2007 International Symposium on Nonlinear Theory and its Applications
NOLTA'07, Vancouver, Canada, September 16-19, 2007

A Primal-Dual Beneath-Beyond Method

Yuzo Ohta* and Masashi Tsumura*

* School of Engineering, Kobe University, 1-1 Rokkodai, Nada, Kobe 657-8501, Japan
Email: ohta@cs.kobe-u.ac.jp

Abstract—The purpose of this paper is to propose an improved primal-dual beneath–beyond method to solve dynamic convex hull problem in d-dimensional space efficiently. The traditional beneath–beyond method requires the whole data of its faces and their inclusion relations. However, in the application of the dynamic convex hull problem, we usually do not need data of all faces. When the dimension d becomes large, then the computing time to maintain these unnecessary data becomes very large. In this respect, we propose a method which need data of facets and subfacets, edges, and nodes. This is useful in saving not only the storage but also the computing time.

1. Introduction

The convex hull problem is one of the most fundamental problems in the computational geometry [1]. In particular, in stability analysis of dynamical systems using computer [2]–[8], and in the construction of the Maximal Admissible Sets (MASs) for constrained systems [9]–[14], the dynamic convex hull algorithm plays a crucial role. The beneath–beyond (BB) method [1] is one of the most powerful method to solve dynamic convex hull problem in d-dimensional spaces. It maintains data of all k-faces, $0 \leq k \leq d-1$, where d is the dimension of the space we are concern. To construct Polytopic Lyapunov Functions [2]–[5], we only need data of facets (($d-1$)-faces) and nodes (0-faces), and, hence, the original BB method is not efficient for this application, and an improved BB method was proposed in [6], which maintains only data of facets, subfacets (($d-2$)-faces), and nodes.

On the other hand, in the construction of Piecewise Linear Lyapunov Functions (PLLFs) [7], [8], we also need data of edges (1-faces). In [13], we proposed to adopt concept of the dual polytope in the construction of MASs. There is one to one corresponding between a k-face of a given primal d-polytope P and a ($d-1-k$)-face of its dual polytope P^D, and, hence, facets, subfacets, edges and nodes of P and nodes, edges, subfacets, and facets corresponds, respectively. The original BB method has data of all k-faces, and, hence, it can represent both the primal polytope and the dual polytope in a single polytope data structure. On the other hand, the modified BB method can not do this, since it has not data of edges.

In this paper, we modify the BB method in [6] so that it also maintains data of edges too. By this, this new BB method not only recovers symmetry in data structure, but also keep the superiority in efficiency.

Notation: In this paper, $\mathbf{R}$ denotes the real number system, and $\mathbf{R}^d$ is the usual vector space of real d–dimensional vectors $x = [x_1, x_2, ..., x_d]^\top$. All vectors are to be regarded as column vectors for the purpose of matrix multiplications. The inner product of two vectors x and y in $\mathbf{R}^d$ is expressed by $(x|y) = \sum_{i=1}^{d} x_i y_i$. For a set P in $\mathbf{R}^d$, the interior, the affine hull and the convex hull of P are denoted by int P, aff P and co P, respectively. For a set V having a finite number of elements $|V|$ denotes the cardinality of V.

2. Faces, Dual polytopes and Coloring

A polytope $P \subseteq \mathbf{R}^d$ is usually give by

$$P = \text{co } (x_1, x_2, \cdots, x_n), \tag{1}$$

where $x_k \in \mathbf{R}^d$ for all k.

When P has an interior point, dim aff $P = d$, and P is said a d-polytope. In the following, for simplicity, we assume that 0 is an interior point of P. Then, P is represented also by

$$P = \{x : (h_i|x) \leq 1, \quad i = 1, 2, \cdots, m\}, \tag{2}$$

where $h_i \in \mathbf{R}^d$ for all i.

Let us consider a hyper plane $\mathcal{H}_1(\eta) = \{x : (\eta|x) = 1\}$ in $\mathbf{R}^d$ which does not intersect 0, and define open half spaces $S_-(\eta)$ and $S_+(\eta)$ by $S_-(\eta) = \{x \in \mathbf{R}^d : (\eta|x) < 1\}$, and $S_+(\eta) = \{x \in \mathbf{R}^d : (\eta|x) > 1\}$. A hyper plane $\mathcal{H}_1(\eta)$ is a supporting hyper plane of the polytope P if and only if conditions int $P \subseteq S_-(\eta)$ and $F = P \bigcap \mathcal{H}_1(\eta) \neq \emptyset$ hold, and F is a k-face of P, where $k =$ dim aff F. Note that any k-face is a polytope in $\mathbf{R}^k$. Let us denote the set of all faces of P and the set of all k-faces of P by $\mathcal{F}(P)$ and $\mathcal{F}_k(P)$, respectively. In particular, we denote $\mathcal{F}_0(P)$ by node P. A ($d-1$)-face $F \in \mathcal{F}_{d-1}(P)$, a ($d-2$)-face $G \in \mathcal{F}_{d-2}(P)$, a 1-face $l \in \mathcal{F}_1(P)$, and a 0-face $x \in$ node P are called a facet, a subfacet, an edge, a node, respectively.

The vector h_i in (2) is called the normalized normal vector of a facet F_i, since it satisfies

$$(h_i|x) = 1 \ \ \forall\, x \in F_i. \tag{3}$$

The dual polytope P^D of the polytope P can be defined using normalized normal vectors of P as follows.

$$P^D = \text{co } (h_1, h_2, \cdots, h_m), \quad m = |\mathcal{F}_{d-1}(P)|. \tag{4}$$

Definition 1 *Coloring of faces. Let $p \in \mathbf{R}^d$, $F_i \in \mathcal{F}_{d-1}(P)$, and let h_i be the normalized normal vector of F_i. The facet F_i is colored as follows:*
1) F_i is yellow if $p \in$ aff F_i, i.e., $(h_i|p) = 1$,
2) F_i is blue if $p \in S_-(F_i)$, i.e., $(h_i|p) < 1$, and
3) F_i is red if $p \in S_+(F_i)$, i.e., $(h_i|p) > 1$.

Traditionally, it is said that p is beneath (beyond, on, respectively) the facet F_i if F_i is blue (red, yellow, respectively), but we use color since it is convenient to define colors k-faces ($k < d-1$).

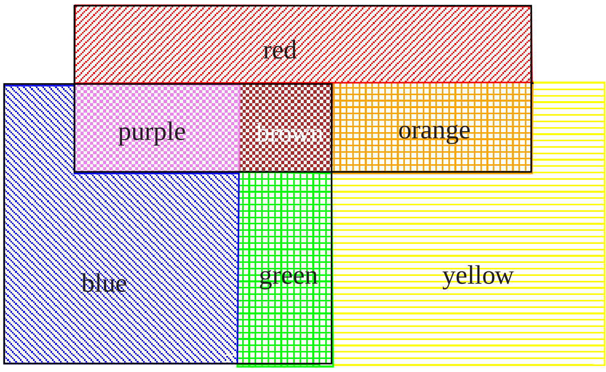


Fig. 1 The three primary colors and mixture of them.

We code colors by using 3-bit data: We correspond yellow, blue and red to 001, 010 and 100, respectively. We define the logical OR operation $\vee$ to these 3-bit data as follows: $(b_1\ b_2\ b_3) = (b_1^1\ b_2^1\ b_3^1) \vee (b_1^2\ b_2^2\ b_3^2)$ is defined by $b_j = b_j^1 \vee b_j^2$, $j = 1, 2, 3$. Moreover, we define $(b_1\ b_2\ b_3) = \bigvee_{k=1}^m (b_1^k\ b_2^k\ b_3^k)$ by $b_j = \bigvee_{k=1}^m b_j^k$, $j = 1, 2, 3$, and

$$\bigvee_{k=1}^{m} b_j^k = (\cdots((b_j^1 \vee b_j^2) \vee b_j^3)\cdots) \vee b_j^m)$$

Definition 2 *Coloring of k-faces.*
Let $G_j \in \mathcal{F}_k(P)$, $k < \dim P - 1$, and let $\{F_{i_j}\}_{j=1}^{m_i}$ be the set of all facets including G_j. Suppose that F_{i_j} has been classified as $(b_1^{i_j}, b_2^{i_j}, b_3^{i_j})$. Set $(b_1\ b_2\ b_3) = \bigvee_{j=1}^m (b_1^{i_j}\ b_2^{i_j}\ b_3^{i_j})$. Then, G_j is colored as follows:
1) G_j is yellow if $(b_1\ b_2\ b_3) = (001)$,
2) G_j is blue if $(b_1\ b_2\ b_3) = (010)$,
3) G_j is red if $(b_1\ b_2\ b_3) = (100)$,
4) G_j is orange if $(b_1\ b_2\ b_3) = (101)$,
5) G_j is green if $(b_1\ b_2\ b_3) = (011)$,
6) G_j is purple if $(b_1\ b_2\ b_3) = (110)$, and
7) G_j is brown if $(b_1\ b_2\ b_3) = (111)$.
We denote the set of all yellow, blue, red, orange, green, purple, brown k-faces of P by $\mathcal{F}_k^Y(P)$, $\mathcal{F}_k^{Bl}(P)$, $\mathcal{F}_k^R(P)$, $\mathcal{F}_k^O(P)$, $\mathcal{F}_k^G(P)$, $\mathcal{F}_k^P(P)$, $\mathcal{F}_k^{Br}(P)$, respectively.

Lemma 1 *Beneath–Beyond Theorem[6].*
Let us consider d-polytope $P \in \mathbf{R}^n$ and a point $p \in \mathbf{R}^d$ such that $p \notin P$, and let $\hat{P} =$ co $(P \cup \{p\})$. Let $k \in \{0, 1, ..., d-1\}$ and let

$$\mathcal{F}_k^{\overline{B}}(P) = \mathcal{F}_k^{Bl}(P) \cup \mathcal{F}_k^{G}(P) \cup \mathcal{F}_k^{P}(P) \cup \mathcal{F}_k^{Br}(P), \tag{5}$$

$$\mathcal{F}_{k-1}^{PBr}(P) = (\mathcal{F}_{k-1}^{P}(P) \cup \mathcal{F}_{k-1}^{Br}(P)), \tag{6}$$

$$\hat{\mathcal{F}}_k^{PBr}(P) = \{\text{ co } (F \cup \{p\}) \mid F \in \mathcal{F}_{k-1}^{PBr}(P)\}, \tag{7}$$

and

$$\overline{\mathcal{F}}_k^{Y}(P) = \{\text{ co } (F \cup \{p\}) \mid F \in \mathcal{F}_k^{Y}(P)\} \tag{8}$$

Then, the set $\mathcal{F}_k(\hat{P})$ of all k-faces of $\hat{P}$ is given by

$$\mathcal{F}_k(\hat{P}) = \mathcal{F}_k^{\overline{B}}(P) \cup \hat{\mathcal{F}}_k^{PBr}(P) \cup \overline{F}_k^{Y}(P) \tag{9}$$

and

$$\mathcal{F}_k^{\overline{B}}(P) \cap \hat{\mathcal{F}}_k^{PBr}(P) = \mathcal{F}_k^{\overline{B}}(P) \cap \overline{\mathcal{F}}_k^{Y}(P) = \emptyset, \tag{10}$$

$$\hat{\mathcal{F}}_k^{PBr}(P) \cap \overline{\mathcal{F}}_k^{Y}(P) = \emptyset. \tag{11}$$

In [1], it is not mentioned about the dimension of faces, it is added in [6]. The Beneath–Beyond Method is an algorithm to compute $\hat{P}$ based on Lemma 1. Lemma 1 means that we can compute $\mathcal{F}_k(\hat{P})$ if we know $\mathcal{F}_k(P)$, $\mathcal{F}_{k-1}(P)$ and color of them. In other words, if we want to solve the dynamic convex hull problem by applying Lemma 1 directly, then we need to update all $\mathcal{F}(\hat{P})$ and their color.

3. Further Improvement of BB Method

In this section, we consider a dynamic convex hull problem, that is, for a given d-polytope $P_0 \in \mathbf{R}^d$ and a set of points $\{p_n \in \mathbf{R}^d\}_{n=0}^{N-1}$, we want to compute nodes and edges of $\{P_n\}_{n=1}^{N}$, where

$$P_{n+1} = \text{ co } (P_n \cup \{p_n\}), \quad n = 0, 1, 2, \cdots, N-1. \tag{12}$$

We will propose a method such that we compute P_{n+1} in (12) using data of $\mathcal{F}_k(P_n)$, where $k = 0, 1, d-2, d-1$.

3.1. Data Structure and Coloring

Outline of data structure of Polytope is the following:

- Polytope P_n has lists of $\{x_k \in \mathcal{F}_0(P_n)\}$, $\{e_l \in \mathcal{F}_1(P_n)\}$, $\{G_j \in \mathcal{F}_{d-2}(P_n)\}$, and $\{F_i \in \mathcal{F}_{d-1}(P_n)\}$.
- Facet F_i has the normalized normal vector h_i, the color bits (b_1, b_2, b_3), and lists of pointers of $\{x_{k_i} \in \mathcal{F}_0(F_i)\}$, $\{e_{l_i} \in \mathcal{F}_1(F_i)\}$, and $\{G_{j_i} \in \mathcal{F}_{d-2}(F_i)\}$. When $\mathcal{F}_0(F_1) = \{x_1,\ x_2,\ \cdots,\ x_m\}$, the normalized normal vector h_i is computed by

$$h_i = (XX^\top)^{-1}Xe, \quad e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top \in \mathbf{R}^m, \tag{13}$$

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \in \mathbf{R}^{d \times m}. \tag{14}$$

- Subfacet G_j has the color bits (b_1, b_2, b_3), pointers of Facets $F_{i_j}, F_{i'_j} \in \mathcal{F}_{d-1}(P_n)$ such that $G_j = F_{i_j} \cap F_{i'_j}$ and pointers of Nodes $\mathcal{F}_0(G_j)$.
- Edge e_l has the color bits (b_1, b_2, b_3), and pointers of Nodes $x_{k_l}, x_{k'_l} \in$ node P_n such that $e_l =$ co $(x_{k_l}, x_{k'_l})$.

- Node x_k has the d-dimensional vector x_k which represent the location of the node, the color bits (b_1, b_2, b_3), and the list of pointers of Facets F_{i_k} such that $x_i \in \mathcal{F}_0(F_{i_k})$.

If we insist to keep symmetry in data structure, we need to assume that Nodes also has a list of pointers of $\{G_{j_i} \in \mathcal{F}_{d-2}(F_i)\}$ and that Edge e_l also has a list of pointers of Facets F_{i_l}, $F_{i_l} \in \mathcal{F}_{d-1}(P_n)$ such that $e_l \in \mathcal{F}_1(F_{i_l})$. However, we omitted them, since these data are not used in our applications.

The color bits (b_1, b_2, b_3) of a Facet F_i, a Subfacet G_j, an Edge e_l, and a Node x_k are represented by $F_i.(b_1, b_2, b_3)$, $G_j.(b_1, b_2, b_3)$, $e_l.(b_1, b_2, b_3)$, and $x_k.(b_1, b_2, b_3)$, respectively. We assume color bits of Facets, Subfacets, Edges and Nodes are initialized as (000).

When P_n and p_n are given, we examine that whether there is a red Facet or not by the brute force method.

If there is not red Facet, then $p_n \in P_n$ and we just set $P_{n+1} = P_n$.

In the following, we consider the case when there are red facets. When we find a red Facet, we switch to a sophisticated method, which uses the facts that Facets which will be colored red or yellow are adjacent and that any Subfacet is the intersection of exactly 2 Facets. This is a modification of Procedure 8.7 (Coloring-Phase 1)([1], p. 154).

Suppose that $F_i.(b_1, b_2, b_3)$ has been determined just now. Then, do the following.
1) For all Edges $e_l \in \mathcal{F}_1(F_i)$ compute $e_l.(b_1, b_2, b_3) := e_l.(b_1, b_2, b_3) \vee F_i.(b_1, b_2, b_3)$.
2) For all Nodes $x_k \in \mathcal{F}_0(F_i)$ compute $x_k.(b_1, b_2, b_3) := x_k.(b_1, b_2, b_3) \vee F_i.(b_1, b_2, b_3)$.
3) For all Subfacets $G_i \in \mathcal{F}_{d-2}(F_i)$, visit G_i and compute $G_j.(b_1, b_2, b_3) := G_j.(b_1, b_2, b_3) \vee F_i.(b_1, b_2, b_3)$. Suppose that $G_j = F_i \cap F_{i'}$. If $F_i.(b_1, b_2, b_3) = (010)$, then do nothing. If $F_i.(b_1, b_2, b_3) \neq (010)$ and if $F_{i'}.(b_1, b_2, b_3) \neq (010)$, then do nothing since $F_{i'}$ has been already colored. If $F_i.(b_1, b_2, b_3) \neq (010)$ and if $F_{i'}.(b_1, b_2, b_3) = (010)$, then determine $F_{i'}.(b_1, b_2, b_3)$ and do the above processes 1)–3) recursively.

In this way, colors of Facets, Subfacets, Edges, Nodes are determined, and we understand that $(b_1, b_2, b_3) = (000)$ means that its color is blue.

We memorize all faces whose color code are modified, and we reset them to (000) before we compute $P_{n+2} = \text{co}\,(P_{n+1}) \cup \{p_{n+1}\})$. By memorizing them, we can save computing time very much.

3.2. Computation of $\mathcal{F}_{d-1}(P_{n+1})$

We note that any facet $F_i \in \mathcal{F}_{d-1}(P_n)$ is blue, red or yellow and that there is no subfacet $G_j \in \mathcal{F}_{d-2}(P_n)$ which is brown. Therefore, by Lemma 1, we have

$$\mathcal{F}_{d-1}(P_{n+1}) = \mathcal{F}^{Bl}_{d-1}(P_n) \cup \hat{\mathcal{F}}^{P}_{d-1}(P_{n+1}) \cup \overline{\mathcal{F}}^{Y}_{d-1}(P_{n+1}), \quad (15)$$

$$\hat{\mathcal{F}}^{P}_{d-1}(P_{n+1}) = \{\,\text{co}\ (F \cup \{p_n\}) \,|\, F \in \mathcal{F}^{P}_{d-2}(P_n)\}, \quad (16)$$

$$\overline{\mathcal{F}}^{Y}_{d-1}(P_{n+1}) = \{\,\text{co}\ (F \cup \{p_n\}) \,|\, F \in \mathcal{F}^{Y}_{d-1}(P_n)\}. \quad (17)$$

If $F \in \mathcal{F}^{P}_{d-2}(P_n)$, then $p_n \notin \text{aff}\ F$, and, hence, we have node $\hat{F} = \{\,\text{node}\ F \cup \{p_n\}\}$, where $\hat{F} = \text{co}\ (F \cup \{p_n\})$. Therefore, node $\hat{F}$, $\hat{F} \in \hat{\mathcal{F}}^{P}_{d-1}(P_{n+1})$, is easily computed.

On the other hand, if $F \in \mathcal{F}^{Y}_{d-1}(P_n)$, then $p_n \in \text{aff}\ F$, and, in general, we have node $\overline{F} \neq \{\,\text{node}\ F \cup \{p_n\}\}$, where $\overline{F} = \text{co}\ (F \cup \{p_n\}) \in \overline{\mathcal{F}}^{Y}_{d-1}(P_{n+1})$. In this case, we need to determine and eliminate nodes which are not necessary to represent $\overline{F}$. In [6], the following relation is shown

$$\text{node}\ \overline{F} = \{p_n\} \cup (\bigcup_{G_j \in \mathcal{F}^{G}_{d-2}(P;F)} \text{node}\ G_j), \quad (18)$$

where $\mathcal{F}^{G}_{d-2}(P_n; F) = \{G_j \in \mathcal{F}^{G}_{d-2}(P_n) \,|\, G_j \subseteq F\}$.

In the right side of (18), there is no node which is not necessary to represent $\overline{F}$. But node $G_j \cap$ node $G_{j'} \neq \emptyset$, that is, we need to eliminate duplicate elements and leave exactly one of them. We can do this by a similar method with the merge sort.

3.3. Computation of $\mathcal{F}_{d-2}(P_{n+1})$

Since we do not have data of $\mathcal{F}^{PBr}_{d-3}(P_n)$, we can not apply Lemma 1 to compute $\mathcal{F}_{d-2}(P_{n+1})$. Therefore, we need an alternative method to compute it. To compute $\mathcal{F}_{d-2}(P_{n+1})$ we use the following [6]:

Lemma 2 *Let* $\tilde{\mathcal{F}}^{PY}_{d-1}(P_{n+1}) = \hat{\mathcal{F}}^{P}_{d-1}(P_{n+1}) \cup \overline{\mathcal{F}}^{Y}_{d-1}(P_{n+1})$, *where* $\hat{\mathcal{F}}^{P}_{d-1}(P_{n+1})$ *and* $\overline{\mathcal{F}}^{Y}_{d-1}(P_{n+1})$ *are given by* (16), (17)*. Then,* $\mathcal{F}_{d-2}(P_{n+1})$ *is given by*

$$\mathcal{F}_{d-2}(P_{n+1}) = \mathcal{F}^{\overline{B}}_{d-2}(P_n) \cup \tilde{\mathcal{F}}^{PY}_{d-2}(P_{n+1}) \quad (19)$$

where

$$\tilde{\mathcal{F}}^{PY}_{d-2}(P_{n+1}) = \bigcup_{\substack{F_1, F_2 \in \tilde{\mathcal{F}}^{PY}_{d-1}(P_{n+1}) \\ \dim(F_1 \cap F_2) = d-2}} F_1 \cap F_2. \quad (20)$$

3.4. Computation of $\mathcal{F}_1(P_{n+1})$ and $\mathcal{F}_0(P_{n+1})$

Since P is a polytope such that $0 \in \text{int}\ P$ and since $p \notin P$, there exist both red facets and blue facets, and, hence, there is no yellow node. Therefore, $\overline{\mathcal{F}}^{Y}_{0}(P) = \emptyset$. On the other hand, $\mathcal{F}^{PBr}_{-1}(P) = \phi$, which is colored purple or brown, and, hence, $\hat{\mathcal{F}}^{PBr}_{0}(P) = \{p\}$. Therefore, by Lemma 1, we have

$$\mathcal{F}_1(P_{n+1}) = \mathcal{F}^{\overline{B}}_1(P_n) \cup \hat{\mathcal{F}}^{PBr}_1(P_n) \cup \overline{F}^{Y}_1(P_n), \quad (21)$$

$$\mathcal{F}^{\overline{B}}_1(P_n) = \mathcal{F}^{Bl}_1(P_n) \cup \mathcal{F}^{G}_1(P_n) \cup \mathcal{F}^{P}_1(P_n) \cup \mathcal{F}^{Br}_1(P_n), \quad (22)$$

$$\hat{\mathcal{F}}^{PBr}_1(P_n) = \{\,\text{co}\ (F \cup \{p_n\}) \,|\, F \in \mathcal{F}^{PBr}_0(P_n)\}, \quad (23)$$

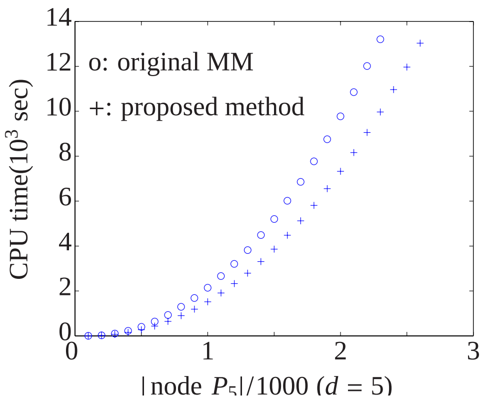
$$\mathcal{F}^{PBr}_0(P_n) = \mathcal{F}^{P}_0(P_n) \cup \mathcal{F}^{Br}_0(P_n), \quad (24)$$

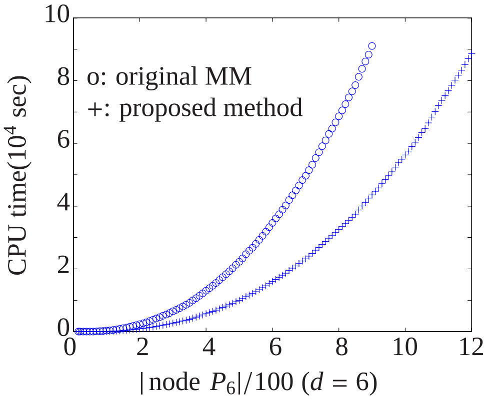
$$\overline{\mathcal{F}}^{Y}_1(P_n) = \{\,\text{co}\ (F \cup \{p_n\}) \,|\, F \in \mathcal{F}^{Y}_1(P_n)\}, \quad (25)$$

$$\mathcal{F}_0(P_{n+1}) = \mathcal{F}^{Bl}_0(P_n) \cup \mathcal{F}^{G}_0(P_n) \cup \mathcal{F}^{P}_0(P_n) \cup \mathcal{F}^{Br}_0(P_n) \cup \{p_n\}. \quad (26)$$

3.5. Numerical Experiments

When $d = 3$, both Subfacet and Edge in our data structure are $\mathcal{F}_1(P)$, and, hence, the original BB method may be more efficient. Similarly, when $d = 4$, we may have similar result. Therefore, we will examine the case when $d \geq 5$. Let P_0 be a very small polytope and we will generate vectors $\{p_n \in \mathbf{R}^d\}_{i=1}^N$ on the unit sphere, and solve convex hull problem. Computing time is the following:





We also apply the proposed method to construct MASs. Let us consider a position servomechanism consists of a DC motor, a gear-box, an elastic shaft and an uncertain load. This plant is modeled as a 4 dimensional system. Controller is a 4 dimensional dynamic controller. Thus, we consider 8 dimensional systems See [15] for details. We construct MASs for this system. If we use the original BB method, it takes about 1 hour to compute a MAS, while we can construct this in 6 seconds by the proposed method.

4. Conclusion

In this paper, we we modified the BB method in [6] so that it also maintains data of edges too. By this, this new BB method not only recovers symmetry in data structure, but also keep the superiority in efficiency. The computing time of the proposed method is 1/1.3 (1/2, respectively) that of the original beneath–beyond method when $d = 5$ ($d = 6$, respectively). We expect that this ratio will be small as g becomes large.

References

[1] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer-Verlag Berlin Heidelberg, 1987.

[2] H. H. Rosenbrock: A method of investigating stability, Proc. 2nd IFAC World Congress, Basel, Switzerland, pp. 590-594, 1963.

[3] R. K. Brayton and C. H. Tong: Stability of dynamical systems: a constructive approach, IEEE Trans. Circuits and Systems, Vol.CAS-26, No.4, pp. 224-234, 1979.

[4] Y. Ohta, H. Imanishi, L. Gong and H. Haneda: Computer generalized Lyapunov functions for a class of nonlinear systems, IEEE Trans. on Circuits and Systems-I, Vol.40, No.5, pp. 343-354, 1993.

[5] F. Blanchini: Nonquadratic Lyapunov functions for robust control, Automatica, Vol. 31, No. 3, pp. 451-461, 1995.

[6] Y. Ohta, Y. Nagai and L. Gong, "Beneath-Beyond Method and Construction of Lyapunov Functions," *Proc. NOLTA'97*, pp. 353-356 (1997).

[7] Y. Ohta and K. Yamamoto, "Stability Analysis of Nonlinear Systems via Piecewise Linear Lyapunov Functions," *Proc. of ISCAS 2000*, II, pp. 208–211, 2000.

[8] Yuzo Ohta, "On the Construction of Piecewise Linear Lyapunov Functions," *Proc. of CDC 2001*, pp. 2173-2178, 2001.

[9] E. G. Gilbert and K. T. Tan, "Linear system with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Trans. on Automatic Control*, vol. 36, pp. 1008–1020, 1991.

[10] E. G. Gilbert and I. Kolmanovsky, "Nonlinear tracking control in the presence of state and control constraints : a generalized reference governor," *Automatica*, vol. 38, pp. 2063–2073, 2002.

[11] K. Hirata and M. Fujita, "Set of admissible reference signals and control of systems with state and control constraints," *Proc. of CDC*, pp. 1427–1432, 1999.

[12] K. Kogiso and K. Hirata, "A reference governor in a piecewise state affine function," *Proc. of CDC*, pp. 1747–1752, 2003.

[13] Y. Ohta, T. Taguchi and T. Yamaguchi, "On-line reference inputs management for constrained servo systems," *Proc. of NOLTA'06*, pp. 891–894, 2006.

[14] Y. Ohta and H.Tanizawa, "On approximation of maximal admissible sets for nonlinear continuous-time systems with constraints," *Proc. of ACC07* (to appear), 2007.

[15] A. Bemporad and E. Mosca, "Fulfilling hard constraints in uncertain linear systems by reference managing," *Automatica*, vol. 34, pp. 451-461, 1998.