Optimal Complexity Reduction of Polyhedral Piecewise Affine Systems *

Tobias Geyer [1,2], Fabio D. Torrisi [3], Manfred Morari

Automatic Control Laboratory, ETH Zurich, 8092 Zurich, Switzerland

Abstract

This paper focuses on the $\mathcal{NP}$-hard problem of reducing the complexity of piecewise polyhedral systems (e.g. polyhedral piecewise affine (PWA) systems). The results are fourfold. Firstly, the paper presents two computationally attractive algorithms for optimal complexity reduction that, under the assumption that the system is defined over the cells of a hyperplane arrangement, derive an *equivalent* polyhedral piecewise system that is *minimal* in the number of polyhedra. The algorithms are based on the cells and the markings of the hyperplane arrangement. In particular, the first algorithm yields a set of disjoint (non overlapping) merged polyhedra by executing a branch and bound search on the markings of the cells. The second approach leads to non-disjoint (overlapping) polyhedra by formulating and solving an equivalent (and well-studied) logic minimization problem. Secondly, the results are extended to systems defined on general polyhedral partitions (and not on cells of hyperplane arrangements). Thirdly, the paper proposes a technique to further reduce the complexity of piecewise polyhedral systems if the introduction of an adjustable degree of error is acceptable. Fourthly, the paper shows that based on the notion of the hyperplane arrangement PWA state feedback control laws can be implemented efficiently. Three examples, including a challenging industrial problem, illustrate the algorithms and show their computational effectiveness in reducing the complexity by up to one order of magnitude.

Key words: Optimal complexity reduction; Model reduction; Controller reduction; Piecewise affine system; Hybrid system; Boolean minimization; Hyperplane arrangement

1 Introduction

This paper focuses on the problem of finding a *minimal* representation of polyhedral piecewise systems. More specifically, for a given polyhedral piecewise system, we solve the problem of deriving a polyhedral piecewise system that is both equivalent to the former and minimal in the number of polyhedra. We refer to this as *optimal complexity reduction*.

Polyhedral piecewise systems are defined by partitioning a (polyhedral) input-space into polyhedra and associating with each polyhedron a function. Major subclasses of polyhedral systems are polyhedral piecewise *polynomial* systems (with polynomial functions) and polyhedral piecewise *affine* (PWA) systems (Sontag, 1981), where the functions are affine. In particular, PWA systems represent a universal modelling framework to describe hybrid systems (systems with continuous and discrete components and variables). Concerning synthesis, constrained optimal control problems, which traditionally could be solved only on-line, can be pre-solved off-line, too. If the underlying (prediction) model is of PWA form (and a linear objective function is used), this leads to an explicit state-feedback control law that is also PWA (Borrelli, 2003; Borrelli et al., 2005).

In the sequel, we will mostly consider PWA models and PWA state-feedback control laws. The motivation for deriving a PWA model minimal in the number of polyhedra is twofold. When pre-computing the optimal control law off-line to derive the state-feedback controller, an internal PWA model is required. Due to the combinatorial nature of the problem, both the computation time and the controller complexity are (in the worst case) exponential in the number of polyhedra of the PWA model (Borrelli, 2003).

On the other hand, once the PWA state-feedback control law has been derived, the memory requirement and

* This paper was not presented at any IFAC meeting.
[1] Corresponding author. Tel.: +49-89-5528-3435; fax: +49-89-5528-3180; email address: geyer@control.ee.ethz.ch
[2] T. Geyer is currently with GE Global Research Europe, Munich, Germany
[3] F.D. Torrisi is currently with McKinsey & Company, Zurich, Switzerland

the on-line computation time are linear in the number of polyhedra of the feedback law when using standard brute force search. When using a binary search tree as proposed in Tøndel et al. (2003), the computational burden can be reduced at the expense of enlarging the memory requirement. More precisely, the computation time becomes sublinear in the number of polyhedra, while the memory requirement becomes superlinear.

If the number of polyhedra with the same function is large, the number of possible polyhedral combinations for merging explodes; as shown in Chazelle (1984) and references therein this task is $\mathcal{NP}$-hard. Since most of these unions are not convex or even not connected and thus cannot be merged, trying all combinations using standard techniques based on *linear programming* (LP) as suggested in Bemporad et al. (2001) is prohibitive. Furthermore, our objective here is not only to reduce the number of polyhedra but rather to find the minimal and thus optimal number of polyhedra. To the best of our knowledge, the derivation of such algorithms is still an open problem.

The problem can be tackled by using the notion of cells and markings in a hyperplane arrangement. The cells are the polyhedra generated by the hyperplane arrangement. These cells can be uniquely identified by their markings, i.e. their relative positions with respect to the hyperplanes. Using the markings enables us to determine *a priori* – i.e. without solving any LP – if a given combination of polyhedra is convex.

Exploiting this fact, we propose in this paper two algorithms that yield the minimal number of polyhedra without solving any LP. The first algorithm executes a branch and bound search on the markings yielding a set of disjoint (non-overlapping) merged polyhedra. Additional heuristics on the branching strategy are used to reduce the computation time. The second approach relies on the fact that the optimal complexity reduction problem can be reformulated as a logic minimization problem by replacing the markings by Boolean variables and minterms. Logic minimization is a fundamental issue in digital circuit design, and efficient tools have been developed to successfully tackle these problems, which often encounter hundreds or thousands of variables. The resulting polyhedra, however, are not disjoint in general, but overlapping. We would like to stress that – since both algorithms refrain from solving additional LPs – they are not only optimal but also computationally feasible. In many cases, the hyperplane arrangement and its markings are available. This is the case, for example, when a PWA model has been derived using the mode enumeration algorithm (Geyer et al., 2003). Nevertheless, the applicability of the algorithms can be extended to polyhedral piecewise systems lacking the hyperplane arrangement (like PWA state-feedback control laws) by first computing the hyperplane arrangement and the markings. For this, standard and efficient techniques are available including reverse search (Avis and Fukuda, 1996).

This paper is organized as follows. Section 2 recalls basic terminology, defines polyhedral PWA systems, introduces hyperplane arrangements, and summarizes the concept of cell enumeration. The (disjoint and non-disjoint) optimal complexity reduction problems are formally stated in Section 3. The key lemma to evaluate convexity of polyhedra (using only their markings) is proven in Section 4. Algorithms for optimal complexity reduction based on branch and bound, and logic minimization are proposed in Sections 5 and 6, respectively. Section 7 summarizes an algorithm to derive the hyperplane arrangement. By simplifying the hyperplane arrangement the complexity of the solution can be further reduced at the expense of an adjustable degree of error. The optimality of the algorithms is shown and further elaborated on in Section 8. In Section 9, the effectiveness of the approaches is demonstrated by three examples including a large industrial problem. Techniques to efficiently implement PWA state-feedback control laws are outlined in Section 10, and conclusions are drawn in Section 11.

In the sequel, we will often abbreviate *optimal complexity reduction* with OCR. The OCR algorithms have been implemented in Matlab and are included in the multi-parametric toolbox (MPT) developed by Kvasnica et al. (2004). The toolbox is freely available from `http://control.ee.ethz.ch/~mpt/`.

2 Preliminaries

2.1 Basic Terminology

Definition 1 (Polyhedron) *A convex set $\mathcal{P}$ in the d-dimensional Euclidian space $\mathbb{R}^d$ given by $\mathcal{P} = \{x \in \mathbb{R}^d \mid a^T x \leq b\}$ is called a* polyhedron *with $a \in \mathbb{R}^{d\times n}$ and $b \in \mathbb{R}^n$. The operator $\leq$ denotes an element-wise comparison of two vectors.*

Equivalently, the polyhedron $\mathcal{P}$ can be considered as the intersection of a finite number (here n) of half spaces $\{x \in \mathbb{R}^d \mid a_i^T x \leq b_i\}$ induced by the hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$. In particular, a and b hold the n hyperplanes, i.e. $a = [a_1, \ldots, a_n]$ and $b = [b_1, \ldots, b_n]^T$.

Definition 2 (Facet) *If $\mathcal{P} \cap H_i$ is $(d-1)$-dimensional then $\mathcal{P} \cap H_i$ is called a* facet *of the polyhedron $\mathcal{P}$.*

Definition 3 (Polyhedral Partition) *A collection of polyhedra $\mathcal{P}_i \subseteq \mathcal{R}$, $i \in \mathcal{I} \subset \mathbb{N}$, is a* polyhedral partition *of the polyhedron $\mathcal{R}$, iff*

$$(i) \quad \bigcup_{i \in \mathcal{I}} \mathcal{P}_i = \mathcal{R}\,, \tag{1a}$$

$$(ii) \quad \mathcal{P}_i \cap \mathcal{P}_j \text{ is lower-dimensional} \, \forall i, j \in \mathcal{I}, \; i \neq j \tag{1b}$$

2.2 Polyhedral Piecewise Affine Systems

Polyhedral Piecewise Affine (PWA) systems (Sontag, 1981; Heemels et al., 2001) are defined by partitioning the polyhedral input-space $\mathcal{X}$ into polyhedra and associating with each polyhedron an affine output function

$$y = f_j(x) \tag{2a}$$

$$\text{with } j \text{ such that } x \in \mathcal{P}_j\,, \tag{2b}$$

where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ denote the (vector-valued) input and output, respectively, the polyhedra $\mathcal{P}_j$ define a set of polyhedra $\{\mathcal{P}_j\}_{j \in \mathcal{J}}$ on $\mathcal{X}$, and f_j is a vector-valued affine function. We refer to $j \in \mathcal{J}$, with $\mathcal{J}$ finite, as the *mode* of the system and to $\#\mathcal{J}$ as the number of modes.

Remark 4 *Strictly speaking* (1b) *gives rise to double definitions of the PWA function* (2) *over the boundaries of the polyhedra. This issue can be accommodated by using a disambiguation rule for the cases where there exist i, j such that $f_i(x) \neq f_j(x)$ for $x \in \mathcal{P}_i \cap \mathcal{P}_j$. In any case, merging is only possible when $f_i(x) = f_j(x)$ for $x \in \mathcal{P}_i \cup \mathcal{P}_j$. Therefore, since $\mathcal{P}_i \cap \mathcal{P}_j \subset \mathcal{P}_i \cup \mathcal{P}_j$ holds, boundaries that would require disambiguation are clearly preserved by the algorithms presented in the remainder of the paper and can be handled by trivial modifications to the algorithms. These modifications are omitted for ease of reading.*

Throughout this paper, we will consider two forms of polyhedral PWA systems. *PWA models* are dynamical system representations with a PWA state and a PWA output equation, which are functions of the state and the input. *PWA state-feedback control laws* are a controller representation, where the control input is a PWA function of the state.

Since our line of research is mostly concerned with polyhedral PWA systems, we (artificially) restrict ourselves in this paper to such systems. Nevertheless, the whole framework including the proposed algorithms and theorems also holds for polyhedral piecewise systems in general including e.g. polyhedral piecewise *polynomial* systems. Specifically, f_j in (2a) is not required to be affine.

2.3 Hyperplane Arrangements

Let $\mathcal{A}$ be a collection of n distinct hyperplanes $\{H_i\}_{i=1,\ldots,n}$ in the d-dimensional Euclidian space $\mathbb{R}^d$. We say that the hyperplanes of $\mathcal{A}$ are in *general position*, if there exists no pair of parallel hyperplanes, and

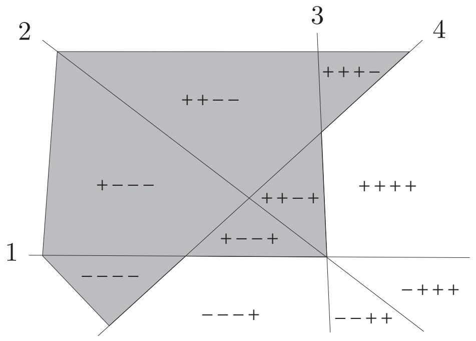


Fig. 1. Arrangement of four hyperplanes (lines) in $\mathcal{R} = \mathbb{R}^2$ with the corresponding markings $m \in M(\mathcal{R})$. Regarding Example 16, the polyhedra corresponding to M_w are white and the polyhedra corresponding to M'_b are grey shaded, respectively

if any point of $\mathbb{R}^d$ belongs at most to d hyperplanes. Let $\mathrm{SV} : \mathbb{R}^d \to \{-, +\}^n$ be the simplified sign vector [4] defined as

$$\mathrm{SV}_i(x) = \begin{cases} - & \text{if } a_i^T x \leq b_i, \\ + & \text{if } a_i^T x \geq b_i \end{cases} \quad \text{for } i \in \{1, 2, \ldots, n\}\,. \tag{3}$$

Consider the set $\mathcal{P}_m = \{x \in \mathbb{R}^d \mid \mathrm{SV}(x) = m\}$ for a given sign vector m. This set is called a *cell* of the arrangement and is according to Definition 1 a polyhedron as it is defined by linear inequalities. We will refer to m as the *marking* of the polyhedron (or cell) $\mathcal{P}_m$ in the *hyperplane arrangement* $\mathcal{A}$ (see Fig. 1). Let $M(\mathcal{R})$ be the image of the function $\mathrm{SV}(x)$ for $x \in \mathcal{R} \subseteq \mathbb{R}^d$, namely the collection of all possible markings of all points in $\mathcal{R}$.

Let the '$*$' element extend the sign vector in the sense that it denotes the union of cells, where the corresponding hyperplane is not a facet of the associated polyhedron $\mathcal{P}_m$. As an example, consider in Fig. 1 the two polyhedra with the markings $m_1 = ----$ and $m_2 = +---$. Then, $m = *---$ is equivalent to $\{m_1, m_2\}$ and refers to $\mathcal{P}_{m_1} \cup \mathcal{P}_{m_2}$.

2.4 Cell Enumeration in Hyperplane Arrangements

The cell enumeration problem in a hyperplane arrangement amounts to enumerate all the elements of the set $M(\mathcal{R})$. Let $\#M(\mathcal{R})$ be the number of cells identified by $M(\mathcal{R})$. For n hyperplanes in a d-dimensional space Buck

[4] Note that in general, the sign vector is defined such that its image is $\{-, 0, +\}$, where the '0' element corresponds to $a_i^T x = b_i$. As noted in Remark 4 complexity reduction will only attempt to remove hyperplanes separating polyhedra such that $f_i(x) = f_j(x)$ for $x \in \mathcal{P}_i \cup \mathcal{P}_j$. Therefore, the double definition of $\mathrm{SV}_i(x)$ for $a_i^T x = b_i$ is not an issue.

(1943) defines the upper bound

$$\#M \leq \sum_{i=0}^{d} \binom{n}{i} = O(n^d), \tag{4}$$

with the equality satisfied if the hyperplanes are in general position and $\mathcal{R} = \mathbb{R}^d$.

Edelsbrunner (1987) showed that the cell enumeration problem admits an optimal solution with time and space complexity $O(n^d)$. An alternative approach based on reverse search was presented by Avis and Fukuda (1996), improved by Ferrez et al. (2001) and implemented by Ferrez and Fukuda (2002). Reverse search is an exhaustive search technique that can be considered as a special graph search. This search technique has been used to design efficient algorithms for various enumeration problems such as enumeration of all spanning trees and cells in hyperplane arrangements.

Proposition 5 (Ferrez et al. (2001, Theorem 4.1)) *There exists a reverse search algorithm for enumerating hyperplane arrangements that runs in $O(n\,\mathrm{lp}(n,d)\,\#M)$ time and $O(nd)$ space, where $\mathrm{lp}(n,d)$ denotes the complexity of solving a Linear Program (LP) with n constraints and d variables.*

Note that in many cases of interest, the hyperplanes are not in general position and $\#M$ is considerably smaller than the theoretical upper bound. Moreover, reverse search is a standard search algorithm for which efficient parallel implementations exist (Brungger et al., 1999).

From the definition of $\mathcal{P}_m$ and (3) follows directly that the collection of polyhedral sets $\{\mathcal{P}_m\}_{m \in M(\mathcal{R})}$ is a polyhedral partition of $\mathcal{R}$.

3 Problem Statement

In the following, we assume that besides the PWA representation a corresponding hyperplane arrangement $\mathcal{A}$ is available together with the markings $M(\mathcal{R})$[5]. Specifically, we assume the following.

Assumption 1 *The polyhedra of the given PWA system are cells in a hyperplane arrangement, of which the markings are available.*

For a given PWA representation the aim of the OCR algorithms is to derive an equivalent representation that is minimal in the number of polyhedra by replacing polyhedra with the same (affine) function by new sets of polyhedra of minimal cardinality. For clarity of exposition, we associate with each (affine) function a different color, and we collect the polyhedra with the same color. Then, for a given color, we pose the following three problems, where we distinguish between results formed by disjoint and non-disjoint polyhedra.

[5] In Section 7, we will relax this assumption and extend the OCR algorithms to general PWA system not defined in hyperplane arrangements.

Problem 6 (Disjoint Optimal Complexity Reduction (DOCR)) *Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color satisfying Assumption 1, the* disjoint *optimal complexity reduction problem amounts to deriving a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with the following properties: (i) the union of the new polyhedra is equal to the union of the original ones, i.e. $(\bigcup_{i=1}^{q} \mathcal{Q}_i) = (\bigcup_{i=1}^{p} \mathcal{P}_i)$, (ii) q is minimal, i.e. there exists no set $\{\mathcal{Q}'_i\}_{i=1,\dots,q'}$ with a smaller number of polyhedra, (iii) the new polyhedra are mutually disjoint, i.e. $\mathcal{Q}_i \cap \mathcal{Q}_j$ is lower-dimensional for all $i, j \in \{1, \dots, q\}$, $i \neq j$, and (iv) the new polyhedra are formed as unions of the old ones, i.e. for each $\mathcal{Q}_j$, $j \in \{1, \dots, q\}$, there exists an index set $\mathcal{I} \subseteq \{1, \dots, p\}$, such that $\mathcal{Q}_j = \bigcup_{i \in \mathcal{I}} \mathcal{P}_i$.*

This problem is equivalent to an optimal merging problem. Next, we remove Requirements (iii) and (iv) thus allowing overlaps in the resulting polyhedra. Additionally, we require that each polyhedron is represented by a minimal number of facets.

Problem 7 (Non-Disjoint Optimal Complexity Reduction (NOCR)) *Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color satisfying Assumption 1, the* non-disjoint *optimal complexity reduction problem amounts to deriving a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with Properties (i) and (ii) as in Problem 6. A secondary objective is to (iii) minimize the number of facets on $\mathcal{Q}_i$.*

Strictly speaking, the second problem is not a merging problem, but a more general optimal set covering problem, which is (as shown later) equivalent to logic minimization frequently used in digital circuit design.

Problem 8 (General Non-Disjoint Optimal Complexity Reduction (GNOCR)) *Given an initial set of polyhedra $\{\mathcal{P}_i\}_{i=1,\dots,p}$ with the same color, the* general non-disjoint *optimal complexity reduction problem amounts to deriving a new set of polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ with Properties (i), (ii) and (iii) as in Problem 7.*

All three tasks are non-trivial, because the union of polyhedra with the same color is in general non-convex and we are aiming at deriving the optimal solution, or more specifically, the set of polyhedra with the minimal cardinality. Indeed, the problems are $\mathcal{NP}$-hard (see Chazelle (1984) and references therein). As a direct consequence, fast algorithms are unlikely to exist leaving us either with rather long computation times or suboptimal solutions. Our goal is to design algorithms that are applicable to problems of meaningful size but nevertheless yield the global optimum.

4 Convexity and Connectivity of Polyhedral Sets

Definition 9 (Separating Hyperplane) *Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ are two (convex) polyhedra that do not intersect, i.e. $\mathcal{P}_1 \cap \mathcal{P}_2$ is lower-dimensional. A hyperplane $\{x \mid c^T x = d\}$ with $c \neq 0$ and d, such that $c^T x \leq d$ for all $x \in \mathcal{P}_1$ and $c^T x \geq d$ for all $x \in \mathcal{P}_2$ is called a* separating hyperplane *for the polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$.*

The proof of the following lemma follows directly from the definition of the markings.

Lemma 10 (Separating Hyperplane) *Given the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ consisting of n distinct hyperplanes, the set of markings $M(\mathcal{R})$, and the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ with the corresponding markings $m_1, m_2 \in M(\mathcal{R})$ that differ in the j-th component, then H_j is a separating hyperplane for $\mathcal{P}_1$ and $\mathcal{P}_2$.*

Definition 11 (Envelope) *Given two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, the* envelope $\text{env}(\mathcal{P}_1, \mathcal{P}_2)$ *of the two polyhedra is defined as the intersection of half spaces that contain both polyhedra, where the half spaces are induced by the facets of the polyhedra.*

Lemma 12 (Envelope) *Consider the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ consisting of n distinct hyperplanes, the set of markings $M(\mathcal{R})$, and the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$. Let these two polyhedra have the markings $m_1, m_2 \in M(\mathcal{R})$, where $m_1(i) = m_2(i)$ for $i \in \mathcal{I}$ and $m_1(i) \neq m_2(i)$ for $i \in \mathcal{I}'$ with $\mathcal{I}' = \{1, \dots, n\} \setminus \mathcal{I}$. We construct the marking m_e as follows: $m_e(i) = m_1(i)$ for $i \in \mathcal{I}$ and $m_e(i) =$'$*$' for $i \in \mathcal{I}'$. Then the envelope env$(\mathcal{P}_1, \mathcal{P}_2)$ of the two polyhedra is given by m_e.*

Proof Recall that a '$*$' in a marking means that the corresponding hyperplane does not define the polyhedron. As all the facets of $\mathcal{P}_1$ and $\mathcal{P}_2$ are subsets of the hyperplanes in the arrangement, and as the hyperplanes with indices $\mathcal{I}'$ are separating hyperplanes for $\mathcal{P}_1$ and $\mathcal{P}_2$ according to Lemma 10, the proof follows from the definition of the envelope. $\square$

The proof can be easily generalized to envelopes of more than two polyhedra.

Theorem 13 (Convexity, Bemporad et al. (2001, Theorem 3)) *Given the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, their union $\mathcal{P}_1 \cup \mathcal{P}_2$ is convex if and only if $\mathcal{P}_1 \cup \mathcal{P}_2 =$ env$(\mathcal{P}_1, \mathcal{P}_2)$.*

The following lemma allows us to determine the convexity of two polyhedra by only evaluating their markings. This lemma constitutes the basis for the OCR algorithms.

Lemma 14 (Convexity) *Given the collection of markings $M(\mathcal{R})$, the union of the two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ with the markings $m_1, m_2 \in M(\mathcal{R})$, $m_1 \neq m_2$, is convex, if and only if the markings differ in exactly one component.*

Proof As we have Theorem 13 at our disposal, we only need to prove that $\mathcal{P}_1 \cup \mathcal{P}_2 = \text{env}(\mathcal{P}_1, \mathcal{P}_2)$ if and only if m_1 and m_2 differ in exactly one component. The "$\Leftarrow$" part follows directly from Lemma 12. The "$\Rightarrow$" part follows by contradiction. Recall, that $\mathcal{P}_1 \cup \mathcal{P}_2 \subseteq \text{env}(\mathcal{P}_1, \mathcal{P}_2)$, and assume that $\mathcal{P}_1 \cup \mathcal{P}_2 \neq \text{env}(\mathcal{P}_1, \mathcal{P}_2)$, i.e. there are points $x \in \text{env}(\mathcal{P}_1, \mathcal{P}_2) \setminus (\mathcal{P}_1 \cup \mathcal{P}_2)$. Then there exists at least one hyperplane that is separating x from $\mathcal{P}_1$ or x from $\mathcal{P}_2$ besides the one that is separating $\mathcal{P}_1$ from $\mathcal{P}_2$. Thus m_1 and m_2 differ in at least two components. $\square$

The concept of markings in a hyperplane arrangement allows us to evaluate the convexity of polyhedra by applying Lemma 14 to their associated set of markings. The algorithms refrain from solving LPs – in fact, they extract the information from the markings that in turn summarize the result of the LPs solved to compute the cells of the hyperplane arrangement. Even though we will design algorithms assuring optimality, the computation times to solve the OCR problems are rather small (provided that Assumption 1 holds) making the algorithms applicable to problems of meaningful size.

Definition 15 (Connectivity) *Two polyhedra are called* neighboring *polyhedra if they share a common facet. A set of polyhedra $\{\mathcal{P}_i\}_{i \in \mathcal{I}}$ is* connected *if for each $\mathcal{P}_i$, $i \in \mathcal{I}$, there exists a $\mathcal{P}_j$, $i \neq j$, $j \in \mathcal{I}$ such that $\mathcal{P}_i$ and $\mathcal{P}_j$ are neighboring polyhedra.*

Obviously, a necessary condition for the convexity of a union of a set of polyhedra is that the set of polyhedra is connected. Connectivity can be easily determined using the markings. Given the set of markings $M(\mathcal{R})$ and the set of polyhedra $\{\mathcal{P}_{m_i}\}_{m_i \in M(\mathcal{R})}$ with markings in the set $M(\mathcal{R})$, the polyhedra are connected if and only if for each polyhedron $\mathcal{P}_{m_i}$ with marking $m_i \in M(\mathcal{R})$, there exists a polyhedron $\mathcal{P}_{m_j}$ with marking $m_j \in M(\mathcal{R})$, such that m_i and m_j differ in exactly one component. To reduce the computation time of the OCR algorithms, we will exploit this fact by further partitioning the set of polyhedra with the same color into connected subsets.

5 Disjoint Optimal Complexity Reduction

Let the set M_w denote the markings of a connected subset with the same color. We refer to the corresponding polyhedra as *white* polyhedra. As the color of the remaining polyhedra is not relevant at this stage, we assume that the remaining markings $M'_b = M(\mathcal{R}) \setminus M_w$ correspond to *black* polyhedra. The basic concept of the algorithm is to derive a minimal representation of the white polyhedra by dividing their envelope sequentially into polyhedra using the hyperplanes of the hyperplane arrangement.

5.1 Algorithm 1 – Branch and Bound

Let the envelope of the white polyhedra with markings M_w be denoted by $\mathcal{P}_{m_e}$. It is given by the marking m_e, which is constructed as in Lemma 12. Slightly abusing the notation we will write $m_e = \text{env}(M_w)$. As all white polyhedra are contained in their envelope, we can formulate an equivalent problem with reduced complexity that considers only the black polyhedra contained in this envelope, i.e. $M_b = \{m_b \in M_b' \mid \mathcal{P}_{m_b} \subseteq \mathcal{P}_{m_e}\}$, where $\mathcal{P}_{m_b}$ denotes the polyhedron with marking m_b.

Let $\mathcal{I} \in \{1, \ldots, n\}$ denote the index set of hyperplanes in $\mathcal{A}$ that are separating hyperplanes for polyhedra in the envelope $\mathcal{P}_{m_e}$. According to Lemma 10, $\mathcal{I}$ is simply the collection of indices i with $m_e(i)$ ='$*$'. Then, we can choose any hyperplane H_i, $i \in \mathcal{I}$, to divide $\mathcal{P}_{m_e}$ into two polyhedra. H_i also divides the sets of white and black markings respectively into two subsets. We denote the subset of M_w that holds those markings whose i-th element is a '$-$' with $M_{w|m(i)=-}$, i.e. $M_{w|m(i)=-} = \{m \in M_w \mid m(i) = '-'\}$. $M_{w|m(i)=+}$ and the partition of M_b are defined accordingly. Clearly, the unions of each pair of subset equal the original sets M_w and M_b, respectively. Next, the algorithm branches on the i-th hyperplane by calling itself twice – first with the arguments M_w and M_b restricted to possessing a '$-$' as i-th element, and then correspondingly with the arguments restricted to a '$+$'. Both function calls return sets of markings M_m corresponding to merged white polyhedra. This is repeated for all the remaining hyperplanes with indices $i \in \mathcal{I}$.

A branch terminates if one of the following two cases occurs. First, if the set of markings corresponding to black polyhedra is empty, i.e. $M_b = \emptyset$. This implies, that at this point the envelope contains only white polyhedra. Hence, the envelope represents the union of the set of white polyhedra with markings in M_w, and it is convex by construction. We will refer to this convex set as a *merged white* polyhedron. Second, a branch terminates if the set of markings corresponding to white polyhedra is empty, i.e. $M_w = \emptyset$, as this implies that no more white polyhedra are available for merging.

The algorithm uses standard bound techniques to cut off suboptimal branches by using the two variables z and $\bar{z}$. z denotes the current number of merged white polyhedra and $\bar{z}$ is the local upper bound on z. Initially, z is set to 0, and $\bar{z}$ is initialized to the number of original white polyhedra. Branching is only performed if $z < \bar{z}$, as branches with $z \geqslant \bar{z}$ are either equivalent to or worse than the current optimum.

The above described branch and bound algorithm is summarized in the following, where $\#M$ denotes the number of elements in the set M.

Algorithm 1
```
function $M_m = mrg(M_w, M_b', z, \bar{z})$
$m_e = env(M_w)$; $M_b = \{m_b \in M_b' \mid \mathcal{P}_{m_b} \subseteq \mathcal{P}_{m_e}\}$
if $M_w = \emptyset$ then $M_m = \emptyset$
elseif $M_b = \emptyset$ then $M_m = m_e$
else
    $\mathcal{I} = \{i \mid m_e(i) = '*'\}$; $M_m = \emptyset$
    for $i \in \mathcal{I}$
        if $z < \bar{z}$ then
            $M_{m_1} = mrg(M_{w|m(i)=-}, M_{b|m(i)=-}, z, \bar{z})$
            $M_{m_2} = mrg(M_{w|m(i)=+}, M_{b|m(i)=+}, z + \#M_{m_1}, \bar{z})$
            if $M_m = \emptyset$ or $\#M_{m_1} + \#M_{m_2} < \#M_m$ then
                $M_m = M_{m_1} \cup M_{m_2}$; $\bar{z} = \min(\bar{z}, z + \#M_m)$
return $M_m$
```

Example 16 *As an example with four hyperplanes in a two-dimensional space consider Fig. 1. The envelope of the white polyhedra is given by the positive half space of H_4 and the marking $m_e = ***+$. Thus, only the black polyhedra with markings $M_b = \{+--+, ++-+\}$ are considered, and branching is only performed on the hyperplanes in $\mathcal{I} = \{1, 2, 3\}$. Branching on H_1 leads in one step to the two merged (white) polyhedra with $M_m = \{-**+, ++++\}$. This is already the optimal solution. Nevertheless, the algorithm also branches on the two remaining hyperplanes in $\mathcal{I}$ and finds two additional solutions that are equivalent to the first one in terms of the number of polyhedra.*

Lemma 17 *Algorithm 1 solves the Disjoint Optimal Complexity Reduction Problem 6.*

Proof The proof follows in a constructive way from the algorithm. When branching on the i-th hyperplane H_i, the set of white markings is divided into the two sets $M_{w|m(i)=-}$ and $M_{w|m(i)=+}$ according to the two half spaces induced by H_i. This operation assures that the merged polyhedra are mutually disjoint. In particular, as no white polyhedra are discarded during the operation and since $M_w = (M_{w|m(i)=-}) \cup (M_{w|m(i)=+})$, the union of the merged polyhedra equals the union of the white polyhedra. The minimality of the number of merged polyhedra is ensured by branching on all hyperplanes unless bound techniques cut off suboptimal branches. □

We conclude that the proposed algorithm is computational efficient in the sense that the convexity recognition is performed only by comparing the markings rather than by solving LPs, it is optimal as the branch and bound algorithm guarantees that the global minimum is found, and it is a top down approach based on the notion of the envelope of white polyhedra that is sequentially divided into subsets up to the point where the subset contains either only white polyhedra or where it is empty.

5.2 Branching Heuristics

Apart from bound techniques, additional heuristics can be used to greatly reduce the computation time. These heuristics provide the hyperplanes with branching priorities according to their expected benefit in the OCR process and allow deciding on which hyperplane to branch first. The heuristics are intended to quickly find a solution equal or close to the optimal one thus allowing the effective pruning of suboptimal branches.

Specifically, we associate with the hyperplanes the following (descending) branching order:

(1) Hyperplanes that separate two non-connected groups of white polyhedra thus allowing us to divide the problem into two subproblems. Connectivity can be easily determined as described in Section 4.
(2) Hyperplanes, such that one half space contains only white polyhedra. If so, we choose the hyperplane yielding the maximal number of white polyhedra.
(3) Any remaining hyperplane.

6 Non-Disjoint Optimal Complexity Reduction

In this section, we reformulate the complexity reduction problem as a logic minimization problem. Thus, instead of markings with $\{-,+\}$ elements, we will use Boolean vectors with $\{0,1\}$ components.

Logic minimization is commonly used in digital circuit design, where a given Boolean function is to be minimized in terms of the number of *literals* (Boolean variables or their complement in a Boolean expression) and the number of *product terms* (expressions made of literals and the AND operation). Logic minimization was initiated in the 1950s by Veitch (1952) and Karnaugh (1953) who introduced the K-map to manually minimize (two-level) Boolean functions. In the 1980s, ESPRESSO-II (Brayton et al., 1984) was designed that allows one to attain global optimality even for large problems. This guarantees that the minimum number of product terms is derived, while the number of literals is also minimized (with second priority). The tool is readily available from the Departement of EECS, University of California, Berkeley (1982), and has been employed for the examples presented in the remainder of this paper.

6.1 Problem Formulation with Boolean Logic

For a hyperplane arrangement with n hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$, $i \in \{1,\dots,n\}$, we redefine the sign vector as the function $\mathrm{SV}' : \mathbb{R}^d \to \{0,1\}^n$ that maps x into a Boolean vector with components

$$\mathrm{SV}'_i(x) = \begin{cases} 0 & \text{if } a_i^T x \le b_i, \\ 1 & \text{if } a_i^T x \ge b_i \end{cases} \quad \text{for } i \in \{1,2,\dots,n\}\,. \quad (5)$$

Here, we use the prime to distinguish it from the original sign vector (3). Accordingly, a polyhedral cell is defined as $\mathcal{P}_\delta = \{x \in \mathbb{R}^d \mid \mathrm{SV}'(x) = \delta\}$ for a given Boolean vector δ, which replaces the marking m. Let $\Delta(\mathcal{R})$ be the image of $\mathrm{SV}'(x)$ for $x \in \mathcal{R} \subseteq \mathbb{R}^d$, namely the collection of all the possible Boolean vectors of all the points in $\mathcal{R}$.

The '$*$' element, which extends the sign vector by denoting hyperplanes that are not a facet of the associated polyhedron, is translated into Boolean variables that are removed from the Boolean vector δ. Thus in general, δ has a variable number of components. Obviously, the definitions and lemmas of Section 4 can be directly applied to the Boolean problem formulation with $\mathrm{SV}'(x)$ and δ.

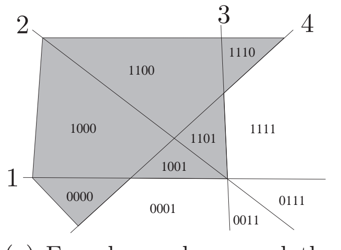
6.2 Algorithm 2 – Logic Minimization

We start by introducing the Boolean function f_{W} that – given the Boolean vector δ – evaluates the color of the polyhedron (white, black or undecided). The color is undecided if the corresponding polyhedron is not a cell in the hyperplane arrangement, i.e. the corresponding δ is not contained in $\Delta(\mathcal{R})$ and the polyhedron has an empty interior. Specifically, f_{W} yields for δ corresponding to white polyhedra a '1', for black ones a '0' and for empty ones (with an empty interior) an 'X', which is usually referred to as a *don't care* in digital circuit design.

We write f_{W} in *disjunctive normal* form, which is also referred to as *sum of product* form. In f_{W}, each *minterm* (product term, in which all variables appear exactly once) represents a white polyhedron. Each literal refers to a facet of such a polyhedron, and f_{W} represents the union of all white polyhedra. Logic minimization can be used to reduce the number of terms in f_{W}. This is equivalent to reducing the number of white polyhedra. Additionally the number of literals of each term can be reduced. The latter refers to reducing the number of facets per polyhedron. In general, these objectives lead to overlapping polyhedra. This is a desired feature, since – as will be shown in Section 8 – overlaps allow reducing the overall number of product terms and literals.

Alternatively, one may represent f_{W} in form of a truth table. A truth table carries the main advantage that it allows for providing the logic minimization tool with additional structural information, namely empty polyhedra can be specified with an 'X'. During the minimization process, the tool assigns to don't cares minterms (empty polyhedra) a color such that the overall number of product terms and literals becomes minimal. The result is either a simplified truth table or a reduced disjunctive normal form. Both representations directly translate into the new set of (overlapping) polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$.

We refer to the logic minimization as Algorithm 2. Summing up, for a given color, the truth table with the Boolean function f_{W} is built, a logic minimization tool



(a) Four hyperplanes and the Boolean vectors δ

δ_1	δ_2	δ_3	δ_4	f_{W}
0	0	0	0	0
0	0	0	1	1
0	0	1	0	X
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	1	1	1

(b) Truth table for white polyhedra

Fig. 2. Revisited Example 16 with the hyperplane arrangement, the Boolean variables and the truth table

(here ESPRESSO-II) is used to derive a reduced disjunctive normal form that is minimal in the number of product terms (which refer to polyhedra) and, with second priority, minimal in the number of literals (which refer to facets).

Example 18 *Reconsider Example 16 with the hyperplanes and markings as in Fig. 1. We associate with each hyperplane a Boolean variable δ_i, which we collect in the Boolean vector δ. As shown in Fig. 2(a), we restate the problem in terms of δ. The Boolean function for the white polyhedra follows immediately to*

$$f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\bar{\delta}_3\delta_4 + \bar{\delta}_1\bar{\delta}_2\delta_3\delta_4 + \bar{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4 \,. \quad (6)$$

Simplifying this function algebraically and without exploiting the don't cares leads to $f_{\mathrm{W}} = \bar{\delta}_1\bar{\delta}_2\delta_4 + \delta_2\delta_3\delta_4$.

Alternatively, we may translate Fig. 2(a) into the truth table for white polyhedra, which is shown in Table 2(b). Here, the empty polyhedra are listed with an 'X'. Using ESPRESSO-II, this additional information allows one to obtain the representation $f_{\mathrm{W}} = \bar{\delta}_1\delta_4 + \delta_3\delta_4$ that is minimal in the number of product terms (polyhedra) and the number of literals (facets). In terms of markings, this result corresponds to $M_m = \{-{*}{*}+, {*}{*}{+}{+}\}$. Compared to Example 16, where the disjoint OCR Algorithm based on the markings yielded $M_m = \{-{*}{*}+, {+}{+}{+}{+}\}$, the solution here is reduced by two facets. In general, as will be seen in Section 8, allowing non-disjoint polyhedra leads to solutions with fewer polyhedra and fewer facets compared to the case where we restrict ourself to disjoint polyhedra.

Lemma 19 *Algorithm 2 solves the Non-Disjoint Optimal Complexity Reduction Problem 7.*

Proof Given the resulting white polyhedra $\{\mathcal{Q}_i\}_{i=1,\dots,q}$ the proof contains three parts. Firstly, we need to prove that adding additional hyperplanes to the arrangement does not improve the solution by reducing q. This follows directly from the fact that only facets separating black and white polyhedra are needed as facets for $\mathcal{Q}_i$, and that all these facets are subsets of the hyperplanes contained in the arrangement. Secondly, recall the equivalence between polyhedra and product terms, and facets and literals, respectively. As the logic minimization tool yields the minimal number of product terms (assuming that empty polyhedra are included in the minimization process as don't cares), q is minimal, too. Furthermore, the equivalence ensures that the union of the resulting polyhedra $\mathcal{Q}_i$ equals the union of the original white polyhedra. Thirdly, the minimization of the number of literals leads to the minimal number of facets.

7 Computation of Hyperplane Arrangement

Algorithms 1 and 2 in the proposed form are only applicable to PWA systems with a hyperplane arrangement, i.e. PWA systems whose polyhedra are cells in a hyperplane arrangement of which the markings are available. In this section, we remove Assumption 1 and outline two algorithms that compute the hyperplane arrangement and its cells.

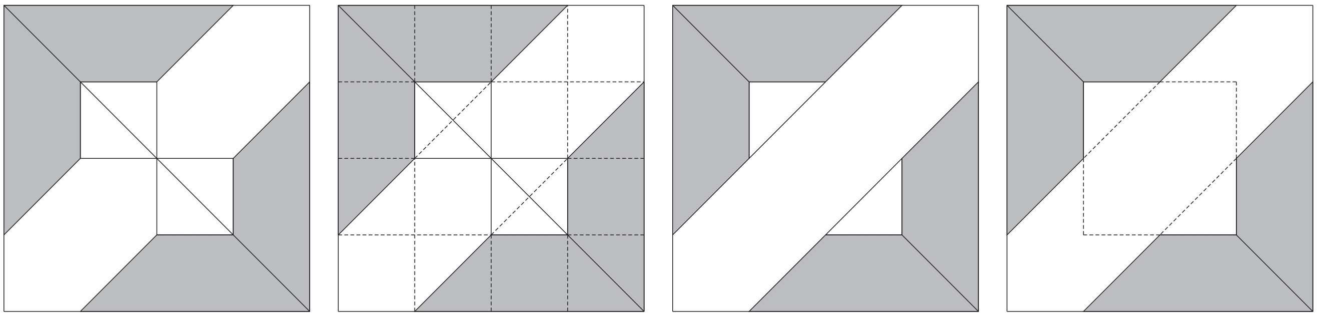
7.1 Algorithm 3 – Computation of Full Hyperplane Arrangement

The first algorithm, to which we refer as Algorithm 3, computes the full (as opposed to the simplified) hyperplane arrangement. Given a set of polyhedra, the algorithm consists of two major steps.

(1) *Hyperplane arrangement:* Collect the facets of all polyhedra and remove all duplicates. This leads to the hyperplane arrangement.
(2) *Markings and Colors:* Determine the relative position of each polyhedron with respect to each hyperplane. This yields a preliminary set of markings, where an additional symbol is used to denote polyhedra whose interior intersects with a hyperplane. Resolve the latter markings by sequentially dividing the corresponding polyhedra into two. Propagate the color information to the markings.

The first step is computationally very cheap and involves only vector comparisons. The second operation, however, involves solving LPs and increases the number of polyhedra significantly. Therefore, such an algorithm is computational tractable only for problems with a limited complexity. Yet, a number of enhancements, namely the exploitation of parallel hyperplanes and the removal of redundant hyperplanes reduces the computation time remarkably. In particular, the reverse search algorithm (see Proposition 5) can be used advantageously in the second step.

Example 20 *Consider the sets of white and black polyhedra in Fig. 3(a). The first step of Algorithm 3 identifies 13 different facets. Since the facets constraining the convex hull of the polyhedra are not considered, the hyperplane arrangement encompasses only nine hyperplanes shown as dashed lines in Fig. 3(b). In the second step, the*



(a) Original white and black polyhedra (b) Modified polyhedra in a hyperplane arrangement (c) Polyhedra resulting from disjoint OCR (d) Polyhedra resulting from non-disjoint OCR

Fig. 3. Derivation of cells defined in a hyperplane arrangement and OCR in Example 20

polyhedra are divided into cells in the hyperplane arrangement. As a result, the number of white polyhedra is increased from 6 *to* 16 *(see Fig. 3(b)). OCR restricted to disjoint polyhedra (Algorithm 1) yields three white polyhedra depicted in Fig. 3(c), whereas OCR based on logic minimization (Algorithm 2) yields only two white polyhedra, which are overlapping as indicated by the dashed lines in Fig. 3(d).*

It is particularly interesting to observe that merging the original white polyhedra in Fig. 3(a) in a optimal way without using a hyperplane arrangement would lead to four white polyhedra. In general, such an approach would require to determine the convexity of each union (each pair, triple, etc.) of white polyhedra by using the algorithms of Bemporad et al. (2001), which resort to solving LPs, and to choose among the convex unions a combination that yields the minimal number of unions and covers all white polyhedra. Despite the fact that such an approach is computationally intractable even for very small problems, it is also in general inferior to the OCR algorithms in terms of the number of resulting polyhedra as the example demonstrates.

Thus deriving the hyperplane arrangement first and reducing the complexity subsequently in an optimal way yields in general a lower number of polyhedra compared to the case, where the original polyhedra are merged optimally without the notion of a hyperplane arrangement.

The following key lemma follows directly from Lemma 19 and the fact that Algorithm 3 is run first.

Lemma 21 *Algorithm 3 followed by Algorithm 2 solves the General Non-Disjoint Optimal Complexity Reduction Problem 8.*

Clearly, a corresponding lemma could be established for the combination of Algorithms 3 and 1, too.

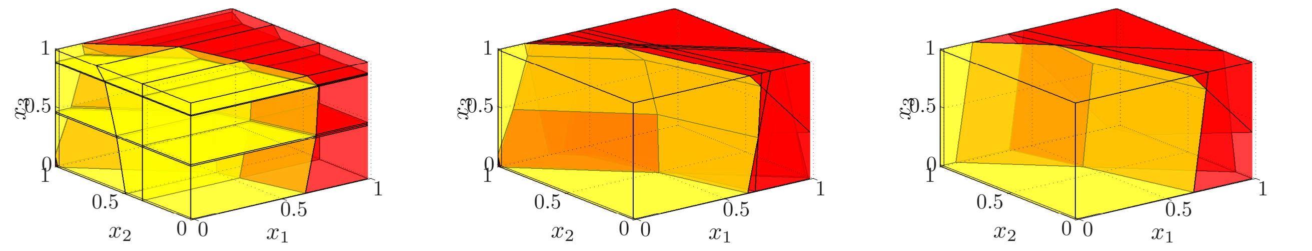
7.2 Algorithm 4 – Computation of Simplified Hyperplane Arrangement

In this section, we outline a technique to reduce the number of hyperplanes in the hyperplane arrangement. Since the complexity of the OCR algorithms depends exponentially on the number of hyperplanes in the arrangement, such a reduction extends the applicability of the OCR algorithms to problems of larger size. Moreover, the complexity of the solution can be further reduced at the expense of an adjustable degree of error.

In many cases, the hyperplane arrangement contains numerous hyperplanes that are almost identical, at least very similar. Assume that all hyperplanes $H_i = \{x \in \mathbb{R}^d \mid a_i^T x = b_i\}$ are normed, i.e. $a_i^T a_i = 1$. Given the two hyperplanes H_i and H_j, we use $\mu = ||[a_i^T \; b_i]^T - [a_j^T \; b_j]^T||_1$ as a measure for similarity. We say that the hyperplanes H_i and H_j are similar, if μ is below a given threshold.

Next, we outline Algorithm 4 that derives a simplified hyperplane arrangement by replacing clusters of similar hyperplanes by their weighted average. This approach is an extension of Algorithm 3.

(1) *Hyperplane arrangement:* Collect the facets of all polyhedra and remove all duplicates. This leads to the hyperplane arrangement.
(2) *Clusters:* For a given μ, identify groups of similar hyperplanes. Replace the groups by one hyperplane given by the weighted average of the group. This yields the simplified hyperplane arrangement.
(3) *Markings:* Compute the cells and their markings of the simplified arrangement as in Step 2 of Algorithm 3.
(4) *Colors:* Identify the color of each cell by intersecting it with all given initial polyhedra. If all non-empty intersections are with polyhedra of the same color, assign this color to the cell. Otherwise, either use the color of the largest intersection, or assign a don't care to the cell.



(a) Original sets of white and black polyhedra (b) Polyhedra resulting from non-disjoint OCR with $\mu = 0$ (c) Polyhedra resulting from non-disjoint OCR with $\mu = 0.35$

Fig. 4. OCR and simplification of the hyperplane arrangement in Example 22

In general, the result will have (small) color errors. We define the color error as the radius of the largest Chebycheff ball that can be inscribed in the intersection of the two polyhedra $\mathcal{P}$ and $\mathcal{Q}$ of different colors, where $\mathcal{P}$ is in the set of original polyhedra and $\mathcal{Q}$ is in the set of resulting polyhedra. Often, small errors in the color of the resulting polyhedra can be tolerated, particularly in the presence of model uncertainties (in case of PWA models) or measurement noise (in case of PWA state-feedback control laws). Moreover, the complexity of the solution based on a simplified hyperplane arrangement is in general significantly reduced and the solution is well-defined, i.e. each point $x \in \mathcal{R}$ is included in (at least) one polyhedron and is associated with exactly one color. The design parameter μ can be considered as a tuning knob to reduce the complexity of the solution (while increasing the color error).

Example 22 *Consider in the three-dimensional space the sets of white (yellow) and black (red) polyhedra in Fig. 4(a), where we aim at minimizing both the 15 white and the 28 black polyhedra. Thus, after running Algorithm 3 to obtain the hyperplane arrangement, we execute Algorithm 2 twice. When refraining from simplifying the hyperplane arrangement, one white and six black polyhedra result as shown in Fig. 4(b). Alternatively, simplifying the hyperplane arrangement with $\mu = 0.04$ leads to one white and five black polyhedra and a color error below 0.004. Increasing μ to 0.1 and 0.35 reduces the number of black polyhedra to four and three, respectively, and increases the maximal color error to 0.01 and 0.037, respectively. As the polyhedra are scaled to $[0, 1]$, even the absolute error of 0.037 corresponds to an inaccuracy of only 3.7 percent. Observe that increasing μ not only reduces the number of resulting polyhedra, it also greatly reduces the number of facets per polyhedron.*

Summing up this example, the non-disjoint OCR algorithm with $\mu = 0$ reduced the number of polyhedra by 84 percent. Setting $\mu = 0.35$ additionally reduced the number of polyhedra by another 43 percent while introducing a color error of 3.7 percent.

8 Optimality of Algorithms

In the following, we compare the OCR Algorithms 1 and 2 with each other. Both are optimal in the sense that they yield the minimum number of polyhedra for the specific problem they solve (Problems 6 and 7). Yet, as the problems differ regarding the property whether the resulting polyhedra are required to be disjoint or not, the complexity of the solution in terms of the number of polyhedra and facets differs in general, too.

Recall that in Problem 6, the resulting polyhedra are required to be disjoint and unions of the original polyhedra. Thus, Problem 6 is an optimal *merging* problem, which can be also considered as a *specific* optimal *set partitioning* problem. The problem is specific in the sense that the hyperplanes along which the set can be partitioned are restricted to the hyperplanes given by the facets of the original polyhedra to be merged. This issue is rather subtle, yet we would like to clarify it with the following example.

Example 23 *For given sets of white and black polyhedra, assume we have derived the hyperplane arrangement, split the polyhedra into cells defined in this arrangement, and run subsequently Algorithm 1 that yields the three white polyhedra shown in Fig. 5(a). This solution is optimal with respect to Problem 6. Yet, adding to the hyperplane arrangement an additional vertical hyperplane that cuts through the center of the figure would reduce the solution to only two white polyhedra. On the other hand, Algorithm 2 leads to the two white polyhedra depicted in Fig. 5(b), where the dashed lines indicate the overlaps. Adding additional hyperplanes to the arrangement before running Algorithm 2 would not improve the solution. This holds in general due to Lemma 21.*

We conclude that even though Algorithm 1 solves Problem 6 by deriving a solution that is minimal in the number of *merged* polyhedra, the number of polyhedra might be further reduced by introducing additional facets. Thus, in general, the merged polyhedra constitute only a suboptimal solution to the (more general) optimal set partitioning problem. Nevertheless, even though such a case has been constructed here, they appear to be rare in practice.

In Problem 7, the restriction requiring the resulting polyhedra to be disjoint and unions of the original polyhedra is dropped. Hence, strictly speaking, the second problem is not a merging problem but a more general optimal set covering problem. As Problem 7 is less restrictive than Problem 6, we expect Algorithm 2 to generally yield a lower number of polyhedra and facets than Algorithm 1. This is confirmed by Examples 20 and 23. In particular, as already mentioned above, adding hyperplanes does not improve the solution.

9 Examples

In this section we present several examples showing how the OCR algorithms can be applied to PWA models as well as to PWA state-feedback control laws with the aim to efficiently derive equivalent minimal representations. All experiments were run on a Pentium IV 2.8 GHz machine with MATLAB 6.5.

9.1 PWA Model with Hyperplane Arrangement

In Geyer et al. (2003), we have detailed the model of a paperboy delivering mail items to households within a PWA neighborhood. The model has a PWA characteristic, and it has two real inputs, four real states and two binary states. To facilitate the modelling, we have described the model in a high-level textual form in the modelling language HYSDEL (Torrisi and Bemporad, 2004). The mode enumeration algorithm (Geyer et al., 2003) transforms the textual model description into the equivalent PWA model. Since this algorithm is based on the cell enumeration in hyperplane arrangements, it also provides the corresponding hyperplane arrangement encompassing 11 hyperplanes and the set of markings. The model encompasses 168 polyhedra in the eight-dimensional state-input space.

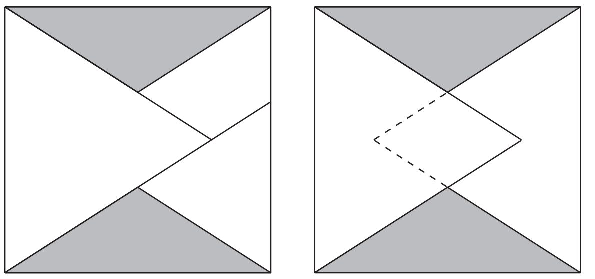

(a) Polyhedra resulting from disjoint OCR
(b) Polyhedra resulting from non-disjoint OCR

Fig. 5. OCR in Example 23 visualizing the consequence of restricting the hyperplane arrangement to hyperplanes given by facets of the original white polyhedra

The disjoint OCR Algorithm 1 reduces the number of polyhedra from 168 down to 36 within 0.22 s. This is a reduction of the complexity by roughly 80 percent. In particular, both PWA models are equivalent, meaning that for every given state and input they yield the same state-updates and outputs.

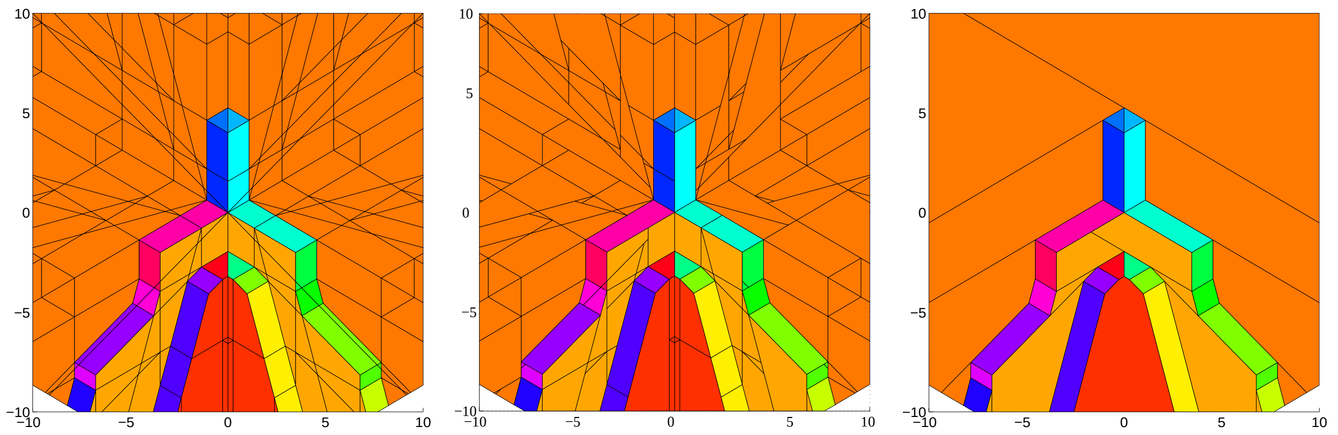
9.2 PWA State-Feedback Control Law for Toy Example

Next, we perform OCR to a PWA state-feedback control law. For a PWA model with two modes, Baotić et al. (2003) have formulated and solved a constrained infinite time optimal control problem. The resulting polyhedral partition of the state-space $\mathcal{X} = [-10, 10] \times [-10, 10]$ is shown in Fig. 6(a), where each color relates to a different affine control law. Note that there exist 19 different control laws and 252 polyhedra.

For this example, we compare the following three algorithms with each other: (i) general disjoint OCR (Algorithms 3 and 1), (ii) general non-disjoint OCR (Algorithms 3 and 2) and (iii) *greedy merging*, which we outline next. In a first step – to speed up the computations – the greedy merging algorithm builds a sparse matrix indicating whether two polyhedra are neighbors according to Definition 15 using the algorithms described in Bemporad et al. (2001). Based on this list, it determines in a second step by solving LPs if a pair of neighboring polyhedra forms a convex union. If so, the pair is replaced by its union and the list is updated accordingly. This is done sequentially until no pair is left for merging. The merging procedure is done in a greedy way (optimality is not pursued) and additional facets are not introduced.

Algorithm 3 derives a hyperplane arrangement with 135 hyperplanes containing 5200 polyhedra within 34 s. The disjoint OCR Algorithm 1 leads to 39 polyhedra, which are shown in Fig. 6(c). Compared to the initial 252 polyhedra, this is a reduction of 84 percent. The computation time is 3 min. The non-disjoint OCR Algorithm 2 also leads to 39 polyhedra. Even though these are overlapping, the polyhedral partition is very similar to the one in Fig. 6(c). Yet the computation time is with 5 s very small. Greedy merging fails to reach similar levels of simplification. It leads to the result shown in Fig. 6(b) with 189 polyhedra computed in 17 s.

Based on our experience, we conclude the following. The disjoint OCR algorithm based on branch and bound is rather slow limiting its applicability mostly to problems with a few thousand polyhedra defined in a two- or three-dimensional space. The non-disjoint OCR algorithm based on logic minimization, however, is generally faster by two orders of magnitude and it also scales better as the problem size increases. This is mainly due to the fact that a state-of-the-art logic minimization tool (ESPRESSO-II) with advanced heuristics is used. OCR problems with hyperplane arrangements comprising hundreds of hyperplanes with some 100′000 cells



(a) Original set of 252 polyhedra

(b) Set of 189 polyhedra resulting from greedy merging

(c) Set of 39 polyhedra resulting from OCR

Fig. 6. Polyhedral partitions of the PWA state-feedback control law, where each color relates to a different affine feedback law, using different complexity reduction schemes

have been tackled successfully within a few minutes. For larger problems, the memory requirement for storing the truth table becomes a limiting factor. The main bottleneck, however, is the computation of the cells in the hyperplane arrangement, namely Algorithm 3. The main problem here is the significant increase in the number of polyhedra when deriving the hyperplane arrangement (in the above example from 252 to 5200 polyhedra). This can be overcome by applying the techniques presented in Section 7.2, namely by simplifying the hyperplane arrangement or using a divide and conquer strategy as will be shown in the next example.

9.3 PWA State-Feedback Control Law for Industrial Problem

In the last example, we apply the non-disjoint OCR algorithm to an industrial problem in the field of power electronics. We consider a three-phase two-level DC-AC inverter driving an induction motor using the control methodology Direct Torque Control. For a comprehensive treatment of this problem, the reader is referred to Geyer (2005).

For this setup, we have formulated an optimal (direct torque) control problem. Even though the model involves only three real and three binary states, the control problem is highly challenging due to the model nonlinearities, the large control horizon required and the very short sampling interval of 25 μs. As a result, the PWA state-feedback control law is very complex and comprises for each of the eight binary state combinations up to 8500 polyhedra. Taking into account the short sampling interval such a control law is hardly implementable.

To make our OCR schemes applicable, we employ a *divide and conquer* strategy. Specifically, we divide the original large problem sequentially into pairs of smaller subproblems, which can be solved efficiently by one of the OCR algorithms. Subsequently, the solutions of the subproblems are recombined. To assure optimality, an OCR algorithm is run also on the unions of the solutions of the subproblems. In general, this is computationally feasible as the complexity of the subproblems has been greatly reduced in the first step. Such a scheme is particularly useful in our context here, since the computational burden and memory requirement is in the worst case exponential in the number of hyperplanes. Care has to be taken when dividing a problem with the hyperplane arrangement $\{H_i\}_{i=1,\dots,n}$ into two subproblems. As the computational burden mostly depends on n, it is beneficial to divide the problem such that the number of hyperplanes n_1 and n_2 in the subproblems are both minimal and balanced ($n_1 \approx n_2$).

As a result, the complexity of the control laws is reduced by roughly 90 percent. The computation times are in the range of seven to 11 hours and thus large. Yet, we would like to recall that the OCR needs to be performed only once off-line. For the particular application the reduction of the number of polyhedra by an order of magnitude is expected to be decisive for a successful hardware implementation, given the very short sampling interval.

10 Efficient Implementation of PWA State-Feedback Control Laws

When implementing a PWA state-feedback control law, one faces the problem of finding the control law for the given state in an efficient way. The standard solution is to determine the polyhedron that the state lies in by cycling through (in the worst case all) the polyhedra and checking if the corresponding inequalities hold. Even though these operations involve only matrix multiplications, for small sampling times and/or large numbers of polyhedra this approach becomes prohibitive.

The OCR algorithms not only allow one to derive a representation of the control law that is minimal in the number of polyhedra, but also provide a simple and efficient way to implement it. After the OCR step, we propose to compute again the hyperplane arrangement and the markings for the reduced problem. Based on the markings, the controller can be implemented either as a collection of Boolean functions or as a binary search tree.

10.1 Collection of Boolean Functions

Each marking or Boolean vector is associated with a certain color, where each color represents a feedback law. Similar to Section 6, we build for each color a Boolean function with the Boolean vector as argument. This yields a collection of Boolean functions. Thus online, for a given state, one only needs to determine the Boolean vector based on the modified sign vector and evaluate which Boolean function is true. The latter directly relates to the feedback law. Hence, only the sign vector together with the Boolean functions needs to be implemented. In particular, polyhedra do not need to be stored and evaluated thus reducing the memory requirement and the on-line computation time. Such an approach is particularly suitable for a hardware implementation, since the Boolean functions can be easily implemented as a two-level disjunctive normal form using AND, OR and NOT gates.

10.2 Binary Search Tree

Alternatively, one may build a binary search tree similar to Tøndel et al. (2003), which allows one to determine for a given state the polyhedron and the associated control law efficiently. Specifically, given a state, the control law is found by traversing from the root node to the leafs. Each node is associated with a hyperplane; branching at the nodes is done according to the half space (induced by the hyperplane) the state lies in. Each leaf is associated with a control law (and a polyhedron).

A non-trivial task is to build a search tree of minimal depth. Such a search tree minimizes the worst-case computational burden, which is proportional to the maximal number of hyperplanes to be evaluated. Tøndel et al. (2003) use heuristics to derive a tree of small depth. Using the markings, however, enables us to derive a binary search tree of minimal depth by setting up a branch and bound algorithm similar to Algorithm 1.

11 Conclusions

Exploiting the markings of the hyperplane arrangement allowed us to build an equivalent polyhedral piecewise system minimal in the number of polyhedra by using either branch and bound techniques to derive a disjoint set of polyhedra, or logic minimization to obtain polyhedra that are in general overlapping. Compared to the disjoint OCR approach, in general, logic minimization leads to solutions with fewer polyhedra (due to the possibility of overlaps) and it performs significantly faster. If the markings and the hyperplane arrangement are given (e.g. from a preceding run of the mode enumeration algorithm), it is not necessary to solve additional LPs. This allows a significant reduction of the computational requirements while retaining optimality of the resulting simplified partitions.

By computing the hyperplane arrangement, the applicability of the algorithms was extended to derive minimal polyhedral piecewise representations of general polyhedral piecewise systems not defined over the cells of a hyperplane arrangement. To reduce the computational burden of large problems, a divide and conquer strategy can be used, and/or the hyperplane arrangement can be simplified. The latter approach is particularly attractive if the solution complexity needs to be further reduced and small errors can be tolerated.

The notion of markings in hyperplane arrangements also allows an efficient implementation of polyhedral control laws – either as a collection of Boolean functions or as a binary search tree of minimal depth.

References

Avis, D., Fukuda, K., 1996. Reverse search for enumeration. Discr. App. Math. 65, 21–46.

Baotić, M., Christophersen, F., Morari, M., Dec. 2003. Infinite time optimal control of hybrid systems with a linear performance index. In: Proc. 42nd IEEE Conf. on Decision and Control. USA, pp. 3191–3196.

Bemporad, A., Fukuda, K., Torrisi, F., Apr. 2001. Convexity recognition of the union of polyhedra. Comp. Geometry: Theory and Applications 18, 141–154.

Borrelli, F., 2003. Constrained Optimal Control of Linear and Hybrid Systems. Vol. 290 of LNCIS. Springer.

Borrelli, F., Baotić, M., Bemporad, A., Morari, M., Oct. 2005. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. Automatica 41 (10), 1709–1721.

Brayton, R., Hachtel, G., McMullen, C., Sangiovanni-Vincentelli, A., 1984. Logic Minimization Algorithms for VLSI Synthesis. Kluwer Academic Publishers.

Brungger, A., Marzetta, A., Fukuda, K., Nievergelt, J., 1999. The parallel search bench zram and its applications. Annals of Operations-Research 90, 45–63.

Buck, R., 1943. Partition of space. American Math. Monthly 50, 541–544.

Chazelle, B., Aug. 1984. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. SIAM J. of Computing 13, 488–507.

Departement of EECS, University of California, Berkeley, 1982. webpage of ESPRESSO-II. online document, `http://www-cad.eecs.berkeley.edu/Software/software.html`.

Edelsbrunner, H., 1987. Algorithms in Combinatorial Geometry. Springer.

Ferrez, J., Fukuda, K., 2002. Implementations of lp-based reverse search algorithms for the zonotope construction and the fixed-rank convex quadratic maximization in binary variables using the ZRAM and the cddlib libraries. Tech. rep., Mcgill.

Ferrez, J., Fukuda, K., Liebling, T., Nov. 2001. Cuts, zonotopes and arrangements. Tech. rep., EPF Lausanne, Switzerland.

Geyer, T., 2005. Low complexity model predictive control in power electronics and power systems. Dr. sc. tech. thesis, Automatic Control Laboratory ETH Zurich.

Geyer, T., Torrisi, F., Morari, M., 2003. Efficient mode enumeration of compositional hybrid systems. In: Pnueli, A., Maler, O. (Eds.), Hybrid Systems: Computation and Control. Vol. 2623 of LNCS. Springer, pp. 216–232.

Heemels, W., Schutter, B. D., Bemporad, A., Jul. 2001. Equivalence of hybrid dynamical models. Automatica 37 (7), 1085–1091.

Karnaugh, M., Nov. 1953. A map method for synthesis of combinational logic circuits. AIEE Transactions on Communications and Electronics 72, 593–599.

Kvasnica, M., Grieder, P., Baotić, M., Morari, M., 2004. Multi parametric toolbox (MPT). In: Alur, R., Pappas, G. (Eds.), Hybrid Systems: Computation and Control. Vol. 2993 of LNCS. Springer, pp. 448–462, `http://control.ee.ethz.ch/~mpt`.

Sontag, E., Apr. 1981. Nonlinear regulation: The piecewise linear approach. IEEE Trans. Automat. Contr. 26 (2), 346–358.

Tøndel, P., Johansen, T., Bemporad, A., May 2003. Evaluation of piecewise affine control via binary search tree. Automatica 39 (5), 945–950.

Torrisi, F., Bemporad, A., Mar. 2004. Hysdel — a tool for generating computational hybrid models for analysis and synthesis problems. IEEE Trans. Contr. Syst. Technol. 12 (2), 235–249.

Veitch, E., May 1952. A chart method for simplifying boolean functions. In: Proceedings of the Association for Computing Machinery. pp. 127–133.