2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

Optimized Decision Trees for Point Location in Polytopic Data Sets - Application to Explicit MPC

A.N. Fuchs, C.N. Jones and M. Morari

Abstract— **Explicit Model Predictive Control (EMPC) produces control laws defined over a set of polytopic regions in the state space. In this paper we present a method to create a binary search tree for point location in such polytopic sets, in order to provide a fast lookup of the control law corresponding to a given state. We use hyperplanes as decision criteria that are, contrary to previous works, not constrained to the boundaries of the polytopes. Each hyperplane is the solution of a mixed-integer optimization problem with two objectives: having the same number of polytopes on either side of the hyperplane and minimizing the number of polytopes cut by the hyperplane. Contrary to previous approaches, the method can be applied to polytopic sets where the polytopes are either adjacent with common facets (for classical EMPC) or separated in space (for suboptimal EMPC). There are two benefits using this approach: First, the method optimizes the balance of the tree. If a tree of the theoretically lowest possible depth (i.e. $\log_2$ depth) exists, the algorithm will find it, although the time to solve the optimization problem may become prohibitive for large problems. Second, the method provides an efficient evaluation of suboptimal EMPC policies since it allows to maximize the distance of the hyperplane to the closest polytope that is not intersected.**

I. Introduction

A. Context

This paper deals with the construction of binary decision trees for point location in polytopic data sets. A control relevant application of the point location problem is the evaluation of Explicit Model Predictive control (EMPC) policies as introduced in [1] for the optimal control of constrained linear systems with quadratic cost functions. The control policy has the form of a piecewise affine (PWA) function defined over a polytopic partition of the state space, where each polytopic region is assigned to a linear control law. Optimal control then reduces to determining which region (polytope) contains the current state and to picking the corresponding control law. This problem is known as the *point location problem*.

Recent results on suboptimal robust EMPC study controllers that only use a subset of the polytopic regions of the original EMPC solution as an approximation of the optimal solution [2], while still giving some stabilizing guarantees [3]. Suboptimal control then reduces to determining a region (polytope) that is close to the current state to picking the corresponding control law.

Both optimal and suboptimal EMPC hence require to assign a point of the state space to an element of a polytopic set.

B. The point location problem

We first consider ordinary EMPC policies. The simplest approach to the point location problem - checking all polytopes whether they contain a given point - is very time-consuming and can even take longer than a direct online-optimization as is standard practice in ordinary MPC where a quadratic program is solved at every time step.

A binary search tree based on hyperplanes reduces the online computation of the controller to a sequence of inner product evaluations as the decision criteria of the tree. The worst evaluation time of the tree depends on the maximum number of decisions to take, called depth. Geometrically, one can interpret the decision criterion as a hyperplane separating the polytopes into three groups - (a) polytopes entirely in the positive halfspace, (b) polytopes entirely in the negative halfspace and (c) polytopes with points in both halfspaces. Clearly, if a given point (state) lies in the positive halfspace, it can not be contained in a polytope which lies entirely in the negative halfspace and vice versa. Let $\mathcal{P}_l$ denote the union of all polytopes in group (a) and (c), and $\mathcal{P}_r$ the union of those in (b) and (c). The hyperplane allows one to distinguish whether a given point lies in $\mathcal{P}_l$ or $\mathcal{P}_r$.

Tondel et al. [4] proposed a search tree whose decision hyperplanes lie in the set of all irredundant bounding hyperplanes of all polytopes (i.e. the affine hulls of the facets). The algorithm evaluates how well each hyperplane serves as a decision criterion for a binary search tree and picks the best one. The process is repeated recursively in order to build the tree, we refer to the result as *facet tree*. One drawback of this approach is illustrated in Fig. 1, showing an example with $N = 9$ polytopes. The facet tree does not balance the number of elements in $\mathcal{P}_l$ and $\mathcal{P}_r$ very well and would result in a tree of a depth linear in N. We will propose an alternative algorithm that is not restricted to the use of facet hyperplanes and for the example returns an optimal tree of depth $\log_2 N$. Although the facet tree can in principle be applied to sets of polytopes separated in space, it has not been designed for that and potentially produces trees of larger depth. We show that our method naturally extends to suboptimal EMPC policies, maximizing at each decision level the distance between the hyperplane and the closest polytope on either side. For point location in classical EMPC solutions several other methods have been proposed, that we will not further discuss. They include the evaluation of the value function of all regions [5], nearest neighbour search [6] and subdivision walking [7]. During on-line evaluation, [6] and [7] are logarithmic time in the number of polytopes, but restricted to classical MPC with the polytopes not separated in space. While [5] could in principle also be applied to suboptimal EMPC, it is time-linear in the number of polytopes. The decision tree proposed in this paper is time-logarithmic *and* can be applied

978-1-4244-7427-1/10/$26.00 ©2010 AACC

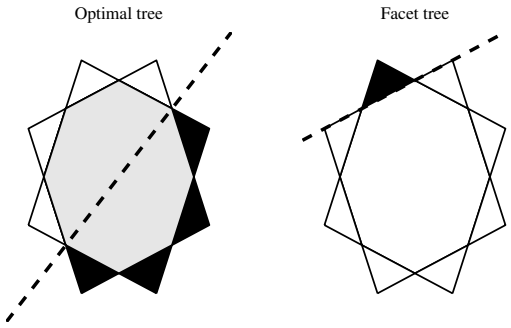

Fig. 1. Comparison of the optimized and the facet tree after one decision: $\mathcal{P}_l$ (black and gray), $\mathcal{P}_r$ (white and gray), hyperplane (dashed)

to suboptimal EMPC solutions.

Point location methods are also known in computer graphics for the hidden-surface problem [8].To the knowledge of the authors, none of the methods is directly applicable to higher dimensional polytopic sets, see [9], [10] or [11].

C. Outline of the paper

Section II defines the tree building algorithm and shows it terminates. The main contribution of the paper in Section III is the formulation of the optimal cutting problem, a Mixed Integer Linear Programs (MILP) which determines the best hyperplane at every node of the tree construction. In Section IV we show how to extend the method to suboptimal EMPC policies and how to impose explicit bounds on the depth of the tree. Section V demonstrates the performance of the algorithm with some test systems.

II. The search tree algorithm

This section gives some necessary definitions on polytopes, point location and search trees together with the basic algorithm to build the search tree and some simple properties.

Definition 1: A *polytopic set* in $\mathbb{R}^n$, is a finite collection $\mathcal{P} = \{\mathrm{P}_1, \mathrm{P}_2, ..., \mathrm{P}_N\}$ of polytopes P_i (bounded intersections of a finite number of halfspaces) given in inequality form

$$\mathrm{P}_i = \{x \in \mathbb{R}^n : \quad A_i x \leq b_i\} \tag{1}$$

where $A_i \in \mathbb{R}^{m_i \times n}$ and $b_i \in \mathbb{R}^n$.

We will here only consider polytopic sets that are *non-overlapping*, i.e. $\mathrm{int}(\mathrm{P}_i) \cap \mathrm{int}(\mathrm{P}_j) = \emptyset \quad \forall i \neq j$. For overlapping polytopes, the method can be extended using slackvariables.

Definition 2: A *point location function (PLF)* for a polytopic set $\mathcal{P}$ is a function $g : \mathbb{R}^n \to \mathcal{P}$ that returns a polytope $\mathrm{P} = g(x) \in \mathcal{P}$ such that $x \in \mathrm{P}$, or $\emptyset$ if no such polytope exists.

We want to construct a PLF with a complexity logarithmic in N. Next we recursively define the term *tree*.

Definition 3: A *binary seach tree of hyperplanes*, or simply a *tree*, $\mathcal{T}$ is either a polytope P or a triplet $(\mathcal{T}_l, \mathcal{T}_r, \mathrm{H})$ where $\mathcal{T}_l, \mathcal{T}_r$ are also trees and H is a hyperplane.

A hyperplane is given by

$$\mathrm{H} = \mathrm{H}(\alpha, \beta) = \left\{x \in \mathbb{R}^n : \quad \alpha^T x = \beta\right\} \tag{2}$$

with the normal vector $\alpha \in \mathbb{R}^n$ and the offset $\beta \in \mathbb{R}$. We say that a point x lies to the *left* of the hyperplane H, if $\alpha^T x \geq \beta$, otherwise it lies to the *right*.

Definition 4: The *tree-PLF*, is a function $g_{\mathcal{T}} : \mathbb{R}^n \to \mathcal{P}$ that uses Algorithm 1 to compute its value.

Algorithm 1 Algorithm for tree based point location

Require: point x, search tree $\mathcal{T}$
Ensure: polytope P such that $x \in \mathrm{P}$

```
function EVALTREE(T, x)
    if ISPOLYTOPE(T) then
        return T
    else
        (T_l, T_r, H(α, β)) ← T
        if α^T x ≥ β then
            return EVALTREE(T_l, x)
        else
            return EVALTREE(T_r, x)
        end if
    end if
end function
```

Each function call of the recursive function in Algorithm 1 is referred to as $node$. Starting with the tree $\mathcal{T} = (\mathcal{T}_l, \mathcal{T}_r, \mathrm{H})$ as root node, the algorithm jumps to $\mathcal{T}_l$ if the given point x lies to the left of H, otherwise it jumps to $\mathcal{T}_r$. Reaching the last level of the tree, called the *leaf*, the algorithm returns a polytope P. The largest possible number of nodes that the tree-PLF algorithm has to visit before returning a polytope P is the *depth* of the tree.

For a given polytopic set $\mathcal{P}$, we want to construct a tree $\mathcal{T}$ of minimum depth that is *correct*, i.e.

$$\forall x \in \cup_{i=1}^{N} \mathrm{P}_i : \qquad x \in g_{\mathcal{T}}(x) \tag{3}$$

Algorithm 2 Recursive function to build the tree

Require: polytopic set $\mathcal{P}$
Ensure: tree $\mathcal{T}$

```
function BUILDTREE(P)
    if card(P) = 1 then
        return P
    else
        (P_l, P_r, H) ← CUTPOLYTOPES(P)
        return (BUILDTREE(P_l),BUILDTREE(P_r),H)
    end if
end function
```

Algorithm 2 shows pseudocode of the recursive function used to build such a tree. The function card($\mathcal{P}$) denotes the number of elements of the polytopic set $\mathcal{P}$. If the polytopic set $\mathcal{P}$ contains only one polytope, a terminal node is reached, and $\mathcal{T} = \mathcal{P} = \{\mathrm{P}\}$. Otherwise, $\mathcal{P}$ is split into two polytopic sets, $\mathcal{P}_l$ and $\mathcal{P}_r$, the trees are calculated, and combined to a

larger tree, including the decision criterion of this node, the hyperplane H.

Lemma 1: Algorithm 2 will terminate in a finite amount of iterations if

$$\text{card}(\mathcal{P}_i) < \text{card}(\mathcal{P}) \qquad i \in \{l, r\} \tag{4}$$

Proof: Since $\mathcal{P}_l$ and $\mathcal{P}_r$ are strictly smaller than $\mathcal{P}$, the number of elements of the argument to the recursively called function BUILDTREE is strictly decreasing and will eventually consist of one element, terminating the recursion and thereby the Algorithm. ∎

Lemma 2: Algorithm 2 is correct in the sense of (3) if the function CUTPOLYTOPES satisfies:

$$\begin{aligned} \forall \mathrm{P} \in \mathcal{P}: \quad & \{\exists x \in \mathrm{P} : \alpha^T x > \beta\} \rightarrow \{\mathrm{P} \in \mathcal{P}_l\} \\ \forall \mathrm{P} \in \mathcal{P}: \quad & \{\exists x \in \mathrm{P} : \alpha^T x < \beta\} \rightarrow \{\mathrm{P} \in \mathcal{P}_r\} \end{aligned} \tag{5}$$

where $\mathrm{H} = \mathrm{H}(\alpha, \beta)$.

Proof: Consider a single function call in Algorithm 1 with a point $x \in \mathrm{P} \in \mathcal{P}$. If $\alpha^T x \geq \beta$ then we need $\mathrm{P} \in \mathcal{T}_l$ for the decision to be correct. Since the decision has to be correct for any $x \in \mathrm{P}$, the first line of (5) follows. The argument for $\mathcal{T}_r$ is identical. ∎

Note that with (5) a polytope P_k may belong to both $\mathcal{P}_l$ and $\mathcal{P}_r$. In that case, the polytope is intersected by the hyperplane and must be kept in both branches of the decision tree. The optimizaion problem will include the objective to minimize the number of such regions.

Lemma 3: If the function CUTPOLYTOPES guarantees

$$\begin{aligned} &\text{card}(\mathcal{P}_l) \leq (1 - t_{min})\text{card}(\mathcal{P}) \\ &\text{card}(\mathcal{P}_r) \leq (1 - t_{min})\text{card}(\mathcal{P}) \\ &\quad \text{for} \quad t_{min} \in (0, 0.5] \end{aligned}$$

then Algorithm 2 is guaranteed to return a tree of depth $d \leq d_{max}$ where

$$d_{max} = \log_{\frac{1}{1-t_{min}}} N = -\frac{\ln N}{\ln(1 - t_{min})} \tag{6}$$

Proof: On the last level after d recursions the algorithm terminates and card($\mathcal{P}$)$= 1 = (1-t)^d N$ with $t \geq t_{min}$. Solving this equation for d yields the result. ∎

Note that 0.5 is a hard upper bound on t, otherwise the natural requirement $\mathcal{P} = \mathcal{P}_l \cup \mathcal{P}_r$ which follows from (5) could not be satisfied. In general, it is not possible to generate a PLF meeting this hard bound due to the geometry of the polytopic set.

III. The optimal cutting problem

The Lemmas of the previous section used some properties of the function CUTPOLYTOPES. This section presents a candidate for this function and shows that the properties hold. One looks for a hyperplane that will balance between the polytopes on the left and on the right. The search for the *best* hyperplane is captured in an optimization problem with the hyperplane parameters α and β as variables.

A. Problem Statement

We begin with a mathematical formulation of the problem of finding a good hyperplane balancing the polytopes on each side. First observe that a polytope P_i can not be entirely on both sides of a hyperplane $\mathrm{H}(\alpha, \beta)$, i.e. at most one of the following holds:

$$\forall x \in \mathrm{P}_i : \alpha^T x \geq \beta \tag{7}$$

$$\forall x \in \mathrm{P}_i : \alpha^T x < \beta \tag{8}$$

To characterize these cases, we introduce binary variables $L_i, R_i \in \{0, 1\} \quad i = 1, 2, ..., N$ and their composition to binary vectors $L, R \in \{0, 1\}^N$. We assign $L_i = 1$ if (7) holds and $L_i = 0$ otherwise. Similarly, let $R_i = 1$ when (8) holds and $R_i = 0$ otherwise. Then the *optimal cutting problem for polytopic sets* (OC) is:

$$\begin{aligned} J^* = \min_{\alpha, \beta, L, R} \; & J \\ \text{s.t.} \quad & \forall i = 1, 2, ..., N \\ & L_i = 1 \leftrightarrow \left\{\forall x \in \mathrm{P}_i : \alpha^T x \geq \beta\right\} && \text{(9a)} \\ & R_i = 1 \leftrightarrow \left\{\forall x \in \mathrm{P}_i : \alpha^T x \leq \beta\right\} && \text{(9b)} \\ & \mathrm{P}_i = \{x \in \mathbb{R}^n : \quad A_i x \leq b_i\} && \text{(9c)} \\ & \|\alpha\|_\infty = 1 && \text{(9d)} \end{aligned}$$

with

$$J = \left| \left(\sum_{i=1}^{N} L_i \right) - \left(\sum_{i=1}^{N} R_i \right) \right| + \omega_1 \cdot \sum_{i=1}^{N} |L_i + R_i - 1| \tag{10}$$

The first term of the cost function J describes the balance of the decision by penalizing the difference of the number of polytopes entirely on the left and on the right side of the hyperplane. The second term, weighted with a positive scalar ω_1, penalizes the polytopes with $L_i = R_i = 0$ which are not completely on either side of the hyperplane and are undecided after the hyperplane decision of the search tree. Note that if the polytope P_i is full-dimensional at most one of (9a) and (9b) can be true, so

$$\forall i : \qquad L_i + R_i \leq 1 \tag{11}$$

The unit norm requirement on α has been introduced to ensure boundedness of the solution: Let $\{\alpha^*, \beta^*, L^*, R^*\}$ be the optimal solution to (OC) *without* the unit norm constraint on α. Then $\{k\alpha^*, k\beta^*, L^*, R^*\}, k \in \mathbb{R}, k > 0$ would be another solution with the same cost. This confirms the geometric interpretation that a scaling of both α and β does not change the hyperplane. Also, the trivial solution $\{0, 0, L^*, R^*\}$ with all zero entries of appropriate dimensions would be another feasible solution for any value L^*, R^*. To avoid an unbounded or trivial solution, an explicit constraint on either α or β needs to be introduced. Setting $\beta = 1$ would cause α still to be unbounded when the origin is located close to the optimal hyperplane. This leaves the non-convexity unit norm constraint (9d), that can be easily incorporated in the MILP formulation as shown in the next section.

B. Formulation as an MILP

We will first state the result before deriving it from (OC). Consider the *MILP formulation of the optimal cutting problem* (OCM) :

$$
\begin{aligned}
J(j) = \min_{y,\alpha,\beta,L,R} \; & J \\
\text{s.t.} \quad & \forall i = 1,2,...,N \\
& A_i^T y_i = \alpha && \text{(12a)} \\
& -M(1-L_i) \le b_i^T y_i - \beta \le M(1-R_i) && \text{(12b)} \\
& -M(1-R_i) \le y_i \le M(1-L_i) && \text{(12c)} \\
& L_i, R_i \in \{0,1\} && \text{(12d)} \\
& \alpha_j = 1 \qquad \|\alpha\|_\infty \le 1 && \text{(12e)}
\end{aligned}
$$

where $y = \{y_1, y_2, ..., y_N\} \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \ldots \times \mathbb{R}^{m_N}$. Note that the MILP is solved once for each dimension with only the last constraint on α changing. This is one of many possible implementations of the unit norm constraint in (OC). Alternatively one could introduce d extra integer variables indicating which element of α is constrained to unity.

Proposition 1: If $\{\alpha^*, \beta^*, L^*, R^*\}$ is the optimal solution to (OCM) over all j then they are also optimizers to (OC) and $J^* = \min_{j=1,2,...,d} J(j)$.

Proof: With an approach from robust optimization [12], one can remove the quantifiers in (OC) by introducing a second optimization level of linear programs. This gives the equivalent formulation

$$
\begin{aligned}
\min_{\alpha,\beta,L,R} \; & J \\
\text{s.t.} \quad & \forall i = 1,2,...,N \\
& L_i = 1 \leftrightarrow \left\{ \left(\min \left\{ \alpha^T x : A_i x \le b_i \right\} \right) \ge \beta \right\} \\
& R_i = 1 \leftrightarrow \left\{ \left(\max \left\{ \alpha^T x : A_i x \le b_i \right\} \right) \le \beta \right\} && \text{(13)}
\end{aligned}
$$

Assuming that all polytopes have non-empty interiors, strong linear programming duality yields for the first constraint

$$
\begin{aligned}
& \min \left\{ \alpha^T x : A_i x \le b_i \right\} \\
= & - \max \left\{ -\alpha^T x : A_i x \le b_i \right\} \\
= & - \min \left\{ b_i^T y : A_i^T y = -\alpha, y \ge 0 \right\} \\
= & - \min \left\{ -b_i^T y : A_i^T y = \alpha, y \le 0 \right\} && \text{(14)}
\end{aligned}
$$

and the logical expression of the optimization problem becomes a feasibility problem:

$$
\begin{aligned}
& \left\{ \left(\min \left\{ \alpha^T x : A_i x \le b_i \right\} \right) \ge \beta \right\} \\
\leftrightarrow & \left\{ \left(\min \left\{ -b_i^T y : A_i^T y = \alpha, y \le 0 \right\} \right) \le -\beta \right\} \\
\leftrightarrow & \left\{ \exists y : y \le 0, A_i^T y = \alpha, -b_i^T y \le -\beta \right\} \\
\leftrightarrow & \left\{ \exists y : y \le 0, A_i^T y = \alpha, b_i^T y - \beta \ge 0 \right\}. && \text{(15)}
\end{aligned}
$$

This allows one to rewrite the expression as linear constraints using the so-called big-M technique [13]:

$$
\begin{aligned}
b_i^T y - \beta \ge -M(1-L_i) \\
y \le M(1-L_i) && \text{(16)}
\end{aligned}
$$

where M is a positive scalar constant larger than any value the absolute value of the left side of the inequality can attain. Relation (16) enforces only the implication $\rightarrow$ in (9a). The converse $\leftarrow$ follows from the minimization of the cost function, which penalizes unassigned polytopes with $L_i = 0$. Note that we use only one M variable for simplicity of notation. In practice, one has to carefully select a different bound for each inequality and region to have the best possible numerical properties [14].

With a similar derivation for the second logical expression, we obtain (OCM). ■

In contrast to the original formulation (OC) with universal quantifiers ($\forall$), the optimal cutting problem is now formulated as d one-level MILPs, linear in all optimization variables. Choosing

$$
\begin{aligned}
\mathcal{P}_l &= \{ \mathrm{P}_i : L_i = 1 \quad \text{or} \quad L_i = R_i = 0 \} \\
\mathcal{P}_r &= \{ \mathrm{P}_i : R_i = 1 \quad \text{or} \quad L_i = R_i = 0 \} && \text{(17)}
\end{aligned}
$$

one obtains the prerequisites for Lemmas 1 and 2:

Lemma 4: (a) If $\mathcal{P}$ has more than one element, $\mathcal{P}_l$ and $\mathcal{P}_r$ are strictly smaller than $\mathcal{P}$.

(b) Equation (5) holds.

Proof: (a) Non-overlapping polytopes always admit a separating hyperplane due to their convexity. Hence, at least one polytope is assigned exclusively to each side, left and right, and the result follows.

(b) With (17) and (11), $\mathcal{P}_l$ contains all polytopes $\mathrm{P}_i \in \mathcal{P}$, except those with $R_i = 1$. Since (9b) holds, the first equations of (5) follows. The argument for $\mathcal{P}_r$ is identical. ■

IV. Modifications and extensions

This section lists some useful modification of (OCM) and derives the large margin extension needed for suboptimal EMPC.

A. Modifications

Optimization algorithms can be terminated early to return suboptimal solutions. In order to still give guarantees on how much the tree differs from optimum using Lemma 3, tightening constraints can be added to (OCM):

$$
\sum_{i=1}^{N} L_i \ge t \cdot N \qquad \sum_{i=1}^{N} R_i \ge t \cdot N \qquad \text{(18)}
$$

For numerical stability of (OCM) and for overlapping polytopes, slack variables can be introduced. The costfunction of (OCM) can be modified to account for regions with identical control laws, in order to reduce the depth of the tree.

B. Large margin and relation to support vector machines

Recent results on suboptimal robust EMPC study controllers that only use a subset of the polytopic regions of the original EMPC solution as an approximation of the optimal solution [2], while still giving some stabilizing guarantees [3]. The resulting control policy is defined over a set of non-overlapping polytopes that are possibly seperated in space.

During the online evaluation, points outside the polytopic set are assigned to one of the polytopes nearby. A binary hyperplane decision tree for that task can be obtained with a small modification of (OCM). The separating hyperplane $\mathrm{H}(\alpha, \beta)$ is chosen to maximize the distance to the closest polytope entirely on the left or right of the hyperplane. Such a "best fit" of the decision criterion is closely related to *support vector machines* (SVM), see [15]. The solution to the optimal cutting problem for polytopic sets seperated in space can be seen as a type of SVM for polytopic data instead of point data. One crucial difference is, that the assignement to the two sets is not known beforehand but determined through an optimization problem, captured in the variables L_i and R_i in (12d). The following derivation reformulates the optimal cutting problem (OCM) to maximize the margin around the hyperplane containing no polytopes assigned to either left or right. Let x_+ (x_-) denote the extreme point of the polytope in question closest to the left (right) of the hyperplane and let x_+^P (x_-^P) be the projection of this point onto $\mathrm{H}(\alpha, \beta)$. If the points x_+ and x_- do not have the same distance to the hyperplane H, then H can be shifted until they do.
Define $\mathrm{H}_+(\alpha, \beta + 1)$ ($\mathrm{H}_-(\alpha, \beta - 1)$) as the hyperplanes intersecting x_+ (x_-). We have

$$x_+ = x_+^P + \frac{\alpha}{\|\alpha\|_2}\|x_+ - x_+^P\|_2 \tag{19}$$

and want to maximize the distance $\|x_+ - x_+^P\|_2$. Taking the inner product with α gives

$$\begin{aligned} \alpha^T x_+ &= \alpha^T x_+^P + \frac{\alpha^T \alpha}{\|\alpha\|_2}\|x_+ - x_+^P\|_2 \\ \beta + 1 &= \beta + \|\alpha\|_2\|x_+ - x_+^P\|_2 \\ \|x_+ - x_+^P\|_2 &= \frac{1}{\|\alpha\|_2} \end{aligned} \tag{20}$$

Hence, maximizing the thickness of the margin around the hyperplane is equivalent to minimizing the 2-norm of α and can be captured through an additional term in the cost function and the constraints of (OCM), resulting in the *Large Margin Optimal Cutting problem* (LM-OCM):

$$\begin{aligned} J(j) = \min_{y,\alpha,\beta,L,R}\ & J_a + \omega_2 \cdot \alpha^T \alpha \\ \text{s.t.}\quad & \forall i = 1, 2, ..., N \\ & \text{(12a), (12c), (12d)} \\ & -M(1 - L_i) \le b_i^T y_i - \beta - L_i \\ & b_i^T y_i - \beta + R_i \le M(1 - R_i) \end{aligned} \tag{21}$$

where the central term of the inequality follows from the hyperplane with $\beta + 1 (\beta - 1)$ whenever $L_i = 1 (R_i = 1)$. As for the (OCM), it is possible to introduce slack variables to the (LM-OCM) . In SVMs, slack variables are also well known and are used to characterize the tolerance for misclassifications [16].

V. Simulation Results

This section compares the optimized tree to the facet tree for an ordinary and a suboptimal EMPC solution.

	Optimized tree	Facet tree
depth of the tree	16	15
non-terminal nodes	4385	4218
terminal nodes (leafs)	4386	4219

TABLE I

Trees obtained for a set of 487 3D-polytopes (Optimization terminated early)

A. Ordinary EMPC

The test data is a polytopic set from the EMPC solution for a LTI system with statefeedback:

$$\begin{aligned} x_{k+1} &= \begin{pmatrix} 0.7 & -0.1 & 0 \\ 0.2 & -0.5 & 0.1 \\ 0 & 0.1 & 0.1 \end{pmatrix} x_k + \begin{pmatrix} 0.1 & 0 \\ 0.1 & 0.1 \\ 0.1 & 0 \end{pmatrix} u_k \\ -500 &\le x_i \le 480 \quad i = 1, 2, 3 \\ -5 &\le u_j \le 5 \quad j = 1, 2 \end{aligned} \tag{22}$$

The EMPC solution is calculated with the Multi-Parametric Toolbox (MPT) [17] in MATLAB, no preproccessing step like the removal of redundant control laws are applied. The resulting polytopic set consists of $N = 487$ polytopes in three dimensions. Algorithm 2 has been implemented in MATLAB, with the YALMIP-interface [18] using the ILOG-CPLEX Optimizer [19]. The assignment ratio was chosen to be $t = 0.1$. For the facet tree, the standard MPT implementation (MPT_SEARCHTREE) has been used.

In Table V-A, the depth of both trees is larger than the theoretical depth of $\lceil \log_2 N \rceil = 9$, the optimized tree having a slightly larger depth and number of nodes than the facet tree. This is because the first MILPs have up to 1000 integer variables and were terminated early with a suboptimal tree.

On small problems in two dimensions, the optimized tree had the same depth and size as the facet tree. As for now, no higher dimensional problem could be solved with a significant improvement over the facet tree.

B. Suboptimal EMPC

The case study to present (OC-LM) is based on a random subset of the EMPC solution of a 2D plant, obtaining the blue regions in Fig. 2. Together with Fig. 3 the plots illustrate the trees obtained with the two methods. The red cells are the sets of points that are assigned to the same region when evaluating the search tree - namely to the blue polytope contained in the cells. For the facet tree in Fig. 2, it can be seen that the hyperplanes were generated by the facets of the polytopes. Points very close to a polytope but on the other side of the hyperplane are assigned to a different polytope further away. The optimized tree, Fig. 3, fits the hyperplanes better between the polytopes. Also, the additional freedom when choosing the hyperplanes decreases the size of the tree, reflected in the total number of nodes: 13 for the optimized vs. 21 for the facet tree.
To implement suboptimal EMPC controllers, it is therefore preferable to use the optimized search trees with the large margin extension.

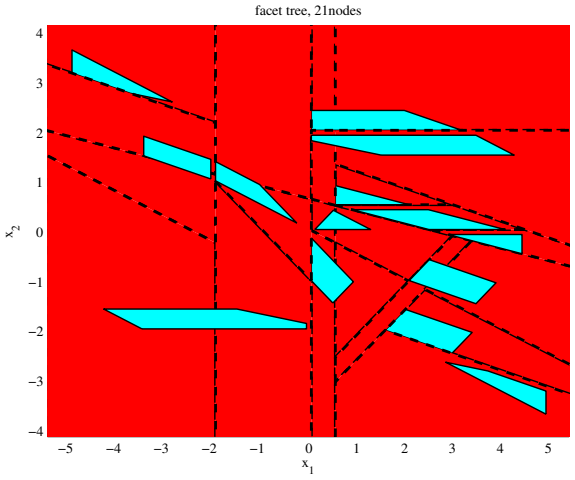

Fig. 2. Facet tree for a 2D polytopic set: polytopes (blue), hyperplane segments (dashed lines), cells of points (red) assigned to the polytope contained in the cell

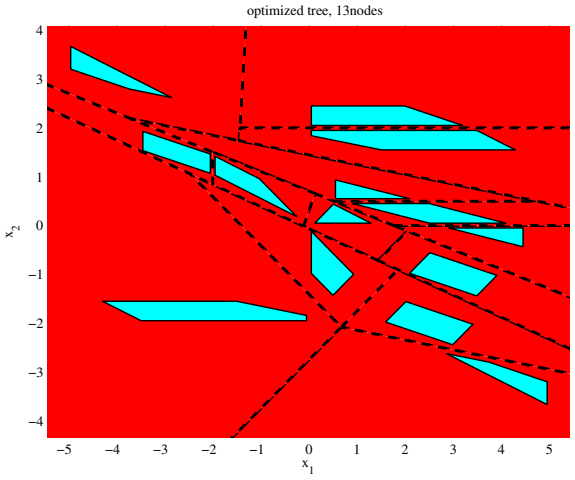

Fig. 3. Optimized tree for a 2D polytopic set: polytopes (blue), hyperplane segments (dashed lines), cells of points (red) assigned to the polytope contained in the cell

VI. Conclusion

We showed how to build a search tree for point-location in polytopic sets using MILPs to determine the best hyperplanes used as decision criteria. The method performes slightly worse than the facet tree on a 3D example of ordinary EMPC because the MILPs could not be solved to optimality. For a random 2D polytopic set separated in space, as could occur in suboptimal EMPC, the method significantly reduced the number of nodes of the tree. Besides, the hyperplanes are spread between the polytopes because they are not constrained to the facets.

For future work, semidefinite relaxations are suggested to obtain optimized trees for higherdimensional polytopic sets.

References

[1] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The Explicit Solution of Model Predictive Control via Multiparametric Quadratic Programming. In *American Control Conference*, pages 872–876, Chicago, USA, June 2000.

[2] G. Pannocchia, J. B. Rawlings, and S. J. Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43:852–860, 2007.

[3] H. Manum, C.N. Jones, J. Loefberg, M. Morari, and S. Skogestad. Bilevel programming for analysis of low-complexity control of linear systems with constraints. 2009.

[4] P.Tondel, T.A.Johansen, and A.Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, 2003.

[5] M. Baotic, F. Borrelli, A. Bemporad, and M. Morari. Efficient On-Line Computation of Constrained Optimal Control. *SIAM Journal on Control and Optimization*, 47(5):2470–2489, September 2008.

[6] C.N. Jones, P. Grieder, and S. Rakovic. A Logarithmic-Time Solution to the Point Location Problem for Parametric Linear Programming. *Automatica*, 42(12):2215–2218, December 2006.

[7] Y. Wang, C.N. Jones, and Jan M. Maciejowski. Efficient point location via subdivision walking with application to explicit MPC. In *European Control Conference*, Kos, Greece, July 2007.

[8] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, 1974.

[9] Yi jen Chiang and Roberto Tamassia. Dynamic algorithms in computational geometry. volume 80, pages 1412–1434, 1992.

[10] Schwarzkopf and Otfried. Ray shooting in convex polytopes. In *SCG '92: Proceedings of the eighth annual symposium on Computational geometry*, pages 286–295, New York, NY, USA, 1992. ACM.

[11] B. Chazelle and J. Friedman. Point location among hyperplanes and unidirectional ray-shooting. volume 4, pages 53–62. Elsevier, 1994.

[12] A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[13] J. Löfberg. Big-M reformulation - a tutorial, http://control.ee.ethz.ch/joloef/wiki/pmwiki.php?n=Tutorials.Big-M, 10/2009.

[14] Lou Hafer. Tightening big M constraints. *DIMACS Workshop on COIN-OR, July 17-20 2009*.

[15] Guosheng Wang. A survey on training algorithms for support vector machine classifiers. *Networked Computing and Advanced Information Management, International Conference on*, 1:123–128, 2008.

[16] C. Cortes and V. Vapnik. Support-vector networks. volume 20, pages 273–297. Kluwer Academic Publishers, 1995.

[17] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-Parametric Toolbox (MPT). In *HSCC (Hybrid Systems: Computation and Control)*, pages 448–462, March 2004.

[18] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD*, Taipei, Taiwan, 2004.

[19] ILOG CPLEX webpage. http://www.ilog.com/products/cplex/. 2009.