ISSN 0005-1144
ATKAFF 50(3–4), 119–134(2009)

Mato Baotić

Polytopic Computations in Constrained Optimal Control

UDK 681.513.52
IFAC 5.5;2.8.1

Original scientific paper

In the last decade a lot of research has focused on the explicit solution of optimal and robust control problems for the class of constrained discrete-time systems. Many newly developed control algorithms for such control problems internally use operations on polytopic sets. We review basic polytopic manipulations and analyze them in the context of the computational effort.

We especially consider the so-called *regiondiff* problem where the set difference between a polyhedron and union of polyhedra needs to be computed. Regiondiff problem and related *polycover* problem – checking if a polytope is covered by the union of other polytopes – are utilized very often in derivation of the explicit solutions to the constrained finite time optimal control problems for piecewise affine systems. Similar observation holds for the computation of the (positive) controlled invariant sets, infinite time optimal control solution and/or controllers with reduced complexity for piecewise affine systems.

We describe an in-place depth-first exploration algorithm that solves the regiondiff problem in an efficient manner. We derive strict upper bound for the computational complexity of the described algorithm. In extensive testing we show that our algorithm is superior to the mixed integer linear programming approach when solving the polycover problem.

Key words: Polytope, Set difference, Set cover, Constrained optimal control, Discrete-time systems

Operacije nad politopskim skupovima kod optimalnog upravljanja sustava s ograničenjima. U posljednjih desetak godina znatna istraživačka aktivnost usmjerena je na pronalaženje eksplicitnih rješenja optimalnog i robusnog upravljanja za klasu vremenski diskretnih sustava s ograničenjima. Brojni razvijeni algoritmi interno koriste operacije nad politopskim skupovima. U ovom radu analiziramo osnovne operacije nad politopskim skupovima sa stajališta njihove računske kompleksnosti.

Naročita pozornost dana je takozvanom *regiondiff* problemu, odnosno problemu proračuna razlike poliedarskog skupa i unije poliedara. Isto tako je analiziran i srodni *polycover* problem – provjera je li poliedarski skup u potpunosti prekriven unijom poliedara. Oba ova problema često se sreću pri konstruiranju ekplicitnih rješenja optimalnog upravljanja po dijelovima afinih sustava uz konačan horizont predikcije, kao i pri proračunu pozitivnih invarijantnih skupova, optimalnog upravljanja uz beskonačan horizont predikcije i/ili proračunu regulatora smanjene kompleksnosti za po dijelovima afine sustave.

Razvijen je efikasan algoritam za rješenje *regiondiff* problema zasnovan na dubinskom pretraživanju stablaste strukture problema. Izvedena je teoretska gornja ograda za kompleksnost dobivenog algoritma, i pokazano je zašto je takva ograda konzervartivna u praksi. Na nizu simulacija pokazana je računsku superiornost razvijenog algoritam za *polycover* problem u odnosu na pristup zasnovan na rješavanju mješovitog cjelobrojnog programa.

Ključne riječi: politopski skupovi, razlika skupova, pokrivenost skupa, optimalno upravljanje sustava s ograničenjima, vremenski diskretni sustavi

1 INTRODUCTION

Set-theoretic methods for analysis and design of control systems have a long history since the first contributions trace back to the early 70s (cf. [1–5]). The theory was almost abandoned once it became clear that, although the techniques are appropriate to solve many theoretical and practical problems, the complexity of the required algorithms was not compatible with the computer technology of the time. In recent years, the theory is receiving a renewed interest due the current computer performances which are suitable to face non-trivial problems [5].

In the last decade a lot of research has focused on the solution of optimal and robust control problems for the class of discrete-time constrained linear systems [6–11]. Furthermore many algorithms have been extended to the more general class of so-called PieceWise Affine (PWA) systems [12–15].

Motivation for this paper is the computation of the explicit state-feedback optimal controllers. The term "explicit" means that such design yields the optimal controller in a closed form, i.e. as an explicit function of the system states [6]. Constrained optimal and/or robust control methods ensure systematic design of high-quality control systems for demanding applications. The optimal control systems satisfies prescribed constraints while achieving the performance that is optimal with respect to the design objective. Robustly controlled system can cope with the measurement/modelling uncertainty i.e. it can achieve desired goal while being exposed to the adverse effects of the environment. The computation of the optimal or robust controller for a general constrained non-linear system is a very difficult problem. Consequently, with the existing computational capabilities, the solution to the optimal control problem is tractable for a limited class of systems/constraints.

Several optimal control formulations may be used in explicit controller design. The most common formulations are the Constrained Finite Time Optimal Control (CFTOC) and the Constrained Time Optimal Control (CTOC). The performance objective of the CFTOC is the sum over a finite prediction horizon of piecewise linear or quadratic functions of the control inputs and (predicted values of) system states. An attractive property of the CFTOC problem is the fact that the optimizer, i.e. the optimal control input, is a piecewise affine function of the (current) system states [6,13]. In the CTOC problem formulation the goal is to find the control inputs that in a minimal number of time steps move the (current) system states into a controlled invariant set around the origin. The computation of explicit controller for the CTOC problem can be solved by manipulations of polytopes, and solution can be stored in a form of a PWA function of the current system states [14]. Both CFTOC and CTOC methodology rely on availability of a valid discrete-time model of the system. In practice this means that we need a systematic procedure for discrete-time PWA model identification of nonlinear process [15].

Almost all developed CFTOC and CTOC algorithms in their core use some type of operations on polytopes and unions of polytopes. Furthermore, many of them rely on extensive computation of set difference between polytope and union of polytopes [16]. In this paper we review basic polytopic manipulations and analyze them in the context of the computational effort. We derive efficient algorithms for solving two specific problems that emerge very often when deriving the explicit solution to the constrained optimal control problem for PWA systems: (i) *regiondiff* problem – computation of the set difference between a polyhedron and union of polyhedra; (ii) *polycover* problem – checking if a polytope is covered by the union of other polytopes.

2 NOTATION AND DEFINITIONS

$\mathbb{N}$ is the set of positive integers, $\mathbb{R}^{m\times n}$, with $m,n\in\mathbb{N}$, is the space of m by n real matrices, $\mathbb{R}^n$ is the space of n-dimensional real column vectors, $\emptyset$ denotes the empty set, and $\mathbb{N}_{1:m}$ is the shorthand notation for the set $\{1,\dots,m\}$. The symbol ':=' denotes that the left-hand side is defined by the right-hand side, and '=:' is used for the reverse logic (e.g., $\mathbb{N}_{1:m} := \{1,\dots,m\}$). Let $r\in\mathbb{R}^n$, $R\in\mathbb{R}^{m\times n}$, $i\in\mathbb{N}_{1:m}$, $\mathcal{I}\subseteq\mathbb{N}_{1:m}$, then $R_{(i)}$ denotes the i-th row of R; $R_{(\mathcal{I})}$ denotes the matrix formed from the rows of R indexed by $\mathcal{I}$; R^T denotes the matrix transpose of R; $\|\cdot\|$ denotes the Euclidean vector norm, i.e., $\|r\| := \sqrt{r^T r}$, and $\mathrm{card}(\cdot)$ denotes cardinality of the set (e.g. $\mathrm{card}(\mathbb{N}_{1:m}) = m$).

Throughout the paper all vector inequalities are considered component-wise (e.g., $a\le b$, with $a,b\in\mathbb{R}^m$, is equivalent to $a_{(i)}\le b_{(i)}$, $\forall i\in\mathbb{N}_{1:m}$). As a general rule vectors and matrices are denoted with italic letters (e.g., $a,b,R,\dots$) and sets are denoted with the upper case calligraphic letters (e.g., $\mathcal{I}$, $\mathcal{Q},\dots$).

We use the shorthand notation $\{\mathcal{Q}_i\}_{i=1}^{N_Q}$ for a collection of sets, i.e., $\{\mathcal{Q}_i\}_{i=1}^{N_Q} := \{\mathcal{Q}_1,\mathcal{Q}_2,\dots,\mathcal{Q}_{N_Q}\}$. Note that for $\mathcal{Q} = \{\mathcal{Q}_i\}_{i=1}^{N_Q}$ we have $\mathrm{card}(\mathcal{Q}) = N_Q$.

Definition 1 (Linear Program) *A* linear program *(LP) is a convex optimization problem that can be expressed in the form*

$$\begin{aligned} \min_{x} \quad & c^T x \\ \textit{subj. to} \quad & Gx \le g, \end{aligned} \tag{LP}$$

where $x\in\mathbb{R}^n$ is the optimization variable, and matrices $c\in\mathbb{R}^n$, $G\in\mathbb{R}^{m\times n}$, $g\in\mathbb{R}^m$ are given problem parameters. □

Theoretically every LP (with rational parameters) is solvable in polynomial time by both the ellipsoid method of Khachiyan [17, 18] and various interior point methods [19, 20]. A practical algorithm to solve an LP with n variables and m constraints requires roughly $\mathcal{O}(n^3 m^{0.5} + n^2 m^{1.5})$ operations [21]. In this paper the computational expense of a single LP has similar meaning like single addition or multiplication have in algebraic computations. In many instances we will express complexity of algorithms in terms of the number of LPs one needs to solve, thus establishing the basis for relative comparison between different algorithms. Henceforth with $\mathrm{lp}(n,m)$ we denote the complexity of a single LP with n variables and m constraints.

Most of the following definitions are standard. For additional details the reader is referred to [22, 23].

Definition 2 *A* hyperplane *in* $\mathbb{R}^n$ *is a set of the form*

$$\{x \in \mathbb{R}^n \mid a^T x = b\},$$

where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\|a\| > 0$. $\square$

Definition 3 *A* half-space *in* $\mathbb{R}^n$ *is a set of the form*

$$\mathcal{H} = \{x \in \mathbb{R}^n \mid a^T x \le b\},$$

where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\|a\| > 0$. $\square$

Definition 4 (Polyhedron and polytope) *A convex set*

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Px \le p\}, \tag{1}$$

with $P \in \mathbb{R}^{n_p \times n}$, $p \in \mathbb{R}^{n_p}$, $n_p < \infty$, *is called* polyhedron. *Bounded polyhedron is called* polytope. *The vector inequality in* (1) *is considered component-wise.* $\square$

We say that a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Px \le p\}$ is *full-dimensional* if it is possible to fit a non-empty n-dimensional ball in $\mathcal{P}$, i.e.,

$$\exists x_0 \in \mathbb{R}^n, \epsilon > 0 \; : \; \mathcal{B}(x_0, \epsilon) \subset \mathcal{P}, \tag{2}$$

where

$$\mathcal{B}(x_0, \epsilon) := \{x \in \mathbb{R}^n \mid \|x - x_0\| \le \epsilon\}. \tag{3}$$

Equivalently, polytope $\mathcal{P}$ is full-dimensional if

$$\exists x_0 \in \mathbb{R}^n, \; \epsilon > 0 \; : \|\delta\| \le \epsilon \Rightarrow P(x_0 + \delta) \le p. \tag{4}$$

Otherwise, we say that polytope $\mathcal{P}$ is *lower-dimensional*. A polytope $\mathcal{P}$ is referred to as *empty* if

$$\nexists x \in \mathbb{R}^n : Px \le p. \tag{5}$$

Furthermore, if $\|P_{(i)}\| = 1$, $i = 1, \ldots, n_p$, we say that the polytope $\mathcal{P}$ is *normalized*. One of the fundamental properties of a polytope is that it can be described in half-space representation (1) or in vertex representation (cf. [22]), as given below,

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_P} \alpha_i V_i^P, \; 0 \le \alpha_i \le 1, \; \sum_{i=1}^{v_P} \alpha_i = 1\}, \tag{6}$$

where $V_i^P \in \mathbb{R}^n$ denotes the i-th vertex of $\mathcal{P}$, and v_P is the total number of vertices of $\mathcal{P}$.

We will henceforth refer to the half-space representation (1) and vertex representation (6) as $\mathcal{H}$- and $\mathcal{V}$-representation respectively (see Fig. 1).

Definition 5 (Face) *A linear inequality* $a^T x \le b$ *is called* valid *for a polyhedron* $\mathcal{P}$ *if* $a^T x \le b$ *holds for all* $x \in \mathcal{P}$. *A subset of a polyhedron is called a* face *of* $\mathcal{P}$ *if it can be represented as*

$$\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid a^T x = b\}, \tag{7}$$

for some valid inequality $a^T x \le b$. *The faces of polyhedron* $\mathcal{P}$ *of dimension* 0, 1, $(n-2)$ *and* $(n-1)$ *are called vertices, edges, ridges and facets, respectively.* $\square$

We see that a set $\mathcal{F} \subseteq \mathcal{P}$ is called a face of $\mathcal{P}$ if it is either $\emptyset$, $\mathcal{P}$ itself or the intersection of $\mathcal{P}$ with a hyperplane derived from a valid inequality. An empty set $\emptyset$ is a face of every polyhedron and by convention it has dimension -1.

According to our definition every polytope represents a convex, compact (i.e. bounded and closed) set. We will see later that polytopes – as simple objects as they are – play an instrumental role in the derivation of optimal control strategies for constrained linear and piecewise affine systems. Very often, however, we also encounter sets that are disconnected or non-convex but can be represented as a union of finite number of polytopes. Therefore, it is useful to define the following mathematical concept.

Definition 6 *A* polytopal complex $\mathcal{C}$ *is a finite collection of polytopes in* $\mathbb{R}^n$ *such that*

- *the empty set is in* $\mathcal{C}$
- *if* $\mathcal{P} \in \mathcal{C}$, *then all the faces of* $\mathcal{P}$ *are also in* $\mathcal{C}$,
- *the intersection* $\mathcal{P} \cap \mathcal{Q}$ *of two polytopes* $\mathcal{P}, \mathcal{Q} \in \mathcal{C}$ *is a face both of* $\mathcal{P}$ *and* $\mathcal{Q}$. $\square$

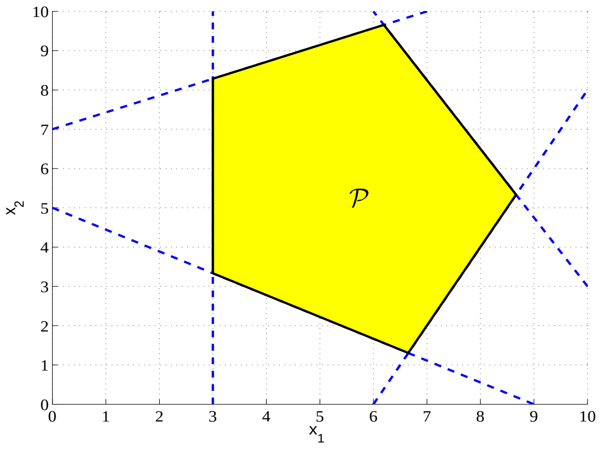
Example 1 *Here are several polytopal complexes in* $\mathbb{R}^1$: $\{\emptyset, 0, 2, [0, 2]\}$, $\{\emptyset, 0, 2, 4, [0, 2]\}$. *Contrary to those, the following sets are not polytopal complexes in* $\mathbb{R}^1$: $\{\emptyset, 0, [0, 2]\}$ *– because of a missing polytope* $\{2\}$, *and* $\{\emptyset, -2, 0, 2, [-2, 0], [-2, 2], [0, 2]\}$ *– because intersection of two polytopes* $[-2, 0]$ *and* $[-2, 2]$ *is not a face of* $[-2, 2]$. $\square$

Characterization of a polytopal complex is quite expensive. For every polytope in the complex we need to compute all the faces (i.e. lower-dimensional polytopes). Since we usually work only with full-dimensional polytopes (cf. Remark 1) it is reasonable to introduce a more practical concept.

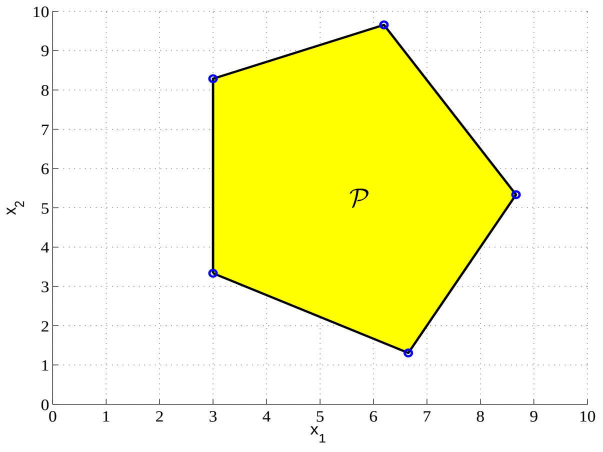
Definition 7 (P-collection) *A* P-collection $\mathcal{C}$ *is a finite collection of full-dimensional polytopes in* $\mathbb{R}^n$, *i.e.,*

$$\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{N_C}, \tag{8}$$

where $N_C = \operatorname{card}(\mathcal{C}) < \infty$, $\mathcal{C}_i := \{x \in \mathbb{R}^n \mid C_i x \le c_i\}$, $\mathcal{C}_i$ *full-dimensional*, $i = 1, \ldots, N_C$. $\square$



(a) $\mathcal{H}$-representation of a polytope in $\mathbb{R}^2$. The hyperplanes $\{x \mid P_{(i)}x = p_{(i)}\}$, $i = 1, \dots, 5$, are depicted with dashed lines



(b) $\mathcal{V}$-representation of a polytope in $\mathbb{R}^2$. The vertices V_i^P, $i = 1, \dots, 5$, are depicted with circles

Fig. 1. Illustration of a polytope in $\mathcal{H}$-representation and $\mathcal{V}$- representation

Definition 8 (Underlying Set) *The* underlying set *of a P-collection $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{N_C}$ is the point set in $\mathbb{R}^n$*

$$\underline{\mathcal{C}} := \bigcup_{i=1}^{N_C} \mathcal{C}_i. \tag{9}$$

□

Example 2 *A collection $\mathcal{R} = \{[-2,-1],[0,2],[2,4]\}$ is a P-collection in $\mathbb{R}^1$ with the underlying set $\underline{\mathcal{R}} = [-2,-1] \cup [0,4]$. As another example, $\mathcal{R} = \{[-2,0],[-1,1],[0,2]\}$ is a P-collection in $\mathbb{R}^1$ with underlying set $\underline{\mathcal{R}} = [-2,2]$. Clearly, polytopes that define P-collection can overlap, while the underlying sets can be disconnected and non-convex.* □

Usually it is clear from the context if we are talking about the P-collection or we are referring to the underlying set of a P-collection, in which case, for simplicity, we use the same notation for both.

Definition 9 (Partition) *A collection of sets $\{\mathcal{R}_i\}_{i=1}^{N_R}$ is a* partition *of a set $\mathcal{P}$ if: (i) $\mathcal{P} = \cup_{i=1}^{N_R} \mathcal{R}_i$, and (ii) $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i \neq j$, with $i, j \in \{1, \dots, N_R\}$.* □

Definition 10 (Polyhedral Partition) *A collection of sets $\{\mathcal{R}_i\}_{i=1}^{N_R}$ is a* polyhedral partition *of a set $\mathcal{P}$ if $\{\mathcal{R}_i\}_{i=1}^{N_R}$ is a partition of $\mathcal{P}$ and the sets $\bar{\mathcal{R}}_i$ are polyhedra, where $i = 1, \dots, N_R$, and $\bar{\mathcal{R}}_i$ denotes the closure of $\mathcal{R}_i$.* □

3 OPERATIONS ON POLYTOPES

We will now define some basic operations and functions on polytopes. Note that although we focus on polytopes and polytopic objects most of the operations described here are directly (or with minor modifications) applicable to polyhedral objects. Additional details on polytope computation can be found in [22, 23]. All operations and functions described in this section are contained in the MPT toolbox [24].

3.1 Minimal Representation

We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid Px \leq p\}$ is in a *minimal representation* if the removal of any row in $Px \leq p$ would change it (i.e., there are no redundant half-spaces). The computation of minimal representation of polytopes is discussed in [25] and generally requires solution of one LP for each half-space defining the non-minimal representation of $\mathcal{P}$. We report one straightforward implementation of the minimal representation computation in Algorithm 1.

Algorithm 1 Minimal representation: **minrep**($\mathcal{P}$)

```
Input: $\mathcal{P} := \{x \in \mathbb{R}^n \mid Px \leq p\}, p \in \mathbb{R}^{n_p}$
Output: Minimal representation of $\mathcal{P}$
1. $\mathcal{I} \leftarrow \{1, \dots, n_p\}$
2. for $i = 1$ to $n_p$ do
3.     $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$
4.     $f^* \leftarrow \max_x \{P_{(i)}x \mid P_{(\mathcal{I})}x \leq p_{(\mathcal{I})},\ P_{(i)}x \leq p_{(i)}+1\}$
5.     if $f^* > p_{(i)}$ then
6.         $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
7.     end if
8. end for
9. return $\tilde{\mathcal{P}} := \{x \mid P_{(\mathcal{I})}x \leq p_{(\mathcal{I})}\}$
```

Remark 1 *It is straightforward to see that a normalized, full-dimensional polytope $\mathcal{P}$ has a* unique *minimal representation.*[1] *This fact is very useful in practice. Normalized, full-dimensional polytopes in a minimal representation allow us to avoid any ambiguity when comparing them and very often speed-up other polytope manipulations. For simplicity, and without loss of generality, in the rest of this paper we consider only full-dimensional polytopes. Lower-dimensional polytopes (with the exception of an empty polytope) are not considered. The reasoning is similar to the one in the MPT toolbox [24] – this simplification reduces the complexity of the computations, while being sufficient for description and solution of the control problems we consider in the paper.* $\square$

3.2 Chebyshev Ball

The Chebyshev ball of a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Px \leq p\}$, with $P \in \mathbb{R}^{n_p \times n}$, $p \in \mathbb{R}^{n_p}$, corresponds to the largest radius ball in $\mathbb{R}^n$ such that $\mathcal{B}(x_c, R_c) \subset \mathcal{P}$. The center x_c and radius R_c of the Chebyshev ball can be easily found by solving the following LP [26]

$$\begin{aligned} \max_{x_c, R_c} \quad & R_c \\ \text{subj. to} \quad & P_{(i)} x_c + \|P_{(i)}\| R_c \leq p_{(i)},\ i = 1, \ldots, n_p. \end{aligned} \tag{10}$$

If the solution to (10) is $R_c = 0$, then the polytope $\mathcal{P}$ is lower-dimensional; if $R_c < 0$, then the polytope is empty. Therefore, an answer to the question "is polytope $\mathcal{P}$ full-dimensional/empty?" is obtained at the *expense* of only one linear program of complexity $\text{lp}(n+1, n_p)$. Furthermore, for a full-dimensional polytope we also get a point x_c that is in the interior of $\mathcal{P}$ (see Fig. 2 for illustration). One word of caution: the center of a Chebyshev ball x_c in (10) is not necessarily unique (e.g. when $\mathcal{P}$ is a rectangle). There are other types of unique interior points one could compute for a full-dimensional polytope (e.g., analytic center, center of the largest volume ellipsoid, etc.) but those computations are usually formulated as semidefinite programming problems [26] and hence they are more expensive to carry out than the Chebyshev ball computation via LP (10).

3.3 Projection

Given an $(n+m)$-polytope $\mathcal{P} = \{x \in \mathbb{R}^{n+m} \mid Px \leq p\} \subset \mathbb{R}^{n+m}$, $n, m \in \mathbb{N}$, the projection onto the $\mathbb{R}^n$ is defined as an n-polytope

$$\text{proj}_n(\mathcal{P}) := \{y \in \mathbb{R}^n \mid \exists z \in \mathbb{R}^m : P[y^T\, z^T]^T \leq p\}. \tag{11}$$

An illustration of a projection operation is given in Fig. 3. In general, the projection methods (for arbitrary dimensions m and n) reported in literature can be grouped into four classes: Fourier elimination [27, 28], block elimination [29], vertex based approaches [30] and wrapping-based techniques [31]. For a good introduction to projection, we refer the reader to [31] and the references therein.

[1]The term 'unique' here means that for a polytope $\mathcal{P} := \{x \in \mathbb{R}^n \mid Px \leq p\}$ the matrix $[P\ p]$ comprises unique set of row vectors. The order of those rows is irrelevant.

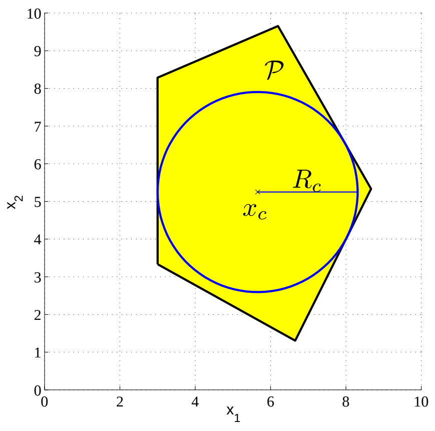


Fig. 2. Chebyshev ball $\mathcal{B}(x_c, R_c)$ contained in a polytope $\mathcal{P} \subset \mathbb{R}^2$

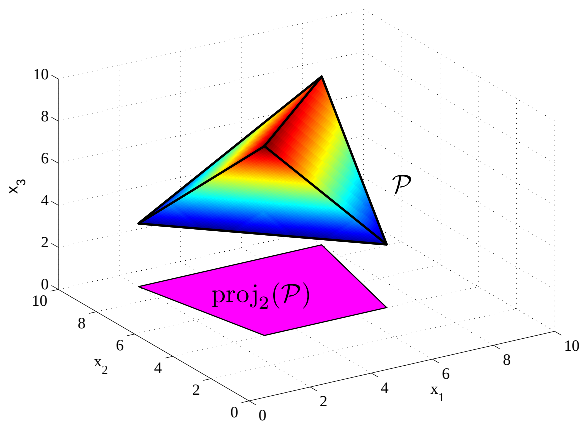


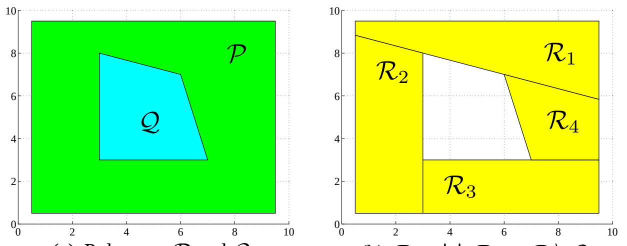
Fig. 3. Projection of a 3-dimensional polytope $\mathcal{P}$ onto $\mathbb{R}^2$

3.4 Set Difference

The Set difference of two polytopes $\mathcal{P}$ and $\mathcal{Q}$

$$\mathcal{R} = \mathcal{P} \setminus \mathcal{Q} := \{x \in \mathbb{R}^n \mid x \in \mathcal{P}, x \notin \mathcal{Q}\}, \tag{12}$$

is, in general, given as a P-collection $\mathcal{R} = \bigcup_i \mathcal{R}_i$, which can be computed by consecutively inverting the half-spaces defining $\mathcal{Q}$ as described in [6] (see Fig. 4). Note that here we use the term P-collection in the dual context of both P-collection and its underlying set (cf. Definition 7 and Definition 8). The precise statement would say that $\underline{\mathcal{R}} = \mathcal{P} \setminus \mathcal{Q}$, where $\underline{\mathcal{R}}$ is the underlying set of P-collection $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{N_R}$. However, whenever it is clear from context what are we referring to, as is the case here, we will abuse the notation and use the former, more compact form.



(a) Polytopes $\mathcal{P}$ and $\mathcal{Q}$ (b) $\mathcal{R} = \bigcup_i \mathcal{R}_i = \mathcal{P} \setminus \mathcal{Q}$

Fig. 4. Illustration of the set difference operation

Remark 2 *The set difference of two intersecting polytopes (or any closed sets) $\mathcal{P}$ and $\mathcal{Q}$ is, in general, not a closed set. This means that some borders of polytopes $\mathcal{R}_i$ from a P-collection $\mathcal{R} = \mathcal{P} \setminus \mathcal{Q}$ are open, while other borders are closed. Even though it is possible to keep track of the origin of particular borders of $\mathcal{R}_i$, thus specifying if they are open or closed, we are not doing it in the algorithms described in this paper nor in MPT [24], cf. Remark 1. In computations, we will henceforth only consider the closure of sets $\mathcal{R}_i$.* □

Related to the above operation are the following two computationally very demanding problems/questions: what is the set difference between a polytope $\mathcal{P}$ and a P-collection $\mathcal{Q} := \cup_i \mathcal{Q}_i$; and is a polytope $\mathcal{P}$ fully covered by a P-collection $\mathcal{Q} := \cup_i \mathcal{Q}_i$? For instance, the latter operation is needed to check if two unions of polyhedra cover the same non-convex set [11, 14] (e.g., step 5 of Algorithm 4.1 in [14]). In Section 4 and Section 5 we analyze these problems in more details, and derive efficient algorithms that perform these tasks.

3.5 Convex Hull

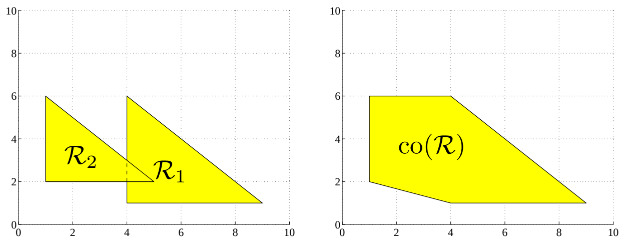
The convex hull of a set of points $\mathcal{V} = \{V_i\}_{i=1}^{N_V}$, with $V_i \in \mathbb{R}^n$, is a polytope defined as

$$\mathrm{co}(\mathcal{V}) = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{N_V} \alpha_i V_i,\ 0 \le \alpha_i \le 1,\ \sum_{i=1}^{N_V} \alpha_i = 1\}. \tag{13}$$

The convex hull operation is used to switch from a $\mathcal{V}$-representation of a polytope to a $\mathcal{H}$-representation. The convex hull of a union of polytopes $\mathcal{R}_i \subset \mathbb{R}^n$, $i = 1, \ldots, N_R$, is a polytope

$$\mathrm{co}\left(\bigcup_{i=1}^{N_R} \mathcal{R}_i\right) := \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{N_R} \alpha_i x_i,\ x_i \in \mathcal{R}_i,\ 0 \le \alpha_i \le 1,\ \sum_{i=1}^{N_R} \alpha_i = 1\}. \tag{14}$$

An illustration of the convex hull operation is given in Fig. 5. Construction of the convex hull is an expensive op-



(a) P-collection $\mathcal{R} = \bigcup_i \mathcal{R}_i$ (b) Convex hull of $\mathcal{R}$

Fig. 5. Illustration of the convex hull operation

eration which is exponential in the number of facets of the original polytope. An efficient software implementation is available from [32, 33].

3.6 Envelope

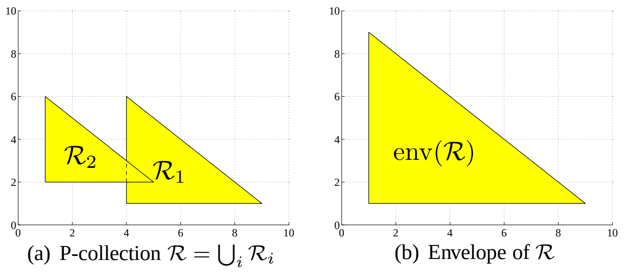
The envelope of two $\mathcal{H}$-polyhedra $\mathcal{P} = \{x \in \mathbb{R}^n \mid Px \le p\}$ and $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Qx \le q\}$ is an $\mathcal{H}$-polyhedron

$$\mathrm{env}(\mathcal{P}, \mathcal{Q}) = \{x \in \mathbb{R}^n \mid \tilde{P}x \le \tilde{p},\ \tilde{Q}x \le \tilde{q}\}, \tag{15}$$

where $\tilde{P}x \le \tilde{p}$ is the subsystem of $Px \le p$ obtained by removing all the inequalities not valid for the polyhedron $\mathcal{Q}$, and $\tilde{Q}x \le \tilde{q}$ are defined in a similar way with respect to $Qx \le q$ and $\mathcal{P}$ [34]. In a similar fashion definition can be extended to the case of the envelope of a P-collection. An illustration of the envelope operation is depicted in Fig. 6. The computation of the envelope is relatively cheap since it only requires the solution to one LP for each facet of $\mathcal{P}$ and $\mathcal{Q}$. Note that envelope of two (or more) polytopes is not necessarily bounded set (e.g. when $\mathcal{P} \cup \mathcal{Q}$ is shaped like a star).

3.7 Vertex Enumeration

The operation of extracting the vertices of a polytope $\mathcal{P}$ given in $\mathcal{H}$-representation is referred to as vertex enumeration. This operation is the dual to the convex hull operation and the algorithmic implementation is identical to a convex hull computation, i.e. given a set of extreme



(a) P-collection $\mathcal{R} = \bigcup_i \mathcal{R}_i$ (b) Envelope of $\mathcal{R}$

Fig. 6. Illustration of the envelope operation

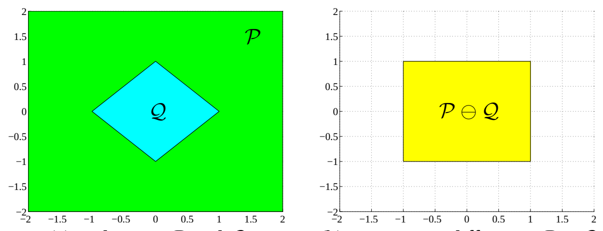
points $\mathcal{V} = \{V_i\}_{i=1}^{N_V} = \mathrm{vert}(\mathcal{P})$ of a polytope $\mathcal{P}$ given in $\mathcal{H}$-representation it holds that $\mathcal{P} = \mathrm{co}(\mathcal{V})$, where the operator vert denotes the vertex enumeration. The necessary computational effort is in the worst case exponential in the number of the polytope facets. Two different approaches to vertex enumeration exist: the double description method [35] and reverse search [36]. An efficient implementation of the double description method is available in [32, 33].

3.8 Pontryagin Difference

The Pontryagin difference (also known as Minkowski difference) of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$\mathcal{P} \ominus \mathcal{Q} := \{x \in \mathbb{R}^n \mid x + q \in \mathcal{P},\ \forall q \in \mathcal{Q}\}. \quad (16)$$

The Pontryagin difference can be efficiently computed for polytopes by solving a sequence of LPs [37]. For special cases (e.g. when $\mathcal{Q}$ is a hypercube), even more efficient computational methods exist [38]. An illustration of the Pontryagin difference is given in Fig. 7.



(a) Polytopes $\mathcal{P}$ and $\mathcal{Q}$ (b) Pontryagin difference $\mathcal{P} \ominus \mathcal{Q}$

Fig. 7. Illustration of the Pontryagin difference operation

3.9 Minkowski Sum

The Minkowski sum of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$\mathcal{P} \oplus \mathcal{Q} := \{x + q \in \mathbb{R}^n \mid x \in \mathcal{P},\ q \in \mathcal{Q}\}. \quad (17)$$

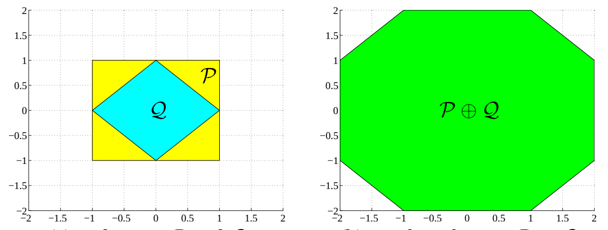
The Minkowski sum is a computationally expensive operation which requires either vertex enumeration and convex hull computation in $\mathbb{R}^n$ or a projection from $\mathbb{R}^{2n}$ down to $\mathbb{R}^n$. The implementation of the Minkowski sum via projection is based on the following observation. For

$$\mathcal{P} = \{y \in \mathbb{R}^n \mid Py \le p\}, \qquad \mathcal{Q} = \{z \in \mathbb{R}^n \mid Qz \le q\},$$

it holds that

$$\begin{aligned}
\mathcal{R} &= \mathcal{P} \oplus \mathcal{Q} \\
&= \left\{x = y + z \mid y, z \in \mathbb{R}^n,\ Py \le p,\ Qz \le q\right\} \\
&= \left\{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n : Py \le p,\ Q(x - y) \le q\right\} \\
&= \left\{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n : \begin{bmatrix} 0 & P \\ Q & -Q \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} p \\ q \end{bmatrix}\right\} \\
&= \mathrm{proj}_n\left(\left\{w \in \mathbb{R}^{2n} \mid \begin{bmatrix} 0 & P \\ Q & -Q \end{bmatrix} w \le \begin{bmatrix} p \\ q \end{bmatrix}\right\}\right).
\end{aligned}$$

Both the projection and vertex enumeration based methods are implemented in the MPT toolbox [24]. An illustration of the Minkowski sum is given in Fig. 8.



(a) Polytopes $\mathcal{P}$ and $\mathcal{Q}$ (b) Minkowski sum $\mathcal{P} \oplus \mathcal{Q}$

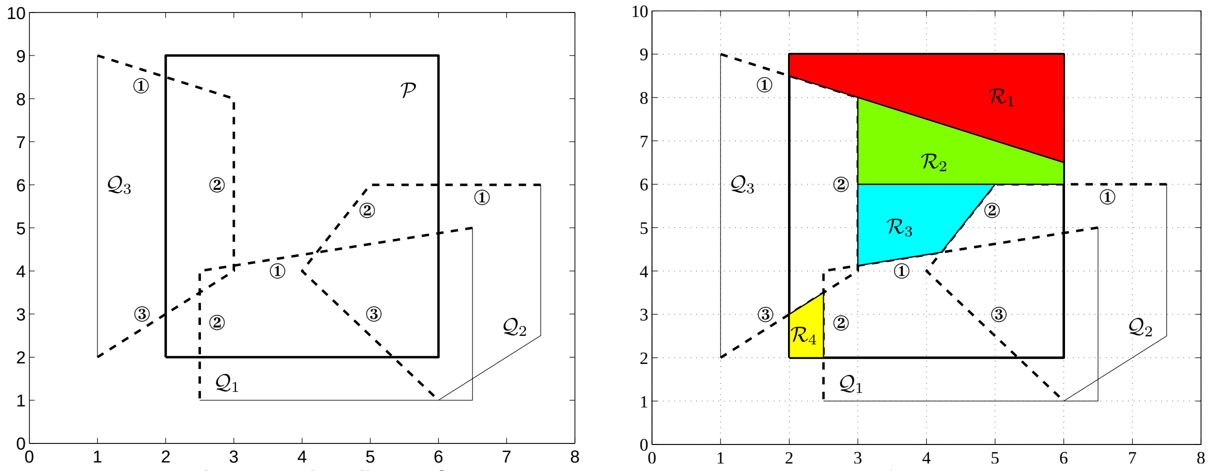
Fig. 8. Illustration of the Minkowski sum operation

Remark 3 *The Minkowski sum is* **not** *the complement of the Pontryagin difference. For two polytopes $\mathcal{P}$ and $\mathcal{Q}$, it holds that $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q} \subseteq \mathcal{P}$ (cf. Fig. 7 and Fig. 8).* □

4 REGIONDIFF PROBLEM

Let $\mathcal{P} = \{x \mid Px \le p\}$ be a polyhedron in $\mathbb{R}^n$ given as the intersection of n_p half-spaces and $\mathcal{Q}_i = \{x \in \mathbb{R}^n \mid Q_i x \le q_i\}$ be N_Q polyhedra in $\mathbb{R}^n$ given as the intersection of n_{q_i} half-spaces (i.e. $Q_i \in \mathbb{R}^{n_{q_i} \times n}$). We want to determine the set difference $\mathcal{P} \setminus (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$. We refer to this problem as the **regiondiff** problem (to differentiate it from the problem of computing the set difference between two polytopes, cf. Section 3.4).

We outline the procedure of finding the exact solution to the **regiondiff** problem in Algorithm 2. It computes the set $\mathcal{R} = \mathcal{P} \setminus (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$, where $\mathcal{R}$ (if not empty) is given as union of non-overlapping polyhedra.



(a) Polytope $\mathcal{P}$ and P-collection $\mathcal{Q}$ (b) $\mathcal{R} := \{\mathcal{R}_i\}_{i=1}^4 =$ **regiondiff**$(\mathcal{P}, \mathcal{Q})$

Fig. 9. Example for the set difference computation with Algorithm 2 in $\mathbb{R}^2$. Polytope $\mathcal{P}$ is depicted with full line, active constraints (see (18) *for definition) of $\mathcal{Q}_i$ with dashed lines, and non-active constraints of $\mathcal{Q}_i$ with thin lines. The indices of corresponding active constraints of each $\mathcal{Q}_i$ are denoted as encircled numbers.*

One example of solving the **regiondiff** problem with Algorithm 2 for 2-dimensional polytope $\mathcal{P}$ and P-collection $\{\mathcal{Q}_i\}_{i=1}^3$ is illustrated in Fig. 9.

Algorithm 2 solves the **regiondiff** problem by basically implementing an in-place depth-first exploration of the tree of feasible constraint combinations, where each branch/node corresponds to the inversion of a facet of $\mathcal{Q}_i$ (step 12 of Algorithm 2). The approach used to partition the space is based on [6].

Note that the output of Algorithm 2 is a P-collection $\mathcal{R}$ that comprises only non-overlapping polyhedra, regardless of possible overlaps in the input P-collection $\mathcal{Q}$. We also note that Algorithm 2 would work even if steps 2-10 were not present. However, those steps potentially have a huge impact on the speed of computation since they exclude unnecessary checks at later stages (i.e. lower branches) of the algorithm.

Remark 4 *Step 5 of Algorithm 2 is a simple feasibility LP of complexity* $\mathrm{lp}(n+1, n_p + n_{q_k})$. *Similarly, step 12 is a feasibility LP of complexity* $\mathrm{lp}(n+1, n_p+1)$. □

Remark 5 (Preprocessing) *From a practical point of view it makes sense to preprocess the data before using Algorithm 2 to compute the set difference. Namely, we can speed up the computation by ensuring that $\mathcal{P}$ and $\mathcal{Q}_i$, $i = 1, \dots, N_Q$, are full-dimensional polyhedra in $\mathbb{R}^n$ given in the minimal $\mathcal{H}$-representation: $\mathcal{P} = \{x \mid Px \le p\}$, $\mathcal{Q}_i = \{x \mid Q_i x \le q_i\}$, $P \in \mathbb{R}^{n_p \times n}$, $p \in \mathbb{R}^{n_p}$, $Q_i \in \mathbb{R}^{n_{q_i} \times n}$, $q_i \in \mathbb{R}^{n_{q_i}}$. Note that it is always possible to obtain normalized polyhedron in minimal representation (see Section 3.1) and to remove those $\mathcal{Q}_i$ that do not intersect $\mathcal{P}$, e.g. by checking if Chebyshev ball (see Section 3.2) of a joint polyhedra $\{x \mid Px \le p, Q_i x \le q_i\}$ is non-empty.*

Furthermore, it is useful to pre-compute the set of so-called active constraints*:*

$$\mathcal{A}_i = \left\{ j \in \{1, \dots, n_{q_i}\} \mid \exists x : Px \le p, [Q_i]_{(j)} x > [q_i]_{(j)} \right\}, \tag{18}$$

for each $\mathcal{Q}_i$, $i = 1 \dots, N_Q$. This can be done at the expense of at most $\sum_{i=1}^{N_Q} n_{q_i}$ LPs:

$$\sum_{i=1}^{N_Q} n_{q_i} \cdot \mathrm{lp}(n+1, n_p + n_{q_i}). \tag{19}$$

We note that the removal of non-active constraints from the description of each $\mathcal{Q}_i$ does not change the result of Algorithm 2, i.e.,

$$\mathcal{P} \setminus \{\mathcal{Q}_i\}_{i=1}^{N_Q} = \mathcal{P} \setminus \{\tilde{\mathcal{Q}}_i\}_{i=1}^{N_Q}, \tag{20}$$

where

$$\tilde{\mathcal{Q}}_i := \{x \in \mathbb{R}^n \mid \tilde{Q}_i x \le \tilde{q}_i\}, \tag{21}$$

$$\tilde{Q}_i = [Q_i]_{(\mathcal{A}_i)}, \; \tilde{q}_i = [q_i]_{(\mathcal{A}_i)}. \tag{22}$$

As a matter of fact, since $\mathrm{card}(\mathcal{A}_i) \le n_{q_i}$, *this removal of non-active constraints can significantly reduce dimension of the problem parameters in Algorithm 2 and thus drastically reduce complexity of Algorithm 2.* □

Remark 6 (Postprocessing) *Note that Algorithm 2 returns P-collection $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{N_R}$, with $\mathcal{R}_i$ that are in (possibly) non-minimal $\mathcal{H}$-representation. As a postprocessing step one might call Algorithm 1 to obtain the minimal representation for each $\mathcal{R}_i$ of the result.* □

Algorithm 2 Set difference: **regiondiff**($\mathcal{P}, \mathcal{Q}$)

Input: $\mathcal{P} := \{x \in \mathbb{R}^n \mid Px \le p\}$, $p \in \mathbb{R}^{n_p}$, $\mathcal{Q} := \{\mathcal{Q}_i\}_{i=1}^{N_Q}$, $\mathcal{Q}_i := \{x \in \mathbb{R}^n \mid Q_i x \le q_i\}$, $q_i \in \mathbb{R}^{n_{q_i}}$, $i = 1, \dots, N_Q$

Output: P-collection representing $\mathcal{P} \setminus \mathcal{Q}$

```
1.  $\mathcal{R} \leftarrow \emptyset$, $k \leftarrow 1$
2.  if $\exists i \in \mathbb{N}_{1:N_Q} : n_{q_i} = 0$ then
3.      return $\mathcal{R}$
4.  end if
5.  while $\nexists x \in \mathbb{R}^n : Px < p,\ Q_k x < q_k$ do
6.      $k \leftarrow k + 1$
7.      if $k > N_Q$ then
8.          return $\mathcal{P}$
9.      end if
10. end while
11. for $j = 1$ to $n_{q_k}$ do
12.     if $\exists x \in \mathbb{R}^n : Px \le p,\ [Q_k]_{(j)} x > [q_k]_{(j)}$ then
13.         $\tilde{\mathcal{P}} = \mathcal{P} \cap \{x \mid [Q_k]_{(j)} x \ge [q_k]_{(j)}\}$
14.         if $k < N_Q$ then
15.             $\mathcal{R} \leftarrow \mathcal{R} \cup$ regiondiff$(\tilde{\mathcal{P}}, \{\mathcal{Q}_i\}_{i=k+1}^{N_Q})$
16.         else
17.             $\mathcal{R} \leftarrow \mathcal{R} \cup \tilde{\mathcal{P}}$
18.         end if
19.     end if
20.     $\mathcal{P} \leftarrow \mathcal{P} \cap \{x \mid [Q_k]_{(j)} x \le [q_k]_{(j)}\}$
21. end for
22. return $\mathcal{R}$
```

The most time consuming part of Algorithm 2 lies in recursive calls to itself (step 15). Therefore in the following computational complexity analysis we neglect the cost of removal of the *non-active constraints* from $\mathcal{Q}_i$ (cf. Remark 5) or computing the minimal representation of the regions $\mathcal{R}_i$ that describe the set difference (cf. Remark 6).

To establish the worst case complexity of Algorithm 2 one needs to look at its tree-like exploration of the space. The depth of the tree is equal to N_Q (i.e., the number of regions $\mathcal{Q}_i$). Clearly, every node on the level k has at most $n_{q_{k+1}}$ children nodes on the level $k+1$, where n_{q_k} denotes the number of constraints of polytope $\mathcal{Q}_k$ (if preprocessing step from Remark 5 is done then n_{q_k} is the number of active constraints) . Therefore the number of nodes at level k is bounded from above by

$$\prod_{i=1}^{k} n_{q_i} \le \left(\frac{H_k}{k}\right)^k, \tag{23}$$

where

$$H_k := \sum_{i=1}^{k} n_{q_i}, \tag{24}$$

i.e., H_k is the cumulative number of constraints up to the level k of the tree, and $k = 1, \dots, N_Q$.

One should note that (23) gives an overly conservative upper bound for the number of nodes at level k. For example, if (23) was a strict upper bound, it would imply that the number of nodes at the bottom of the tree is of the order

$$\mathcal{O}\left(\left(\frac{M}{N_Q}\right)^{N_Q}\right), \tag{25}$$

where M denotes the total number of constraints

$$M := \sum_{i=1}^{N_Q} n_{q_i}. \tag{26}$$

To better estimate the upper bound on the number of nodes at level k we recall the Buck's formula [39] for the hyperplane arrangement problem. Buck's formula gives an upper bound for the maximal number of cells created by H_k hyperplanes in $\mathbb{R}^n$. The upper bound is obtained by the hyperplanes in the so-called general position [22]. Therefore, we can obtain the strict upper bound on the number of nodes at level k

$$T_k \le \sum_{i=0}^{n} \binom{H_k}{i} = \mathcal{O}\left(\frac{{H_k}^n}{n!}\right), \tag{27}$$

where T_k denotes the number of nodes at level k of the tree for Algorithm 2, and $k = 1, \dots, N_Q$.

Since every region in the output of Algorithm 2 corresponds to exactly one feasible node at the bottom of the exploration tree, we clearly see that the number of nodes T_{N_Q} is an upper bound on the number of regions generated by Algorithm 2

$$N_R \le T_{N_Q}. \tag{28}$$

Therefore, by taking into account that $H_{N_Q} = M$, the following strict upper bound on N_R can be computed from (27)

$$N_R \le \sum_{i=0}^{n} \binom{M}{i} = \mathcal{O}\left(\frac{M^n}{n!}\right). \tag{29}$$

Feasible nodes of level $k+1$, with $k = 1, \dots, N_Q - 1$, are computed from feasible nodes at level k. The computation at every node of level k involves solution of at most $n_{q_{k+1}}$ LPs with $n+1$ variables and at most $n_p + H_{k+1}$ constraints (cf. Remark 4). Therefore, the strict upper bound on the overall complexity of Algorithm 2 is given by

$$\sum_{k=1}^{N_Q-1} T_k \cdot n_{q_{k+1}} \cdot \mathrm{lp}(n+1, n_p + H_{k+1}), \tag{30}$$

i.e., the overall complexity of Algorithm 2 is of the order

$$\mathcal{O}\left(\frac{M^{n+1}}{n!} \cdot \mathrm{lp}(n+1, n_p + M)\right). \tag{31}$$

From (31) it is clear that for $n > 1$ the overall complexity of Algorithm 2 is hugely affected by M – the total number of (active) constraints for the problem at hand. Therefore, for specific applications, where **regiondiff** problem is solved via Algorithm 2, one should investigate if it is possible to split the original problem in subproblems (assuming that the dimension n remains fixed) that would result in calls to Algorithm 2 with a reduced value of M (see [40] for one application of this idea).

Remark 7 *The expression* (29) *represents the strict upper bound on the output complexity (i.e. the set difference representation) for Algorithm 2, and therefore it cannot be further improved. The same claim holds also for the expression* (30) *(and, consequently, for* (31)*) that defines the strict upper bound on the computational complexity of Algorithm 2. However, in practice ($n \ll N_Q \ll M$) these expressions are found to be very conservative (cf. testing results in Section 6).* □

Remark 8 *Extension of the **regiondiff** problem to the case where both $\mathcal{P}$ and $\mathcal{Q}$ are P-collections is straightforward. Namely, the set difference is a P-collection $\mathcal{R} = \{\mathcal{P}_i\}_{i=1}^{N_P} \setminus \{\mathcal{Q}_j\}_{j=1}^{N_Q}$ that can be computed by cycling through all polytopes $\mathcal{P}_i$. We have*

$$\mathcal{R} = \bigcup_{i=1}^{N_P} \mathcal{P}_i \setminus \{\mathcal{Q}_j\}_{j=1}^{N_Q}. \tag{32}$$

Note that here the output P-collection $\mathcal{R}$ may contain overlapping polyhedra if the input P-collection $\mathcal{P}$ has some overlapping polyhedra. □

5 POLYCOVER PROBLEM

The problem of checking if some polytope is covered with the union of other polytopes emerges very often in the construction of explicit solution for the constrained finite time optimal control of piecewise affine systems. The need for solving this type of a problem is also common in the computation of the (positive) controlled invariant sets, infinite time solution or controllers with reduced complexity for PWA systems. Thus, it is important to have an algorithm that checks if a polytope $\mathcal{P}$ is fully covered by a P-collection $\mathcal{Q} := \cup_i \mathcal{Q}_i$ in an efficient manner. We refer to this problem as the **polycover** problem.

One idea of solving the **polycover** problem is inspired by the following observation

$$\mathcal{P} \subseteq \cup_i \mathcal{Q}_i \quad \Leftrightarrow \quad \mathcal{P} = \cup_i (\mathcal{P} \cap \mathcal{Q}_i).$$

Therefore, we could create $\mathcal{R}_i = \mathcal{Q}_i \cap \mathcal{P}$, for $i = 1, \dots, N_Q$ and then compute the union of the collection of polytopes $\{\mathcal{R}_i\}$ by using the *polyunion* algorithm for computing the convex union of $\mathcal{H}$-polyhedra reported in [34]. If approach [34] succeeds (i.e., the union is a convex set) and the resulting polytope is equal to $\mathcal{P}$ then $\mathcal{P}$ is covered by $\{\mathcal{Q}_i\}_{i=1}^{N_Q}$, otherwise it is not. However, this approach is computationally very expensive. The algorithm in [34] is based on the idea that whenever $\mathcal{P} := \cup_i \mathcal{Q}_i$ is a polyhedron then $\mathcal{P} = \text{env}(\{\mathcal{Q}_i\}_{i=1}^{N_Q})$. Unfortunately the underlying computation is very expensive, and one needs to solve $\mathcal{O}(\Pi_{i=1}^{N_Q} n_{q_i})$ LPs with n variables and $M = \sum_{i=1}^{N_Q} n_{q_i}$ constraints. Thus, the polyunion computation from [34] becomes quickly prohibitive with the increasing number of polytopes N_Q and constraints M.

In the following we propose two different approaches.

5.1 Polycover: Regiondiff Based Algorithm

Clearly, **polycover** is just a special case of **regiondiff**, where resulting P-collection $\mathcal{R} = \emptyset$. In a special case when we only want to check if $\mathcal{P} \subseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ finding any feasible $\mathcal{R}_j$ in Algorithm 2 provides a negative answer and we can abort further search. Implementation of this strategy is given in Algorithm 3.

Similarly to Algorithm 2 the worst case complexity of Algorithm 3 is bounded by (30) (see also Remark 5 and Remark 7).

Remark 9 *Note that it is straightforward to extend the **polycover** problem to the case where both $\mathcal{P}$ and $\mathcal{Q}$ are P-collections:*

- *Verification of $\{\mathcal{P}_i\}_{i=1}^{N_P} \subseteq \{\mathcal{Q}_j\}_{j=1}^{N_Q}$:*

  $$\{\mathcal{P}_i\}_{i=1}^{N_P} \subseteq \{\mathcal{Q}_j\}_{j=1}^{N_Q} \Leftrightarrow \{\mathcal{P}_i\}_{i=1}^{N_P} \setminus \{\mathcal{Q}_j\}_{j=1}^{N_Q} = \emptyset.$$

- *Verification of $\{\mathcal{P}_i\}_{i=1}^{N_P} = \{\mathcal{Q}_j\}_{j=1}^{N_Q}$:*

  $$\{\mathcal{P}_i\}_{i=1}^{N_P} = \{\mathcal{Q}_j\}_{j=1}^{N_Q} \Leftrightarrow$$
  $$\{\mathcal{P}_i\}_{i=1}^{N_P} \setminus \{\mathcal{Q}_j\}_{j=1}^{N_Q} = \emptyset \ \& \ \{\mathcal{Q}_j\}_{j=1}^{N_Q} \setminus \{\mathcal{P}_i\}_{i=1}^{N_P} = \emptyset.$$

Here the set difference between two P-collections can be computed according to (32). □

5.2 Polycover: MILP formulation

When some of the optimization variables in a linear program (see Definition 1) are constrained to integer values the ensuing problem is called a *mixed integer linear program* (MILP).

Definition 11 *A* mixed integer linear program *(MILP) is a non-convex optimization problem that can be expressed in the form*

$$\begin{aligned} \min_x \quad & f^T x \\ \textit{subj. to} \quad & Gx \le g, \end{aligned} \tag{MILP}$$

Algorithm 3 Set cover: **polycover**$(\mathcal{P}, \mathcal{Q})$

Input: $\mathcal{P} := \{x \in \mathbb{R}^n \mid Px \leq p\}$, $p \in \mathbb{R}^{n_p}$, $\mathcal{Q} := \{\mathcal{Q}_i\}_{i=1}^{N_Q}$, $\mathcal{Q}_i := \{x \in \mathbb{R}^n \mid Q_i x \leq q_i\}$, $q_i \in \mathbb{R}^{n_{q_i}}$, $i = 1, \ldots, N_Q$

Output: Check if $\mathcal{P} \subseteq \cup_{i=1}^{N_Q} \mathcal{Q}_i$, {**true**, **false**}

```
 1. $k \leftarrow 1$
 2. if $\exists i \in \mathbb{N}_{1:N_Q} : n_{q_i} = 0$ then
 3.   return $\mathcal{R}$
 4. end if
 5. while $\nexists x \in \mathbb{R}^n : Px < p,\ Q_k x < q_k$ do
 6.   $k \leftarrow k + 1$
 7.   if $k > N_Q$ then
 8.     return false
 9.   end if
10. end while
11. for $j = 1$ to $n_{q_k}$ do
12.   if $\exists x \in \mathbb{R}^n : Px \leq P^c,\ [Q_k]_{(j)} x > [q_k]_{(j)}$ then
13.     if $k = N_Q$ then
14.       return false
15.     end if
16.     $\tilde{\mathcal{P}} = \mathcal{P} \cap \{x \mid [Q_k]_{(j)} x \geq [q_k]_{(j)}\}$
17.     if polycover$(\tilde{\mathcal{P}}, \{\mathcal{Q}_i\}_{i=k+1}^{N_Q})$ = false then
18.       return false
19.     end if
20.     $\mathcal{P} \leftarrow \mathcal{P} \cap \{x \mid [Q_k]_{(j)} x \leq [q_k]_{(j)}\}$
21.   end if
22. end for
23. return true
```

where $x \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is the optimization variable, n_r is the number of real valued variables, n_b is the number of binary (or, in general, integer) variables, and matrices $f \in \mathbb{R}^n$, $G \in \mathbb{R}^{n_g \times n}$, $g \in \mathbb{R}^{n_g}$, with $n = n_r + n_b$, are given problem parameters. □

We note that $\mathcal{P}$ is not fully covered by $\mathcal{Q}$, i.e.

$$\mathcal{P} \nsubseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i) \tag{33}$$

if and only if there is a point x inside of $\mathcal{P}$ that violates at least one of the constraints of each $\mathcal{Q}_i$, $i = 1, \ldots, N_Q$. This is equivalent to the following set of conditions

$$\exists x \in \mathcal{P} : \exists j_i \in \{1, \ldots, n_{q_i}\},\ [Q_i]_{(j_i)} x - [q_i]_{(j_i)} > 0, \\ i = 1, \ldots, N_Q. \tag{34}$$

To express this violation of constraints we introduce slack variables

$$y_{i,j}(x) = \begin{cases} [Q_i]_{(j)} x - [q_i]_{(j)} & \text{if} \quad [Q_i]_{(j)} x - [q_i]_{(j)} \geq 0, \\ 0 & \text{if} \quad [Q_i]_{(j)} x - [q_i]_{(j)} \leq 0, \end{cases} \\ j = 1, \ldots, n_{q_i}, \quad i = 1, \ldots, N_Q. \tag{35}$$

The expression (34) can now be posed as a feasibility question in x and $y_{i,j}$

$$\begin{array}{rcl} Px & \leq & p, \\ \sum_{j=1}^{n_{q_i}} y_{i,j} & > & 0, \quad i = 1, \ldots, N_Q \end{array} \tag{36}$$

Checking the condition (36) is still not possible with standard solvers, since the relation (35) describes a non-linear function. However, by introducing auxiliary binary variables one can rewrite (35) as the following equivalent set of linear inequalities (cf. [41])

$$\begin{bmatrix} 0 & -m_L \\ 0 & -M_U \\ 1 & -M_U \\ -1 & m_L \\ 1 & -m_L \\ -1 & M_U \end{bmatrix} \begin{bmatrix} y_{i,j} \\ \delta_{i,j} \end{bmatrix} \leqslant \begin{bmatrix} [Q_i]_{(j)} x - [q_i]_{(j)} - m_L \\ -[Q_i]_{(j)} x + [q_i]_{(j)} \\ 0 \\ 0 \\ [Q_i]_{(j)} x - [q_i]_{(j)} - m_L \\ -[Q_i]_{(j)} x + [q_i]_{(j)} + M_U \end{bmatrix} \\ \delta_{i,j} \in \{0,1\}, \\ j = 1, \ldots, n_{q_i}, \\ i = 1, \ldots, N_Q, \tag{37}$$

where $\delta_{i,j}$ are auxiliary binary variables and $m_L, M_U \in \mathbb{R}$ are bounds on constraint expressions that can be precomputed (or overestimated) beforehand

$$\begin{array}{rcl} m_L \leq & \min\limits_{x,i,j} & [Q_i]_{(j)} x - [q_i]_{(j)} \\ & \text{subj. to} & Px \leq p \\ & & j \in \{1, \ldots, n_{q_i}\} \\ & & i \in \{1, \ldots, N_Q\} \end{array} \tag{38}$$

$$\begin{array}{rcl} M_U \geq & \min\limits_{x,i,j} & [Q_i]_{(j)} x - [q_i]_{(j)} \\ & \text{subj. to} & Px \leq p \\ & & j \in \{1, \ldots, n_{q_i}\} \\ & & i \in \{1, \ldots, N_Q\} \end{array} \tag{39}$$

Actually, in terms of the number of inequalities that are used, (37) can be further simplified to

$$\begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 1 & -m_L \\ 1 & -M_U \end{bmatrix} \begin{bmatrix} y_{i,j} \\ \delta_{i,j} \end{bmatrix} \leqslant \begin{bmatrix} 0 \\ -[Q_i]_{(j)} x + [q_i]_{(j)} \\ [Q_i]_{(j)} x - [q_i]_{(j)} - m_L \\ 0 \end{bmatrix} \\ \delta_{i,j} \in \{0,1\}, \\ j = 1, \ldots, n_{q_i}, \\ i = 1, \ldots, N_Q. \tag{40}$$

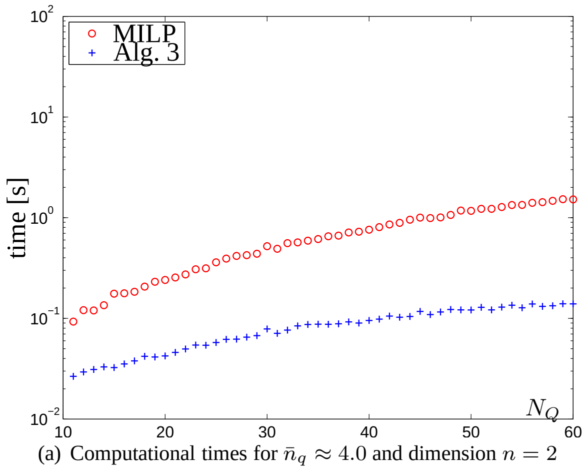


Fig. 10. Setup 1. Comparison of MILP (41) *and Algorithm 3 when solving* **polycover**$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$*, with* $\mathcal{P} \subset \mathbb{R}^n$*,* $n \in \{2,3\}$*, and varying* $N_Q = 11,\dots,60$*. Reported computational times are average values for* 100 *random tests per each scenario, i.e., value of* N_Q*.*

Since (36) and (40) describe a Mixed Integer Linear Programming (MILP) feasibility problem it follows that we can check if $\mathcal{P} \not\subseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ by solving an MILP feasibility problem.

However, instead of solving a feasibility MILP problem with (36) it may be more useful (and numerically robust) to solve the following optimality MILP problem

$$\max_{\lambda, x, \delta_{i,j}, y_{i,j}} \quad \lambda$$
$$\text{subj. to} \quad \left\{ \begin{array}{rcll} Px & \leq & p, & \\ \sum_{j=1}^{n_{q_i}} y_{i,j} & \geq & \lambda, & i = 1,\dots,N_Q \\ \sum_{j=1}^{n_{q_i}} \delta_{i,j} & \geq & 1, & i = 1,\dots,N_Q \\ & & \text{constraints (40)} & \end{array} \right. \tag{41}$$

Effectively, the optimal value λ^* is related to the size of the largest non-covered part of $\mathcal{P}$.

Theorem 1 *Let* λ^* *be the solution to the problem* (41)*, then* $\mathcal{P} \not\subseteq (\cup_{i=1}^{N_Q} \mathcal{Q}_i)$ *if and only if* $\lambda^* > 0$. □

Proof: Follows from the construction of the MILP problem (41). ■

Remark 10 *Strictly speaking, condition* $\sum_{j=1}^{n_{q_i}} \delta_{i,j} \geq 1$ *in* (41) *is redundant, but it reduces the problems with the integrality tolerances in existing MILP solvers. Also note that when solving* (41) *there is no need for condition* $y_{i,j} \geq 0$ *(first row in constraints* (40)*).* □

The MILP problem (41) has $n_p + 2N_Q + 3\sum_{i=1}^{N_Q} n_{q_i}$ constraints, $n + 1 + \sum_{i=1}^{N_Q} n_{q_i}$ real variables and $\sum_{i=1}^{N_Q} n_{q_i}$ binary variables.

6 TESTING OF ALGORITHMS

In this section we compare performances of the two approaches proposed in Section 5 for computing **polycover**$(\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q})$, i.e., checking if $\mathcal{P} \subseteq \{\mathcal{Q}_i\}_{i=1}^{N_Q}$.

Testing was carried out on a 2.4 GHz Pentium 4 machine with 1.5 GB RAM. The **polycover** problem was solved with MPT 2.6 [24] under MATLAB 7.01 in case of Algorithm 3, and with CPLEX 9.0 [42] in case of the MILP formulation (41).

Setup 1

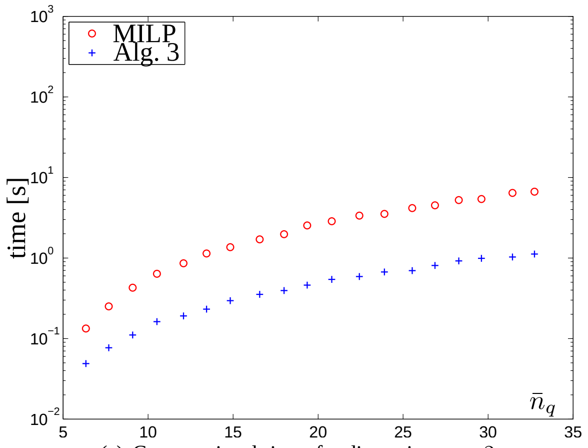
Series of random tests were performed in several dimensions n, with varying number of regions N_Q and average number of constraints

$$\bar{n}_q := \frac{\sum_{i=1}^{N_Q} n_{q_i}}{N_Q} = \frac{M}{N_Q}. \tag{42}$$

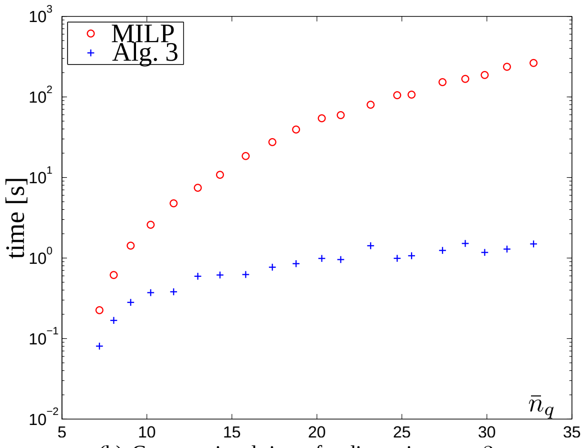
In all testing instances $\mathcal{P} \subset \mathbb{R}^n$ was chosen as a full-dimensional polytope centered at the origin with (a random) Chebyshev radius between 10 and 15. P-collection $\mathcal{Q}$ is such that $\mathcal{Q}_i \subset \mathbb{R}^n$, $i = 1,\dots,N_Q$, are full-dimensional polytopes with an average Chebyshev radius equal to 10 and Chebyshev centers randomly placed (with uniform distribution) within a hypercube $[-10, 10]^n$. Testing results for various scenarios are reported in Fig. 10 and Fig. 11. We note that Algorithm 3 is always better than the MILP formulation (41). This superiority is more pronounced for larger dimensions of the space.

Setup 2

To illustrate effectiveness of Algorithm 3 we consider the following **polycover** problem setup that should always

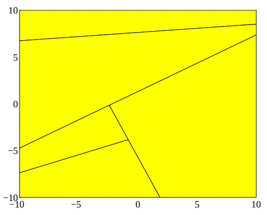


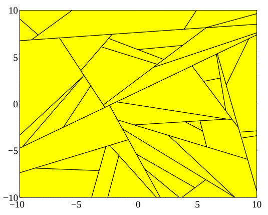
(a) Computational times for dimension $n = 2$

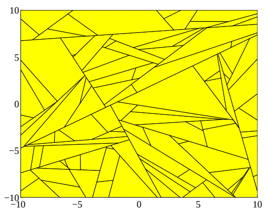


(b) Computational times for dimension $n = 3$

Fig. 11. Setup 1. Comparison of MILP (41) *and Algorithm 3 when solving* **polycover**($\mathcal{P}, \{\mathcal{Q}_i\}_{i=1}^{N_Q}$)*, with* $\mathcal{P} \subset \mathbb{R}^n$*,* $n \in \{2, 3\}$*,* $N_Q = 20$*, and varying* $\bar{n}_q \in [6.8\ 33]$*. Reported computational times are average values for* 100 *random tests per each scenario, i.e., value of* $\bar{n}_q$*.*


(a) Partition for $N_Q = 5$


(b) Partition for $N_Q = 40$


(c) Partition for $N_Q = 100$

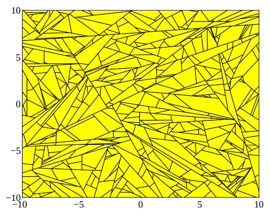

(d) Partition for $N_Q = 500$

Fig. 12. Setup 2. Illustration of a P-collection $\mathcal{Q}$ *generation in* $\mathbb{R}^2$*.*

give the value **true**. Polytope $\mathcal{P} \subset \mathbb{R}^n$ is chosen as a hypercube $[-10, 10]^n$ centered at the origin. For a fixed dimension $n \in \{2, \dots, 12\}$ and successive values of N_Q the P-collection $\mathcal{Q}$ is constructed as a polytopic partition of $\mathcal{P}$ in the following (iterative) manner: for $N_Q = 1$, $\mathcal{Q}_1 = \mathcal{P}$; for $N_Q > 1$ a randomly generated point x_{N_Q} in $[-10, 10]^n$ decides which polytope Q_i, $i \in \mathbb{N}_{1:N_Q-1}$, is split into two polytopes by a random hyperplane passing trough x_{N_Q} (i.e., through the interior of $\mathcal{Q}_i$) thus creating new polytopic partition $\mathcal{Q}$ of $\mathcal{P}$ with N_Q polytopes. See Fig. 12 for illustration of the above procedure in $\mathbb{R}^2$.

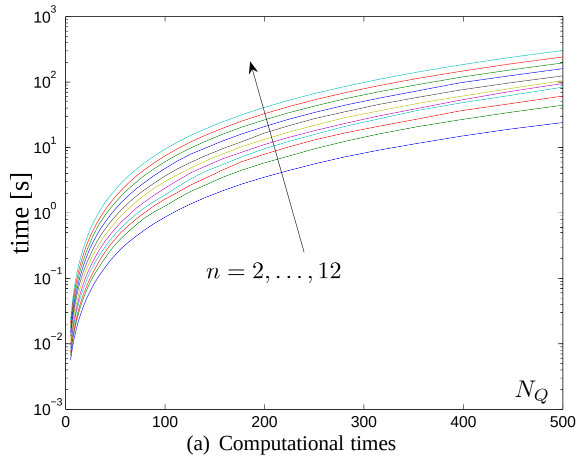
Testing results for varying N_Q and different dimensions of the space n are reported in Fig. 13. We point out that for most of the instances reported in Fig. 13 the MILP formulation (41) is practically intractable (i.e., it produces no solution even after 24 hours of computation). Perhaps the most informative view of the testing results is reported in Fig. 14, which indicates that the computational time of Algorithm 3 (for all dimensions considered in Setup 2) grows approximately with the square of the number of regions in P-collection $\mathcal{Q}$.

7 CONCLUSION

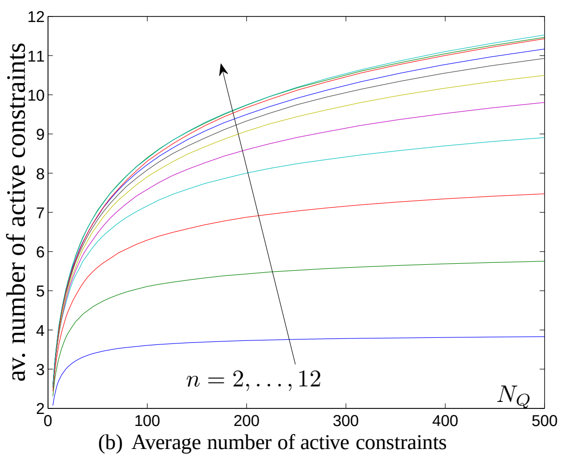
We have reviewed standard polytopic operations that are utilized when deriving explicit form of the optimal and robust control for discrete-time systems.

We have analyzed in great detail two special classes of these polytopic operations: the so-called *regiondiff* problem where the set difference between a polyhedron and union of polyhedra is computed, and the *polycover* problem where one is interested in checking if a polytope is covered by the union of other polytopes.

We have devised an in-place depth-first exploration algorithm that solves in an efficient manner both the regiondiff problem and, as a special case, the polycover problem. Furthermore, we have derived the strict upper bound for the computational complexity of this algorithm. In large number of random test we have shown that our polycover algorithm is superior to the mixed integer linear programming solution to the same problem.



(a) Computational times



(b) Average number of active constraints

Fig. 13. Setup 2. Computational times and average number of active constraints in Algorithm 3 for different dimensions of the space $n = 2, \dots, 12$*, and varying number of polytopes* $N_Q \in \{5, \dots, 500\}$*. Reported values are average values computed over* 100 *random tests per each scenario, i.e., value of* n *and* N_Q*.*

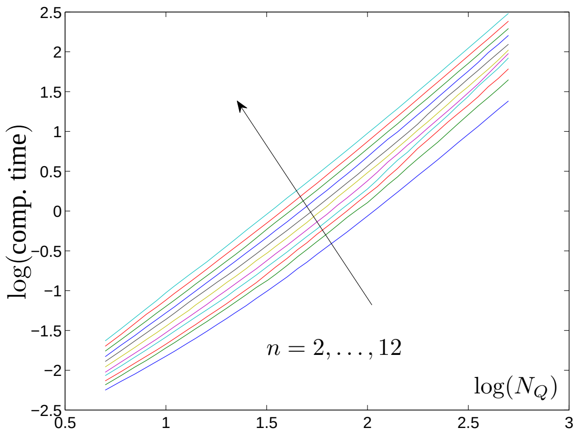


Fig. 14. Setup 2. Logarithmic scale representation of computational times reported in Fig. 13

ACKNOWLEDGMENT

A large portion of reported research was carried out while the author was working at the Automatic Control Laboratory, ETH Zurich, Switzerland. Support of the Croatian Ministry of Science under grant No. 036-0361621-3012 and grant No. 036-0363078-3018 is kindly acknowledged. The author would also like to thank reviewers for helpful suggestions for improvement of the paper.

REFERENCES

[1] H. S. Witsenhausen, **A Minimax Control Problem for Sampled Linear Systems**, IEEE Trans. on Automatic Control, vol. 13, no. 1, pp. 5–21, 1968.

[2] D. P. Bertsekas and I. B. Rhodes, **On the Minimax Reachability of Target Sets and Target Tubes**, Automatica, vol. 7, no. 2, pp. 233–247, March 1971.

[3] J. Glover and F. Schweppe, **Control of linear dynamic systems with set constrained disturbances**, IEEE Trans. on Automatic Control, vol. 16, no. 5, pp. 411–423, 1971.

[4] D. P. Bertsekas, **Infinite-time Reachability of State-Space Regions by Using Feedback Control**, IEEE Trans. on Automatic Control, vol. 17, no. 5, pp. 604–613, 1972.

[5] F. Blanchini and S. Miani, **Set-Theoretic Methods in Control**. Berlin: Birkhäuser, 2007.

[6] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, **The explicit linear quadratic regulator for constrained systems**, Automatica, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[7] A. Bemporad, F. Borrelli, and M. Morari, **Model Predictive Control Based on Linear Programming—The Explicit Solution**, IEEE Trans. on Automatic Control, vol. 47, no. 12, pp. 1974–1985, Dec. 2002.

[8] A. Bemporad, F. Borrelli, and M. Morari, **Min-Max Control of Constrained Uncertain Discrete-Time Linear Systems**, IEEE Trans. on Automatic Control, vol. 48, no. 9, pp. 1600–1606, Sept. 2003.

[9] F. Borrelli, **Constrained Optimal Control of Linear and Hybrid Systems**, ser. Lecture Notes in Control and Information Sciences. Springer, 2003, vol. 290.

[10] M. Baotić, F. Borrelli, A. Bemporad, and M. Morari, **Efficient On-Line Computation of Constrained Optimal Control**, SIAM Journal on Control and Optimization, vol. 47, no. 5, pp. 2470–2489, Sept. 2008. [Online]. Available: http://control.ee.ethz.ch

[11] S. V. Raković, E. C. Kerrigan, D. Q. Mayne, and J. Lygeros, **Reachability Analysis of Discrete-Time Systems With**

Disturbances, IEEE Trans. on Automatic Control, vol. 51, no. 4, pp. 546–561, 2006.

[12] M. Baotić, F. J. Christophersen, and M. Morari, **Constrained Optimal Control of Hybrid Systems With a Linear Performance Index**, IEEE Trans. on Automatic Control, vol. 51, no. 12, pp. 1903–1919, Dec. 2006.

[13] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, **Dynamic programming for constrained optimal control of discrete-time linear hybrid systems**, Automatica, vol. 41, pp. 1709–1721, Oct. 2005.

[14] P. Grieder, M. Kvasnica, M. Baotić, and M. Morari, **Stabilizing low complexity feedback control of constrained piecewise affine systems**, Automatica, vol. 41, pp. 1683–1694, Oct. 2005.

[15] M. Vašak and N. Perić, **Combining identification and constrained optimal control of piecewise affine systems**, Automatika, Journal for Control, Measurement, Electronics, Computing and Communications, vol. 48, no. 3–4, pp. 145–160, 2007.

[16] M. Baotić, **Optimal Control of Piecewise Affine Systems – a Multi-parametric Approach**, Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, Mar. 2005. [Online]. Available: http://control.ee.ethz.ch

[17] L. Khachiyan, **A polynomial algorithm in linear programming**, Soviet Mathematics Doklady, vol. 20, pp. 191–194, 1979.

[18] A. Schrijver, **Theory of Linear and Integer Programming**. New York: John Wiley & Sons, Inc., 1986.

[19] N. Karmarkar, **A new polynomial-time algorithm for linear programming**, Combinatorica, vol. 4, pp. 373–395, 1984.

[20] C. Roos, T. Terlaky, and J. Vial, **Theory and Algorithms for Linear Optimization: An Interior Point Approach**. New York: John Wiley & Sons, Inc., 1997.

[21] D. den Hertog, **Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity**, ser. Mathematics and its Applications. Dordrecht: Kluwer Academic Publishers, 1994, no. 277.

[22] G. M. Ziegler, **Lectures on Polytopes**. Springer, 1994.

[23] B. Grünbaum, **Convex Polytopes**, 2nd ed. Springer-Verlag, 2000.

[24] M. Kvasnica, P. Grieder, and M. Baotić, **Multi-Parametric Toolbox (MPT)**, 2004. [Online]. Available: http://control.ee.ethz.ch/~mpt

[25] K. L. Clarkson, **More output-sensitive geometric algorithms**, In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 695–702.

[26] S. Boyd and L. Vandenberghe, **Convex Optimization**. Cambridge University Press, 2004.

[27] S. Cernikov, **Contraction of finite systems of linear inequalities**, Doklady Akademiia Nauk SSSR, vol. 152, no. 5, pp. 1075–1078, 1963, in Russian (English translation in Societ Mathematics Doklady, Vol. 4, No. 5 (1963), pp.1520–1524).

[28] S. S. Keerthi and K. Sridharan, **Solution of parametrized linear inequalities by Fourier elimination and its applications**, J. Opt. Theory and Applications, vol. 65, no. 1, pp. 161–169, 1990.

[29] E. Balas, **Projection with a minimum system of inequalities**, Computational Optimization and Applications, vol. 10, pp. 189–193, 1998.

[30] K. Fukuda, T. M. Liebling, and C. Lütolf, **Extended convex hull**, in 12th Canadian Conference on Computational Geometry, July 2000, pp. 57–64.

[31] C. Jones, E. Kerrigan, and J. Maciejowski, **Equality set projection: A new algorithm for the projection of polytopes in halfspace representation**, Department of Engineering, Cambridge University, UK, Tech. Rep. CUED Technical Report CUED/F-INFENG/TR.463, 2004, http://www-control.eng.cam.ac.uk/~cnj22.

[32] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, **The quickhull algorithm for convex hulls**, ACM Trans. on Mathematical Software, vol. 22, no. 4, pp. 469–483, Dec. 1996. [Online]. Available: http://www.qhull.org

[33] K. Fukuda, **Cdd/cdd+ Reference Manual**, Dec. 1997, www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html.

[34] A. Bemporad, K. Fukuda, and F. Torrisi, **Convexity recognition of the union of polyhedra**, Computational Geometry, vol. 18, pp. 141–154, Apr. 2001.

[35] K. Fukuda and A. Prodon, **Double description method revisited**, Combinatorics and Computer Science, vol. 1120, pp. 91–111, 1996.

[36] D. Avis, **lrs: A revised implementation of the reverse search vertex enumeration algorithm**, Polytopes, Combinatorics and Computation, pp. 177–198, 2000.

[37] I. Kolmanovsky and E. G. Gilbert, **Theory and computation of disturbance invariant sets for discrete-time linear systems**, Mathematical Problems in Egineering, vol. 4, pp. 317–367, 1998.

[38] E. C. Kerrigan and J. M. Maciejowski, **On robust optimization and the optimal control of constrained linear systems with bounded state disturbances**, in Proc. 2003 European Control Conference, Cambridge, UK, Sept. 2003.

[39] R. Buck, **Partition of space**, American Mathematical Monthly, vol. 50, pp. 541–544, 1943.

[40] M. Vašak, M. Baotić, and N. Perić, **Efficient Computation of the One-Step Robust Sets for Piecewise Affine Systems With Polytopic Additive Uncertainties**, in Proceedings of the European Control Conference 2007, Kos, Greece, July 2007, pp. 1430–1435.

[41] A. Bemporad and M. Morari, **Control of systems integrating logic, dynamics, and contraints**, Automatica, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[42] ILOG, Inc., **CPLEX 9.0 User Manual**, Gentilly Cedex, France, 2005, http://www.ilog.fr/products/cplex.

Mato Baotić received the B.Sc. and M.Sc. degrees, both in Electrical Engineering, from the Faculty of Electrical Engineering and Computing (FER Zagreb), University of Zagreb, Croatia, in 1997 and 2000, respectively. As a recipient of the ESKAS scholarship of the Swiss Government he was a visiting researcher at the Automatic Control Lab, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, during the academic year 2000/2001. In 2005 he received the Ph.D. from the ETH Zurich, Switzerland. Currently he is Assistant Professor at the Department of Control and Computer Engineering, FER Zagreb, Croatia. His research interests include mathematical programming, hybrid systems, optimal control and model predictive control.

AUTHOR'S ADDRESS
Asst. Prof. Mato Baotić, Ph.D.
Department of Control and Computer Engineering,
Faculty of Electrical Engineering and Computing,
University of Zagreb,
Unska 3, HR-10000 Zagreb, Croatia
email: mato.baotic@fer.hr

Received: 2009-11-02
Accepted: 2009-11-27