ourselves to Ω_i with indices in

$$I_{jump} = \{ i \mid -\rho_{\Omega_i}(-\bar{a}) \le \bar{b} \}. \tag{25}$$

The result of the discrete image computation is

$$\bigcup_{i \in I} \widehat{post}_d(\Omega_i). \tag{26}$$

According to (24), we need to compute for each $\widehat{post}_d(\Omega_i)$ the support $\rho_{\Omega_i \cap \mathcal{G}^*}(\ell)$ for all $\ell \in R^\intercal L$. Applying the approach from Sect 2.4, this involves minimizing for each Ω_i an instance of (6), i.e.,

$$f^i(\lambda) = \rho_{\Omega_i}(\ell - \lambda \bar{a}) + \lambda \bar{b}. \tag{27}$$

Recall that according to Assumption 1, computing $\rho_{\Omega_i}(\ell)$ requires computing $\rho_{\Omega_j}(\ell)$ for $j = 0, \ldots, i-1$. Therefore running a minimization algorithm on $f^i(\lambda)$ also produces samples of $f^j(\lambda)$, $j = 0, \ldots, i-1$. These samples can be used to improve the estimate of the minimum of $f^j(\lambda)$. In addition, one can aim at choosing the next λ such that as many of these estimates as possible benefit from the new sample.

Our algorithm proceeds as follows to compute the min $f^i(\lambda)$ up to a given error $\varepsilon \ge 0$:

(1) We start with a work list $I_{work} = I_{jump}$, and a first sample at $\lambda_{next} = 0$.
(2) Compute $\rho_{\Omega_i}(\ell - \lambda_{next}\bar{a})$ for $i = 0, \ldots, \max(I_{work})$.
(3) Update bounds on minima and requests of next λ for each $f^i(\lambda)$: $(r_-^i, r_+^i, \lambda_{next}^i)$ for $i \in I_{work}$
(4) $\lambda_{next} = select(\{\lambda_{next}^i\}_i)$
(5) Keep only problems on work list whose bounds exceed error: $I_{work} = \{ i \mid r_+^i - r_-^i > \varepsilon \}$.
(6) If $I_{work} \neq \emptyset$, go to (2).

The output of the algorithm are the r_+^i, i.e., an upper bound on the minimum for each $f^i(\lambda)$. The funtion *select* chooses one of the candidates λ_{next}^i for the next sampling point. We consider picking the first, the last, the median and a random candidate.

Clustering: Computing the one-to-one image of the sets covering the flowpipe, as in (26), can have the devastating effect of increasing the number of convex sets exponentially with the search depth. To avoid an explosion in the number of sets and gain efficiency, we compute the convex hull of subsets of these sets instead. This is referred to as convex hull *clustering*, for details see Frehse et al. [2011]. We now discuss how to compute the support function of these clusters efficiently using a branch-and-bound approach.

Let $I_0, \ldots, I_K$ be the maximal connected subsets of I_{jump} i.e., each I_j is the set of indices of a connected subsequence of Ω_i that can take the transition. We compute the outer approximation of the convex hull of these sets,

$$\mathcal{Y}_k = \mathrm{CH}\big(\textstyle\bigcup_{i \in I_k} \Omega_i\big). \tag{28}$$

The final result is the outer approximation of the image of the discrete transition,

$$\mathcal{Z}_k = \widehat{post}_d(\mathcal{Y}_k). \tag{29}$$

With (1),(7),(24) and (27), this requires computing for all directions $\ell \in R^{\intercal}L$

$$\rho_{\mathcal{Y}_k}(\ell) = \max_{i \in I_k} \inf_{\lambda \in \mathbb{R}^{\geq 0}} f^i(\lambda). \quad (30)$$

Similarly to above, we accelerate the computation of (30) by solving the minimization problems for $i \in I_k$ in parallel, and applying the following improvements:

- updating the estimates for all $f^i(\lambda)$ with every sample, as warranted by Assumption 1,
- using a branch-and-bound algorithm to eliminate the f^i for which the upper bound is lower than the currently largest lower bound.

We provide experimental comparison of the different techniques in Sect. 4.

3.5 Intersecting a Flowpipe with a Polyhedron

The results from the previous section can be used to compute the intersection of a flowpipe with a polyhedron. To trade accuracy against performance, we follow the lines of Sect. 2.4 and intersect with each halfspace of the polyhedron separately. Our goal is to intersect the convex hull of the flowpipe Ω_i in the kth interval I_k with the set

$$\mathcal{G}^* = \Big\{ x \;\Big|\; \bigwedge_{j=1}^{m} \bar{a}_j^{\intercal} x \leq \bar{b}_j \Big\},$$

similar to the intersection with a single halfspace in (30). For the intersection of Ω_i the jth halfspace in $\mathcal{G}^*$, we must minimize the function

$$f_j^i(\lambda) = \rho_{\Omega_i}(\ell - \lambda \bar{a}_j) + \lambda \bar{b}_j. \quad (31)$$

Applying the same approximation for polyhedron intersection as in (12), we obtain the overapproximation

$$\rho_{\mathcal{Y}_k}(\ell) \leq \max_{i \in I_k} \min_{j=1,\ldots,m} \inf_{\lambda \in \mathbb{R}^{\geq 0}} f_j^i(\lambda). \quad (32)$$

As with (30), we can use a branch-and-bound algorithm to eliminate instances of i, j for which the upper bound of $f_j^i(\lambda)$ is lower than the largest lower bound.

4. EXPERIMENTAL RESULTS

Two benchmarks shall illustrate the performance and precision of the proposed algorithms.

Timed bouncing ball: We take advantage of the simplicity of the timed bouncing ball to compare accuracy and speed of the different intersection variants. The timed bouncing ball has the state variables position x, velocity v, and time t. Its hybrid automaton model consists of a single location with invariant $x \geq 0$ and continuous dynamics

$$\dot{x} = v, \qquad \dot{v} = -g, \qquad \dot{t} = 1,$$

where g is the gravitational constant (here normed to 1). A discrete transition from the location to itself changes the sign of the velocity when the ball touches the ground. Here we chose the guard constraints $x \leq 0$ and $v < 0$ (the latter to keep the velocity from flipping when the ball goes upwards), and the assignment

$$x' = x, \qquad v' = -cv, \qquad t' = t,$$

where c is a constant for the loss of speed (here 0.75). The initial states are

$$10 \leq x \leq 10.2, v = 0, t = 0.$$

Table 3. Speed versus accuracy comparison of different variants of the discrete image computation, applied to the timed bouncing ball example. The accuracy is measured in x at the highest point of the last jump

direction	ε	clustering	runtime(s)	err$_x$
standard discrete image computation				
box		TH$^+$	1.3	2.480
box		TH&CH$^+$	2.6	1.635
box		CH$^+$	31.2	1.442
oct		TH$^+$	3.0	0.398
oct		TH&CH$^+$	12.7	0.327
oct		CH$^+$	36.6	0.270
discrete image with precise intersection				
box	0.0	TH$^+$	1.3	0.506
box	0.0	TH&CH$^+$	3.4	0.443
box	0.0	CH$^+$	55.9	0.330
precise intersection of convex hull with branch & bound				
box	1.0	CH$^-$	0.8	0.601
box	0.1	CH$^-$	0.6	0.241
box	0.0	CH$^-$	0.6	0.233

Depending on the technique used, we cluster the convex sets either before or after we compute the image of the discrete transition. This is indicated in the clustering column of Table 3 by (-) and (+), respectively. We consider as alternatives the template hull of all sets (TH), the convex hull of all sets (CH), and a mix of both (template hull of about 30%, then convex hull). These alternatives have different speed/accuracy trade-offs. Table 3 shows the performance results for computing the reachable states over five jumps. The error threshold used in our Lower Bound Search is indicated by ε. As a measure of accuracy, we take the difference of the height of the last jump with the accurate value, which was obtained manually. The time step is fixed at $\delta = 0.01$, i.e., each convex set Ω_i in the flowpipe approximations covers the flowpipe over the time span $[t_i, t_i + \delta]$.

The standard variant applies the outer approximation before the intersection, as in (19). Using box directions, the error is so large that the flowpipe of the 5th jump is reaches higher than the 4th, and adding further jumps the computed sets diverge to infinity. Using octagonal directions improves the precision, but slows down the analysis as 18 directions have to be computed instead of 6. Note that the convex hull clustering for octagonal directions is not much slower than for box directions because fewer sets intersect. But even with octagonal directions and very small time steps, the precision leaves to be desired.

The precise variant of the image computation consists of the image operator (23) using Lower Bound search with error bound ε. It shows better accuracy than the standard variant, but convex hull clustering comes at a loss in speed. As we do not detect a significant gain in speed from increasing the error bound, only results for $\varepsilon = 0$ are shown.

The precise branch & bound variant shows both the highest accuracy and the greatest speed. It uses the convex approximation of (23) indicated in (30). Its major gain in performance comes from applying (16) and (17), which reduces the number of sets in the computation. Increasing

Table 4. Speed versus accuracy comparison of different variants of the discrete image computation, for computing a fixed-point of the filtered oscillator example. The accuracy shows in the max amplitude of the output signal z

vars	δ	ε	clustering	runtime(s)	max. z	iter
standard discrete image computation						
6	0.01		TH$^+$	0.3	0.570	5
18	0.01		TH$^+$	2.1	0.361	9
34	0.01		TH$^+$	8.7	0.243	13
66	0.05		TH$^+$	17.4	0.291	23
130	0.05		TH$^+$	132.7	0.569	39
130	0.025		TH$^+$	206.0	0.166	41
precise intersection of convex hull with branch & bound						
6	0.01	0	CH$^-$	0.4	0.567	5
18	0.01	0	CH$^-$	2.4	0.356	9
34	0.01	0	CH$^-$	9.0	0.237	14
66	0.05	0.1	CH$^-$	17.3	0.243	23
66	0.05	0.01	CH$^-$	18.1	0.232	24
66	0.05	0.001	CH$^-$	27.4	0.192	37
66	0.05	0	CH$^-$	55.6	0.190	71
130	0.05	0.1	CH$^-$	126.0	0.339	39
130	0.05	0.01	CH$^-$	126.5	0.314	39
130	0.05	0.001	CH$^-$	205.5	0.190	39
130	0.025	0.01	CH$^-$	174.2	0.128	65

the error of the intersection computation reduces somewhat the number of samples, but the time gain is not substantial, see also Remark 1.

Filtered Oscillator: For a scalability comparison we turn to the filtered oscillator from Frehse et al. [2011]. It consists of a switched linear oscillator with two state variables x, y and four locations that is attached to K first-order filters put in sequence. The total number of state variables is therefore $K+2$. The filter stack produces the smoothened output signal z.

Table 4 shows results for up to 130 state variables, for both standard discrete image computation and the proposed variant with precise intersection. All instances are computed using box directions. For all except the lower dimensional versions, the precise intersection variant outperforms the standard operator in precision, and often also in speed. In this example, the capacity to compute the intersection up to a given error (column 3) shows its benefits: a small but not too small error greatly reduces the analysis time, at an acceptable loss in accuracy.

ACKNOWLEDGEMENTS

The authors thank Colas Le Guernic for fruitful discussions on the intersection problem.

REFERENCES

R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

R. E. Burkard, H. W. Hamacher, and G. Rote. Sandwich approximation of univariate convex functions with an application to separable convex programming. *Naval Res. Logistics*, 38:911–924, 1991.

George B. Dantzig and Mukund N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer, 2003.

Goran Frehse and Rajarshi Ray. Design principles for an extendable verification tool for hybrid systems. In *ADHS09 : 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, 2009.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011. ISBN 978-3-642-22109-5.

Colas Le Guernic. *Reachability analysis of hybrid systems with linear continuous dynamics*. PhD thesis, Université Grenoble 1 - Joseph Fourier, 2009.

Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In Ahmed Bouajjani and Oded Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 540–554. Springer, 2009. ISBN 978-3-642-02657-7.

Rajarshi Ray and Goran Frehse. An approach to direct minimization of convex piecewise linear functions. Technical report, Verimag, November 2011. URL `http://www-verimag.imag.fr/~ray/technical_reports/minimizing_convex_functions.pdf`.