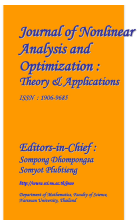
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 3, No. 1, (2012), 1-12
ISSN : 1906-9685
http://www.math.sci.nu.ac.th



AN OUTER APPROXIMATION METHOD UTILIZING THE RELATION OF CONNECTIONS AMONG VERTICES FOR SOLVING A DC PROGRAMMING PROBLEM

SYUUJI YAMADA[1,*], YUSUKE TOKUSHIGE[2], TAMAKI TANAKA[3] AND TETSUZO TANINO[4]

[1,2,3]Graduate School of Science and Technology, Niigata University, 8050 Ikarashi-2nocho, Niigata-City 950-2181, Japan

[4] Graduate School of Engineering, Osaka University, Yamada-Oka 2-1, Suita, Osaka, 565-0871, Japan

ABSTRACT. In this paper, we improve the outer approximation method proposed by Tuy [10] for solving a dc programming problem. The improved algorithm has the global convergence by generating a sequence of polytopes approximating a compact convex set from outside. Moreover, by incorporating a procedure for calculating the vertex sets utilizing the relations of connections among vertices by edges, the improved algorithm can calculate an approximate solution effectively.

KEYWORDS : Global optimization; Dc programming problem; Outer approximation method.

1. INTRODUCTION

Dc programming is one of the important subjects in global optimization and has been studied by Avriel and Williams [1], Hillestad and Jacobsen [2], Meyer [3], Rosen [5], Tuy [7] and Ueing [11]. It is known that many global optimization problems can be transformed into dc programming problems. In particular, Avriel and Williams [1] and Zaleesky [12] have shown that dc programming problems often occur in certain engineering design and economic management applications. Moreover, iterative solution methods for solving dc programming problems have been proposed by Thoai[6], Tuy [9, 10] and many other researchers. Many algorithm of them all are based on outer approximation methods. Outer approximation is one of the powerful procedures in global optimization and can solve various global optimization problems. The algorithms have the global convergence by generating a sequence of convex polyhedral sets. Therefore, it is shown that every accumulation

[1] Supported by the Commission on Higher Education and the Thailand Research Fund (Project No. MRG5380247).
[2] Supported by the Centre of Excellence in Mathematics, the Commission on Higher Education, Thailand.
[*] *Corresponding author.*
Email address : yamada@math.sc.niigata-u.ac.jp(S. Yamada), tamaki@math.sc.niigata-u.ac.jp(T. Tanaka) and tanino@eei.eng.osaka-u.ac.jp(T. Tanino).
Article history : Received 24 December 2011. Accepted 27 December 2011.

point of the sequence of the provisional solutions generated by the algorithm is a globally optimal solution.

In this paper, we consider a problem (DC) to minimize a dc (difference of two convex) function over a compact convex set. To calculate an approximate solution of (DC) effectively, we improve the outer approximation method proposed by Tuy [10]. The proposed algorithm has the global convergence by generating a sequence of polytopes approximating the intersection of a closed half space and the epigraph of one convex function constructing the objective function of (DC) from outside. Moreover, we propose a procedure for calculating all verteices of polytopes by utilizing the relation of connections among vertices by edges. By incorporating the proposed procedure for calculating the vertex sets into the improved outer approximation method, the efficiency of the algorithm is upgraded.

The organization of this paper is as follows: In Section 2, we consider a dc programming problem. In Section 3, we explain the outer approximation algorithm proposed by Tuy [10]. In Section 4, we improve Tuy's outer approximation algorithm. In Section 5, we propose a procedure for calculating the vertex sets utilizing the relation of connections among vertices by edges. In Section 6, to verify the effectively of the algorithm integrated the outer approximation method and the procedure for calculating the vertex sets proposed in Sections 4 and 5, we show computational experiments.

Throughout this paper, we use the following notation: $\mathbb{R}$ denotes the set of all real numbers. For a subset $X \subset \mathbb{R}^n$, $\mathrm{int}\ X$, $\mathrm{bd}\ X$ and $\mathrm{co}\ X$ denote the interior, the boundary and the convex hull of X, respectively. For a finite set $X \subset \mathbb{R}^n$, $|X|$ denotes the number of elements of X. Given a convex polyhedral set (or polytope) $X \subset \mathbb{R}^n$, $V(X)$ denotes the set of all vertices of X. For vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, we use two kinds of symbols for open and closed line segments: $]\boldsymbol{a}, \boldsymbol{b}[:= \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} = \boldsymbol{a}+\delta(\boldsymbol{b}-\boldsymbol{a}),\ 0 < \delta < 1\}$ and $[\boldsymbol{a}, \boldsymbol{b}] := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} = \boldsymbol{a}+\delta(\boldsymbol{b}-\boldsymbol{a}),\ 0 \leq \delta \leq 1\}$. Given a vector $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a}^\top$ denotes the transposed vector of $\boldsymbol{a}$. Given a convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $\partial f(\boldsymbol{x})$ denotes the subdifferential of f at $\boldsymbol{x}$, that is, $\partial f(\boldsymbol{x}) := \{\boldsymbol{u} \in \mathbb{R}^n : f(\boldsymbol{y}) \geq \langle \boldsymbol{u}, \boldsymbol{y} - \boldsymbol{x}\rangle + f(\boldsymbol{x}),\ \boldsymbol{y} \in \mathbb{R}^n\}$.

2. DC PROGRAMMING PROBLEM

Let us consider the following dc programming problem:

$$(\mathrm{DC}) \begin{cases} \text{minimize} & f(\boldsymbol{x}) - g(\boldsymbol{x}) \\ \text{subject to} & h_j(\boldsymbol{x}) \leq 0,\ j = 1, \ldots, m, \end{cases}$$

where $f, g, h_j : \mathbb{R}^n \longrightarrow \mathbb{R}$ $(j = 1, \ldots, m)$ are continuously differentiable convex functions. Since the objective function of (DC) is defined as the difference of two convex functions, the objective function is called dc function. Let $X := \{\boldsymbol{x} \in \mathbb{R}^n : h_j(\boldsymbol{x}) \leq 0,\ j = 1, \ldots, m\}$. Then, from the convexity of h_j, X is a convex set.

For (DC), we assume the following conditions.

(A1): $\mathrm{int}\ X = \{\boldsymbol{x} \in \mathbb{R}^n : h_j(\boldsymbol{x}) < 0,\ j = 1, \ldots, m\} \neq \emptyset$.

(A2): X is compact.

(A3): A real number r is given and satisfies $r \geq \max\{\|\boldsymbol{x} - \boldsymbol{y}\| : \boldsymbol{x}, \boldsymbol{y} \in X\}$.

From Assumption (A2) and the continuity of the objective function, (DC) has a globally optimal solution. Now, we notice that (DC) is a convex (concave) programming problem if g (f) is linear. Then, the useful solution methods have been already proposed. Therefore, we add the following assumption for (DC).

(A4): f and g are not linear.

3. TUY'S OUTER APPROXIMATION ALGORITHM

In order to calculate an approximate solution of (DC), the following outer approximation algorithm has been proposed by Tuy [10].

Algorithm OA

Step 0:

Step 0-1: Find a feasible solution $\boldsymbol{y}^1 \in X$ and set $\omega_1 := f(\boldsymbol{y}^1) - g(\boldsymbol{y}^1)$. Go to Step 0-2.

Step 0-2: Construct a polytope $S \subset \mathbb{R}^n$ satisfying $S \supset X$. Calculate the vertex set $V(S)$. Choose $\boldsymbol{x}' \in V(S)$ satisfying $f(\boldsymbol{x}') = \max\{f(\boldsymbol{x}) : \boldsymbol{x} \in V(S)\}$. Set $\tilde{t} := f(\boldsymbol{x}')$ and $D(\boldsymbol{y}^1) := \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \boldsymbol{x} \in X,\ f(\boldsymbol{x}) - t \le \omega_1,\ t \le \tilde{t}\}$. Go to Step 0-3.

Step 0-3: Construct a polytope $P_1 \subset \mathbb{R}^{n+1}$ satisfying $P_1 \supset D(\boldsymbol{y}^1)$ and $P_1 \subset \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : t \le \tilde{t}\}$. Calculate the vertex set $V(P_1)$. Go to Step 0-4.

Step 0-4: Find $(\bar{\boldsymbol{y}}, \bar{t}) \in \operatorname{int} D(\boldsymbol{y}^1)$. Set a tolerance $\tau \ge 0$ and $k = 1$. Go to Step 1.

Step 1: Choose $(\boldsymbol{x}^k, t_k) \in \arg\min\{-g(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in V(P_k)\}$. Go to Step 2.

Step 2: If $-g(\boldsymbol{x}^k) + t_k \ge -\tau$, then stop: $\boldsymbol{y}^k$ is an approximate solution of (DC). Otherwise, go to Step 3.

Step 3: Set $\boldsymbol{y}^{k+1}, \omega_{k+1}, P_{k+1}$ as follows:

$$\boldsymbol{y}^{k+1} := \begin{cases} \boldsymbol{x}^k & \text{if } f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) < 0 \text{ and } \boldsymbol{x}^k \in X, \\ \boldsymbol{y}^k & \text{otherwise,} \end{cases}$$

$$\omega_{k+1} := \begin{cases} f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) & \text{if } f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) < 0 \text{ and } \boldsymbol{x}^k \in X, \\ \omega_k & \text{otherwise,} \end{cases}$$

$$P_{k+1} := P_k \cap \{(\boldsymbol{x}, t) \in R^n \times R : \ell_k(\boldsymbol{x}, t) \le 0\},$$

where

$$\ell_k(\boldsymbol{x}, t) := \langle \boldsymbol{d}^k, (\boldsymbol{x}^\top, t)^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle,$$
$$\boldsymbol{d}^k \in \partial\phi(\boldsymbol{z}^k, \theta_k),$$
$$(\boldsymbol{z}^k, \theta_k) \in [(\boldsymbol{x}^k, t_k), (\bar{\boldsymbol{y}}, \bar{t})] \cap \operatorname{bd} D(\boldsymbol{y}^k).$$

Calculate the vertex set $V(P_{k+1})$. Set $k \leftarrow k + 1$ and return to Step 1.

The sequences $\{D(\boldsymbol{y}^k)\}$ and $\{P_k\}$ satisfy the following conditions.

$$D(\boldsymbol{y}^1) \supset D(\boldsymbol{y}^2) \supset \cdots \supset D(\boldsymbol{y}^k) \supset D(\boldsymbol{y}^{k+1}) \supset \cdots,$$
$$P_1 \supset P_2 \supset \cdots \supset P_k \supset P_{k+1} \supset \cdots,$$
$$P_k \supset D(\boldsymbol{y}^k) \text{ for each } k = 1, 2, \ldots.$$

Moreover, we have the following inequality.

$$\omega_1 \ge \omega_2 \ge \cdots \ge \omega_k \ge \omega_{k+1} \ge \cdots \ge \min(\mathrm{DC}),$$

where $\min(\mathrm{DC})$ denotes the optimal value of (DC). By approximating $\{D(\boldsymbol{y}^k\}$ by $\{P_k\}$ from outside, Algorithm OA generates the provisional solution sequence $\{\boldsymbol{y}^k\}$. Moreover, it is shown that every accumulation point $\{\boldsymbol{y}^k\}$ is a globally optimal solution of (DC). Hence, by setting $\tau > 0$, Algorithm OA calculates an approximate solution of (DC). However, $\{P_k\}$ often do not converge effectively, because the shape of $D(\boldsymbol{y}^k)$ approximated by P_k is changed as the number of iterations increases. Therefore, we improve Algorithm OA by fixing the target approximated by $\{P_k\}$ in Section 4. By such the improvement, the convergence of $\{P_k\}$ becomes more efficient.

4. IMPROVEMENT OF TUY'S OUTER APPROXIMATION ALGORITHM

In order to enhance the computational efficiency of the algorithm, we improve Tuy's outer approximation algorithm as follows:

Algorithm IOA

Step 0:

Step 0-1: Find a feasible solution $\boldsymbol{y}^1 \in X$ and set $\omega_1 := f(\boldsymbol{y}^1) - g(\boldsymbol{y}^1)$. Go to Step 0-2.

Step 0-2: Construct a polytope $S \subset \mathbb{R}^n$ satisfying $S \supset X$. Calculate the vertex set $V(S)$. Choose $\boldsymbol{x}' \in V(S)$ satisfying $f(\boldsymbol{x}') = \max\{f(\boldsymbol{x}) : \boldsymbol{x} \in V(S)\}$. Set $\tilde{t} := f(\boldsymbol{x}')$ and $D := \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \boldsymbol{x} \in X,\ f(\boldsymbol{x}) \le t \le \tilde{t}\}$. Go to Step 0-3.

Step 0-3: Construct a polytope $P_1 \subset \mathbb{R}^{n+1}$ satisfying $P_1 \supset D$ and $P_1 \subset \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : t \le \tilde{t}\}$. Calculate the vertex set $V(P_1)$. Go to Step 0-4.

Step 0-4: Find $(\bar{\boldsymbol{y}}, \bar{t}) \in \operatorname{int} D$. Set a tolerance $\tau \ge 0$ and $k = 1$. Go to Step 1.

Step 1: Choose $(\boldsymbol{x}^k, t_k) \in \arg\min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in V(P_k)\}$, where $g_k(\boldsymbol{x}) := g(\boldsymbol{x}) + \omega_k$. Go to Step 2.

Step 2: If $-g_k(\boldsymbol{x}^k) + t_k \ge -\tau$, then stop: $\boldsymbol{y}^k$ is an approximate solution of (DC). If $\phi(\boldsymbol{x}^k, t_k) \le \tau$, then stop: $\boldsymbol{x}^k$ is an approximate solution of (DC), where

$$\phi(\boldsymbol{x}, t) := \max\{h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}), f(\boldsymbol{x}) - t\}.$$

Otherwise, go to Step 3.

Step 3: Set $\boldsymbol{y}^{k+1}, \omega_{k+1}, P_{k+1}$ as follows:

$$\boldsymbol{y}^{k+1} := \begin{cases} \boldsymbol{x}^k & \text{if } f(\boldsymbol{x}^k) - g_k(\boldsymbol{x}^k) < 0 \text{ and } \boldsymbol{x}^k \in X, \\ \boldsymbol{y}^k & \text{otherwise,} \end{cases}$$

$$\omega_{k+1} := \begin{cases} f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) & \text{if } f(\boldsymbol{x}^k) - g_k(\boldsymbol{x}^k) < 0 \text{ and } \boldsymbol{x}^k \in X, \\ \omega_k & \text{otherwise,} \end{cases}$$

$$P_{k+1} := P_k \cap \{(\boldsymbol{x}, t) \in R^n \times R : \ell_k(\boldsymbol{x}, t) \le 0\},$$

where

$$\ell_k(\boldsymbol{x}, t) := \langle \boldsymbol{d}^k, (\boldsymbol{x}^\top, t)^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle,$$
$$\boldsymbol{d}^k \in \partial\phi(\boldsymbol{z}^k, \theta_k),$$
$$(\boldsymbol{z}^k, \theta_k) \in [(\boldsymbol{x}^k, t_k), (\bar{\boldsymbol{y}}, \bar{t})] \cap \operatorname{bd} D.$$

Calculate the vertex set $V(P_{k+1})$. Set $k \leftarrow k + 1$ and return to Step 1.

From Assumption (A4), $\operatorname{int} D \neq \emptyset$ and hence $(\bar{\boldsymbol{y}}, \bar{t}) \in \operatorname{int} D$ can be found. For each k, since $D = \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \phi(\boldsymbol{x}, t) \le 0, t \le \tilde{t}\}$ and $\phi(\boldsymbol{x}, t)$ is convex on $\mathbb{R}^{n+1}$, we have $D \subset \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \ell_k(\boldsymbol{x}, t) \le 0\}$. Hence,

$$P_1 \supset P_2 \supset \cdots \supset P_k \supset \cdots \supset D.$$

By the definition of $\boldsymbol{y}^k$ at Steps 0-1 and 3, $\{\boldsymbol{y}^k\} \subset X$. At iteration k, if $f(\boldsymbol{x}^k) - g_k(\boldsymbol{x}^k) < 0$,

$$f(\boldsymbol{x}^k) - g_k(\boldsymbol{x}^k) = f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) - \omega_k = f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) - (f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k)) < 0,$$

that is, $f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) < f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k)$. Therefore, for each k, we have the following inequalities.

$$f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k) \ge f(\boldsymbol{y}^{k+1}) - g(\boldsymbol{y}^{k+1}) \ge \min(\mathrm{DC}),$$
$$\omega^k \ge \omega^{k+1} \ge \min(\mathrm{DC}).$$

Moreover, the following theorems hold.

Theorem 4.1. *Assume that $-g_k(\boldsymbol{x}^k) + t_k \geq 0$ at iteration k of Algorithm* IOA. *Then, $\boldsymbol{y}^k$ is a globally optimal solution of* (DC).

Proof. At iteration k of Algorithm IOA, it follows from the definition of $\boldsymbol{y}^k$ that $\boldsymbol{y}^k \in X$. Hence, in order to complete the proof, we shall show that $f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k) \leq f(\boldsymbol{x}) - g(\boldsymbol{x})$ for each $\boldsymbol{x} \in X$. Let $\boldsymbol{x}' \in X$. Then, $(\boldsymbol{x}', f(\boldsymbol{x}')) \in D$. Since $D \subset P_k$ and g_k is convex, from the definition of $(\boldsymbol{x}^k, t_k)$, we have

$$\begin{aligned} -g_k(\boldsymbol{x}') + f(\boldsymbol{x}') &\geq \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in D\} \\ &\geq \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in P_k\} \\ &= \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in V(P_k)\} \\ &= -g_k(\boldsymbol{x}^k) + t_k \geq 0 \end{aligned}$$

Hence,

$$f(\boldsymbol{x}') - g_k(\boldsymbol{x}') = f(\boldsymbol{x}') - g(\boldsymbol{x}') - \omega_k = f(\boldsymbol{x}') - g(\boldsymbol{x}') - (f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k)) \geq 0.$$

Therefore,

$$f(\boldsymbol{x}') - g(\boldsymbol{x}') \geq f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k).$$

Consequently, $\boldsymbol{y}^k$ is a globally optimal solution of (DC). □

Theorem 4.2. *Assume that $\phi(\boldsymbol{x}^k, t_k) \leq 0$ at iteration k of Algorithm* IOA. *Then, $\boldsymbol{x}^k$ is a globally optimal solution of* (DC).

Proof. Since $\phi(\boldsymbol{x}^k, t_k) \leq 0$, $(\boldsymbol{x}^k, t_k) \in D$. Hence,

$$\begin{aligned} -g_k(\boldsymbol{x}^k) + t_k &\geq \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in D\} \\ &\geq \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in P_k\} \\ &= \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in V(P_k)\} \\ &= -g_k(\boldsymbol{x}^k) + t_k. \end{aligned}$$

This implies that $-g_k(\boldsymbol{x}^k) + t_k \geq \min\{-g_k(\boldsymbol{x}) + t : (\boldsymbol{x}, t) \in D\}$. Since $(\boldsymbol{x}^k, f(\boldsymbol{x}^k)) \in D$, $-g_k(\boldsymbol{x}^k) + t_k \leq -g_k(\boldsymbol{x}^k) + f(\boldsymbol{x}^k)$, that is, $t_k \leq f(\boldsymbol{x}^k)$. Moreover, since $(\boldsymbol{x}^k, t_k) \in D$, $t_k \geq f(\boldsymbol{x}^k)$. Hence, $t_k = f(\boldsymbol{x}^k)$. Since $-g_k(\boldsymbol{x}) + t \geq -g_k(\boldsymbol{x}) + f(\boldsymbol{x})$ for each $(\boldsymbol{x}, t) \in D$, we have

$$\begin{aligned} -g(\boldsymbol{x}^k) + f(\boldsymbol{x}^k) - \omega_k &= \min\{-g(\boldsymbol{x}) + t - \omega_k : (\boldsymbol{x}, t) \in D\} \\ &= \min\{-g(\boldsymbol{x}) + f(\boldsymbol{x}) - \omega_k : \boldsymbol{x} \in X\}. \end{aligned}$$

Since ω_k is constant, $f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) = \min\{f(\boldsymbol{x}) - g(\boldsymbol{x}) : \boldsymbol{x} \in X\}$. Consequently, $\boldsymbol{x}^k$ is a globally optimal solution of (DC). □

Corollary 4.3. *Assume that $\phi(\boldsymbol{x}^k, t_k) > 0$ at iteration k of Algorithm* IOA. *Then, there exists $(\boldsymbol{z}^k, \theta_k) \in [(\boldsymbol{x}^k, t_k), (\bar{\boldsymbol{y}}, \bar{t})] \cap \mathrm{bd}\ D$. Moreover, $(\boldsymbol{x}^k, t_k) \notin P_{k+1}$.*

Proof. We note $D = \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \phi(\boldsymbol{x}, t) \leq 0,\ t \leq \tilde{t}\}$. Since $(\bar{\boldsymbol{y}}, \bar{t}) \in \mathrm{int}\ D$, $\phi(\bar{\boldsymbol{y}}, \bar{t}) < 0$. Hence, from the assumption of this corollary, there exists $(\boldsymbol{z}^k, \theta_k) \in [(\boldsymbol{x}^k, t_k), (\bar{\boldsymbol{y}}, \bar{t})]$ such that $\phi(\boldsymbol{z}^k, \theta_k) = 0$, that is, $(\boldsymbol{z}^k, \theta_k) \in \mathrm{bd}\ D$. Moreover, from the convexity of ϕ,

$$\begin{aligned} \ell_k(\bar{\boldsymbol{y}}, \bar{t}) &= \langle \boldsymbol{d}^k, (\bar{\boldsymbol{y}}^\top, \bar{t})^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle = \phi(\boldsymbol{z}^k, \theta_k) + \langle \boldsymbol{d}^k, (\bar{\boldsymbol{t}}^\top, \bar{t})^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle \\ &\leq \phi(\bar{\boldsymbol{y}}, \bar{t}) < 0. \end{aligned}$$

We note that there exists $\mu > 0$ satisfying

$$((\boldsymbol{x}^k)^\top, t_k)^\top - (\boldsymbol{z}^k, \theta_k)^\top = -\mu((\bar{\boldsymbol{y}}^\top, \bar{t})^\top - (\boldsymbol{z}^k, \theta_k)^\top).$$

Therefore,

$$\ell_k(\boldsymbol{x}^k, t_k) = \langle \boldsymbol{d}^k, ((\boldsymbol{x}^k)^\top, t_k)^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle = -\mu \langle \boldsymbol{d}^k, (\bar{\boldsymbol{y}}^\top, \bar{t})^\top - (\boldsymbol{z}^k, \theta_k)^\top \rangle > 0.$$

Consequently, $(\boldsymbol{x}^k, t_k) \notin P_{k+1}$. □

Theorem 4.4. *Assume that $\tau = 0$ and that $\{(\boldsymbol{x}^k, t_k)\}$ is an infinite sequence generated by Algorithm* IOA. *Then, every accumulation point of $\{(\boldsymbol{x}^k, t_k)\}$ is contained in D.*

Proof. Since $\{(\boldsymbol{x}^k, t_k)\} \subset P_1$ and P_1 is bounded, $\{(\boldsymbol{x}^k, t_k)\}$ has an accumulation point. Moreover, since $\{(\boldsymbol{z}^k, \theta_k)\} \subset \mathrm{bd}\ D$ and $\mathrm{bd}\ D$ is bounded, $\{(\boldsymbol{z}^k, \theta_k)\}$ has an accumulation point. Furthermore, since $\{\boldsymbol{d}^k\} \subset \cup_{(x,t)\in \mathrm{bd}\ D} \partial\phi(\boldsymbol{x}, t)$ and $\cup_{(x,t)\in \mathrm{bd}\ D} \partial\phi(\boldsymbol{x}, t)$ is bounded (see Theorem 24.7 in [4]), $\{\boldsymbol{d}^k\}$ has an accumulation point. Let $(\hat{\boldsymbol{x}}, \hat{t})$, $(\hat{\boldsymbol{z}}, \hat{\theta})$ and $\hat{\boldsymbol{d}}$ be accumulation points of $\{(\boldsymbol{x}^k, t_k)\}$, $\{(\boldsymbol{z}^k, \theta_k)\}$ and $\{\boldsymbol{d}^k\}$, respectively. Without loss of generality, we can assume that $(\boldsymbol{x}^k, t_k) \longrightarrow (\hat{\boldsymbol{x}}, \hat{t})$ and $(\boldsymbol{z}^k, \theta_k) \longrightarrow (\hat{\boldsymbol{z}}, \hat{\theta})$ as $k \longrightarrow +\infty$. Then, from the upper semi-continuity of $\partial\phi$, $\hat{d} \in \partial\phi(\hat{\boldsymbol{z}}, \hat{t})$ (see Theorem 24.4 in [4]).

In order to obtain a contradiction, we suppose that $(\hat{\boldsymbol{x}}, \hat{t}) \notin D$. Since $\{(\boldsymbol{x}^k, t_k)\} \subset P_1 \subset \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : t \leq \tilde{t}\}$ and $D = \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \phi(\boldsymbol{x}, t) \leq 0,\ t \leq \tilde{t}\}$, $\hat{t} \leq \tilde{t}$ and hence $\phi(\hat{\boldsymbol{x}}, \hat{t}) > 0$. Then, there exists $\mu > 0$ such that $(\hat{\boldsymbol{x}}, \hat{t}) - (\hat{\boldsymbol{z}}, \hat{\theta}) = -\mu((\bar{\boldsymbol{y}}, \bar{t}) - (\hat{\boldsymbol{z}}, \hat{\theta}))$. Since $\langle \hat{\boldsymbol{d}}, (\bar{\boldsymbol{y}}^\top, \bar{t})^\top - (\hat{\boldsymbol{z}}^\top, \hat{\theta})^\top \rangle \leq \phi(\bar{\boldsymbol{y}}, \bar{t}) < 0$,

$$\langle \hat{\boldsymbol{d}}, (\hat{\boldsymbol{x}}^\top, \hat{t})^\top - (\hat{\boldsymbol{z}}^\top, \hat{\theta})^\top \rangle = -\mu \langle \hat{d}, (\bar{\boldsymbol{y}}^\top, \bar{t})^\top - (\hat{\boldsymbol{z}}^\top, \hat{\theta})^\top \rangle > 0.$$

Moreover, we have

$$\begin{aligned} \lim_{k\longrightarrow+\infty} \ell_k(\boldsymbol{x}^k, t_k) &= \lim_{k\longrightarrow+\infty} \langle \boldsymbol{d}^k, ((\boldsymbol{x}^k)^\top, t_k)^\top - ((\boldsymbol{z}^k)^\top, \theta_k)^\top \rangle \\ &= \langle \hat{\boldsymbol{d}}, (\hat{\boldsymbol{x}}^\top, \hat{t})^\top - (\hat{\boldsymbol{z}}^\top, \hat{\theta})^\top \rangle > 0. \end{aligned}$$

Hence, there exists $k_1, k_2 > 0$ such that $\ell_{k_2}(\hat{\boldsymbol{x}}, \hat{t}) > 0$ and $\ell_{k_2}(\boldsymbol{x}^k, t_k) > 0$ for each $k \geq k_1$. This implies that $\{(\boldsymbol{x}^k, t_k)\}_{k\geq k_1} \cap P_{k_2+1} = \emptyset$. However, from the definitions of P_k and $(\boldsymbol{x}^k, t_k)$, $\{(\boldsymbol{x}^k, t_k)\}_{k>\max\{k_1,k_2\}} \subset P_{k_2+1}$. This is a contradiction. Consequently, $(\hat{\boldsymbol{x}}, \hat{t}) \in D$. □

Theorem 4.5. *Assume that $\tau = 0$ and that $\{(\boldsymbol{x}^k, t_k)\}$ is an infinite sequence generated by Algorithm* IOA. *Then, every accumulation point of $\{\boldsymbol{x}^k\}$ is a globally optimal solution of* (DC).

Proof. In the same way of Theorem 4.4, we can assume that $(\boldsymbol{x}^k, t_k) \longrightarrow (\hat{\boldsymbol{x}}, \hat{t})$ as $k \longrightarrow +\infty$. Since $\{\boldsymbol{y}^k\} \subset X$ and X is bounded, $\{\boldsymbol{y}^k\}$ has an accumulation point. Hence, without loss of generality, we can assume that $\boldsymbol{y}^k \longrightarrow \hat{\boldsymbol{y}}$ as $k \longrightarrow +\infty$. Moreover, since $\omega_k = f(\boldsymbol{y}^k) - g(\boldsymbol{y}^k)$, from the continuity of f and g, we can set $\hat{\omega} := \lim_{k\longrightarrow+\infty} \omega_k$. Then, by the definitions of g_k and D, we have

$$\begin{aligned} \hat{t} - g(\hat{\boldsymbol{x}}) - \hat{\omega} &= \lim_{k\longrightarrow+\infty} t_k - g(\boldsymbol{x}^k) - \omega_k \\ &= \lim_{k\longrightarrow+\infty} \min\{t - g(\boldsymbol{x}) - \omega_k : (\boldsymbol{x}, t) \in V(P_k)\} \\ &= \lim_{k\longrightarrow+\infty} \min\{t - g(\boldsymbol{x}) - \omega_k : (\boldsymbol{x}, t) \in P_k\} \\ &\leq \lim_{k\longrightarrow+\infty} \min\{t - g(\boldsymbol{x}) - \omega_k : (\boldsymbol{x}, t) \in D\} \\ &= \lim_{k\longrightarrow+\infty} \min\{f(\boldsymbol{x}) - g(\boldsymbol{x}) - \omega_k : (\boldsymbol{x}, t) \in D\} \\ &= \min(\mathrm{DC}) - \hat{\omega}. \end{aligned}$$

Hence, $\hat{t} - g(\hat{\boldsymbol{x}}) \leq \min(\mathrm{DC})$. Moreover, from Theorem 4.4, $(\hat{\boldsymbol{x}}, \hat{t}) \in D$. By the definition of D, $\hat{t} \geq f(\hat{\boldsymbol{x}})$. Therefore, $\hat{t} - g(\hat{\boldsymbol{x}}) \geq f(\hat{\boldsymbol{x}}) - g(\hat{\boldsymbol{x}}) \geq \min(\mathrm{DC})$. Thus, we have

$$\hat{t} - g(\hat{\boldsymbol{x}}) = f(\hat{\boldsymbol{x}}) - g(\hat{\boldsymbol{x}}) = \min(\mathrm{DC}).$$

Consequently, $\hat{\boldsymbol{x}}$ is a globally optimal solution of (DC). □

Corollary 4.6. *Assume that $\tau = 0$ and that $\{\boldsymbol{y}^k\}$ is an infinite sequence generated by Algorithm* IOA. *Then, every accumulation point of $\{\boldsymbol{y}^k\}$ is a globally optimal solution of* (DC).

Proof. In the same way of Theorem 4.5, we can assume that $\boldsymbol{y}^k \longrightarrow \hat{\boldsymbol{y}}$ as $k \longrightarrow +\infty$. Since $\{\boldsymbol{y}^k\} \subset X$ and X is compact, $\hat{\boldsymbol{y}} \in X$. Hence, $f(\hat{\boldsymbol{y}}) - g(\hat{\boldsymbol{y}}) \geq \min(\mathrm{DC})$. Moreover, by Theorem 4.5 and the definition of $\boldsymbol{y}^k$, we have

$$\min(\mathrm{DC}) == \lim_{k \longrightarrow +\infty} f(\boldsymbol{x}^k) - g(\boldsymbol{x}^k) \geq \lim_{k \longrightarrow +\infty} f(\boldsymbol{y}^{k+1}) - g(\boldsymbol{y}^{k+1}) = f(\hat{\boldsymbol{y}}) - g(\hat{\boldsymbol{y}}).$$

Therefore, $f(\hat{\boldsymbol{y}}) - g(\hat{\boldsymbol{y}}) = \min(\mathrm{DC})$. Consequently, $\hat{\boldsymbol{y}}$ is a globally optimal solution of (DC). □

From Theorems 4.1 and 4.2, it is shown that the stopping criteria of Algorithm IOA are valid. In the case where infinite sequences $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{y}^k\}$ are generated by Algorithm IOA, by Theorem 4.5 and Corollary 4.6, every accumulation point of $\{\boldsymbol{x}^k\}$ and $\{\boldsymbol{y}^k\}$ is a globally optimal solution of (DC). Moreover, by setting $\tau > 0$, it follows from Theorem 4.4 that Algorithm IOA terminates within a finite number of iterations.

5. PROCEDURE FOR CALCULATING THE VERTEX SET

At Step 3 of Algorithm IOA proposed in Section 4, the vertex set $V(P_k)$ is calculated. In the classical methods, to obtain the vertex set, many systems of linear equations are solved. However, it is known that such procedures are inefficient. In this section, we propose a procedure for calculating the vertex set $V(P_k)$ utilizing the relation of connections among vertices by edges. Hence, in order to construct such a procedure, we introduce the following definitions on convex analysis.

Definition 5.1. ([8]) Let P be an n-dimensional polyhedral set and let H be a supporting hyperplane of P. Then, the intersection $F := H \cap P$ is called a *face* of P. If $\dim F = 1$, F is called an *edge* of P. If $\dim F = n - 1$, F is called a *facet* of P.

We remember that P_k is given at iteration k of Algorithm IOA. Let $(\boldsymbol{v}(i), t(i))$ ($i \in \Delta_k \subset \{1, \ldots, \alpha(k)\}$) be vertices of P_k and let F_j ($j \in \Gamma_k \subset \{1, \ldots, \beta(k)\}$) be facets of P_k, where

- $\alpha(k)$ and $\beta(k)$ denote the numbers of vertices and facets generated by iteration k, respectively,
- Δ_k is the index set of all vertices of P_k, that is, $\{(\boldsymbol{v}(i), t(i)) : i \in \Delta_k\} = V(P_k)$
- Γ_k is the index set of all facets of P_k, that is, $F_j \cap P_k \neq \emptyset$ for each $j \in \Gamma_k$ and $F_{j'} \cap P_k = \emptyset$ for any $j' \in \{1, \ldots, \beta(k)\} \backslash \Gamma_k$.

For each $i \in \Delta_k$, we set the index sets $\mathcal{V}_i$ and $\mathcal{F}_i$ as follows:

$$\mathcal{V}_i := \{j : \mathrm{co}\,\{(\boldsymbol{v}(i), t(i)), (\boldsymbol{v}(j), t(j))\} \text{ is an edge of } P_k,\ j \in \Delta_k \backslash \{i\}\},$$

$$\mathcal{F}_i := \{j : (\boldsymbol{v}(i), t(i)) \in F_j,\ j \in \Gamma_k\}.$$

The procedure proposed in this section consists of the following three parts.

Section 5.1: Search for all vertices of P_k to remove.

Section 5.2: Calculation of all vertices of P_{k+1} to generate.
Section 5.3: Check for the availability of F_i ($i \in \Gamma_k$).
Section 5.4: Update the index sets $\mathcal{V}_i$.

5.1. **SEARCH FOR ALL VERTICES TO REMOVE.** Since $P_{k+1} = P_k \cap \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \ell_k(\boldsymbol{x}, t) \leq 0\}$ and $\ell_k(\boldsymbol{v}(i_k), t(i_k)) > 0$, there exist vertices of P_k which are not contained in P_{k+1}. Hence, to construct the vertex set $V(P_{k+1})$, it is necessary to search for all $(\boldsymbol{v}, t) \in V(P_k)$ satisfying $\ell_k(\boldsymbol{v}, t) > 0$. For this reason, we propose the following procedure.

Procedure A

Step 0: Set $\mathcal{M} := \mathcal{M}' = \{i_k\}$. Go to Step 1.
Step 1: If $\mathcal{M}' = \emptyset$, then stop. Otherwise, choose $j \in \mathcal{M}'$ and go to Step 2.
Step 2:
  Step 2-0: Set $\mathcal{T} := \mathcal{V}_j$ and go to Step 2-1.
  Step 2-1: If $\mathcal{T} = \emptyset$, then go to Step 3. Otherwise, choose $q \in \mathcal{T}$ and go to Step 2-2.
  Step 2-2: If $\ell_k(\boldsymbol{v}(q), t(q)) > 0$ and $q \notin \mathcal{M}$, set $\mathcal{M} \leftarrow \mathcal{M} \cup \{q\}$ and $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{q\}$. Go to Step 2-3.
  Step 2-3: Set $\mathcal{T} \leftarrow \mathcal{T} \backslash \{q\}$ and return to Step 1.
Step 3: Set $\mathcal{M}' \leftarrow \mathcal{M}' \backslash \{j\}$ and return to Step 1.

Procedure A attain the following.

- $\mathcal{M}$ listed the indices of all vertex of P_k which are not contained in P_{k+1}. Hence, for each $i \in \mathcal{M}$,

$$(\boldsymbol{v}(i), t(i)) \in V(P_k) \text{ and } (\boldsymbol{v}(i), t(i)) \notin V(P_{k+1}).$$

5.2. **CALCULATION OF ALL VERTICES TO GENERATE.** Let $(\boldsymbol{v}(i_1), t(i_1)), (\boldsymbol{v}(i_2), t(i_2)) \in V(P_k)$ satisfy

$$i_2 \in \mathcal{V}_{i_1},\ \ell_k(\boldsymbol{v}(i_1), t(i_1)) > 0 \text{ and } \ell_k(\boldsymbol{v}(i_2), t(i_2)) < 0.$$

Then, $\mathrm{co}\ \{(\boldsymbol{v}(i_1), t(i_1)), (\boldsymbol{v}(i_2), t(i_2))\}$ is an edge of P_k and there exists a vertex $(\boldsymbol{v}', t')$ of P_{k+1} generated at iteration k of Algorithm IOA, that is, $(\boldsymbol{v}', t') \in V(P_{k+1}) \backslash V(P_k)$. Moreover, the following assertions hold.

- $(\boldsymbol{v}', t') \in F_j$ for all $j \in \mathcal{F}_{i_1} \cap \mathcal{F}_{i_2}$.
- $(\boldsymbol{v}', t') \in F_{\beta(k)+1}$.

Here, $F_{\beta(k)+1} := P_{k+1} \cap \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \ell_k(\boldsymbol{x}, t) = 0\} = P_k \cap \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \ell_k(\boldsymbol{x}, t) = 0\}$. We note that $F_{\beta(k)+1}$ is a unique facet of P_{k+1} at iteration k of Algorithm IOA. Hence, we can set $\beta(k+1) := \beta(k) + 1$.

In order to calculate all vertices generated at iteration k of Algorithm IOA, we proposed the following procedure.

Procedure B

Step 0: Set $\mathcal{W} := \emptyset$, $\alpha(k+1) := \alpha(k)$ and $\mathcal{M}' := \mathcal{M}$, where $\mathcal{M}$ is generated by Procedure A. Go to Step 1.
Step 1: If $\mathcal{M}' = \emptyset$, then set $\Delta_{k+1} := (\Delta_k \cup \{\alpha(k)+1, \ldots, \alpha(k+1)\}) \backslash \mathcal{M}$ and $\beta(k+1) := \beta(k)+1$, and stop. Otherwise, choose $j \in \mathcal{M}'$ and go to Step 2.
Step 2:
  Step 2-0: Set $\mathcal{T} := \mathcal{V}_j$ and go to Step 2-1.
  Step 2-1: If $\mathcal{T} = \emptyset$, then go to Step 3. Otherwise, choose $q \in \mathcal{T}$ and go to Step 2-2.

Step 2-2: If $\ell_k(\boldsymbol{v}(q), t(q)) < 0$, then go to Step 2-3. If $\ell_k(\boldsymbol{v}(q), t(q)) = 0$, then go to Step 2-4. Otherwise, go to Step 2-5.

Step 2-3: Set $(\boldsymbol{v}(\alpha(k+1)+1), t(\alpha(k+1)+1)$, $\mathcal{F}_{\alpha(k+1)+1}$ and $\mathcal{V}_{\alpha(k+1)+1}$ as follows:

$$(\boldsymbol{v}(\alpha(k+1)+1), t(\alpha(k+1))) := (1-\lambda)(\boldsymbol{v}(j), t(j)) + \lambda(\boldsymbol{v}(q), t(q)),$$
$$\mathcal{F}_{\alpha(k+1)+1} := (\mathcal{F}_j \cap \mathcal{F}_q) \cup \{\beta(k)+1\},$$
$$\mathcal{V}_{\alpha(k+1)+1} := \{q\},$$
$$\mathcal{V}_q \leftarrow (\mathcal{V}_q \backslash \{j\}) \cup \{\alpha(k+1)+1\},$$
$$\mathcal{W} \leftarrow \mathcal{W} \cap \{\alpha(k+1)+1\},$$
$$\alpha(k+1) \leftarrow \alpha(k+1)+1,$$

where

$$\lambda := \frac{\ell_k(\boldsymbol{v}(q), t(q))}{\ell_k(\boldsymbol{v}(j), t(j)) - \ell_k(\boldsymbol{v}(q), t(q))}.$$

The renewal of $\mathcal{V}_{\alpha(k+1)+1}$ are incomplete at this step. The index set $\mathcal{V}_{\alpha(k+1)+1}$ will be corrected by Procedure D proposed in Section 5.4. Go to Step 2.

Step 2-4: Update $\mathcal{F}_q$ and $\mathcal{V}_q$ as follows:

$$\mathcal{F}_q \leftarrow \mathcal{F}_q \cup \{\beta(k)+1\},$$
$$\mathcal{V}_q \leftarrow \mathcal{V}_q \backslash \{j\},$$
$$\mathcal{W} \leftarrow \mathcal{W} \cup \{q\}.$$

Go to Step 2-5.

Step 2-5: Set $\mathcal{T} \leftarrow \mathcal{T} \backslash \{q\}$, and return to Step 2-1.

Step 3: Set $\mathcal{M}' \leftarrow \mathcal{M}' \backslash \{j\}$ and return to Step 1.

By Procedure B, we have

- Δ_{k+1}, $\alpha(k+1)$ and $\beta(k+1)$ are calculated. Hence, $V(P_{k+1}) := \{(\boldsymbol{v}(i), t(i)) : i \in \Delta_{k+1}\}$.
- $\{(\boldsymbol{v}(i), t(i)) : i = \alpha(k)+1, \ldots, \alpha(k+1)\} = V(P_{k+1}) \backslash V(P_k)$ is calculated.
- $\mathcal{W}$ listed the indices of all vertices of P_{k+1} contained in $\{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \ell_k(\boldsymbol{x}, t) = 0\}$ is constructed. Hence, $\mathcal{W} = \{\alpha(k)+1, \ldots, \alpha(k+1)\} \cup \{i \in \Delta_k : \ell_k(\boldsymbol{v}(i), t(i)) = 0\}$. We note that for each $i \in \mathcal{W}$, $\mathcal{V}_i$ needs to be generated or updated.
- For each $i \in \Delta_k \backslash \{i \in \Delta_k : \ell_k(\boldsymbol{v}(i), t(i)) = 0\}$, $\mathcal{V}_i$ is updated completely.
- For each $i \in \Delta_{k+1}$, $\mathcal{F}_i$ is updated.

5.3. **CHECK FOR THE AVAILABILITY OF FACETS.** In this section, we propose a procedure for checking the availability of facets F_i ($i \in \Gamma_k \cup \{\beta(k+1)\}$) for P_{k+1}. Then, it is clear that the following theorem holds.

Theorem 5.2. *Let $P \subset \mathbb{R}^n$ be an n-dimensional convex polyhedral set and let $F(1), \ldots, F(\beta)$ be facets of P. Then, for each $i \in \{1, \ldots, \beta\}$ and $j \in \{1, \ldots, \beta\} \backslash \{i\}$, $F_i \cap P_k \not\subset F_j$.*

By Theorem 5.2, we propose a procedure for omitting $i \in \Gamma_k$ satisfying $F_i \cap V(P_{k+1}) \subset F_j$ for some $j \in (\Gamma_k \cup \{\beta(k+1)\}) \backslash \{i\}$ from $\Gamma_k \cup \{\beta(k+1)\}$ as follows.

Procedure C

Step 0: Set $\Gamma_{k+1} := \Gamma_k \cup \{\beta(k+1)\}$ and $i := 1$. Go to Step 1.

Step 1: If $i = \beta(k+1)$, then stop. If $i \notin \Gamma_{k+1}$, go to Step 4. Otherwise, go to Step 2.

Step 2: Set $\Psi_i := \{q \in \Delta_{k+1} : i \in \mathcal{F}_q\}$. Go to Step 3.
Step 3:
Step 3-0: Set $j := i + 1$ and go to Step 3-1.
Step 3-1: If $j = \beta(k+1)+1$, then go to Step 4. If $j \notin \Gamma_{k+1}$, then go to Step 3-4. Otherwise, go to Step 3-2.
Step 3-2: Set $\Psi_j := \{q \in \Delta_{k+1} : j \in \mathcal{F}_q\}$. Go to Step 3-3.
Step 3-3: If $\Psi_i \subset \Psi_j$, then set $\Gamma_{k+1} \leftarrow \Gamma_{k+1}\backslash\{i\}$, $\mathcal{F}_q \leftarrow \mathcal{F}_q\backslash\{i\}$ for each $q \in \Delta_{k+1}$, and go to Step 4. Otherwise, go to Step 3-4.
Step 3-4: Set $j \leftarrow j + 1$ and return to Step 3-1.
Step 4: Set $i \leftarrow i + 1$ and return to Step 1.

From Procedure C, for each $i \in \Gamma_{k+1}$,

$$\dim\,(P_{k+1} \cap F_i) = n - 1.$$

5.4. **UPDATE THE INDEX SETS.** Procedure B generates $V(P_k)$ by utilizing $\mathcal{V}_i$ ($i \in \mathcal{M}$), where $\mathcal{M}$ is constructed by Procedure A. Hence, in order to generate $V(P_{k+2})$ at iteration $k+1$ of Algorithm IOA in the same way, it is necessary to construct $\mathcal{V}_{\alpha(k)+1}, \ldots, \mathcal{V}_{\alpha(k+1)}$, and to update $\mathcal{V}_i$ for each $i \in \{i \in \Gamma_k : \ell_k(\boldsymbol{v}(i), t(i)) = 0\}$. Then, the following theorem holds.

Theorem 5.3. *Let $P \subset \mathbb{R}^n$ be an n-dimensional convex polyhedral set and let $\{F_i :\in \Gamma\}$ be the set of all facets of P, where Γ is the index set. The line segment* $\mathrm{co}\,\{\boldsymbol{v}', \boldsymbol{v}''\}$ $(\boldsymbol{v}', \boldsymbol{v}'' \in V(P),\ \boldsymbol{v}' \neq \boldsymbol{v}'')$ *is an edge of P if and only if there exist $i_1, \ldots, i_{n-1} \in \Gamma$ such that* $\mathrm{co}\,\{\boldsymbol{v}', \boldsymbol{v}''\} \subset F_{i_j}$ *for each $j = 1, \ldots, n-1$.*

By Procedure C, $\{F_i : i \in \Gamma_{k+1}\}$ is the set all facets of P_{k+1}. Hence, to construct $\mathcal{V}_i$ for each $i \in \mathcal{W}$ ($\mathcal{W}$ is generated by Procedure B), we propose the following procedure.

Procedure D

Step 0: Set $j = 1$ and go to Step 1.
Step 1: If $j = \alpha(k+1)$, then stop. If $j \notin \mathcal{W}$, then go to Step 3. Otherwise, go to Step 2.
Step 2:
Step 2-0: Set $q := j + 1$ and go to Step 2-1.
Step 2-1: If $q = \alpha_{k+1}$, then go to Step 3. If $q \notin \mathcal{W}$, then go to Step 2-3. Otherwise, go to Step 2-2.
Step 2-2: If $|\mathcal{F}_j \cap \mathcal{F}_q| = n - 1$, set $\mathcal{V}_j \leftarrow \mathcal{V}_j \cup \{q\}$ and $\mathcal{V}_q \leftarrow \mathcal{V}_q \cup \{j\}$. Go to Step 2-3.
Step 2-3: Set $q \leftarrow q + 1$ and return to Step 2-1.
Step 3: Set $j \leftarrow j + 1$ and return to Step 1.

By Procedures B and D, all index sets $\mathcal{V}_i$ ($i \in \Delta_{k+1}$) are updated completely.

6. COMPUTATIONAL EXPERIMENTS

In order to investigate the efficiency of Algorithm IOA, we did numerical experiments for the following problem:

$$\begin{cases} \text{minimize} & \left(\dfrac{1}{2}\displaystyle\sum_{i=1}^{n} f_i^1 x_i^2 - \sum_{j=1}^{n} f_j^2 x_j + f\right) - \left(\dfrac{1}{2}\displaystyle\sum_{i=1}^{n} g_i^1 x_i^2 - \sum_{j=1}^{n} g_j^2 x_j + g\right) \\ \text{subject to} & \dfrac{1}{2}\displaystyle\sum_{i=1}^{n} a_i (x_i - b_i)^2 - c \leq 0,\ \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{6.1}$$

where $f_j^i, g_j^i, a_j, b_j, f, g, c \in [1, 10]$ ($i = 1, 2$, $j = 1, \ldots, n$) are defined by a random number generator. For each n satisfying $1 \leq n \leq 8$, we generated 60 problems in the form of (6.1). The numerical experiments were performed by a computer (CPU: Xeron MP 3.33GHz–8MB×3, RAM: 6GB). Algorithm IOA was encoded by the C language on Linux. The computational result of Algorithms IOA for such problems are written in Table 1, where the tolerance is $\tau = 0.001$. Here, $|V(P_{\text{last}})|$ in Table 1 denotes the number of vertices of P_k at the last iteration.

TABLE 1. Computational results of Algorithm IOA

n	Number of iteration		CPU-time(sec)		$\lvert V(P_{\text{last}})\rvert$	
	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
1	3.379	1.495	0.009	0.032	6.379	1,473
2	15.917	5.460	0.000	0.001	34.933	10.535
3	50.950	87.218	0.002	0.011	256.017	511.184
4	68.617	26.764	0.009	0.009	907.633	448.863
5	151.879	98.777	0.351	0.608	7166.828	6302.436
6	227.076	159.698	10.464	26.017	40650.830	41346.200
7	318.729	172.307	586.560	1005.900	233899.100	186763.300
8	243.350	75.510	4626.178	4006.415	540451.800	264155.460

The computational result shows that the proposed algorithm is effective for not so large size problems.

7. CONCLUSIONS

In this paper, we have improved Tuy's outer approximation algorithm for solving (DC). Both Tuy's algorithm and the improved algorithm generate the sequences of polytopes to guarantee the global convergence. The sequence of polytopes generated by Tuy's algorithm approximates the sequence of compact convex sets. On the other hands, the improved algorithm generates the sequence of polytopes approximating a compact convex set from outside. By fixing the target set approximated by the sequence of polytopes, the efficiency of the proposed algorithm has been upgraded. Moreover, to improve the computational efficiency of the algorithm, we have proposed a procedure for calculating the vertex sets. By utilizing the relation of connections among vertices by edges, the proposed procedure calculates the vertex sets without solving systems of linear equations.

REFERENCES

1. M. Avriel and A.C. Williams, Complementary Geometric Programming, SIAM J. Appl. Math. 19 (1970) 125–141.
2. R.J. Hillestad and S.E. Jacobsen, Reverse Convex Programming, Appl. Math. Optim. 6 (1980) 63–78.
3. R. Meyer, The Validity of a Family of Optimization Methods, SIAM J. Control 8 (1970) 42–54.
4. R.T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, N.J., 1970.
5. J.B. Rosen, Iterative Solution of Nonlinear Optimal Control Problems, SIAM J. Control 4 (1966) 223–244.
6. N.V. Thoai, A Modified Version of Tuy's Method for Solving DC Programming, Optim. 19 (1988) 665–674.
7. H. Tuy, Convex Programs with an Additional Reverse Convex Constraint, J. Optim. Theory Appl. 52 (1987) 463–486.
8. H. Tuy, Convex Analysis and Global Optimization, Kluwer Academic Publishers, 1998.

9. H. Tuy, Canonical DC Programming: Outer Approximation Methods Revisited, Oper. Res. Let. 18 (1995) 99-106.
10. H. Tuy, On Global Optimality Conditions and Cutting Plane Algorithms, J. Optim. Theory Appl. 118 (2003) 201–216.
11. U.A. Ueing, A Combinatorial Method to Compute a Global Solution of Certain Nonconvex Optimization Problems, Numerical Methods for Nonlinear Optimization Edited by F.A. Lootsma, Academic Press, New York (1972) 223–230.
12. A.B. Zaleesky, Nonconvexity of Feasible Domains and Optimization of Management Decision (in Russian), Ekonomika i Matematitcheskie Metody 16 (1980) 1069–1081.