Journal of Nonlinear and Convex Analysis
Volume 12, Number 1, 2011, 17–27

Yokohama Publishers
ISSN 1880-5221 ONLINE JOURNAL

OUTER APPROXIMATION METHOD INCORPORATING THE FERRARI'S METHOD FOR SOLVING A QUADRATIC DC PROGRAMMING PROBLEM

SYUUJI YAMADA, TAMAKI TANAKA, AND TETSUZO TANINO

Abstract. In this paper, we propose two kinds of successive approximation methods for solving a quadratic dc programming problem. One is the algorithm based on an outer approximation method. The algorithm calculates an approximate solution by constructing two cutting planes at each iteration. The second is the outer approximation algorithm incorporating the Ferrari's method. By utilizing the Ferrari's method, provisional solutions are updated effectively.

1. Introduction

To obtain an approximate solution to a global optimization problem, outer approximation methods have been proposed by Cheney and Goldstein [1], Veinot [6], Falk and Hoffman [2], Thieu, Tam and Ban [5], Horst and Tuy [3] and Horst, Thiau and Vries [4]. However, the algorithm does not have a guarantee the finding of an approximate feasible solution. Moreover, at many iterations, the provisional solution might not improve the performance index. This means that an appropriate suboptimal solution might not be found at many iterations of the existing algorithm.

In this paper, we consider an optimization problem with a linear function to be minimized over a set defined by two quadratic functions (one is strictly convex and the other is strictly concave). This problem is called the quadratic dc programming problem and it is known that such problems can be transformed into Problem (DC) written in Section 2. We propose an outer approximation method for Problem (DC). The algorithm utilizes a sequence of polytopes. It is shown that every accumulation point of the sequence of the provisional solutions is an optimal solution of Problem (DC). Moreover, in order to update the provisional solutions effectively, we consider a subproblem of Problem (DC) at each iteration of the algorithm. The subproblem can be solved by calculating all solutions of a biquadratic equation. Therefore, we incorporate the Ferrari's method in the algorithm.

The organization of this paper is as follows: In Section 2, we explain Problem (DC). In Section 3, we formulate an outer approximation algorithm for Problem (DC) and establish the convergence of the algorithm. The difference of the objective function value between an approximate solution calculated by the proposed algorithm and the exact optimal solution is smaller than a tolerance selected at the initialization step. In Section 4, we incorporate the Ferrari's method in the algorithm proposed in Section 3. By incorporating the Ferrari's method, the procedure bounding the feasible set will be expedited.

2010 *Mathematics Subject Classification.* Primary 90C26; Secondary 90C59.

Key words and phrases. Outer approximation method, reverse convex constraints, canonical d.c. programming.

Throughout this paper, we use the following notation: $\mathbb{R}$ and $\mathbb{C}$ denote the sets of all real and complex numbers, respectively. For a subset $X \subset \mathbb{R}^n$, int X, bd X, cl X and co X denote the interior, the boundary, the closure and the convex hull of X, respectively. 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 convex polyhedral set (or polytope) $X \subset \mathbb{R}^n$, $V(X)$ denotes the set of all vertices of X. For a subset $X \subset \mathbb{R}^n$, $X^\circ := \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{y}^\top \boldsymbol{x} \leq 1,\ \forall \boldsymbol{x} \in X\}$ is called the polar set of X. Given a function $f : \mathbb{R} \to \mathbb{R}$, $f'(a)$ denotes the derivative of f at $a \in \mathbb{R}$. Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, $\partial f(\boldsymbol{x})$ denotes the subdifferential of f at $\boldsymbol{x}$, that is, $\partial f(\boldsymbol{x}) := \{\boldsymbol{u} \in \mathbb{R}^n : f(\boldsymbol{y}) \geq \langle \boldsymbol{u}, \boldsymbol{y} - \boldsymbol{x}\rangle + f(\boldsymbol{x}),\ \boldsymbol{y} \in \mathbb{R}^n\}$. Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f(\boldsymbol{x})$ denotes the gradient vector of f at $\boldsymbol{x}$. Given a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla^2 f(\boldsymbol{x})$ denotes the Hessian matrix of f at $\boldsymbol{x}$. Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $\alpha \in \mathbb{R}$, $L_f(\alpha)$ denotes the level set of f corresponding to α, that is, $L_f(\alpha) := \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \leq \alpha\}$. For $\varepsilon > 0$ ($\varepsilon \in \mathbb{R}$) and $\boldsymbol{x} \in \mathbb{R}^n$, $B(\boldsymbol{x}, \varepsilon) := \{\boldsymbol{y} \in \mathbb{R}^n : \|\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon\}$. Let I be the unit matrix on $\mathbb{R}^{n\times n}$. Given a matrix $A \in \mathbb{R}^{m\times n}$, $A^\top$ and A^{-1} denote the transposed matrix and the inverse matrix of A, respectively. Let $\boldsymbol{e}^k := (e^k_1, \ldots, e^k_n)^\top \in \mathbb{R}^n$ ($k \leq n$) be a vector satisfying $e^k_k = 1$ and $e^k_j = 0$ for each $j \neq k$.

2. Quadratic dc programming problem

Let us consider a quadratic dc programming problem as follows:

$$
(\mathrm{DC}) \begin{cases} \text{minimize} & f(\boldsymbol{x}) := (\boldsymbol{e}^n)^\top \boldsymbol{x} = x_n \\ \text{subject to} & g(\boldsymbol{x}) := \dfrac{1}{2}\boldsymbol{x}^\top P \boldsymbol{x} - (\boldsymbol{e}^n)^\top \boldsymbol{x} = \dfrac{1}{2}\boldsymbol{x}^\top P \boldsymbol{x} - x_n \leq 0, \\ & h(\boldsymbol{x}) := \dfrac{1}{2}(\boldsymbol{x} - \boldsymbol{q})^\top (\boldsymbol{x} - \boldsymbol{q}) - r \geq 0, \\ & \boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n, \end{cases}
$$

where $\boldsymbol{c}$ is a vector in $\mathbb{R}^n$ satisfying $\|\boldsymbol{c}\| = 1$, $P = (p_{ij}) \in \mathbb{R}^{n\times n}$ is a positive-definite matrix, $\boldsymbol{q} \in \mathbb{R}^n$, and $r \in \mathbb{R}$ satisfies $r > \dfrac{1}{2}\boldsymbol{q}^\top \boldsymbol{q}$. Then, $g, h : \mathbb{R}^n \to \mathbb{R}$ are strictly convex functions. Let $Y := \{\boldsymbol{x} \in \mathbb{R}^n : g(\boldsymbol{x}) \leq 0\}$ and $X := \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \leq 0\}$. Since $g\left(\dfrac{1}{p_{nn}}\boldsymbol{e}^n\right) = -\dfrac{1}{2p_{nn}} < 0$ and $h(\boldsymbol{0}) = \dfrac{1}{2}\boldsymbol{q}^\top \boldsymbol{q} - r < 0$, int $Y = \{\boldsymbol{x} \in \mathbb{R}^n : g(\boldsymbol{x}) < 0\} \neq \emptyset$ and int $X = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) < 0\} \neq \emptyset$, and hence $Y \backslash \mathrm{int}\ X$ is the feasible set of Problem (DC). From the strict convexities of g and h, Y and X are compact convex sets. Through this paper, we assume the following condition for Problem (DC).

(A) $Y \backslash \mathrm{int}\ X = \mathrm{cl}\ ((\mathrm{int}\ Y) \backslash X) \neq \emptyset$.

By Assumption (A) and the definitions of g and h, $Y \backslash \mathrm{int}\ X$ is nonempty and compact. Hence, Problem (DC) has a globally optimal solution. We note that $\arg\min\{f(\boldsymbol{x}) : \boldsymbol{x} \in Y\} = \{\boldsymbol{0}\}$. However, $\boldsymbol{0} \in \mathrm{int}\ X$, that is, $\boldsymbol{0}$ is not a feasible solution of Problem (DC). Denote by min(DC) the optimal value of Problem (DC). We note that min(DC) > 0. Moreover, the following lemma holds.

Lemma 2.1. (Lemma 2.1, [7]) *Let $\bar{\boldsymbol{x}}$ be a globally optimal solution of Problem* (DC). *Then, $\bar{\boldsymbol{x}} \in Y \cap (\mathrm{bd}\ X)$.*

In the case of $n \geq 2$, it follows from the following proposition that $\min(\mathrm{DC}) = \min\{f(\boldsymbol{x}) : \boldsymbol{x} \in (\mathrm{bd}\ Y) \cap (\mathrm{bd}\ X)\}$.

Proposition 2.1. (Proposition 2.1, [7]) *Let $n \geq 2$. Then, the globally optimal solution $\bar{\boldsymbol{x}}$ of Problem* (DC) *satisfies the relation $\bar{\boldsymbol{x}} \in (\mathrm{bd}\ Y) \cap (\mathrm{bd}\ X)$.*

Remark 2.1. In the case of $n = 1$, let us consider the following dc programming problem:

$$\begin{cases} \text{mininimize} & x \\ \text{subject to} & x \in Y \backslash \mathrm{int}\ X, \end{cases}$$

where $Y = [0, 3]$ and $X = [-1, 1]$. Then, $Y \backslash \mathrm{int}\ X = [1, 3]$. We note that $\{1\} = \arg\min\{x : x \in Y \backslash \mathrm{int}\ X\}$, but $\arg\min\{x : x \in Y \backslash \mathrm{int}\ X\} \cap \mathrm{bd}\ Y = \emptyset$.

3. Outer approximation method

Let $\phi(\boldsymbol{x}) := \max\{g(\boldsymbol{x}), h(\boldsymbol{x})\}$. Then, $\phi : \mathbb{R}^n \to \mathbb{R}$ is a strictly convex function and $Y \cap X = \{\boldsymbol{x} \in \mathbb{R}^n : \phi(\boldsymbol{x}) \leq 0\}$. In this section, in the case of $n \geq 2$, we propose an algorithm based on the outer approximation method presented by Yamada, Tanaka and Tanino [7] for solving Problem (DC) as follows:

Algorithm OA

Step 0: Select a tolerance $\alpha \geq 0$. Set

$$\boldsymbol{x}^1 := r\boldsymbol{e}^1, \tag{3.1}$$

$$\boldsymbol{a}^1 := \min\left\{\frac{1}{p_{n\,n}}, q_n\right\}\boldsymbol{e}^n, \tag{3.2}$$

$$T_1 := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{r}_- \leq \boldsymbol{x} \leq \boldsymbol{r}_+\}, \tag{3.3}$$

$$S_1 := \bar{S}_1 := T_1, \tag{3.4}$$

where $\boldsymbol{r}_- := (q_1 - r, \ldots, q_{n-1} - r, 0)^\top \in \mathbb{R}^n$ and $\boldsymbol{r}_+ := (q_1 + r, \ldots, q_n + r)^\top \in \mathbb{R}^n$. Compute the vertex set $V(S_1)(= V(\bar{S}_1))$. Choose $\boldsymbol{v}^1 \in \arg\max\{\phi(\boldsymbol{v}) : \boldsymbol{v} \in V(S_1)\}$, $\hat{\boldsymbol{v}}^1 = \bar{\boldsymbol{v}}^1 \in \arg\max\{h(\boldsymbol{v}) : \boldsymbol{v} \in V(S_1)\}$ and $\boldsymbol{y}^1 \in [\boldsymbol{a}^1, \boldsymbol{v}^1] \cap \mathrm{bd}\ (Y \cap X)$. Set $k \leftarrow 1$ and go to Step 1.

Step 1: If at least one of the following criteria holds, then stop.

(SC1) $\boldsymbol{y}^k \in Y \backslash \mathrm{int}\ X$ and $y^k_n \leq \alpha$,

(SC2) $\boldsymbol{x}^k \in Y \backslash \mathrm{int}\ X$ and $\hat{\boldsymbol{v}}^k \in X$ (that is, $S_k \subset X$),

(SC3) $\boldsymbol{x}^k \in Y \backslash \mathrm{int}\ X$ and $\bar{\boldsymbol{v}}^k \in X$ (that is, $\bar{S}_k \subset X$).

Otherwise, go to Step 2.

Step 2: Set

$$\boldsymbol{x}^{k+1} := \begin{cases} \boldsymbol{y}^k & \text{if } \boldsymbol{y}^k \in Y \backslash \mathrm{int}\ X, \\ \boldsymbol{x}^k & \text{otherwise}, \end{cases} \tag{3.5}$$

(3.6) $$\boldsymbol{a}^{k+1} := \begin{cases} \dfrac{y_n^k - \alpha}{2a_n^k}\boldsymbol{a}^k & \text{if } a_n^k \geq y_n^k \text{ and } \boldsymbol{y}^k \in Y \backslash \text{int } X, \\ \boldsymbol{a}^k & \text{otherwise.} \end{cases}$$

Generate T_{k+1}, S_{k+1} and $\bar{S}_{k+1}$ as follows:

(3.7) $$T_{k+1} := T_k \cap \{\boldsymbol{x} \in \mathbb{R}^n : \ell(\boldsymbol{x}, \boldsymbol{v}^k) \leq 0\},$$

(3.8) $$S_{k+1} := T_{k+1} \cap L_f\left(x_n^{k+1} - \frac{\alpha}{2}\right),$$

(3.9) $$\bar{S}_{k+1} := T_{k+1} \cap L_f(x_n^{k+1} - \alpha),$$

where

(3.10) $$\ell(\boldsymbol{x}, \boldsymbol{v}^k) := (\boldsymbol{u}^k)^\top(\boldsymbol{x} - \boldsymbol{v}^k) + \phi(\boldsymbol{v}^k),$$

$$\boldsymbol{u}^k := \begin{cases} \nabla g(\boldsymbol{v}^k) & \text{if } g(\boldsymbol{v}^k) > h(\boldsymbol{v}^k), \\ \nabla h(\boldsymbol{v}^k) & \text{otherwise.} \end{cases}$$

Compute the vertex sets $V(S_{k+1})$ and $V(\bar{S}_{k+1})$. Choose

(3.11) $$\boldsymbol{v}^{k+1} \in \arg\max\{\phi(\boldsymbol{v}) : \boldsymbol{v} \in V(S_{k+1})\},$$

(3.12) $$\hat{\boldsymbol{v}}^{k+1} \in \arg\max\{h(\boldsymbol{v}) : \boldsymbol{v} \in V(S_{k+1})\},$$

$$\bar{\boldsymbol{v}}^{k+1} \in \arg\max\{h(\boldsymbol{v}) : \boldsymbol{v} \in V(\bar{S}_{k+1})\},$$

(3.13) $$\boldsymbol{y}^{k+1} \in [\boldsymbol{a}^{k+1}, \boldsymbol{v}^{k+1}] \cap \text{bd } (Y \cap X).$$

Set $k \leftarrow k + 1$ and return to Step 1.

The following lemmas hold.

Lemma 3.1. *At each iteration k of Algorithm* OA, $\boldsymbol{a}^k \in \text{int } (Y \cap X)$.

Proof. Firstly, we shall show that $\boldsymbol{a}^1 \in \text{int } (Y \cap X)$. Since $g\left(\dfrac{1}{p_{n\,n}}\boldsymbol{e}^n\right) = -\dfrac{1}{2p_{n\,n}} < 0$ and $g(\boldsymbol{0}) = 0$, $\boldsymbol{0} \in Y$ and $\dfrac{1}{p_{n\,n}}\boldsymbol{e}^n \in \text{int } Y$. Hence, from the convexity of Y, $\mu\boldsymbol{e}^n \in \text{int } Y$ for each $\mu \in \left]0, \dfrac{1}{p_{n\,n}}\right]$. Moreover, since $h(\boldsymbol{0}) = \dfrac{1}{2}\boldsymbol{q}^\top\boldsymbol{q} - r < 0$ and $h(q_n\boldsymbol{e}^n) = \dfrac{1}{2}(\boldsymbol{q}^\top\boldsymbol{q} - q_n^2) - r < 0$, we have $\boldsymbol{0}, q_n\boldsymbol{e}^n \in \text{int } X$. Therefore, from the convexity of X, $\mu\boldsymbol{e}^n \in \text{int } X$ for each $\mu \in [0, q_n]$. From the definition of $\boldsymbol{a}^1$ in (3.2), we have $\boldsymbol{a}^1 \in \text{int } (Y \cap X)$.

Secondly, we suppose that $\boldsymbol{a}^k \in \text{int } (Y \cap X)$ and that Algorithm OA does not terminate at iteration k. We note that $\boldsymbol{0} \in Y \cap X$. In the case where $a_n^k \geq y_n^k$ and $\boldsymbol{y}^k \in Y \backslash \text{int } X$, since Criteria (SC1) does not hold, $y_n^k - \alpha > 0$. Moreover, since $y_n^k \leq a_n^k$ and $\alpha \geq 0$, $\dfrac{y_n^k - \alpha}{2a_n^k} \in]0, 1[$, that is, it follows from the definition of $\boldsymbol{a}^{k+1}$ in (3.6) that $\boldsymbol{a}^{k+1} \in \text{int } (Y \cap X)$. In the other case, $\boldsymbol{a}^{k+1} = \boldsymbol{a}^k \in \text{int } (Y \cap X)$.

Consequently, $\boldsymbol{a}^k \in \text{int } (Y \cap X)$ at each iteration k of Algorithm OA. □

Lemma 3.2. *At each iteration k of Algorithm* OA, $Y \cap X \subset T_k$.

Proof. Firstly, we shall show that $Y \cap X \subset T_1$. Let $\hat{\boldsymbol{r}}_- := (q_1 - r, \dots, q_n - r)^\top \in \mathbb{R}^n$. Then from the definition of X, $Y \cap X \subset X \subset \{\boldsymbol{x} \in \mathbb{R}^n : \hat{\boldsymbol{r}}_- \le \boldsymbol{x} \le \boldsymbol{r}_+\}$. Since $g(\boldsymbol{0}) = 0$ and $\nabla g(\boldsymbol{0}) = -\boldsymbol{e}^n$, $Y \cap X \subset Y \subset \{\boldsymbol{x} \in \mathbb{R}^n : x_n \ge 0\}$. Hence, by the definition of T_1 in (3.3), $Y \cap X \subset \{\boldsymbol{x} \in \mathbb{R}^n : \hat{\boldsymbol{r}}_- \le \boldsymbol{x} \le \boldsymbol{r}_+,\ x_n \ge 0\} = T_1$.

Secondly, we suppose that $Y \cap X \subset T_k$ and that Algorithm OA does not terminate at iteration k. From the definition of $\boldsymbol{u}^k$, $\boldsymbol{u}^k \in \partial\phi(\boldsymbol{v}^k)$. Hence, by the convexity of ϕ, $Y \cap X \subset \{\boldsymbol{x} \in \mathbb{R}^n : \ell(\boldsymbol{x}, \boldsymbol{v}^k) \le 0\}$. Therefore, by the definition of T_{k+1} in (3.7) $Y \cap X \subset \{\boldsymbol{x} \in \mathbb{R}^n : \ell(\boldsymbol{x}, \boldsymbol{v}^k) \le 0\} \cap T_k = T_{k+1}$.

Consequently, $Y \cap X \subset T_k$ at each iteration k of Algorithm OA. □

By (3.3), (3.4), (3.7) and (3.8), T_k and S_k are polytopes for each k. From Lemma 3.2, $T_k \supset Y \cap X = \{\boldsymbol{x} \in \mathbb{R}^n : \phi(\boldsymbol{x}) \le 0\}$ for each k. By the definition of $\boldsymbol{v}^k$ in (3.11) and the strict convexity of ϕ, $\boldsymbol{v}^k \in \text{bd } T_k$ and hence $\phi(\boldsymbol{v}^k) > 0$. Therefore, the definition of ℓ in (3.10), we have $\ell(\boldsymbol{v}^k, \boldsymbol{v}^k) > 0$ for each k. This implies that for each k,

$$\boldsymbol{v}^k \in T_k,\ \boldsymbol{v}^k \notin T_{k+1} \text{ and hence } T_k \supsetneq T_{k+1}.$$

Under Assumption (A), it is shown that a provisional solution $\boldsymbol{y}^{k'}$ belongs to the feasible set of Problem (DC) for some $k' > 0$ (see Lemma 3.4 in [7]). Then, we note that $\boldsymbol{x}^{k'+1} = \boldsymbol{y}^{k'}$. From the definitions of S_k and y^k in (3.8) and (3.13)

$$\boldsymbol{y}^k \in S_k \subset L_f(x_n^k) \subset L_f(x_n^{k'+1}) \quad \text{for each } k > k' + 1.$$

Moreover, in the case where $\boldsymbol{y}^{k''} \in Y \backslash \text{int } X$ for some $k'' > k'$, since $\boldsymbol{x}^{k''+1} = \boldsymbol{y}^{k''}$ and $\boldsymbol{a}^{k''} \in \text{int } (Y \cap X)$, we have

$$\min(\text{DC}) \le x_n^{k''+1} < x_n^{k'+1} - \frac{\alpha}{2}.$$

Therefore, Algorithm OA calculates an approximate solution of Problem (DC) by the following operations:

Outer Approximation: The polytope sequence $\{T_k\}$ approximates $Y \cap X$ from its outside.

Bound Procedure: The cutting plane sequence $\{L_f(x_n^k)\}$ confines the feasible area of searching for an approximate solution of Problem (DC).

From the definitions of S_k, $\boldsymbol{v}^k$ and $\hat{\boldsymbol{v}}^k$ in (3.8), (3.11) and (3.12), we notice that

$$\begin{aligned}
&S_1 \supset S_2 \supset \cdots \supset S_k \supset \cdots,\\
&\phi(\boldsymbol{v}^1) \ge \phi(\boldsymbol{v}^2) \ge \cdots \ge \phi(\boldsymbol{v}^k) \ge \cdots \ge 0,\\
&h(\hat{\boldsymbol{v}}^1) \ge h(\hat{\boldsymbol{v}}^2) \ge \cdots \ge h(\hat{\boldsymbol{v}}^k) \ge \cdots,\\
&\phi(\boldsymbol{v}^k) \ge h(\hat{\boldsymbol{v}}^k) \text{ for each } k.
\end{aligned}$$

Moreover, in order to terminate within a finite number of iterations, $\bar{S}_k$ and $\bar{\boldsymbol{v}}^k$ are generated at every iterations. For each k, we have

$$\bar{\boldsymbol{v}}^k \in \bar{S}_k \subset S_k, \text{ and } 0 \le h(\bar{\boldsymbol{v}}^k) \le h(\hat{\boldsymbol{v}}^k).$$

Remark 3.1. Assume that Algorithm OA does not generate $\bar{S}_k$ at each iteration. Let $n \ge 3$. Moreover, by Proposition 2.1, let $\boldsymbol{x}^* \in (\text{bd } Y) \cap (\text{bd } X)$ such that $\boldsymbol{x}^*$ is a globally optimal solution of Problem (DC). Suppose that $x_n^k - \dfrac{\alpha}{2} = \min(\text{DC})$.

Then, $x_n^* = x_n^k - \dfrac{\alpha}{2}$. From the definition of $\boldsymbol{y}^k$ in (3.13), $\boldsymbol{y}^{k'} \notin Y \backslash \mathrm{int}\, X$ for each $k' > k$. Hence, $\boldsymbol{x}^* \in S_{k'}$ for each $k' > k$. Since $n \geq 3$ and g, h are differentiable, x^* is not a vertex of $S_{k'}$ for each $k' > k$. Moreover, since g and h are strictly convex functions, for each $k' > k$, there exists $\hat{\boldsymbol{x}}$ contained in a neighborhood of $\boldsymbol{x}^*$ satisfying $\hat{\boldsymbol{x}} \notin X$ and $\hat{\boldsymbol{x}} \in S_{k'}$. Therefore, in this case, such an algorithm does not terminate within a finite number of iterations.

Since $\boldsymbol{x}^{k'+1} \in Y \backslash \mathrm{int}\, X$ for some k', from the definition of $\boldsymbol{x}^k$ in (3.5), the provisional solution $\boldsymbol{x}^k$ is a feasible solution of Problem (DC) for each $k > k'$. Moreover,

$$y_n^{k'} = x_n^{k'+1} \geq x_n^{k'+1} \geq x_n^{k'+2} \geq \cdots \geq \min(\mathrm{DC}).$$

Furthermore, in the case of $\alpha = 0$, it is shown that every accumulation point of $\{\boldsymbol{x}^k\}$ is a globally optimal solution of Problem (DC) (see Theorem 3.1 in [7]).

In the case of $\alpha > 0$, we can show that Algorithm OA terminates within a finite number of iterations and that the following assertions hold (see Section 3.3 in [7]).

- If Criterion (SC1) holds, $y_n^k \leq \min(\mathrm{DC}) + \alpha$.
- If Criterion (SC2) holds, $x_n^k \leq \min(\mathrm{DC}) + \dfrac{\alpha}{2}$.
- If Criterion (SC3) holds, $x_n^k \leq \min(\mathrm{DC}) + \alpha$.

4. Outer approximation method incorporating the Ferrari method

At iteration k of Algorithm OA proposed in Section 3, the provisional solution $\boldsymbol{x}^k$ is updated if $\boldsymbol{y}^k$ is a feasible solution of Problem (DC). However, $\boldsymbol{y}^k$ might not be contained in the feasible set at many iterations. Moreover, $\boldsymbol{y}^k$ is not often a locally optimal solution of Problem (DC). Therefore, in order to update the provisional solution $\boldsymbol{x}^k$ effectively, we consider the following subproblem of Problem (DC):

$$(4.1) \qquad \begin{cases} \text{minimize} & \mu_2 \\ \text{subject to} & g(\mu_1 \hat{\boldsymbol{y}}^k + \mu_2 \boldsymbol{e}^n + \boldsymbol{q}) \leq 0, \\ & h(\mu_1 \hat{\boldsymbol{y}}^k + \mu_2 \boldsymbol{e}^n + \boldsymbol{q}) \geq 0, \\ & \mu_1,\, u_2 \in \mathbb{R}, \end{cases}$$

where $\hat{\boldsymbol{y}}^k := \dfrac{1}{\|\boldsymbol{y}^k - y_n^k \boldsymbol{e}^n\|}(\boldsymbol{y}^k - y_n^k \boldsymbol{e}^n)$ and $\boldsymbol{y}^k$ is obtained at iteration $k-1$ of Algorithm OA. By the definition of T_1 in (3.3), $\boldsymbol{y}^k \notin \{\boldsymbol{x} \in \mathbb{R}^n : x = \mu \boldsymbol{e}^n,\ \mu \in \mathbb{R}\}$ for each k. Hence, the dimension number of Problem (4.1) equals two. Let $(\bar{\mu}_1, \bar{\mu}_2)$ be a globally optimal solution of Problem (4.1). Then, $\bar{\mu}_1 \hat{\boldsymbol{y}}^k + \bar{\mu}_2 \boldsymbol{e}^n$ is a feasible solution of Problem (DC). In the case of $\boldsymbol{y}^k \in Y \backslash \mathrm{int}\, X$, $f(\bar{\mu}_1 \hat{\boldsymbol{y}}^k + \bar{\mu}_2 \boldsymbol{e}^n) = \bar{\mu}_2 \leq y_n^k = f(\boldsymbol{y}^k)$. However, we notice that the local optimality of $\bar{\mu}_1 \hat{\boldsymbol{y}}^k + \bar{\mu}_2 \boldsymbol{e}^n$ for Problem (DC) is not guaranteed. Moreover, in the case of $\boldsymbol{y}^k \notin Y \backslash \mathrm{int}\, X$, the feasible set of Problem (4.1) might be empty.

By Proposition 2.1, Problem (4.1) is equivalent to the following problem:

$$(4.2) \qquad \begin{cases} \text{minimize} & \mu_2 \\ \text{subject to} & g(\mu_1 \hat{\boldsymbol{y}}^k + \mu_2 \boldsymbol{e}^n + \boldsymbol{q}) = 0, \\ & h(\mu_1 \hat{\boldsymbol{y}}^k + \mu_2 \boldsymbol{e}^n + \boldsymbol{q}) = 0, \\ & \mu_1,\, u_2 \in \mathbb{R}. \end{cases}$$

We note that the number of the feasible solutions of Problem (4.2) is less than or equal to four, because the feasible set of Problem (4.2) can be reformulated by the intersection of a circle and an ellipsoid on $\mathbb{R}^n$. Hence, Problem (4.2) can be solved by calculating all points satisfying the following two equations:

$$A_1\mu_1^2 + A_2\mu_2^2 + A_3\mu_1\mu_2 + A_4\mu_1 + A_5\mu_2 + A_6 = 0, \tag{4.3}$$

$$\frac{1}{2}(\mu_1^2 + \mu_2^2) - r = 0, \tag{4.4}$$

where

$$\begin{aligned} A_1 &:= \frac{1}{2}(\hat{\boldsymbol{y}}^k)^\top P\hat{\boldsymbol{y}}^k, \\ A_2 &:= \frac{1}{2}p_{n\,n}, \\ A_3 &:= (\hat{\boldsymbol{y}}^k)^\top P\boldsymbol{e}^n, \\ A_4 &:= \boldsymbol{q}^\top P\hat{\boldsymbol{y}}^k, \\ A_5 &:= \boldsymbol{q}^\top P\boldsymbol{e}^n - 1, \\ A_6 &:= \frac{1}{2}\boldsymbol{q}^\top P\boldsymbol{q} - q_n. \end{aligned}$$

By (4.3) and (4.4), we have the following equation:

$$A_1\mu_1^2 + A_2(2r - \mu_1^2) + A_4\mu_1 + A_6 = \pm(A_3\mu_1 + A_5)\sqrt{2r - \mu_1^2}. \tag{4.5}$$

Moreover, by squaring both sides of (4.5), we get the following biquadratic equation:

$$B_4\mu_1^4 + B_3\mu_1^3 + B_2\mu_1^2 + B_1\mu_1 + B_0 = 0, \tag{4.6}$$

where

$$\begin{aligned} B_4 &:= (A_1 - A_2)^2 + A_3^2, \\ B_3 &:= 2((A_1 - A_2)A_4 + A_3A_5), \\ B_2 &:= 2(A_1 - A_2)(A_6 - 2A_2r) + A_5^2 - 2A_3^2r, \\ B_1 &:= 2(A_4(A_6 - 2A_2r) - 2A_3A_5r), \\ B_0 &:= (A_6 - 2A_2r)^2 - 2A_5^2r. \end{aligned}$$

By using the Ferrari's method written in Appendix, we can obtain all solutions of (4.6). Let $\mu(1)_1, \dots, \mu(4)_1$ be all solutions of (4.6). Then, it is not always true that all solutions are real numbers. If all solutions are not real numbers, then the feasible set of Problem (4.1) is empty. Hence, for each $j \in \{j : \mu(j)_1 \in \mathbb{R},\ j = 1, \dots, 4\}$, we calculate $\mu(j)_2$ such that $(\mu(j)_1, \mu(j)_2)$ satisfies the equations (4.3) and (4.4). Moreover, we chose $j' \in \{j : \mu(j)_1 \in \mathbb{R},\ j = 1, \dots, 4\}$ satisfying $\mu(j')_2 = \min\{\mu(j)_2 : \mu(j)_1 \in \mathbb{R},\ j = 1, \dots, 4\}$. Then, $(\mu(j')_1, \mu(j')_2)$ is a globally optimal solution of Problem (4.1) and $\mu(j')_1\hat{\boldsymbol{y}}^k + \mu(j')_2\boldsymbol{e}^n$ is a feasible solution of Problem (DC). Moreover, in the case of $\boldsymbol{y}^k \in Y\backslash \mathrm{int}\ X$, we have $\mu(j')_2 \le y_n^k$. Therefore, at Step 2 of Algorithm OA, we update the provisional solution $\boldsymbol{x}^k$ as follows:

$$\boldsymbol{x}^{k+1} := \begin{cases} \mu(j')_1\hat{\boldsymbol{y}}^k + \mu(j')_2\boldsymbol{e}^n & \text{if } \{i : \mu(j)_1 \in \mathbb{R},\ j = 1, \dots, 4\} \neq \emptyset, \\ \boldsymbol{x}^k & \text{otherwise.} \end{cases}$$

5. Conclusions

In this paper, we have presented two outer approximation algorithms for Problem (DC).

A characteristic of the first proposed algorithm is to generate two cutting planes when a provisional solution is updated. By generating two cutting planes, the algorithm terminates within a finite number of iterations. Furthermore, by setting a tolerance at the initialization step, we can establish the upper limit of a difference of the objective function value between an approximate solution calculated by the algorithm and the exact optimal solution.

The second suggestion is to introduce the Ferrari's method into the first proposed algorithm. By using a descent method, the provisional solution at each iteration is updated effectively.

Appendix: the Ferrari's method

In the case of $B_4 \neq 0$ in (4.6), all solutions of (4.6) can be obtained by using Procedure 1. In the other case, we can calculate all solutions of (4.6) by using Procedure 2.

Procedure 1 (the Ferrari's method)
Objective: To calculate all solutions of the following biquadratic equation.

$$x^4 + C_3x^3 + C_2x^2 + C_1x + C_0 = 0, \tag{5.1}$$

where x is a variable on $\mathbb{C}$ and $C_0, \ldots, C_3 \in \mathbb{R}$.

Step 1: Equation (5.1) is reformulated as follows:

$$y^4 + D_2y^2 + D_1y + D_0 = 0, \tag{5.2}$$

where

$$\begin{aligned} y &:= x + \frac{C_3}{4}, \\ D_2 &:= C_2 - \frac{3C_3^2}{8}, \\ D_1 &:= C_1 + \frac{C_3^3}{8} - \frac{C_3C_2}{2}, \\ D_0 &:= C_0 - \frac{3C_3^4}{256} + \frac{C_3^2C_2}{16} - \frac{C_3C_1}{4}. \end{aligned}$$

Go to Step 2.

Step 2: By adding a variable λ on $\mathbb{C}$, (5.2) is transformed as follows:

$$\begin{aligned} & y^4 + D_2y^2 + D_1y + D_0 \\ &= y^4 + 2\lambda y^2 + \lambda^2 - 2\lambda y^2 - \lambda^2 + D_2y^2 + D_1y + D_0 \\ &= (y^2 - \lambda)^2 + (D_2 - 2\lambda)y^2 + D_1y + D_0 - \lambda^2 \\ &= (y^2 - \lambda)^2 + (D_2 - 2\lambda)\left(y + \frac{D_1}{2(D_2 - 2\lambda)}\right)^2 \\ &\quad - \frac{D_1^2}{4(D_2 - 2\lambda)} + D_0 - \lambda^2 = 0. \end{aligned} \tag{5.3}$$

Go to Step 3.

Step 3: Let us consider the following equation:

$$-\frac{D_1^2}{4(D_2-2\lambda)}+D_0-\lambda^2=0. \tag{5.4}$$

Equation (5.4) is reformulated as follows:

$$\lambda^3-\frac{D_2}{2}\lambda^2-D_0\lambda+\frac{D_2D_0}{2}-\frac{D_1^2}{8}=0. \tag{5.5}$$

By using Peocedure 2, (5.4) is solvable. Let $\bar{\lambda}$ be a solution of (5.5). Go to Step 4.

Step 4: By replacing λ by $\bar{\lambda}$, the following equation holds.

$$(y^2-\bar{\lambda})^2=E_1(y+E_0)^2, \tag{5.6}$$

where

$$E_1:=-D_2+2\lambda,$$
$$E_0:=\frac{D_1}{2(D_2-2\lambda)}.$$

Hence, we have the following two quadratic equations:

$$y^2-\sqrt{E_1}y-\bar{\lambda}-E_0\sqrt{E_1}=0, \tag{5.7}$$

$$y^2+\sqrt{E_1}y-\bar{\lambda}+E_0\sqrt{E_1}=0. \tag{5.8}$$

Let y_1, y_2 be solutions of equations in (5.7) and y_3, y_4 solutions of (5.8). Then, $y_j-\frac{C_3}{4}$ $(j=1,\dots,4)$ are all solutions of (5.1) and Procedure 1 terminates.

Procedure 2 (the Cardano's method)
Objective: To calculate all solutions of the following cubic equation.

$$x^3+C_2x^2+C_1x+C_0=0, \tag{5.9}$$

where x is a variable on $\mathbb{C}$ and $C_0,\dots,C_2\in\mathbb{R}$.

Step 1: Equation (5.9) is reformulated as follows:

$$y^3+D_1y+D_0=0, \tag{5.10}$$

where

$$y:=x+\frac{C_2}{3},$$
$$D_1:=C_1-\frac{C_2^2}{3},$$
$$D_0:=C_0+\frac{2C_2^3}{27}-\frac{C_2C_1}{3}.$$

Go to Step 2.

Step 2: If $D_0=0$, then $y=0$ and $\pm\sqrt{D_1}$ are all solutions of (5.10) and Procedure 2 terminates. Then $-\frac{C_2}{3}, -\frac{C_2}{3}\pm\sqrt{D_1}$ solve (5.9). Otherwise, go to Step 3.

Step 3: By replacing y with $(u+v)$ where u, v are variables on $\mathbb{C}$, we have the following equation:

$$\begin{aligned}
&y^3 + D_1 y + D_0 \\
&= (u+v)^3 + D_1(u+v) + D_0 \\
&= u^3 + 3uv(u+v) + v^3 + D_1(u+v) + D_0 \\
&= u^3 + v^3 + D_0 + (u+v)(3uv + D_1) \\
&= (u+v)(u^2 - uv + v^2) + D_0 + (u+v)(3uv + D_1) = 0
\end{aligned}$$

Since $D_0 \neq 0$, $u + v \neq 0$. Therefore, we have the following the system of equations:

$$\begin{cases} u^3 + v^3 + D_0 = 0, \\ 3uv + D_1 = 0. \end{cases} \tag{5.11}$$

Moreover, (5.11) is reformulated as follows:

$$\begin{cases} u^3 + v^3 + D_0 = 0, \\ 27u^3v^3 + D_1^3 = 0. \end{cases} \tag{5.12}$$

Let $U := u^3$ and $V := v^3$. Then, (5.12) is transformed as follows:

$$\begin{cases} U + V + D_0 = 0, \\ 27UV + D_1^3 = 0. \end{cases} \tag{5.13}$$

Let $\bar{U} := \dfrac{-D_0 + \sqrt{D_0^2 + \dfrac{4D_1^3}{27}}}{2}$ and $\bar{V} := \dfrac{-D_0 - \sqrt{D_0^2 + \dfrac{4D_1^3}{27}}}{2}$. Then, $(\bar{U}, \bar{V})$ solves (5.13). Moreover, (u_k, v_k) $(k = 1, 2, 3)$ defined as follows solve (5.11).

$$\begin{aligned}
&(u_1, v_1) := (\sqrt[3]{\bar{U}}, \sqrt[3]{\bar{V}}),\ (u_2, v_2) := (\sqrt[3]{\bar{U}}\omega, \sqrt[3]{\bar{V}}\omega^2), \\
&(u_3, v_3) := (\sqrt[3]{\bar{U}}\omega^2, \sqrt[3]{\bar{V}}\omega)
\end{aligned}$$

where $\omega := \dfrac{-1 + \sqrt{3}i}{2}$ and i denotes the imaginary number. Therefore, $y_k := u_k + v_k$ $(k = 1, 2, 3)$ solve (5.10) and hence $x_k := y_k - \dfrac{C_2}{3}$ $(k = 1, 2, 3)$ are all solutions of (5.9). Procedure 2 terminates.

References

[1] E. W. Cheney and A. A. Goldstein, *Newton's method of convex programming and tchebycheff approximation*, Numerische Mathematik **1** (1959), 253–268.

[2] J. E. Falk and K. R. Hoffman, *A successive underestimation method for concave minimization problems*, Mathematics of Operations Research **1** (1976), 251–259.

[3] R. Horst and H. Tuy, *On the convergence of global methods in multiextremal optimization*, Journal of Optimization Theory and Applications **54** (1987), 253–271.

[4] R. Horst, N. V. Thiau and J. De. Vries, *On finding new vertices and redundant constraints in cutting plane algorithms for global optimization*, Operations Research Letters **7** (1988), 85–90.

[5] T. V. Thieu, B. T. Tam and V. T. Ban, *An outer approximation method for globally minimizing a concave function over a compact convex set*, Acta Mathematica Vietnamica **8** (1983), 21–40.

[6] A. F. Veinott, Jr., *The supporting hyperplane method for unimodal programming*, Operations Research **15** (1967), 147–152.

[7] S. Yamada, T. Tanaka and T. Tanino, *Outer approximation method incorporating a quadratic aproximation for a dc programming problem*, Journal of Optimization Theory and Applications **144** (2010), 156–183.

Manuscript received August 19, 2010
revised December 2, 2010

Syuuji Yamada
Department of Information Science and Engineering, Graduate School of Science Technology, Niigata University, 8050 Ikarashi-2nocho, Niigata-City 950-2181, Japan
E-mail address: `yamada@math.sc.niigata-u.ac.jp`

Tamaki Tanaka
Department of Information Science and Engineering, Graduate School of Science Technology, Niigata University, 8050 Ikarashi-2nocho, Niigata-City 950-2181, Japan
E-mail address: `tamaki@math.sc.niigata-u.ac.jp`

Tetsuzo Tanino
Division o Electrical, Electronic and Information Engineering, Graduate School of Engineering, Osaka University, Yamada-Oka 2-1, Suita, Osaka, 565-0871, Japan
E-mail address: `tanino@eei.eng.osaka-u.ac.jp`