Minimal Ellipsoid Circumscribing a Polytope Defined by a System of Linear Inequalities

Jun-ya GOTOH[1] and Hiroshi KONNO[2]

Abstract In this paper, we will propose algorithms for calculating a minimal ellipsoid circumscribing a polytope defined by a system of linear inequalities. If we know all vertices of the polytope and its cardinality is not very large, we can solve the problem in an efficient manner by a number of existent algorithms. However, when the polytope is defined by linear inequalities, these algorithms may not work since the cardinality of vertices may be huge. Based on a fact that vertices determining an ellipsoid are only a fraction of these vertices, we propose algorithms which iteratively calculate an ellipsoid which covers a subset of vertices. Numerical experiment shows that these algorithms perform well for polytopes of dimension up to seven.

Keywords : minimal ellipsoid, computational geometry, global optimization

1 Introduction

This paper is concerned with algorithms for calculating a minimal ellipsoid circumscribing a polytope in a low dimensional Euclidean space. Calculation of a minimal sphere circumscribing a polytope has been studied by many authors. In particular, if we know all extreme points of a polytope and if its cardinality is not excessively large, then the problem can be solved efficiently by a number of algorithms. Among such algorithms are those of Elzinga and Hearn [4], Dyer [3], Skyum [18], Sekitani and Yamamoto [17] and others.

On the other hand, if a polytope is defined by a linear system of inequalities, the problem is NP complete. When the dimension of the underlying space is small, say less than 5 or 6, then the algorithm proposed by Konno et al. [11] can solve the problem in a reasonably efficient manner.

The problem to be discussed in this paper, namely that of calculating a minimal ellipsoid circumscribing a polytope, is a direct extension of a minimal sphere problem. When a polytope is defined by a set of finitely many points, the minimal ellipsoid can be calculated by algorithm of Barnes [1] based on quadratic programming approach and those by Khachiyan and Todd [9], Sun and Freund [19], Zhang [22], Zhang and Gao [23] using interior point algorithms. These can generate a minimal ellipsoid in an efficient way when the cardinality

[1]Institute of Policy and Planning Sciences, University of Tsukuba, 1-1-1 Tennoudai, Tsukuba-City, Ibaraki, 305-8573, JAPAN, e-mail : jgoto@sk.tsukuba.ac.jp.

[2]Department of Industrial and Systems Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, JAPAN, e-mail : konno@indsys.chuo-u.ac.jp.

of the point set is relatively small. Also, there are stochastic algorithms by Welzl [21], Gärtner and Schönherr [7] and others.

However, when the polytope $X \subset \mathbb{R}^n$ is defined by a set of linear inequalities, there exists no practical algorithm since a polytope may contain a huge number of extreme points.

According to a well known theorem by F.John [8], the minimal ellipsoid containing a polytope X is determined by a subset of at most $n(n+3)/2$ points. Hence, we do not have to identify all extreme points.

The algorithm to be proposed in this paper is to generate a sequence of ellipsoids containing a number of extreme points of X and update it by adjoining a new point not contained in it by solving a nonconvex quadratic programming problem.

In the next section, we will give a mathematical representation of the problem, and propose a basic scheme to generate its ε-optimal solution of it. Section 3 will be devoted to a branch and bound algorithm for maximizing a convex quadratic programming problem to find a point not contained in a given ellipsoid. Also, we will propose several schemes to enhance the efficiency of this algorithm.

In Section 4, we present the results of numerical experiments using sample problems in up to seven dimensional space. We will conclude the paper by adding some remarks and possible extensions. Finally, in the appendix, we will give a brief explanation of the algorithm of Sun and Freund [19] for calculating a minimal ellipsoid containing finitely many points, which will be used for implementing our algorithms in Section 4.

2 Minimal Ellipsoid Circumscribing a Polytope

An ellipsoid in n-dimensional Euclidean space $\mathbb{R}^n$ is defined by

$$E(D, \boldsymbol{c}) := \{\ \boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{c})^\top D(\boldsymbol{x} - \boldsymbol{c}) \leq 1\ \},$$

where D is a symmetric positive definite matrix and $\boldsymbol{c} \in \mathbb{R}^n$ is the center of the ellipsoid. It is well known (e.g. [2]) that the volume of an ellipsoid $E(D, \boldsymbol{c})$ is given by $(\det D)^{-\frac{1}{2}}$.

Let us consider a polytope

$$X := \{\ \boldsymbol{x} \in \mathbb{R}^n \mid A\,\boldsymbol{x} \leq \boldsymbol{b}\ \},$$

where $A \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and assume throughout

Assumption 1 : X is non-empty, bounded with an interior.

Then the minimal volume ellipsoid containing X can be obtained by solving the following optimization problem :

$$\text{(MECAP)} \quad \left| \begin{array}{ll} \underset{D,\ \boldsymbol{c}}{\text{minimize}} & \det\left(D^{-\frac{1}{2}}\right) \\ \text{subject to} & (\boldsymbol{x} - \boldsymbol{c})^\top D(\boldsymbol{x} - \boldsymbol{c}) \leq 1, \quad \forall \boldsymbol{x} \in X, \\ & D \succ O, \quad \boldsymbol{c} \in \mathbb{R}^n, \end{array} \right.$$

where $D \succ O$ denotes that D is positive definite.

The problem (MECAP) can be represented as follows :

$$(\text{MECAP}') \left| \begin{array}{ll} \underset{D,\ \boldsymbol{c}}{\text{minimize}} & \det\left(D^{-\frac{1}{2}}\right) \\ \text{subject to} & \max\limits_{\boldsymbol{x}\in X}\left\{(\boldsymbol{x}-\boldsymbol{c})^\top D(\boldsymbol{x}-\boldsymbol{c})\right\} \le 1, \\ & D \succ O, \quad \boldsymbol{c} \in \mathbb{R}^n. \end{array} \right.$$

For a fixed positive definite matrix D, the function $q(\boldsymbol{x}) := (\boldsymbol{x}-\boldsymbol{c})^\top D(\boldsymbol{x}-\boldsymbol{c})$ is convex, so that the problem can be rewritten as follows :

$$(\text{MEC}(V_X)) \left| \begin{array}{ll} \underset{D,\ \boldsymbol{c}}{\text{minimize}} & \det\left(D^{-\frac{1}{2}}\right) \\ \text{subject to} & (\boldsymbol{v}-\boldsymbol{c})^\top D(\boldsymbol{v}-\boldsymbol{c}) \le 1, \quad \forall \boldsymbol{v} \in V_X, \\ & D \succ O, \quad \boldsymbol{c} \in \mathbb{R}^n, \end{array} \right.$$

where V_X is the set of vertices of X.

A minimal ellipsoid containing X can be obtained by existent algorithms, e.g., [1, 19, 21], if we know all extreme points $\boldsymbol{v}_j$, $j = 1, ..., l'$. However, l' can be very large even for small n, so that it may not be easy to solve (MEC(V_X)) by these algorithms.

The algorithm to be proposed below is based upon a branch and bound algorithm for concave minimization problem ([6]) and improved by Phong et al. [15].

At the $(k+1)$-st iteration, we are given the set V_k of $k + k_0$ distinct extreme points $\boldsymbol{v}_j$, $j = 1, 2, ..., k + k_0$. Also, let

$$E(D^k, \boldsymbol{c}^k) := \left\{ \ \boldsymbol{x} \in \mathbb{R}^n \ \middle| \ (\boldsymbol{x}-\boldsymbol{c}^k)^\top D^k(\boldsymbol{x}-\boldsymbol{c}^k) \le 1 \ \right\}$$

be the smallest ellipsoid containing $\boldsymbol{v}_j$'s, which can be calculated by one of the algorithms such as DRN algorithm of Sun and Freund [19].

Let us consider the following quadratic programming problem :

$$(\text{NCQP}(D^k, \boldsymbol{c}^k)) \left| \begin{array}{ll} \underset{\boldsymbol{x}\ \in\ \mathbb{R}^n}{\text{maximize}} & (\boldsymbol{x}-\boldsymbol{c}^k)^\top D^k(\boldsymbol{x}-\boldsymbol{c}^k) \\ \text{subject to} & \boldsymbol{x} \in X. \end{array} \right.$$

Since X is bounded, the problem has an optimal solution $\boldsymbol{v}^k$, which is an extreme point of X. If

$$(\boldsymbol{v}^k-\boldsymbol{c}^k)^\top D^k(\boldsymbol{v}^k-\boldsymbol{c}^k) \le 1, \tag{1}$$

then $E(D^k, \boldsymbol{c}^k)$ is obviously a smallest ellipsoid circumscribing X. If, on the other hand, the inequality (1) is not satisfied, then let

$$V^{k+1} \leftarrow V^k \cup \{\boldsymbol{v}^k\},$$

and continue the process.

Basic Algorithm (Algorithm 1) Let $\varepsilon > 0$ be a tolerance.

Step 1. Let $V^0 = \left\{ \boldsymbol{v}^1, ..., \boldsymbol{v}^{k_0} \right\}$ be a set of vertices of X whose affine hull spans $\mathbb{R}^n$, and set $k \leftarrow 1$.

Step 2. Calculate the smallest ellipsoid $E(D^k, \boldsymbol{c}^k)$ covering V^k by solving (MEC(V^k)).

Step 3. Calculate $\boldsymbol{v}^k$ by solving problem :

$$(\mathrm{NCQP}(D^k, \boldsymbol{c}^k)) \left| \begin{array}{ll} \underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{maximize}} & (\boldsymbol{x} - \boldsymbol{c}^k)^\top D^k (\boldsymbol{x} - \boldsymbol{c}^k) \\ \text{subject to} & \boldsymbol{x} \in X. \end{array} \right.$$

If $(\boldsymbol{v}^k - \boldsymbol{c}^k)^\top D^k (\boldsymbol{v}^k - \boldsymbol{c}^k) \leq 1 + \varepsilon$, then end. Else set $V^{k+1} \leftarrow V^k \cup \{\boldsymbol{v}^k\}$, $k \leftarrow k+1$ and go to **Step 2.**

Theorem 1 *Algorithm 1 terminates in finitely many steps generating a smallest ellipsoid containing X.*

Proof. The vertex $\boldsymbol{v}^k$ is distinct from those in V^{k-1}. The number of vertices of X is finite and the result follows. ‖

For calculating a minimal ellipsoid in Step 2, a number of algorithms are available. We will use an interior point algorithm proposed by Sun and Freund [19] which is considered to be one of the most efficient deterministic algorithms. We will summarize their algorithm in Appendix to make the paper self-contained.

3 Branch and Bound Algorithm and Shortcut Strategy

Let us now turn to the algorithm for solving a convex maximization problem (NCQP($D, \boldsymbol{c}$)) of Section 2. This is admittedly a very difficult global optimization problem. However, when the dimensionality of the underlying space is small, then the problem can be solved in an efficient way by a branch and bound algorithm ([15]).

Let us first define a new set of variables $\boldsymbol{y} = P^\top \boldsymbol{x}$ where P is an orthonormal matrix such that $P^\top D P = \mathrm{diag}\,[\boldsymbol{\lambda}]$, where $\boldsymbol{\lambda} > \boldsymbol{0}$ is the vector of eigenvalues of D. The problem (NCQP($D, \boldsymbol{c}$)) is reduced to the convex maximization problem with separable quadratic objective function. Let us define

$$\left| \begin{array}{ll} \underset{\boldsymbol{y} \in \mathbb{R}^n}{\text{maximize}} & f(\boldsymbol{y}) := \sum_{j=1}^{n} \lambda_j \, y_j^2 - 2\, \boldsymbol{c}^\top P \,\mathrm{diag}\{\boldsymbol{\lambda}\}\, \boldsymbol{y} + \boldsymbol{c}^\top D \boldsymbol{c} \\ \text{subject to} & AP\boldsymbol{y} \leq \boldsymbol{b}, \end{array} \right.$$

where $\lambda_j > 0$, $\forall j$. Let

$$\begin{array}{rcl} f_j(y_j) & := & \lambda_j \, y_j^2 - 2\,[\boldsymbol{c}^\top P \,\mathrm{diag}[\boldsymbol{\lambda}]]_j \, y_j, \quad j = 1, ..., n, \text{ and} \\ Y & := & \{\ \boldsymbol{y} \in \mathbb{R}^n \mid A\,P\boldsymbol{y} \leq \boldsymbol{b}\ \}, \end{array}$$

and let

$$R_0 := \{\ \boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{L}^0 \leq \boldsymbol{y} \leq \boldsymbol{U}^0\ \}$$

be the smallest hyperrectangle containing Y.
Let us define

$$(\mathrm{P}_0) \left| \begin{array}{ll} \underset{\boldsymbol{y} \in \mathbb{R}^n}{\text{maximize}} & f(\boldsymbol{y}) := \sum_{j=1}^{n} f_j(y_j) \\ \text{subject to} & \boldsymbol{y} \in Y \cap R_0. \end{array} \right.$$

We will apply a branch and bound algorithm successfully applied to a number of portfolio optimization problems [12, 13].

Rectangular Subdivision Branch and Bound Algorithm : Let $\varepsilon > 0$ be a tolerance.

Step 1. (initialization) Solve a linear programming problem :

$$\left| \begin{array}{ll} \underset{\boldsymbol{y} \in \mathbb{R}^n}{\text{maximize}} & g^0(\boldsymbol{y}) := \sum_{j=1}^{n} g_j^0(y_j) \\ \text{subject to} & \boldsymbol{y} \in Y \cap R_0, \end{array} \right.$$

where $g_j^0(y_j) = \frac{f_j(U_j^0) - f_j(L_j^0)}{U_j^0 - L_j^0} y_j + \frac{U_j^0 f_j(L_j^0) - L_j^0 f_j(U_j^0)}{U_j^0 - L_j^0}$, $j = 1, ..., n$, and let $\overline{\boldsymbol{y}}^0$ be its optimal solution and $\beta(R_0)$ be its optimal value. Set $\mathcal{P} \leftarrow \{R_0\}, \alpha_0 \leftarrow f(\overline{\boldsymbol{y}}^0)$, $\boldsymbol{y}^0 \leftarrow \overline{\boldsymbol{y}}^0$, and $k \leftarrow 0$.

Step 2. (pruning) Delete all regions $R \in \mathcal{P}$ such that $\beta(R) \leq \alpha_k + \varepsilon$. If $\mathcal{P} = \phi$, terminate : $\hat{\boldsymbol{x}} := P\boldsymbol{y}^k$ is an ε-optimal solution to (NCQP$(D, \boldsymbol{c})$).

Step 3. (branching) Select a region $R_k \in \mathcal{P}$ such that

$$\beta(R_k) = \max\{\ \beta(R) \mid R \in \mathcal{P}\ \},$$

and let

$$s \leftarrow \arg\max_j \left\{\ g_j(\overline{y}_j^k) - f_j(\overline{y}_j^k)\ \right\}.$$

For s, divide the region R_k into the following two regions :

$$\begin{aligned} R_{k_1} &\leftarrow \left\{ \boldsymbol{y} \;\middle|\; \boldsymbol{y} \in R_k,\ y_s \leq y_s^k \right\} =: \left\{ \boldsymbol{y} \;\middle|\; \boldsymbol{L}^{k_1} \leq \boldsymbol{y} \leq \boldsymbol{U}^{k_1} \right\}, \\ R_{k_2} &\leftarrow \left\{ \boldsymbol{y} \;\middle|\; \boldsymbol{y} \in R_k,\ y_s \geq y_s^k \right\} =: \left\{ \boldsymbol{y} \;\middle|\; \boldsymbol{L}^{k_2} \leq \boldsymbol{y} \leq \boldsymbol{U}^{k_2} \right\}. \end{aligned}$$

Solve two linear programming problems :

$$\left|\begin{array}{ll}\underset{\boldsymbol{y}}{\text{maximize}} & g^{k_1}(\boldsymbol{y}) := \sum_{j=1}^{n} g_j^{k_1}(y_j) \\ \text{subject to} & \boldsymbol{y} \in Y \cap R_{k_1}, \end{array}\right| \left|\begin{array}{ll}\underset{\boldsymbol{y}}{\text{maximize}} & g^{k_2}(\boldsymbol{y}) := \sum_{j=1}^{n} g_j^{k_2}(y_j) \\ \text{subject to} & \boldsymbol{y} \in Y \cap R_{k_2}, \end{array}\right.$$

where $g_j^h(y_j) = \frac{f_j(U_j^h)-f_j(L_j^h)}{U_j^h-L_j^h} y_j + \frac{U_j^h f_j(L_j^h)-L_j^h f_j(U_j^h)}{U_j^h-L_j^h}$, $j = 1, ..., n$, for rectangle $R^h = \{ \boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{L}^h \leq \boldsymbol{y} \leq \boldsymbol{U}^h \}$, $h = k_1, k_2$, and let $\overline{\boldsymbol{y}}_{k_1}$, $\overline{\boldsymbol{y}}_{k_2}$ be their optimal solutions, and set $\beta(R_{k_1}) \leftarrow g^{k_1}(\overline{\boldsymbol{y}}_{k_1})$, $\beta(R_{k_2}) \leftarrow g^{k_2}(\overline{\boldsymbol{y}}_{k_2})$. Also, let

$$\boldsymbol{y}^{k+1} \leftarrow \arg\max\{ f(\boldsymbol{y}^k), f(\overline{\boldsymbol{y}}_{k_1}), f(\overline{\boldsymbol{y}}_{k_2}) \}.$$

Set $\alpha_{k+1} \leftarrow f(\boldsymbol{y}^{k+1})$, $\mathcal{P} \leftarrow (\mathcal{P} \setminus R_k) \cap \{R_{k_1}, R_{k_2}\}$, $k \leftarrow k+1$, and goto **Step 2**.

Theorem 2 *$\boldsymbol{y}^k$ converges to an ε-optimal solution of (P_0) as $k \to \infty$.*

proof. See [15, 20]. ‖

Shortcut Strategy

Maximization of a convex quadratic function is usually very time-consuming. However, we do not have to find an optimal solution of the subproblem at each iteration of the branch and bound algorithm. What we need is an extreme point of X which is not contained in the current ellipsoid. Thus we will modify Algorithm 1 as follows.

Algorithm with Shortcut Strategy (Algorithm 2) : Let $\varepsilon > 0$ be a tolerance.

Step 1. Let $V^0 = \left\{ \boldsymbol{v}^1, ..., \boldsymbol{v}^{k_0} \right\}$ be a set of vertices of X whose affine hull spans $\mathbb{R}^n$, and set $k \leftarrow 1$.

Step 2. Calculate the smallest ellipsoid $E(D^k, \boldsymbol{c}^k)$ covering V^k by solving (MEC(V^k)).

Step 3. Apply the branch and bound algorithm to the problem :

$$(\text{NCQP}(D^k, \boldsymbol{c}^k)) \left|\begin{array}{ll}\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{maximize}} & (\boldsymbol{x}-\boldsymbol{c}^k)^\top D^k (\boldsymbol{x}-\boldsymbol{c}^k) \\ \text{subject to} & \boldsymbol{x} \in X. \end{array}\right.$$

If any point $\boldsymbol{v}^k$ satisfying $(\boldsymbol{v}^k-\boldsymbol{c}^k)^\top D^k (\boldsymbol{v}^k-\boldsymbol{c}^k) > 1+\varepsilon$ is found, then abort the branch and bound procedure and set $V^{k+1} \leftarrow V^k \cup \{\boldsymbol{v}^k\}$, $k \leftarrow k+1$ and go to **Step 2.** Else end with an ε-optimal solution $(D^k,\ \boldsymbol{c}^k)$.

Note that points obtained in Step 3 are not necessarily vertices. In practice, almost all these points are expected to be vertices of X.

In addition to the above enhancing scheme, at each application of Step 3 of Algorithm 1 and 2, the initial incumbent value may be replaced the value of some point before the start of the iteration.

4 Computational Experiments

In this section, we will present numerical results of the Algorithm 2 and compare it with the benchmark algorithm defined below.

Benchmark (Algorithm 3) : Combination of Vertex Enumeration and Minimal Covering Ellipsoid Calculation

Step 1. Enumerate all vertices of X.

Step 2. Calculate a minimal ellipsoid by DRN algorithm which is explained in Appendix.

Among a number of vertex enumeration algorithms, we use here the algorithm cddf+ of Fukuda [5].

The polytope X to be used in the numerical experiment is defined as the intersection of an n-dimensional hypercube $[0,5]^n$ and a number of inequalities supporting a sphere $S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{c}\| \leq 2\}$, where $\boldsymbol{c} = (2.5, ..., 2.5)^\top \in \mathbb{R}^n$. (See Figure 1.)

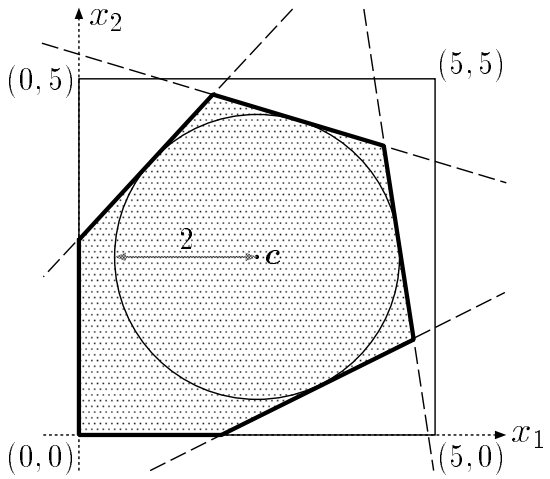

Figure 1 : Illustration of a Polytope Generation Scheme

The polytope generated by this scheme satisfies Assumption 1 and expected to contain a large number of extreme points.

We generated X for m up to 500 and n up to 7. Table 1 shows the average number of vertices of 10 polytopes randomly generated by the scheme above while the numbers in brackets are standard deviation. We see that the number of vertices explodes as n increases.

Table 1 : Average # of Vertices of Generated Problems

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	184.0	547.3	1927.2	6800.8	23344.8
	(0.0)	(12.6)	(32.6)	(115.9)	(352.9)
200	384.0	1165.5	4429.6	16904.1	74096.1
	(0.0)	(9.6)	(69.8)	(3866.8)	(2453.2)
300	584.0	1787.0	6764.6	30429.2	-
	(0.0)	(14.0)	(721.3)	(497.6)	(-)
500	983.6	3080.3	12161.6	54160.7	-
	(1.2)	(12.3)	(101.8)	(4268.9)	(-)

We applied Algorithm 1 and Algorithm 2 using C/C++ on a personal computer with CPU: Pentium 4 processor (2.53GHz), memory: 512MB. We used CPLEX7.1 for solving linear programming problems. DRN algorithm proposed by Sun and Freund [19] is used to calculate a minimal ellipsoid covering a given set of points, and we used CLAPACK for computation of linear algebra in the interior point algorithm. We chose $\varepsilon_{\text{feas}} = \varepsilon_{\text{opt}} = 10^{-7}$ for DRN algorithm and $\varepsilon = 10^{-5}$ for the Algorithm 1 and Algorithm 2.

Tables 2, 3 and 4 show CPU time of Benchmark algorithm, Algorithm 1 and Algorithm 2, respectively, applied to the problems of Table 1.

Table 2 : Average CPU time for Benchmark Algorithm (Algorithm 3) [sec.]

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	0.07	1.29	49.55	5746.93	-
	(0.01)	(0.12)	(5.33)	(958.55)	(-)
200	0.56	10.09	583.26	-	-
	(0.07)	(0.85)	(52.43)	(-)	(-)
300	1.91	36.30	5448.32	-	-
	(0.31)	(2.57)	(1617.17)	(-)	(-)
500	7.89	183.31	-	-	-
	(1.26)	(20.92)	(-)	(-)	(-)

Table 3 : Average CPU time for Basic Algorithm (Algorithm 1) [sec.]

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	0.66	3.82	30.81	244.84	2628.80
	(0.22)	(0.80)	(8.55)	(59.89)	(753.50)
200	1.97	8.65	87.81	812.87	10649.49
	(0.36)	(1.89)	(11.48)	(137.77)	(2884.90)
300	3.73	16.42	133.37	1769.05	18875.53
	(0.91)	(5.54)	(18.94)	(308.64)	(4578.64)
500	10.27	41.38	242.86	2808.67	31309.43
	(2.64)	(13.05)	(40.67)	(563.10)	(5421.97)

Table 4 : Average CPU time for Algorithm with Shortcut Strategy (Algorithm 2) [sec.]

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	0.20	1.01	6.34	46.58	502.99
	(0.05)	(0.17)	(1.00)	(13.01)	(211.93)
200	0.65	2.40	16.94	139.86	1423.48
	(0.16)	(0.39)	(2.89)	(37.87)	(564.78)
300	1.06	4.44	25.43	254.30	3076.94
	(0.25)	(1.41)	(5.82)	(100.50)	(1488.94)
500	2.79	10.76	61.27	501.80	4684.11
	(0.76)	(3.24)	(15.11)	(146.95)	(929.94)

We see that the Benchmark algorithm can solve problem up to $(n, m) = (3, 200)$ very fast. However, when (n, m) is over $(4, 200)$ and $(3, 200)$ it is much slower than Algorithm 1 and 2. This is due to the reason that DRN algorithm becomes less efficient when the number of vertices is large. Also, we see that the Algorithm 2 is superior to Algorithm 1.

Table 5 shows the CPU time for enumerating vertices in Benchmark algorithm.

Table 5 : Average CPU time for Vertex Enumeration of X [sec.]

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	0.01	0.03	0.19	3.74	60.47
	(0.00)	(0.00)	(0.01)	(0.22)	(2.87)
200	0.03	0.14	1.83	48.46	962.61
	(0.00)	(0.01)	(0.09)	(15.45)	(72.87)
300	0.06	0.32	7.43	197.49	-
	(0.00)	(0.01)	(1.67)	(22.91)	(-)
500	0.16	1.26	39.14	804.40	-
	(0.00)	(0.05)	(0.57)	(136.49)	(-)

We see that the vertex enumeration algorithm cddf+ is very efficient. In fact, it shares only a fraction of the total computation time.

Finally, Tables 6 and 7 show the number of iterations of Algorithm 1 and Algorithm 2.

Table 6 : Average # of Iterations of Basic Algorithm (Algorithm 1)

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	8.70	14.70	25.10	35.30	45.90
	(2.15)	(1.62)	(3.78)	(4.38)	(4.21)
200	9.80	13.40	26.00	39.60	51.70
	(1.17)	(1.96)	(1.79)	(2.62)	(5.42)
300	10.60	14.00	24.70	42.80	53.70
	(1.50)	(3.16)	(1.90)	(4.87)	(2.97)
500	11.30	15.10	24.60	40.60	54.80
	(1.55)	(3.53)	(2.24)	(4.18)	(3.31)

Table 7 : Average # of Iterations of Algorithm with Shortcut Strategy (Algorithm 2)

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	11.90	33.30	66.60	111.30	171.70
	(3.67)	(6.56)	(9.76)	(10.46)	(17.38)
200	17.20	29.60	74.70	137.80	222.50
	(2.79)	(5.48)	(8.21)	(15.26)	(16.26)
300	19.70	28.00	73.00	157.50	229.70
	(5.39)	(9.02)	(7.78)	(12.85)	(18.83)
500	25.70	35.70	65.50	151.00	259.00
	(5.97)	(8.98)	(6.77)	(17.82)	(18.91)

We see that the number of iterations of Algorithm 2 is twice or three times more than those of Algorithm 1. However, the total computation time is 5 to 10 times less than Algorithm 1. This proves our conjecture that terminating the branch and bound algorithm as soon as we obtain a new point not contained in the current ellipsoid enhances the overall efficiency of the algorithm, as expected.

Tables 8 and 9 show the average number of points in polytope X generated until ε-optimality is attained. Almost all of these points are thought to be vertices of X. (Accidentally, the value of each cell is equal to the number of iterations shown in Table 6 and 7, respectively, plus $2n$ which is the number of problems solved for obtaining the first rectangle at each branch and bound algorithm.) The last row of each table shows the F. John number, i.e., $n(n+3)/2$, mentioned in the Introduction, indicating that only a fraction of vertices of a polytope are to be required to determine ellipsoids. From these two tables and Table 1, we see that Algorithm 1 and 2 succeeded in identifying a set of vertices which determines minimal ellipsoid.

Table 8 : Average # of Points Gathered until Optimality by Basic Algorithm (Algorithm 1)

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	14.70	22.70	35.10	47.30	59.90
	(2.15)	(1.62)	(3.78)	(4.38)	(4.21)
200	15.80	21.40	36.00	51.60	65.70
	(1.17)	(1.96)	(1.79)	(2.62)	(5.42)
300	16.60	22.00	34.70	54.80	67.70
	(1.50)	(3.16)	(1.90)	(4.87)	(2.97)
500	17.30	23.10	34.60	52.60	68.80
	(1.55)	(3.53)	(2.24)	(4.18)	(3.31)
F.J.#	9	14	20	27	35

Table 9 : Average # of Points Gathered until Optimality with Shortcut Strategy (Algorithm 2)

# of linear inequalities m	n : # of dimensions 3	4	5	6	7
100	17.90 (3.67)	41.30 (6.56)	76.60 (9.76)	123.30 (10.46)	185.70 (17.38)
200	23.20 (2.79)	37.60 (5.48)	84.70 (8.21)	149.80 (15.26)	236.50 (16.26)
300	25.70 (5.39)	36.00 (9.02)	83.00 (7.78)	169.50 (12.85)	243.70 (18.83)
500	31.70 (5.97)	43.70 (8.98)	75.50 (6.77)	163.00 (17.82)	273.00 (18.91)
F.J.#	9	14	20	27	35

Finally, let us mention the effect of the technique explained in the end of Section 3 for improving the first incumbent value of each branch and bound phase. The incumbent value was improved at 70 to 90% iterations of the experiments, and we observe that its percentage of all increases as n and m become larger. For example, by Algorithm 2, 70.6% iterations were improved when $(n, m) = (3, 300)$, while 92.0% when $(n, m) = (7, 500)$.

5 Concluding Remarks

We proposed an algorithm for finding a minimal volume ellipsoid containing a polytope defined by a linear system of inequalities. When the dimension of the polytope and the number of inequalities are small, then we can generate a minimal ellipsoid by first enumerating the extreme points and then apply existent algorithm such as the one proposed by Sun and Freund [19]. However, when the dimension n is over 4, this method tends to become less efficient.

On the other hand, the branch and bound algorithm proposed in this paper can solve problems of dimension up to 7 and the number of inequalities m up to 500 within a practical amount of time. This algorithm is based on the observation that

(i) minimal ellipsoid is determined by at most $n(n+3)/2$ points

(ii) convex quadratic function can be maximized over a polytope by a branch and bound algorithm if the dimension n is less than 10.

Numerical experiments presented in this paper support the validity of these observations. In fact, we can now solve a problem up to $(n, m) = (7, 500)$ within a practical amount of time.

When the number of vertices is not very large, algorithms based on vertex enumeration may be better than the branch-and-bound algorithm. In fact, the polytopes in our experiments tend to have much more vertices than those generated randomly. So, it is fair to say that our algorithms will have advantage when polytope has very large number of vertices.

Computation time can be reduced to about one fifth if we implement the algorithm in a more elaborate way. Unfortunately, it would be very difficult to solve problem when n is over 10 in a deterministic and exact way as observed in a number of computational studies on branch and bound algorithm for global optimization problem with low rank nonconvex structures [10].

Acknowledgement Research of the first author is supported by MEXT Grant-in-Aid for Young Scientists (B) 14780343. Research of the second author is partly supported by MEXT Grant-in-Aid for Scientific Research B(2) 15310122.

References

[1] Barnes, E.R. (1982). "An Algorithm for Separating Patterns by Ellipsoids", *IBM J. Res. Develop.*, vol.26, No.6, pp.759–764.

[2] Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on Modern Convex Optimization – Analysis, Algorithms, and Engineering Applications–*, SIAM.

[3] Dyer, M.E. (1986). "On a Multidimensional Search Technique and its Applications to the Euclidean One-Center Problem", *SIAM J. Computing*, 15, pp.725–738.

[4] Elzinga, J. and Hearn, D.W. (1972). "The Minimal Covering Sphere Problem", *Management Science*, 19, pp.96–104.

[5] Fukuda, K. (1995). "cdd/cdd+ Reference Manual", Institute for Operations Research, Swiss Federal Institute of Technology, Zurich, Switzerland. program available from `http://www.ifor.math.ethz.ch/ fukuda/fukuda.html`

[6] Falk, J. and Soland, R.M. (1969). "An Algorithm for Separable Nonconvex Programming Problems", *Management Science*, 15, pp.550–569.

[7] Gärtner, B. and Schönherr, S. (1997). "Smallest Enclosing Ellipses –Fast and Exact–", *Proc. 13. Annual ACM Symposium on Computational Geometry*, pp.430–432.

[8] John, F. (1948). "Extremum Problems with Inequalities as Subsidiary Conditions", In *Studies and Essays Presented to R.Courant on His Birthday*, pp.187–204, Interscience Publishers, NY.

[9] Khachiyan, L. and Todd, M. (1993). "On the Complexity of Approximating the Maximal Inscribed Ellipsoid for a Polytope", *Mathematical Programming*, 61, pp.137–159.

[10] Konno, H, Thach, P.T. and Tuy, H. (1997). *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers.

[11] Konno, H, Yajima, Y. and Ban, A. (1994). "Calculating a Minimal Sphere Containing a Polytope Defined by a System of Linear Inequalities", *Computational Optimization and Applications*, 3, pp.181–191.

[12] Konno, H, and Wijayanayake, A. (1999). "Mean-Absolute Deviation Portfolio Optimization Model under Transaction Costs", *J. of the Operations Research Society of Japan*, 42, pp.422–435.

[13] Konno, H, and Wijayanayake, A. (2001). "Portfolio Optimization Problem under Concave Transaction Costs and Minimal Transaction Unit Constraints", *Mathematical Programming*, B89, pp.233–250.

[14] Nair, K.P.K. and Chandrasekaran, R. (1971). "Optimal Location of Single-serve Center of Certain Types", *Naval Research Logistics Quarterly*, 18, pp.503–510.

[15] Phong, T.Q., An, L.T.H and Tao, P.D. (1995). "On Globally Solving Linearly Constrained Indefinite Quadratic Minimization Problems by Decomposition Branch and Bound Method", *Operations Research Letters*, 17, pp.215–220.

[16] Rockafellar, R.T. and Wets, R.J-B. (1998). *Variational Analysis*, Springer-Verlag.

[17] Sekitani, K. and Yamamoto, Y. (1993). "A Recursive Algorithm for Finding the Minimal Covering Sphere of a Polytope and the Minimal Covering Spheres of Several Polytopes", *Japan J. of Industrial and Applied Mathematics*, 10, pp.255–273.

[18] Skyum, S. (1991). "A Simple Algorithm for Computing the Smallest Enclosing Circle", *Information Processing Letters*, 37, pp.121–125.

[19] Sun, P. & Freund, R.M. (2002). "Computation of Minimum Volume Covering Ellipsoids", MIT Operations Research Center Working paper OR 064-02, submitted to *Operations Research.*(revised Jun.6,2003)

[20] Tuy, H. (1998). *Convex Analysis and Global Optimization*, Kluwer Academic Publishers.

[21] Welzl, E. (1991). "Smallest Enclosing Disks (Balls and Ellipsoids)", In H.Maurer, editor, *New Results and New Trends in Computer Science*, vol.555 of Lecture Notes in Computer Science, pp.359–370, Springer-Verlag.

[22] Zhang, Y. (1998). "An Interior-Point Algorithm for the Maximum-Volume Ellipsoid Problem", Department of Computational and Applied Mathematics, Rice University, Technical Report TR98-15.

[23] Zhang, Y. and Gao, L. (2003). "On Numerical Solution of the Maximum Volume Ellipsoid Problem", *SIAM Journal of Optimization*, 14, 1, pp.53–76.

Appendix : DRN Algorithm by Sun and Freund [19]

In this Section, we will summarize the algorithm of Sun and Freund [19] for solving the problem (MEC(V)) when the V is given by a finite set of points $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_l\}$.

This algorithm based on interior point approach is perhaps the most efficient deterministic algorithms among others.

We will impose the following standard assumption.

Assumption 2 : The affine hull of $\boldsymbol{v}_1, ..., \ \boldsymbol{v}_l \in \mathbb{R}^n$ spans $\mathbb{R}^n$.

By defining $M := D^{\frac{1}{2}}$ and $\boldsymbol{z} := D^{\frac{1}{2}}\boldsymbol{c}$, the problem (MEC($V$)) can be represented as follows :

$$(\mathrm{MEC2}(V)) \left| \begin{array}{ll} \underset{M,\ \boldsymbol{z}}{\text{minimize}} & -\ln\det M \\ \text{subject to} & (M\boldsymbol{v}_j - \boldsymbol{z})^\top (M\boldsymbol{v}_j - \boldsymbol{z}) \le 1, \quad j = 1, ..., l, \\ & M \succ O, \quad \boldsymbol{z} \in \mathbb{R}^n. \end{array} \right.$$

The objective function is convex (see e.g. [16]) and the feasible set is a quadratic cone, so that (MEC2(V)) is a convex minimization problem. It is straightforward to see that an optimal solution $(D^*, \boldsymbol{c}^*)$ can be recovered by an optimal solution $(M^*, \boldsymbol{z}^*)$ by using the relation $D^* := M^{*2}, \ \boldsymbol{c}^* := (D^*)^{-\frac{1}{2}}\boldsymbol{z}^*$.

Let us introduce a log barrier function and consider the following parametrized problem :

$$(\mathrm{MEC2(V};\theta)) \left| \begin{array}{ll} \underset{M,\ \boldsymbol{z},\ \boldsymbol{t}}{\text{minimize}} & -\ln\det M - \theta \sum_{j=1}^{l} \ln t_j \\ \text{subject to} & (M\boldsymbol{v}_j - \boldsymbol{z})^\top (M\boldsymbol{v}_j - \boldsymbol{z}) + t_j = 1, \quad j = 1, ..., l, \\ & M \succ O, \quad \boldsymbol{t} > \boldsymbol{0}, \quad \boldsymbol{z} \in \mathbb{R}^n. \end{array} \right.$$

For fixed $\theta > 0$, an optimality condition of this problem is given by

$$(\mathrm{Opt}(\theta)) \left\{ \begin{array}{rcl} \sum_{j=1}^{l} u_j \left[(M\boldsymbol{v}_j - \boldsymbol{z})^\top \boldsymbol{v}_j + \boldsymbol{v}_j (M\boldsymbol{v}_j - \boldsymbol{z})^\top \right] - M^{-1} &=& \boldsymbol{0}, \\ \sum_{j=1}^{l} u_j (\boldsymbol{z} - M\boldsymbol{v}_j)^\top &=& \boldsymbol{0}, \\ (M\boldsymbol{v}_j - \boldsymbol{z})^\top (M\boldsymbol{v}_j - \boldsymbol{z}) + t_j &=& 1, \quad j = 1, ..., l, \\ \mathrm{diag}[\boldsymbol{u}]\, \boldsymbol{t} &=& \theta \boldsymbol{1}, \\ M \succ O, \quad \boldsymbol{t} \ge \boldsymbol{0}, \quad \boldsymbol{u} \ge \boldsymbol{0}, & & \end{array} \right.$$

where $\boldsymbol{1} = (1, 1, ..., 1)^\top$. Noting $M = \left[2 \left(B\, \mathrm{diag}[\boldsymbol{u}]\, B^\top - \frac{B\boldsymbol{u}\boldsymbol{u}^\top B^\top}{\boldsymbol{1}^\top \boldsymbol{u}} \right) \right]^{-\frac{1}{2}} =: M(\boldsymbol{u})$ and $\boldsymbol{z} = \frac{M(\boldsymbol{u})B\boldsymbol{u}}{\boldsymbol{1}^\top \boldsymbol{u}} =: \boldsymbol{z}(\boldsymbol{u})$ are the unique solution of the first two equations of (Opt(θ)) for

$B := [\boldsymbol{v}_1|\boldsymbol{v}_2|\cdots|\boldsymbol{v}_l]$ if Assumption 2 and $\boldsymbol{u} > \boldsymbol{0}$ are satisfied, the condition can be reduced to a compact form :

$$(\mathrm{OptR}(\theta)) \left\{ \begin{array}{rcl} \boldsymbol{h}(\boldsymbol{u}) + \boldsymbol{t} & = & \boldsymbol{1}, \\ \mathrm{diag}[\boldsymbol{u}]\,\boldsymbol{t} & = & \theta\boldsymbol{1}, \\ \boldsymbol{u},\ \boldsymbol{t} & \geq & \boldsymbol{0}, \end{array} \right.$$

where $\boldsymbol{h}(\boldsymbol{u}) = (h_1(\boldsymbol{u}), ..., h_l(\boldsymbol{u}))^\top$ and

$$h_j(\boldsymbol{u}) \ := \ (M(\boldsymbol{u})\,\boldsymbol{v}_j - \boldsymbol{z}(\boldsymbol{u}))^\top (M(\boldsymbol{u})\,\boldsymbol{v}_j - \boldsymbol{z}(\boldsymbol{u}))\,, \quad j = 1, ..., l.$$

The Newton equations corresponding to $(\mathrm{OptR}(\theta))$ can be written as follows :

$$(\mathrm{drc}(\boldsymbol{u}, \boldsymbol{t}, \theta)) \left\{ \begin{array}{rcccccl} \nabla_{\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{u})\,\Delta\boldsymbol{u} & + & \Delta\boldsymbol{t} & = & \boldsymbol{r}_1 & := & \boldsymbol{1} - \boldsymbol{t} - \boldsymbol{h}(\boldsymbol{u}) \\ \mathrm{diag}[\boldsymbol{t}]\,\Delta\boldsymbol{u} & + & \mathrm{diag}[\boldsymbol{u}]\Delta\boldsymbol{t} & = & \boldsymbol{r}_2 & := & \theta\boldsymbol{1} - \mathrm{diag}[\boldsymbol{u}]\boldsymbol{t} \end{array} \right.$$

where $\nabla_{\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{u})$ is defined by $[\nabla_{\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{u})]_{ij} = -2\left(\frac{[W(\boldsymbol{u})]_{ij}}{\boldsymbol{1}^\top\boldsymbol{u}} + [W(\boldsymbol{u})]^2_{ij}\right),\ i, j = 1, ..., l$, denoting

$$W(\boldsymbol{u}) := \left(B - \frac{B\boldsymbol{u}}{\boldsymbol{1}^\top\boldsymbol{u}}\right)^\top M^2(\boldsymbol{u}) \left(B - \frac{B\boldsymbol{u}}{\boldsymbol{1}^\top\boldsymbol{u}}\right).$$

Under Assumption 2, $\nabla_{\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{u}) - \mathrm{diag}[\boldsymbol{u}]^{-1}\,\mathrm{diag}[\boldsymbol{t}] \prec O$ if $\boldsymbol{u} > \boldsymbol{0}$ and $\boldsymbol{t} > \boldsymbol{0}$, and therefore, the system $(\mathrm{drc}(\boldsymbol{u}, \boldsymbol{t}, \theta))$ can be solved (See [19]) :

$$\left\{ \begin{array}{rcl} \Delta\boldsymbol{u} & = & (\nabla_{\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{u}) - \mathrm{diag}[\boldsymbol{u}]^{-1}\,\mathrm{diag}[\boldsymbol{t}])^{-1}\,(\boldsymbol{r}_1 - \mathrm{diag}[\boldsymbol{u}]^{-1}\boldsymbol{r}_2)\,, \\ \Delta\boldsymbol{t} & = & \mathrm{diag}[\boldsymbol{u}]^{-1}\boldsymbol{r}_2 - \mathrm{diag}[\boldsymbol{u}]^{-1}\,\mathrm{diag}[\boldsymbol{t}]\,\Delta\boldsymbol{u}. \end{array} \right.$$

DRN Algorithm : Let $\varepsilon_{\mathrm{feas}}$, $\varepsilon_{\mathrm{opt}} > 0$, and let $\boldsymbol{u}^0, \boldsymbol{t}^0 > \boldsymbol{0}$ and $\theta > 0$ be given as inputs.

Step 0: (Initialization) $r \leftarrow 0.99$. $(\boldsymbol{u}, \boldsymbol{t}) \leftarrow (\boldsymbol{u}^0, \boldsymbol{t}^0) > \boldsymbol{0}$

Step 1: (Stopping Criteria) If $\|\boldsymbol{1} - \boldsymbol{t} - \boldsymbol{h}(\boldsymbol{u})\| \leq \varepsilon_{\mathrm{feas}}$ and $\boldsymbol{u}^\top\boldsymbol{t} \leq \varepsilon_{\mathrm{opt}}$, STOP with $\boldsymbol{u},\ D := [M(\boldsymbol{u})]^2,\ \boldsymbol{c} := [M(\boldsymbol{u})]^{-1}\boldsymbol{z}(\boldsymbol{u})$.

Step 2: (Direction) $\theta \leftarrow \frac{\boldsymbol{u}^\top\boldsymbol{t}}{10l}$. Compute $(\Delta\boldsymbol{u}, \Delta\boldsymbol{t})$ by solving the system $(\mathrm{drc}(\boldsymbol{u}, \boldsymbol{t}, \theta))$

Step 3: (Step Size) Compute $\overline{\beta} \leftarrow \max\{\,\beta \,|\, (\boldsymbol{u}, \boldsymbol{t}) + \beta(\Delta\boldsymbol{u}, \Delta\boldsymbol{t}) \geq \boldsymbol{0}\}$ and $\tilde{\beta} \leftarrow \min\{r\overline{\beta}, 1\}$. Set $(\boldsymbol{u}, \boldsymbol{t}) \leftarrow (\boldsymbol{u}, \boldsymbol{t}) + \tilde{\beta}(\Delta\boldsymbol{u}, \Delta\boldsymbol{t})$. Go to **Step 1.**

Remark : For initialization of DRN algorithm, we apply a step proposed by Sun and Freund which guarantees strict positivity of $(\boldsymbol{u}^0, \boldsymbol{t}^0)$ as well as initial feasibility of the equations $\boldsymbol{h}(\boldsymbol{u}) + \boldsymbol{t} = \boldsymbol{1}$ at $(\boldsymbol{u}, \boldsymbol{t}) = (\boldsymbol{u}^0, \boldsymbol{t}^0)$.

Initialization of $(\boldsymbol{u}^0, \boldsymbol{t}^0)$ for DRN Algorithm

i) Set $\boldsymbol{u}^0 := \frac{n}{2l}\boldsymbol{1}$

ii) Check if $\boldsymbol{h}(\boldsymbol{u}^0) \leq 0.95 \cdot \boldsymbol{1}$. If so, set $\boldsymbol{t}^0 := \boldsymbol{1} - \boldsymbol{h}(\boldsymbol{u}^0)$ $(> \boldsymbol{0})$. If $\boldsymbol{h}(\boldsymbol{u}^0) \not\leq 0.95 \cdot \boldsymbol{1}$, rescale $\boldsymbol{u}^0$ as follows :
$\alpha := \max\left\{\ h_1(\boldsymbol{u}^0), ..., h_l(\boldsymbol{u}^0)\ \right\}/0.95,\ \boldsymbol{u}^0 \leftarrow \alpha\boldsymbol{u}^0,\ \boldsymbol{t}^0 \leftarrow \boldsymbol{1} - \boldsymbol{h}(\boldsymbol{u}^0)$