Journal of Global Optimization (2006) 34: 1–14
DOI 10.1007/s10898-005-3883-8

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

JUN-YA GOTOH[1] and HIROSHI KONNO[2]
[1]*Graduate School of Systems and Information Engineering, 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)*

(Received 25 November 2003; accepted in revised form 1 March 2005)

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.

Key words: Computational geometry, Global optimization, Minimal ellipsoid

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 [17], Sekitani and Yamamoto [16] 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 [18], Zhang [21], Zhang and Gao [22] using interior point algorithms. These can generate a minimal ellipsoid in an efficient way when the cardinality of the point set is relatively small. Also, there are stochastic algorithms by Welzl [20], 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 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 Section 2, 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.

2. Minimal Ellipsoid Circumscribing a Polytope

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

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

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 proportional to $(\det D)^{-1/2}$.

Let us consider a polytope

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

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

ASSUMPTION 1. X is nonempty, bounded with an interior.

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

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

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

The problem (MECAP) can be represented as follows:

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

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:

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

where V_X is the set of vertices of X.

A minimal ellipsoid containing X can be obtained by existent algorithms, e.g., [1,18,20], if we know all extreme points $\boldsymbol{v}_j,\ j = 1, \ldots, l'$. However, l' can be very large even for small n, so that it may not be easy to solve $(\text{MEC}(V_X))$ by these algorithms.

The algorithm to be proposed below is based upon a sequence of concave minimization subproblems applied to a branch and bound algorithm [6,14].

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

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

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 [18].

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

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

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 $V^0 = \{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^{k_0}\}$ 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 $(\boldsymbol{v}^k - \boldsymbol{c}^k)^\top D^k (\boldsymbol{v}^k - \boldsymbol{c}^k) \leqslant 1 + \varepsilon$, 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 [18] which is considered to be one of the most efficient deterministic algorithms.

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 [14].

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

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

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

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

and let

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

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 $g_j^0(y_j) = \frac{f_j(U_j^0) - f_j(L_j^0)}{U_j^0 - L_j^0} y_j + \frac{U_j^0 f_j(L_j^0) - L_j^0 f_j(U_j^0)}{U_j^0 - L_j^0}$, $j = 1, \dots, n$, 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 $\beta(R) \leqslant \alpha_k + \varepsilon$. 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:

$$\begin{aligned} R_{k_1} &\leftarrow \left\{ \boldsymbol{y} \mid \boldsymbol{y} \in R_k,\ y_s \leqslant y_s^k \right\} =: \left\{ \boldsymbol{y} \mid \boldsymbol{L}^{k_1} \leqslant \boldsymbol{y} \leqslant \boldsymbol{U}^{k_1} \right\}, \\ R_{k_2} &\leftarrow \left\{ \boldsymbol{y} \mid \boldsymbol{y} \in R_k,\ y_s \geqslant y_s^k \right\} =: \left\{ \boldsymbol{y} \mid \boldsymbol{L}^{k_2} \leqslant \boldsymbol{y} \leqslant \boldsymbol{U}^{k_2} \right\}. \end{aligned}$$

Solve two linear programming problems

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

where $g_j^h(y_j) = \frac{f_j(U_j^h)-f_j(L_j^h)}{U_j^h-L_j^h} y_j + \frac{U_j^h f_j(L_j^h)-L_j^h f_j(U_j^h)}{U_j^h-L_j^h}$, $j=1,\ldots,n$, for rectangle $R^h = \{ \boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{L}^h \leqslant \boldsymbol{y} \leqslant \boldsymbol{U}^h \}$, $h=k_1,k_2$, 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

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

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

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

Proof. See [14, 19]. □

3.1. 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 $V^0 = \{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^{k_0}\}$ 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)).

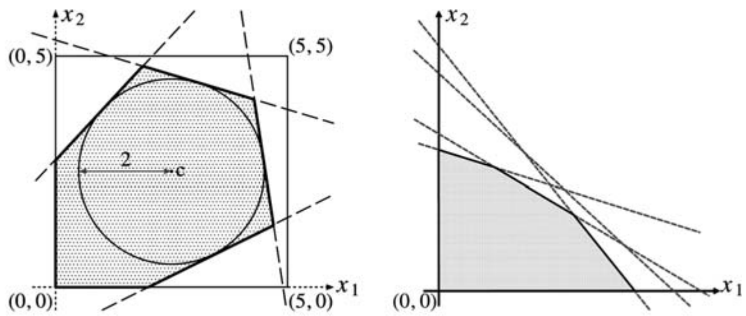
Step 3. Apply the branch and bound algorithm to the problem

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

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 Algorithms 1 and 2, the initial incumbent value of the branch and bound procedure may be replaced by the larger value of some point obtained by then.



(a) Polytope Circumscribing a Sphere (b) Multi-Knapsack type Polytope

Figure 1. Illustration of polytope generation schemes.

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 developed in [18].

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

We will test the algorithms using two different types of polytopes. The first type of polytope X_1 is defined as the intersection of an n-dimensional hypercube $[0, 5]^n$ and $m-2n$ linear inequalities supporting a sphere $S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{c}\| \leqslant 2\}$, where $\boldsymbol{c} = (2.5, \ldots, 2.5)^\top \in \mathbb{R}^n$ (see Figure 1(a)). The second type of polytope X_2 is the multi-knapsack type polytope: $X_2 = \{\boldsymbol{x} \in \mathbb{R}^n | M\boldsymbol{x} \leqslant \mathbf{1}, \boldsymbol{x} \geqslant \mathbf{0}\}$, where $M \in \mathbb{R}^{(m-n)\times n}$ is a matirix whose components are in $(0, 10)$ (see Figure 1(b)).

Intuitively, X_1 is expected to contain much more vertices than X_2 when m and n are fixed. Moreover, vertices of X_2 are expected to be scattered uniformly around the sphere, and consequently, many vertices are expected to support the optimal ellipsoid. On the other hand, X_2 is expected to have less vertices, so that the associated problem is easier.

Polytopes generated by these schemes satisfy Assumption 1 and expected to contain a large number of extreme points. We generated X_1 for m up to 500 and n up to 7 while X_2 for $n = 7$ only.

Table 1. Average number of vertices of generated problems for X_1

Number of linear inequalities (m)	n: Number 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)	(–)

Table 2. Average CPU time of Algorithm 3 for X_1 (sec)

Number of linear inequalities (m)	n: Number 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)	(–)	(–)	(–)

Results for Polytope X_1. Table 1 shows the average number of vertices of 10 polytopes X_1 which are 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.

We applied Algorithms 1 and 2 using C/C++ on a personal computer with CPU: Pentium 4 processor (2.53 GHz), memory: 512 MB. We used CPLEX7.1 for solving linear programming problems. The DRN algorithm proposed by Sun and Freund [18] 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 Algorithms 1 and 2.

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

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 Algorithms 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 3. Average CPU time of Algorithm 1 for X_1 (sec)

Number of linear inequalities (m)	n: Number 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 of Algorithm 2 for X_1 (sec)

Number of linear inequalities (m)	n: Number 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)

Table 5 shows the CPU time for enumerating vertices in Benchmark algorithm for X_1.

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 Algorithms 1 and 2 for X_1.

Table 5. Average CPU time for vertex enumeration of X_1 (sec)

Number of linear inequalities (m)	n: Number 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)	(–)

Table 6. Average number of Iterations of Algorithm 1 for X_1

Number of linear inequalities (m)	n: Number 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 Number of Iterations of Algorithm 2 for X_1

Number of linear inequalities (m)	n: Number 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 5–10 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_1 generated until ε-optimality is attained. Almost all of these points are thought to be vertices of X_1. (Accidentally, the value of each cell is equal to the number of iterations shown in Tables 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 John number, i.e., $n(n+3)/2$, mentioned in Section 1, 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 Algorithms 1 and 2 succeeded in identifying a set of vertices which determines minimal ellipsoid.

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

Table 8. Average number of points gathered until optimality by Algorithm 1 for X_1

Number of linear inequalities (m)	n: Number 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. number	9	14	20	27	35

Table 9. Average number of points gathered until optimality with algorithm 2 for X_1

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

bound phase. The incumbent value was improved at 70–90% 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 $(n, m) = (7, 500)$.

Results for Polytope X_1. Tables 10–12 summarize the numerical results for polytopes X_2 using Algorithm 2. Each table shows the average and the standard deviation of 10 randomly generated instances.

Table 10 shows the average number of vertices of generated polytopes X_2. Compared with Table 1, the average number of vertices is about one tenth of that of X_1's when the number m of constraints is the same. This fact supports our expectation.

Tables 11 and 12 show the average CPU time and the number of iterations, respectively, of Algorithm 2 for solving the instances of X_2, which are comparable to Tables 4 and 7 for X_1. We see from these tables that both the elapsed times and the number of iterations are remarkably smaller compared with X_1. This implies that the algorithm succeeded to gather a set of

Table 10. Average number of Vertices of Polytope X_2

	m: Number of constraints			
	100	200	300	500
av.	2445.6	4107.2	6030.2	8697.0
s.d.	(747.3)	(1213.8)	(1629.2)	(2338.7)

Table 11. Average CPU time with Algorithm2 (for X_2)

	m: Number of constraints			
	100	200	300	500
av.	0.95	1.48	2.05	3.38
s.d.	(0.45)	(0.59)	(0.27)	(1.17)

Table 12. Average number of itrations with Algorithm 2 (for X_2)

	m: Number of constraints			
	100	200	300	500
av.	2.1	1.4	1.3	1.3
s.d.	(1.87)	(1.20)	(0.64)	(0.90)

vertices which are critical to support the minimal ellipsoid at the first stage. In addition, we guess that the distribution of vertices affected the bounding step of the branch and bound algorithm.

From this simple comparison, we expect that the algorithm proposed in this paper performs better for polytopes with a special strucutre.

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 extreme points and then apply existent algorithm such as the one proposed by Sun and Freund [18]. However, when the size of problem is larger, this method tends to become less efficient for those polytopes with a huge number of extreme points such as X_1.

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 X_1 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].

Acknowledgements

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 Journal of Research Development* 26(6), 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 Journal of Computing*, 15 725–738.
4. Elzinga, J. and Hearn, D.W. (1972), The minimal covering sphere problem, *Management Science*, 19, 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, 550–569.
7. Gärtner, B. and Schönherr, S. (1997), Smallest enclosing ellipses –fast and exact–, *Proceedings 13. Annual ACM Symposium on Computational Geometry* 430–432.
8. John, F. (1948), Extremum problems with inequalities as subsidiary conditions, in: *Studiesf and Essays Presented to R.Courant on His Birthday*, Interscience Publishers, NY, pp. 187–204.

9. Khachiyan, L. and Todd, M. (1993), On the complexity of approximating the maximal inscribed ellipsoid for a polytope, *Mathematical Programming* 61, 137–159.
10. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on low rank nonconvex structures*, Kluwer Academic Publishers, Dordrecht.
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, 181–191.
12. Konno, H. and Wijayanayake, A. (1999), Mean-absolute deviation portfolio optimization model under transaction costs, *Journal of the Operations Research Society of Japan* 42, 422–435.
13. Konno, H. and Wijayanayake, A. (2001), Portfolio optimization problem under concave transaction costs and minimal transaction unit constraints, *Mathematical Programming* B89, 233–250.
14. 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, 215–220.
15. Rockafellar, R.T. and Wets, R.J-B. (1998), *Variational Analysis*, Springer-Verlag Berlin.
16. 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 Journal of Industrial and Applied Mathematics* 10, 255–273.
17. Skyum, S. (1991), A simple algorithm for computing the smallest enclosing circle, *Information Processing Letters* 37, 121–125.
18. Sun, P. and Freund, R.M. (2004), Computation of minimum volume covering Ellipsoids, *Operations Research* 52, 690–706.
19. Tuy, H. (1998), *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht.
20. Welzl, E. (1991), Smallest enclosing disks (Balls and Ellipsoids), in: Maurer, H. (ed.), *New Results and New Trends in Computer Science*, of Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 555. pp. 359–370.
21. 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.
22. Zhang, Y. and Gao, L. (2003), On numerical solution of the maximum volume ellipsoid problem, *SIAM Journal of Optimization* 14(1), 53–76.