Linear Programs with an Additional Separable Concave Constraint

Takahito Kuno*

Jianming Shi

July 9, 2001

ISE-TR-01-181

* Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305–8573, Japan
Phone: +81–298–53–5540, Fax: +81–298–53–5206, E-mail: takahito@is.tsukuba.ac.jp

Supported by Grant–in–Aid for Scientific Research of Japan Society for the Promotion of Sciences, C(2) 13680505.

**School of Management
Science University of Tokyo
Kuki, Saitama 346–8512, japan
E–mail: j.shi@ms.kuki.sut.ac.jp

LINEAR PROGRAMS WITH AN ADDITIONAL SEPARABLE CONCAVE CONSTRAINT

Takahito Kuno*
Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
takahito@is.tsukuba.ac.jp

Jianming Shi
School of Management
Science University of Tokyo
Kuki, Saitama 346-8512, Japan
j.shi@ms.kuki.sut.ac.jp

Abstract In this chapter, we develop two algorithms for globally optimizing a special class of linear programs with an additional concave constraint. We assume that the concave constraint is defined by a separable concave function. Exploiting this special structure, we apply Falk-Soland's branch-and-bound algorithm for concave minimization in both direct and indirect manners. In the direct application, we solve the problem alternating local search and branch-and-bound. In the indirect application, we carry out the bounding operation using a parametric right-hand-side simplex algorithm.

Keywords: Reverse convex program, linear program, additional concave constraint, branch-and-bound algorithm, simplex algorithm.

*Supported by Grant-in-Aid for Scientific Research of Japan Society for the Promotion of Sciences, C(2) 13680505.

1. Introduction

In this chapter, we consider a special class of linear programs with an additional concave constraint

$$\left|\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in F \setminus G, \end{array}\right. \tag{1.1}$$

where $\mathbf{c}$ is an n-vector, $F \subset \mathbb{R}^n$ is a polytope and $G \subset \mathbb{R}^n$ is an open convex set. We assume on (1.1) that G possesses a kind of separability, i.e., G can be represented as follows, by means of a sum of functions $g_j : S \to \mathbb{R}$, $j = 1, \ldots, n$, each of which is concave with respect to x_j:

$$G = \left\{ \mathbf{x} \in S^n \;\middle|\; \sum_{j=1}^{n} g_j(x_j) > 0 \right\}.$$

We call the complement of this set G a *separable reverse convex set*, following separable concave functions of the form $g(\mathbf{x}) = \sum_{j=1}^{n} g_j(x_j)$.

The separable concave function is certainly a special class of concave functions, but involves a wide variety of functions unlike its appearance. In fact, it is an elementary exercise in linear algebra that every concave quadratic function can be reduced to a separable form; and the linear multiplicative function $\prod_{j=1}^{n}(\mathbf{c}_j^{\mathsf{T}}\mathbf{x} + d_j)$ can be transformed into $\sum_{j=1}^{n} \log y_j$ with $y_j = \mathbf{c}_j^{\mathsf{T}}\mathbf{x} + d_j$ for $j = 1, \ldots, n$ [11, 15]. These imply that there is a certain amount of demand for the linear program with an additional separable concave constraint (LPASC), as well as for the separable concave minimization problem:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{n} g_j(x_j) \\ \text{subject to} & \mathbf{x} \in F. \end{array}\right. \tag{1.2}$$

The readers should remark that even this well-known global optimization problem belongs to LPASC, because (1.2) is equivalent to

$$\left|\begin{array}{ll} \text{maximize} & -y \\ \text{subject to} & \mathbf{x} \in F, \quad \displaystyle\sum_{j=1}^{n} g_j(x_j) - y \leq 0. \end{array}\right.$$

The research on global optimization of the general linear program with an additional concave constraint (LPAC) can be traced back to 1950's, arising from a location problem by Baumol-Wolfe [1]. The algorithms proposed since then can be classified roughly into four classes. The first class consists of algorithms based on the edge property of $F \setminus G$. As will be shown in Section 2, at least one optimal solution to LPAC lies on the intersection of the edges of F and the boundary of G. Exploiting this property, Hillestad [5] proposed a simplex-type pivoting algorithm for searching an optimal intersection point. Hillestad's algorithm has been modified and still developed by Hillestad-Jacobsen [7] and Thuong-Tuy [17]. The second class is outer approximation algorithms, which involves e.g., Hillestad-Jacobsen [6] and Fülöp [4]. Hillestad-Jacobsen [6] developed a procedure for cutting off a potion from F by a valid cut constructed at an infeasible vertex of F for the associated concave minimization. The convergence of their algorithm is not guaranteed; but Fülöp [4] improved this point later. The third class is conical branch-and-bound algorithms, which involves e.g., a bisection algorithm by Moshirvaziri-Amouzegar [12] and ω-subdivision algorithm by Muu [13]. The last class is algorithms alternating local search and concave minimization. This class is based on the concept of duality between LPAC and its associated concave minimization problem, studied by Tuy [18] and Tuy-Thuong [20]. As reported by Pferschy-Tuy [14], algorithms of this class are very efficient when the dual problem of a given instance is easy to solve.

Since LPASC is a special class of LPAC, algorithms of the above classes are naturally applicable to each instance of LPASC. Unfortunately, however, none of them exploits the special structure of the separable reverse convex set, which must have potential for efficient solution by analogy with the separable concave minimization (1.2). We can only see in a comprehensive survey on d.c. optimization by Tuy [19] a basic subdivision process for minimizing a separable d.c. function over a rectangle. As is well known, the separability of the objective function of (1.2) is one of the most useful structures in designing efficient global optimization algorithms. Since the pioneer work on (1.2) by Falk-Soland [3], a number of efficient rectangular branch-and-bound algorithms has been developed: Soland [16], Horst-Thoai [9], Kuno [11], Ryoo-Sahinidis [15] to name but a few. The purpose of this chapter is to develop practical algorithms by making full of use the separability of LPASC, together with the existing results on LPAC and (1.2).

In Section 2, after describing the problem formally, we give two optimality conditions, both of which play one of the leading parts in the proposed algorithms. Another leading part is played by Falk-Soland's rectangular branch-and-bound algorithm, which is explained in detail, in Section 3. Sections 4 and 5 are devoted to the algorithms for LPASC. In Section 4, on the basis of the duality by Tuy [18] and Tuy-Thuong [20], we develop an algorithm alternating local search and rectangular branch-and-bound. In Section 4, we try solving LPASC using a new rectangular branch-and-bound algorithm. This algorithm has basically the same structure as Falk-Soland's one but contains some devices for improving the efficiency. Some concluding remarks are discussed in Section 5.

2. Problem description and basic properties

The problem we consider in this chapter is of the form

$$\left|\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \sum_{j=1}^{n} g_j(x_j) \leq 0, \end{array}\right. \tag{2.1}$$

where $\mathbf{A} \in \mathrm{I\!R}^{m \times n}$, $\mathbf{b} \in \mathrm{I\!R}^{m}$, $\mathbf{c} \in \mathrm{I\!R}^{n}$, and $\mathbf{l}, \mathbf{u} \in \mathrm{I\!R}^{m}$. For each $j = 1, \ldots, n$, the function $g_j : S \to \mathrm{I\!R}$ is concave, and can be affine or constant. Let

$$C = \{\mathbf{x} \in \mathrm{I\!R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}, \quad D = \{\mathbf{x} \in \mathrm{I\!R}^n \mid \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$$
$$g(\mathbf{x}) = \sum_{j=1}^{n} g_j(x_j), \qquad G = \{\mathbf{x} \in S^n \mid g(\mathbf{x}) > 0\}.$$

Since each component of $\mathbf{l}$ and $\mathbf{u}$ is finite, D represents an n-dimensional rectangle, which we assume to be included in the domain S^n of function g. Using these notations, we can make it clear that the feasible set of (2.1) is a *d.c. set* of the form $C \cap D \setminus G$, i.e., the difference of two convex sets $C \cap D$ and G.

To show some basic properties of (2.1), we first need to make three assumptions, which are not special to our problem but often imposed on d.c. optimization problems (see e.g., [10, 8]). Let $\overline{M}$ denote the closure, and ∂M the boundary of a set M.

Assumption 2.1. *Problem (2.1) is feasible:*

$$C \cap D \setminus G \neq \emptyset.$$

Assumption 2.2. *The side constraint $g(\mathbf{x}) \leq 0$ is essential to (2.1):*

$$\max\{\mathbf{c}^{\mathsf{T}}\mathbf{x} \mid \mathbf{x} \in C \cap D \setminus G\} < \max\{\mathbf{c}^{\mathsf{T}}\mathbf{x} \mid \mathbf{x} \in C \cap D\}.$$

The second assumption is quite natural, as well as the first one. If Assumption 2.2 fails, problem (2.1) is equivalent to an ordinary linear program

$$\left|\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{array}\right. \tag{2.2}$$

We can compute an optimal solution $\mathbf{x}^0$ of (2.2) using any one of ordinary algorithms because $C \cap D$ is nonempty by Assumption 2.1 and bounded.

Assumption 2.3. *Problem (2.1) is regular:*

$$C \cap D \setminus G = \overline{C \cap D \setminus \overline{G}}.$$

Since $C \cap D \setminus G$ is a d.c. set, it might consists of some connected parts. If one of the parts is included in ∂G, it will disappear from $\overline{C \cap D \setminus \overline{G}}$. Assumption 2.3 excludes such a case; in other words, there is a point $\mathbf{y} \in C \cap D$ in any neighborhood of each feasible solution such that $g(\mathbf{y}) < 0$. Although Assumption 2.3 seems somewhat strong, each instance of (2.1) can easily be transformed into a regular one, as will be seen later.

Theorem 2.1. *Under Assumptions 2.1 and 2.2, the boundary of G contains all optimal solutions to (2.1), at least one of which lies on an edge of the polytope $C \cap D$.*

Proof: Since $C \cap D \setminus G$ is a nonempty and closed set, (2.1) has an optimal solution $\mathbf{x}^*$. Assume that $\mathbf{x}^*$ is not a boundary point of G. We see from Assumption 2.2 that $\mathbf{c}^\mathsf{T}\mathbf{x}^0 > \mathbf{c}^\mathsf{T}\mathbf{x}^*$ and $g(\mathbf{x}^0) > 0$ for any optimal solution $\mathbf{x}^0$ to (2.2). Therefore, the line segment joining $\mathbf{x}^*$ and $\mathbf{x}^0$ intersects ∂G at $\mathbf{x}' = (1-\lambda)\mathbf{x}^* + \lambda\mathbf{x}^0$ for some $\lambda \in (0,1)$. This intersection point $\mathbf{x}'$ is feasible to (2.1) and satisfies

$$\mathbf{c}^\mathsf{T}\mathbf{x}' = (1-\lambda)\mathbf{c}^\mathsf{T}\mathbf{x}^* + \lambda\mathbf{c}^\mathsf{T}\mathbf{x}^0 > (1-\lambda)\mathbf{c}^\mathsf{T}\mathbf{x}^* + \lambda\mathbf{c}^\mathsf{T}\mathbf{x}^* = \mathbf{c}^\mathsf{T}\mathbf{x}^*,$$

which contradicts the optimality of $\mathbf{x}^*$.

Now suppose that $\mathbf{x}^*$ is a boundary point of G. Since G is a convex set, it has a supporting hyperplane H at $\mathbf{x}^*$. Let $H' = H \cap C \cap D$. Then H' is a polytope and contains $\mathbf{x}^*$. The minimum of $\mathbf{c}^\mathsf{T}\mathbf{x}$ on H' is achieved at a vertex $\mathbf{v}$ of H'; and hence $\mathbf{c}^\mathsf{T}\mathbf{v} \leq \mathbf{c}^\mathsf{T}\mathbf{x}^*$. However, since $\mathbf{v} \in C \cap D \setminus G$, we have $\mathbf{c}^\mathsf{T}\mathbf{v} = \mathbf{c}^\mathsf{T}\mathbf{x}^*$, which implies that $\mathbf{v}$ is also an optimal solution to (2.1) and must lie on ∂G. Recall that $\mathbf{v}$ is a vertex of H', which is an intersection point of H and some edge of the polytope $C \cap D$. ■

Theorem 2.2. *Under Assumptions 2.1 and 2.2, if $\mathbf{x}^* \in C \cap D \setminus G$ is an optimal solution to (2.1), then*

$$\min\{g(\mathbf{x}) \mid \mathbf{x} \in C \cap D,\ \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^*\} = 0. \tag{2.3}$$

If Assumption 2.3 is fulfilled as well, then (2.3) is sufficient for $\mathbf{x}^*$ to be an optimal solution.

Proof: Let $\mathbf{x}^*$ be an optimal solution to (2.1). Assume that there is a point $\mathbf{x}' \in C \cap D$ such that $g(\mathbf{x}') < 0$ and $\mathbf{c}^\mathsf{T}\mathbf{x}' \geq \mathbf{c}^\mathsf{T}\mathbf{x}^*$. Since $\mathbf{x}' \in C \cap D \backslash G$, this point $\mathbf{x}'$ is an optimal solution to (2.1). From Theorem 2.1, however, every optimal solution to (2.1) lies on ∂G, which contradicts $g(\mathbf{x}') < 0$. Hence, we have

$$\min\{g(\mathbf{x}) \mid \mathbf{x} \in C \cap D,\ \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^*\} \geq 0,$$

where the equality holds when $\mathbf{x} = \mathbf{x}^*$.

Conversely, when (2.3) holds for some $\mathbf{x}^* \in C \cap D \setminus G$, let us assume that there is a point $\mathbf{x}' \in C \cap D \setminus G$ such that $\mathbf{c}^\mathsf{T}\mathbf{x}' > \mathbf{c}^\mathsf{T}\mathbf{x}^*$. Under Assumption 2.3, in any neighborhood of $\mathbf{x}'$ we can find a point $\mathbf{y} \in C \cap D$ with $g(\mathbf{y}) < 0$. If $\mathbf{y}$ is sufficiently near to $\mathbf{x}'$, we can keep $\mathbf{c}^\mathsf{T}\mathbf{y} > \mathbf{c}^\mathsf{T}\mathbf{x}^*$, which contradicts (2.3). Therefore, $\mathbf{c}^\mathsf{T}\mathbf{x} \leq \mathbf{c}^\mathsf{T}\mathbf{x}^*$ holds for any $\mathbf{x} \in C \cap D \setminus G$; and $\mathbf{x}^*$ is an optimal solution to (2.1). ∎

These two optimality conditions are well-known results on (2.1), and still true even when the concave function g is inseparable (see e.g., [10, 8]). They are the key to solution to the class of linear programs with a concave side constraint. Especially when we try to exploit the separability of the side constraint $g(\mathbf{x}) = \sum_{j=1}^{n} g_j(x_j) \leq 0$, the problem given in Theorem 2.2,

$$\mathrm{PD}(\alpha)\ \left|\ \begin{array}{ll} \text{minimize} & g(\mathbf{x}) \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b},\quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{c}^\mathsf{T}\mathbf{x} \geq \alpha, \end{array}\right.$$

plays a crucial role in the algorithms. Since $\mathrm{PD}(\mathbf{c}^\mathsf{T}\mathbf{x}^*)$ has the same optimal solution as the original problem (2.1) though the objective and side constraint are changed, it is called the *dual problem* of (2.1). Here, we should remark that the dual problem of (2.1) is a separable concave minimization.

Before proceeding to the algorithms, we briefly discuss how to transform a given instance of (2.1) into a regular one where Assumption 2.3 is fulfilled. One of the easiest way is to slightly perturb the side constraint function g.

Proposition 2.3. *For sufficiently small $\epsilon > 0$, let*

$$g'(\mathbf{x};\epsilon) = g(\mathbf{x}) - \epsilon(\|\mathbf{x}\|^2 + 1). \tag{2.4}$$

Then the following problem is regular and satisfies Assumption 2.3:

$$\left|\ \begin{array}{ll} \text{maximize} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b},\quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & g'(\mathbf{x};\epsilon) \leq 0. \end{array}\right. \tag{2.5}$$

Proof: See e.g., [10]. ∎

Note that the perturbed function g' is still concave and separable. In fact, letting

$$g'_j(x_j;\epsilon) = g_j(x_j) - \epsilon(x_j^2 + 1/n),$$

then we can represent g' as a sum of n concave functions:

$$g'(\mathbf{x};\epsilon) = \sum_{j=1}^{n} g'_j(x_j;\epsilon).$$

Therefore, (2.5) belongs to the same class as our problem (2.1); and besides, it will provide us with a high-quality approximated solution to (2.1) if ϵ is sufficiently small.

3. Falk-Soland's algorithm for concave minimization

As is well known, the rectangular branch-and-bound algorithm by Falk-Soland [3] is the most powerful tool for minimizing the concave function, though it can only be used when the objective function is separable. In the previous section, however, we have seen that the dual problem of (2.1) is just this case. Therefore, a straightforward way to solve (2.1) by exploiting its separability is to apply the rectangular branch-and-bound algorithm to problem PD(α). In this approach, the point is how to find an appropriate parameter value of α. This has already been resolved by Tuy [18] and Tuy-Thuong [20]. Before describing their approach in detail, we will explain Falk-Soland's branch-and-bound algorithm for a separable concave minimization problem

$$\left|\begin{array}{ll} \text{minimize} & g(\mathbf{x}) \\ \text{subject to} & \mathbf{A}'\mathbf{x} \le \mathbf{b}', \quad \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{array}\right. \tag{3.1}$$

where

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ -\mathbf{c}^{\mathrm{T}} \end{bmatrix}, \quad \mathbf{b}' = \begin{bmatrix} \mathbf{b} \\ -\alpha \end{bmatrix}. \tag{3.2}$$

Let $D = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{l} \le \mathbf{x} \le \mathbf{u}\}$ as in the previous section and

$$C' = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}'\mathbf{x} \le \mathbf{b}\}.$$

3.1 Three basic steps

In the rectangular branch-and-bound algorithm, while subdividing the rectangle D successively into

$$D^k = [l_1^k, u_1^k] \times [l_2^k, u_2^k] \times \cdots \times [l_n^k, u_n^k], \quad k \in \mathcal{K}, \tag{3.3}$$

we solve each subproblem of (3.1) with a feasible set $C' \cap D^k$, i.e., a problem of the following form with $D = D^k$:

$$\mathrm{P}(D) \left| \begin{array}{ll} \text{minimize} & g(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C' \cap D. \end{array} \right.$$

Subdivision of D needs carrying out in such a way that the set of resulting subrectangles D^k's constitutes a partition of D; hence, in the course of the algorithm, we always have

$$D = \bigcup_{k \in \mathcal{K}} D^k, \quad \mathrm{int}D^i \cap \mathrm{int}D^k = \emptyset \ \text{ if } i \neq k, \tag{3.4}$$

where $\mathrm{int}M$ denotes the interior of a set M. The subdivision rules that guarantee the convergence of this algorithm will be discussed later.

The main scheme of the algorithm consists of three basic steps:

Let $D^1 := D$, $\mathcal{K} := \{1\}$ and $k := 1$. Repeat Steps 1 – 3 until $\mathcal{K} = \emptyset$.

Step 1. Take an appropriate index i_k out of $\mathcal{K}$ and let $D := D^{i_k}$.

Step 2. (Bounding operation) Compute a lower bound z^k on the optimal value $z(D)$ of $\mathrm{P}(D)$. If $z^k \geq g(\mathbf{x}^*)$ for the best feasible solution $\mathbf{x}^*$ to (3.1) obtained so far, discard D from further consideration.

Step 3. (Branching operation) Otherwise, divide the rectangle D into two subrectangles D^{2k} and D^{2k+1}. Add $\{2k, 2k+1\}$ to $\mathcal{K}$.

There are two major advantages in this algorithm: (1) we need only two vectors $\mathbf{l}^k = (l_1^k, \ldots, l_n^k)$ and $\mathbf{u}^k = (u_1^k, \ldots, u_n^k)$ to maintain and construct each subproblem $\mathrm{P}(D)$; and (2) we can compute a strong lower bound z^k by solving a linear program. Next, we will see the second point.

3.2 Bounding operation (Step 2)

To compute a lower bound z^k in Step 2, we first determine the convex envelope of g_i on the interval $[l_j, u_j]$ for each $j = 1, \ldots, n$:

$$h_j(x_j) = \frac{g_j(u_j) - g_j(l_j)}{u_j - l_j} x_j + \frac{u_j g_j(l_j) - l_j g_j(u_j)}{u_j - l_j}. \tag{3.5}$$

From the concavity of g_j, we see that

$$h_j(x_j) \leq g_j(x_j) \ \text{ if } x_j \in [l_j, u_j], \quad h_j(x_j) > g_j(x_j) \ \text{ otherwise,}$$

where we should remark that $h_j(x_j) = g_j(x_j)$ if $x_j \in \{l_j, u_j\}$. Therefore, h_j is the *convex envelope* of g_j on the interval $[l_j, u_j]$, i.e., the maximal convex function that never exceeds the value of g_j on $[l_j, u_j]$. Since g is the sum of g_j's, this property is inherited to

$$h(\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j). \tag{3.6}$$

Lemma 3.1. *If* $\mathbf{x} \in D$, *then*

$$h(\mathbf{x}) \leq g(\mathbf{x}),$$

where the equality holds when $x_j \in \{l_j, u_j\}$ *for* $j = 1, \ldots, n$. ■

We should also note on h that it is an affine function of $\mathbf{x}$. Hence, replacing the objective function g of $\mathrm{P}(D)$ by h, we have a linear program that provides a lower bound of its optimal value:

$$\mathrm{PL}(D) \left| \begin{array}{ll} \text{minimize} & h(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C' \cap D. \end{array} \right.$$

Let $\mathbf{x}^k$ denote an optimal solution to $\mathrm{PL}(D)$ when $C' \cap D \neq \emptyset$. Then we see that $h(\mathbf{x}^k)$ can be used as the lower bound z^k in Step 2:

$$z^k = \begin{cases} h(\mathbf{x}^k) & \text{if } C' \cap D \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases}$$

Lemma 3.2. *If* $z^k = +\infty$, *then P(D) is infeasible. Otherwise, among* z^k, $z(D)$ *and* $g(\mathbf{x}^k)$ *there is a relationship:*

$$z^k \leq z(D) \leq g(\mathbf{x}^k),$$

where the equalities hold when $x_j^k \in \{l_j, u_j\}$ *for* $j = 1, \ldots, n$. ■

The rectangular branch-and-bound algorithm requires one to solve a series of linear programs of the form $\mathrm{PL}(D)$. These problems, however, differ from one another in just h and D. Therefore, any optimal basis $\mathbf{B}$ of $\mathbf{A}'$ in the present problem can serve as the initial basis in solution to the succeeding problem. Namely, we first restore $\mathbf{B}$ to a feasible one for the new bounding constraints defining D; then we optimize it according to the new objective function h (see e.g., [2] in further detail). This process can usually be done in quite a few simplex pivoting operations.

3.3 Branching operation (Step 3)

In Step 3, we divide D in such a way that the resulting sets D^{2k} and D^{2k+1} satisfy (3.3) and (3.4). We can carry out this as follows, given an index $j \in \{1, \dots, n\}$ and a number $m_j \in [l_j, u_j]$:

$$\left.\begin{aligned} D^{2k} &= [l_1, u_1] \times \cdots \times [l_{j-1}, u_{j-1}] \times [l_j, m_j] \\ &\qquad \times [l_{j+1}, u_{j+1}] \times \cdots \times [l_n, u_n] \\ D^{2k+1} &= [l_1, u_1] \times \cdots \times [l_{j-1}, u_{j-1}] \times [m_j, u_j] \\ &\qquad \times [l_{j+1}, u_{j+1}] \times \cdots \times [l_n, u_n] \end{aligned}\right\} \tag{3.7}$$

In general, no matter how we select j and m_j, the finiteness of the algorithm cannot be guaranteed without a tolerance for the optimal value of (3.1). In that case, the algorithm generates an infinite sequence of rectangles D^{k_i}, $i = 1, 2, \dots$, such that

$$D^{k_1} \supset D^{k_2} \supset \cdots, \quad C' \cap \left(\bigcap_{i=1}^{\infty} D^{k_i} \right) \neq \emptyset. \tag{3.8}$$

Let us denote D^{k_i} simply by $D^i = [l_1^i, u_1^i] \times \cdots \times [l_n^i, u_n^i]$ and the sequence by the index set $\mathcal{L} = \{1, 2, \dots, i, \dots\}$. We assume that for each $i \in \mathcal{L}$, rectangle D^{i+1} is generated from D^i via (3.7) for some pair $(j_i, m_{j_i}^i)$.

Lemma 3.3. *There is an infinite subsequence $\mathcal{L}_q \subset \mathcal{L}$ such that $j_i = q$ for all $i \in \mathcal{L}_q$ and*

$$l_q^i \to l_q^*, \quad u_q^i \to u_q^*, \quad m_q^i \to m_q^* \in \{l_q^*, u_q^*\} \quad \textit{as } i \to +\infty \textit{ in } \mathcal{L}_q.$$

Proof: Since j_i is an element of the finite set $\{1, \dots, n\}$, we can take an infinite subsequence $\mathcal{L}_q'$ such that $j_i = q$ for all $i \in L_q'$. Assuming $\mathcal{L}_q' = \{1, 2, \dots\}$ without loss of generality, we have

$$l_q^1 \leq l_q^i \leq l_q^{i+1} \leq u_q^{i+1} \leq u_q^i \leq u_q^1, \quad \forall i \in \mathcal{L}_q'.$$

Hence, for some l_q^* and u_q^* such that $l_q^1 \leq l_q^* \leq u_q^* \leq u_q^1$, we have $l_q^i \to l_q^*$ and $u_q^i \to u_q^*$ as $i \to +\infty$ in $\mathcal{L}_q'$. These also implies that l_q^* and u_q^* are accumulation points of m_q^i because m_q^i coincides with either l_q^{i+1} or u_q^{i+1} for each $i \in \mathcal{L}_q'$. Therefore, we can take a subsequence $\mathcal{L}_q \subset \mathcal{L}_q'$ such that $m_q^i \to m_q^* \in \{l_q^*, u_q^*\}$ as $i \to +\infty$ in $\mathcal{L}_q$. ■

When a positive tolerance is allowed for the optimal value of (3.1), the rules below of selecting the pair $(j_i, w_{j_i}^i)$ guarantee the finites of the algorithm.

Bisection. For each $i \in \mathcal{L}$, let us select

$$j_i \in \arg\max\{u_j^i - l_j^i \mid j = 1, \dots, n\}; \tag{3.9}$$

and divide the interval $[l^i_{j_i}, u^i_{j_i}]$ at

$$m^i_{j_i} = (1-\lambda)l^i_{j_i} + \lambda u^i_{j_i}, \tag{3.10}$$

where $\lambda \in (0,1)$ is a constant. We refer to this selection rule of $(j_i, m^i_{j_i})$ as *bisection* of ratio λ.

Lemma 3.4. *If $\mathcal{L}$ is generated according to the bisection rule of ratio $\lambda \in (0,1)$, then there is a subsequence $\mathcal{L}' \subset \mathcal{L}$ such that*

$$g(\mathbf{x}^i) - z^i \to 0 \quad \textit{as } i \to +\infty \textit{ in } \mathcal{L}',$$

where $\mathbf{x}^i$ is an optimal solution to PL(D^i) and z^i the optimal value.

Proof: Let us take the subsequence $\mathcal{L}_q$ given in Lemma 3.3. Then we have $l^i_q \to l^*_q$, $u^i_q \to u^*_q$, $m^i_q \to m^*_q \in \{l^*_q, u^*_q\}$ as $i \to +\infty$ in $\mathcal{L}_q$. From (3.10), however, we have

$$(1-\lambda)l^*_q + \lambda u^*_q = m^*_q \in \{l^*_q, u^*_q\},$$

which holds only if $m^*_q = l^*_q = u^*_q$. This, together with (3.9), implies that D^i shrinks to a point $\mathbf{m}^* = (m^*_1, \ldots, m^*_n) \in D^1$ as $i \to +\infty$ in $\mathcal{L}' = \bigcup_{q=1}^n \mathcal{L}_q$. Hence, from the definition of PL(D^i), we have the desired result. ■

The bisection rule is simple but does not entirely exploit the characteristics of problem PL(D). As stated in Lemma 3.2, the objective function h of PL(D) agrees with the value of g at each vertex of D. The next selection rule of $(j_i, m^i_{j_i})$ uses this property.

ω-division. For each $i \in \mathcal{L}$, let us select

$$j_i \in \arg\max\{g_j(x^i_j) - h_j(x^i_j) \mid j = 1, \ldots, n\}; \tag{3.11}$$

and divide the interval $[l^i_{j_i}, u^i_{j_i}]$ at

$$m^i_{j_i} = x^i_{j_i}. \tag{3.12}$$

We refer to this selection rule of $(j_i, m^i_{j_i})$ as ω-division.

Lemma 3.5. *If $\mathcal{L}$ is generated according to the ω-division rule, then there is a subsequence $\mathcal{L}' \subset \mathcal{L}$ such that*

$$g(\mathbf{x}^i) - z^i \to 0 \quad \textit{as } i \to +\infty \textit{ in } \mathcal{L}'.$$

Proof: Suppose that $m^i_q \to m^*_q = l^*_q$ as $i \to +\infty$ in the sequence $\mathcal{L}_q$ given in Lemma 3.3. Then we have $x^i_q = l^{i+1}_q$ from (3.12), and $h^i_q(x^i_q) = g_q(x^i_q)$ from Lemma 3.1, where h^i_q is the convex envelope of g_q on the interval $[l^i_q, u^i_q]$. If $i \to +\infty$ in $\mathcal{L}_q$, then $h^i_q(x^i_q) - g_q(x^i_q) \to 0$. This can be shown even when $m^*_q = u^*_q$. Hence, by noting (3.11), we have the desired result. ■

3.4 Description of the algorithm

The rest to be discussed is how to select an index i_k from the set $\mathcal{K}$ in Step 1. Usually, either of the following rules is adopted:

Depth first. The set $\mathcal{K}$ is maintained as a list of *stack*. An index i_k is taken from the top of $\mathcal{K}$; and $2k+1$, $2k$ are added in this order to the top.

Best bound. The set $\mathcal{K}$ is maintained as a list of *priority queue*. An index i_k of least z^{i_k} is taken out of $\mathcal{K}$.

Now, we are ready to describe the algorithm completely. Let $\epsilon \geq 0$ be a given tolerance for the optimal value of problem (3.1).

Algorithm 3.1.

```
begin
  $D^1 := D$; $\mathcal{K} := \{1\}$; $k := 1$;
  initialize the incumbent value $z^* := +\infty$;
  while $\mathcal{K} \neq \emptyset$ do
                                                            /* Step 1 */
    select an index $i_k$ from $\mathcal{K}$ by a fixed rule (depth first or best bound);
    $\mathcal{K} := \mathcal{K} \setminus \{i_k\}$; set $D := D^{i_k}$ and define a subproblem P($D$) of (3.1);
                                                            /* Step 2 */
    determine the convex envelope $h_j$ of $g_j$ on $[l_j, u_j]$ for each $j$;
    construct PL($D$) of minimizing $h(\mathbf{x}) := \sum_{j=1}^n h_j(x_j)$;
    compute a feasible solution $\mathbf{x}^k$ and a lower bound $z^k$ on $z(D)$ by
    solving PL($D$);
    if $z^* - z^k > \epsilon$ then begin
                                                            /* Step 3 */
      if $g(\mathbf{x}^k) < z^*$ then update $z^* := g(\mathbf{x}^k)$ and $\mathbf{x}^* := \mathbf{x}^k$;
      select $j \in \{1, \ldots, n\}$ and $m_j \in [l_j, u_j]$ by a fixed rule (bisection
      or $\omega$-division);
      $D^{2k} := [l_1, u_1] \times \cdots \times [l_j, m_j] \times \cdots \times [l_n, u_n]$;
      $D^{2k+1} := [l_1, u_1] \times \cdots \times [m_j, u_j] \times \cdots \times [l_n, u_n]$;
      $\mathcal{K} := \mathcal{K} \cup \{2k, 2k+1\}$; $k := k+1$
    end
  end
end;
```

Theorem 3.6. *When $\epsilon > 0$, Algorithm 3.1 terminates in a finite number of iterations and yields a globally ϵ-optimal solution $\mathbf{x}^*$ to problem (3.1).*

Proof: Let us assume the contrary: Algorithm 3.1 is infinite. Then it generates an infinite sequence $\mathcal{L}$ of D^{k_i} satisfying (3.8). The backtracking criterion $z^* - z^k > \epsilon$ implies that the following inequalities hold at the end of each iteration in which k_i for $i \in \mathcal{L}$ is taken out of $\mathcal{K}$:

$$z^i + \epsilon < z^* \leq g(\mathbf{x}^i),$$

where $\mathbf{x}^i$ is an optimal solution to $\mathrm{PL}(D^{k_i})$ and z^i the optimal value. From Theorem 3.4 and 3.5, however, $g(\mathbf{x}^i) - z^i \to 0$ as $i \to +\infty$ in some $\mathcal{L}' \subset \mathcal{L}$ whichever rule we adopt for selecting (j, w_j) in Step 3. Therefore, we have $\epsilon \leq 0$, which is a contradiction. The ϵ-optimality of $\mathbf{x}^*$ follows from the backtracking criterion. ■

Corollary 3.7. *Suppose $\epsilon = 0$. If the best bound rule is adopted in Step 1, the sequence of $\mathbf{x}^k$'s generated by Algorithm 3.1 has accumulation points, each of which is a globally optimal solution to problem (3.1).*

Proof: If the algorithm happens to be finite, the assertion is obvious from the backtracking criterion. Assume that it is infinite and generates an infinite sequence $\mathcal{L}$ just stated in the proof of the previous theorem. The best bound rule then implies the following at the beginning of each iteration in which j_i for $i \in \mathcal{L}$ is taken out of $\mathcal{K}$:

$$z^i \leq z^k \leq z(D^k), \quad \forall k \in \mathcal{K},$$

where z^i is the optimal value of $\mathrm{PL}(D^{k_i})$. However, $g(\mathbf{x}^i) - z^i \to 0$ as $i \to +\infty$ in some $\mathcal{L}' \subset \mathcal{L}$; and besides $\min\{z(D^k) \mid k \in \mathcal{K}\}$ is nothing but the optimal value of (3.1). Hence, every accumulation point of $\mathbf{x}^k$ is a globally optimal solution to (3.1). ■

4. Direct application of Falk-Soland' algorithm

Let us return to our original problem (2.1), subject to Assumptions 2.1-2.3 given in Section 2. As mentioned at the beginning of Section 3, we can solve (2.1) by applying Algorithm 3.1 to the dual problem $\mathrm{PD}(\alpha)$ if we can find an appropriate value of the parameter α. For this purpose, Tuy [18], Tuy-Thuong [20] and Pferschy-Tuy [14] have proposed the following approach. First, we generate a locally optimal solution $\mathbf{x}^*$ using any one of available algorithms, and then check its global optimality by solving $\mathrm{PD}(\alpha)$ with $\alpha = \mathbf{c}^\mathsf{T}\mathbf{x}^*$. If the optimal value of $\mathrm{PD}(\mathbf{c}^\mathsf{T}\mathbf{x}^*)$ is zero, then $\mathbf{x}^*$ is a globally optimal solution to (2.1); otherwise, we try checking another locally optimal solution.

4.1 Local search

According to their approach, the first thing we have to do is to search a locally optimal solution $\mathbf{x}^*$. We can use the simplex algorithm on problem (2.1) since the objective and constraints except for the last one are all linear. The simplex algorithm may start from any feasible vertex $\mathbf{v}^1 \notin G$ of the polytope $C \cap D$. If no feasible vertex is on hand, we may solve PD($-\infty$), i.e., a concave minimization problem, using Algorithm 3.1:

$$\left| \begin{array}{ll} \text{minimize} & g(\mathbf{x}) \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{array} \right.$$

Since g is a concave function, it achieves the minimum at some vertex $\mathbf{v}^1 \in C \cap D$. By Assumptions 2.1 and 2.3, we must have $g(\mathbf{v}^1) < 0$. To be precise, Algorithm 3.1 might not yield a vertex of $C \cap D$ when $\epsilon > 0$; but, how to cope with that case will be discussed later.

Starting from $\mathbf{v}^1$, we generate a sequence of adjacent vertices $\mathbf{v}^2, \mathbf{v}^3, \ldots$ of $C \cap D$ such that $\mathbf{c}^\mathsf{T}\mathbf{v}^1 < \mathbf{c}^\mathsf{T}\mathbf{v}^2 < \cdots$, using the simplex algorithm, with some anticycling pivoting rule if necessary (see [2]). In this process, we must encounter a vertex $\mathbf{v}^\ell$ satisfying

$$g(\mathbf{v}^i) < 0, \quad i = 1, 2, \ldots, \ell - 1, \quad g(\mathbf{v}^\ell) \geq 0,$$

before reaching an optimal vertex $\mathbf{x}^0$ of the linear program (2.2). Then we compute an intersection point $\mathbf{x}^*$ of the edge $\mathbf{v}^{\ell-1}$–$\mathbf{v}^\ell$ and ∂G. This point $\mathbf{x}^* \in \partial G$ is given as $(1 - \lambda^*)\mathbf{v}^{\ell-1} + \lambda^*\mathbf{v}^\ell$ for

$$\lambda^* = \min\{\lambda \in [0, 1] \mid g[(1 - \lambda)\mathbf{v}^{\ell-1} + \lambda\mathbf{v}^\ell] \geq 0\}, \tag{4.1}$$

Since (4.1) is a convex minimization and besides univariate, computation of λ^* is inexpensive. From Theorem 2.1, we see that $\mathbf{x}^*$ is a nominee for solution to (2.1).

In this local search procedure, we should remark that an edge $\mathbf{v}^{i-1}$–$\mathbf{v}^i$ for some $i < \ell$ can intersects ∂G. More precisely, there might be at most two points $\mathbf{x}'$ and $\mathbf{x}''$ on edge $\mathbf{v}^{i-1}$–$\mathbf{v}^i$ such that $g(\mathbf{x}') = g(\mathbf{x}'') = 0$. In that case, however, both $\mathbf{x}'$ and $\mathbf{x}''$ cannot be optimal for (2.1) because

$$\mathbf{c}^\mathsf{T}\mathbf{v}^{i-1} < \mathbf{c}^\mathsf{T}\mathbf{x}' \leq \mathbf{c}^\mathsf{T}\mathbf{x}'' \leq \mathbf{c}^\mathsf{T}\mathbf{v}^i,$$

and $\mathbf{v}^i \in C \cap D \setminus G$ by assumption. Even if we neglect such intersection points, we never lose any optimal solution to (2.1).

4.2 Global optimality check

If we obtain the nominee $\mathbf{x}^* \in C \cap D \cap \partial G$, the next thing is to check whether $\mathbf{x}^*$ satisfies the optimality condition of (2.1) given by Theorem 2.2. We can accomplish it, in theory, applying Algorithm 3.1 to PD($\mathbf{c}^\mathsf{T}\mathbf{x}^*$). If the algorithm yields $\mathbf{x}^*$ as an optimal solution to PD($\mathbf{c}^\mathsf{T}\mathbf{x}^*$), we can conclude from Theorem 2.2 that $\mathbf{x}^*$ is optimal for (2.1) as well.

In practice, however, Algorithm 3.1 might not terminate in finite time, without a positive tolerance ϵ. We therefore need some additional devices.

For a sufficiently small $\delta > 0$, let $\mathbf{x}'$ be a point on the line including edge $\mathbf{v}^{\ell-1}$–$\mathbf{v}^{\ell}$ such that

$$\mathbf{c}^{\mathsf{T}}\mathbf{x}' = \mathbf{c}^{\mathsf{T}}\mathbf{x}^* + \delta < \mathbf{c}^{\mathsf{T}}\mathbf{x}^0,$$

where $\mathbf{x}^0$ is an optimal solution to the linear program (2.2). Also letting

$$\epsilon = g(\mathbf{x}'),$$

we see that $\epsilon > 0$. Instead of solving PD($\mathbf{c}^{\mathsf{T}}\mathbf{x}^*$), we solve PD($\mathbf{c}^{\mathsf{T}}\mathbf{x}'$) using Algorithm 3.1 with this ϵ as tolerance. Then, by Theorem 3.6, the algorithm must terminates in finite time and yields an ϵ-optimal solution to PD($\mathbf{c}^{\mathsf{T}}\mathbf{x}'$). If the output solution is $\mathbf{x}'$, then it satisfies $g(\mathbf{x}') \leq g(\mathbf{x})+\epsilon$ for any $\mathbf{x} \in C \cap D$ satisfying $\mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \mathbf{c}^{\mathsf{T}}\mathbf{x}^*+\delta$. Since $g(\mathbf{x}') = \epsilon$ by definition, we have

$$g(\mathbf{x}) \geq 0, \quad \forall \mathbf{x} \in C' \cap D,$$

where C' is set to $C \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \mathbf{c}^{\mathsf{T}}\mathbf{x}^* + \delta\}$. If there is a point $\mathbf{x}'' \in C' \cap D$ satisfying this with equality, then x'' is an optimal solution to (2.1) and the optimal value is $\mathbf{c}^{\mathsf{T}}\mathbf{x}^* + \delta$ by Theorem 2.2. Hence, we have

$$\mathbf{c}^{\mathsf{T}}\mathbf{x}^* \geq \mathbf{c}^{\mathsf{T}}\mathbf{x} - \delta, \quad \forall \mathbf{x} \in C \cap D \setminus G,$$

which means that $\mathbf{x}^* \in C \cap D \cap \partial G$ is a globally δ-optimal solution to (2.1).

Next, suppose that Algorithm 3.1 yields $\mathbf{x}^* \neq \mathbf{x}'$ with $g(\mathbf{x}^*) < \epsilon$. This point $\mathbf{x}^*$ is a vertex of $C' \cap D^k$ for some $k \in \mathcal{K}$ but might not be a vertex of $C' \cap D$. However, since g achieves each local minimum at some vertex of $C' \cap D$, finding a vertex $\mathbf{w}^1$ of $C' \cap D$ with $g(\mathbf{w}^1) \leq g(\mathbf{x}^*)$ is not expensive if we locally minimize g on $C' \cap D$ starting from $\mathbf{x}^*$. Once we obtain such a point $\mathbf{w}^1$ with $g(\mathbf{w}^1) \leq g(\mathbf{x}^*) < \epsilon$, we may again generate a sequence of adjacent vertices $\mathbf{w}^2, \mathbf{w}^3, \ldots$ of $C' \cap D$ such that $\mathbf{c}^{\mathsf{T}}\mathbf{w}^1 < \mathbf{c}^{\mathsf{T}}\mathbf{w}^2 < \cdots$, until some edge $\mathbf{w}^{\ell-1}$–$\mathbf{w}^{\ell}$ intersects ∂G.

4.3 Description of the algorithm

Let us summarize the algorithm alternating local search and concave minimization, where $\delta > 0$ is a given tolerance.

Algorithm 4.1.

```
begin
    let $\mathbf{w} \notin G$ be a vertex of $C \cap D$; optimal := false; $C' := C$; $k := 1$;
    while optimal = false do begin
    repeat
      $\mathbf{v} := \mathbf{w}$; find a vertex $\mathbf{w}$ of $C' \cap D$ adjacent to $\mathbf{v}$ such that
      $\mathbf{c}^{\mathsf{T}}\mathbf{v} < \mathbf{c}^{\mathsf{T}}\mathbf{w}$
    until $g(\mathbf{v}) < 0$ and $g(\mathbf{w}) \geq 0$;
    let $\mathbf{x}^k$ be an intersection point of edge $\mathbf{v}$–$\mathbf{w}$ and $\partial G$;
    let $\alpha := \mathbf{c}^{\mathsf{T}}\mathbf{x}^k + \delta$ and $\mathbf{x}'$ be a point on the line including $\mathbf{v}$–$\mathbf{w}$ such
    that $\mathbf{c}^{\mathsf{T}}\mathbf{x}' = \alpha$;
    let $C' := C \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \alpha\}$;
    solve PD($\alpha$) using Algorithm 3.1 with $\epsilon := g(\mathbf{x}')$ and let $\mathbf{x}^*$ denote
    its output;
    if $g(\mathbf{x}^*) < \epsilon$ then
      find a vertex $\mathbf{w}$ of $C' \cap D$ with $g(\mathbf{w}) \leq g(\mathbf{x}^*)$ in a neighborhood
      of $\mathbf{x}^*$
    else optimal := true;
    $k := k + 1$;
  end;
  $\mathbf{x}^* := \mathbf{x}^k$
end;
```

Theorem 4.1. *When $\delta > 0$ is a sufficiently small number, Algorithm 4.1 terminates in a finite number of iterations and yields a globally δ-optimal solution $\mathbf{x}^*$ to (2.1).*

Proof: For each iteration $k > 1$, we see that

$$\mathbf{c}^{\mathsf{T}}\mathbf{x}^k \geq \mathbf{c}^{\mathsf{T}}\mathbf{x}^{k-1} + \delta > \mathbf{c}^{\mathsf{T}}\mathbf{x}^{k-1}.$$

Since $\mathbf{c}^{\mathsf{T}}\mathbf{x}^k$ has an upper bound $\mathbf{c}^{\mathsf{T}}\mathbf{x}^0$, the finiteness of Algorithm 4.1 follows this. ■

5. Indirect application of Falk-Soland's algorithm

In the previous section, to solve (2.1) we apply Falk-Soland's branch-and-bound algorithm in a rather direct manner. This solution method, however, has a weak point that we have to solve a class of concave minimization problems repeatedly, even though the class is fairly easy to solve in comparison with other multiextremal global optimization problems. In this section, we will develop a method that computes an upper bound on the optimal value of (2.1), not a lower bound on that of the dual problem (3.1), in Step 2 of the rectangular branch-and-bound scheme. Therefore, once we call this branch-and-bound algorithm, it generates a globally optimal solution to the original problem (2.1).

As in Falk-Soland's algorithm, we subdivide the rectangle D into subrectangles D^k, $k \in \mathcal{K}$, which constitute a partition of D. However, the problem to be solved for each partition set D^k is not a subproblem of (3.1) but that of (2.1), i.e., we solve a problem of the following form with $D = D^k$:

$$\mathrm{Q}(D) \left| \begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in C \cap D \setminus G. \end{array} \right.$$

Of course, $\mathrm{Q}(D)$ belongs to the same class as (2.1), and hence cannot be solved directly. Instead, we compute an upper bound w^k on $\mathrm{Q}(D)$ in each iteration and narrow partition sets down to the one containing an optimal solution to (2.1):

Let $D^1 := D$, $\mathcal{K} := \{1\}$ and $k := 1$. Repeat Steps 1 – 3 until $\mathcal{K} = \emptyset$.

Step 1. Take an appropriate index i_k out of $\mathcal{K}$ and let $D := D^{i_k}$.

Step 2'. (Bounding operation) Compute an upper bound w^k on the optimal value $w(D)$ of $\mathrm{Q}(D)$. If $w^k \leq \mathbf{c}^{\mathsf{T}}\mathbf{x}^*$ for the best feasible solution $\mathbf{x}^*$ to (2.1) obtained so far, discard D from further consideration.

Step 3. (Branching operation) Otherwise, divide the rectangle D into two subrectangles D^{2k} and D^{2k+1}. Add $\{2k, 2k+1\}$ to $\mathcal{K}$.

We will begin by explaining how to compute the upper bound w^k in Step 2' of this scheme.

5.1 Linearization and its solution

As shown in Lemma 3.1, the convex envelope h of g defined in (3.5) and (3.6) satisfies $h(\mathbf{x}) \leq g(\mathbf{x})$ for all $\mathbf{x} \in D$. Hence, by letting

$$H = \{\mathbf{x} \in R^n \mid h(\mathbf{x}) \leq 0\},$$

we have

$$D \setminus G \subset D \cap H.$$

This immediately implies that the optimal value $w(D)$ of $\mathrm{Q}(D)$ is bounded from above by that of

$$\mathrm{QL}(D) \left| \begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in C \cap D \cap H. \end{array} \right.$$

Let $\mathbf{x}^k$ denote an optimal solution to $\mathrm{QL}(D)$ when it is feasible. Also, let

$$w^k = \begin{cases} \mathbf{c}^{\mathsf{T}}\mathbf{x}^k & \text{if } C \cap D \cap H \neq \emptyset \\ -\infty & \text{otherwise.} \end{cases}$$

Lemma 5.1. *If $w^k = -\infty$, then $Q(D)$ is infeasible. Otherwise, the following relationship holds:*

$$w^k \geq w(D). \tag{5.1}$$

■

Since h is an affine function, QL(D) is a linear program with the set of constraints

$$\mathbf{A}''\mathbf{x} \leq \mathbf{b}'', \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},$$

where

$$\mathbf{A}'' = \begin{bmatrix} & \mathbf{A} & \\ \dfrac{g_1(u_1) - g_1(l_1)}{u_1 - l_1} & \dots & \dfrac{g_n(u_n) - g_n(l_n)}{u_n - l_n} \end{bmatrix}$$

$$\mathbf{b}'' = \begin{bmatrix} \mathbf{b} \\ \displaystyle\sum_{j=1}^{n} \frac{l_j g_j(u_j) - u_j g_j(l_j)}{u_j - l_j} \end{bmatrix}.$$

Therefore, if we try to use w^k as the upper bound in Step 2', we need to solve a series of linear programs different from one another just in the last rows of $\mathbf{A}''$ and $\mathbf{b}''$. However, this structure of $\mathbf{A}''$ causes a serious disadvantage when we solve them using the revised simplex algorithm where the inverse of each basis is maintained in a compact form such as a product of eta matrices (see e.g., [2]). Even if we keep an optimal basis of $\mathbf{A}''$ in such a form for the present problem, it is of no use in solving the succeeding problems. To improve this, we propose to solve QL(D) in two stages starting from an optimal solution $\mathbf{x}^{k-1}$ of the preceding linear program. In both stages, the matrix we mainly deal with is not $\mathbf{A}''$ but $\mathbf{A}'$ defined in (3.2).

First stage of solution to QL(D). Let

$$\phi(\alpha) = \min\{h(\mathbf{x}) \mid \mathbf{x} \in C \cap D,\ \mathbf{c}^{\mathsf{T}}\mathbf{x} = \alpha\}.$$

It is known that ϕ is a convex piecewise affine function of α (see e.g., [2]). We see from the following that QL(D) amounts to locating the maximum of α satisfying $\phi(\alpha) \leq 0$.

Lemma 5.2. *Let $\mathbf{x}'$ be a point in $C \cap D \cap H$. Then $\mathbf{x}'$ is an optimal solution to QL(D) if and only if $\mathbf{c}^{\mathsf{T}}\mathbf{x}'$ is the maximum value of α satisfying $\phi(\alpha) \leq 0$.*

Proof: Note that $\phi(\mathbf{c}^{\mathsf{T}}\mathbf{x}') \leq h(\mathbf{x}') \leq 0$ since $\mathbf{x}' \in C \cap D \cap H$. Therefore, if either holds, we have $\mathbf{c}^{\mathsf{T}}\mathbf{x}' \geq \mathbf{c}^{\mathsf{T}}\mathbf{x}$ for any $\mathbf{x} \in C \cap D$ satisfying $h(\mathbf{x}) \leq 0$, which implies the other. ■

If $\mathbf{c}^\mathsf{T}\mathbf{x} \neq \alpha$ for any $\mathbf{x} \in C \cap D$, then $\phi(\alpha)$ is understood to be $+\infty$. For a given $\mathbf{x}^{k-1}$ optimal for the preceding linearized subproblem, we first check the value of ϕ at $\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$. This can be done by solving the following linear program

$$\left|\begin{array}{ll} \text{minimize} & h(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C \cap D, \quad \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}. \end{array}\right. \tag{5.2}$$

Three cases can occur:

Case 1: Problem (5.2) is infeasible. In this case, $\phi(\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}) = +\infty$ and $\mathrm{QL}(D)$ can provide no better solution than $\mathbf{x}^{k-1}$. Hence, we can exclude the rectangle D from further consideration.

Case 2: Problem (5.2) has an optimal solution $\mathbf{x}'$ such that $\mathbf{c}^\mathsf{T}\mathbf{x}' = \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$. The value $\phi(\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1})$ is given by $h(\mathbf{x}')$. If $h(\mathbf{x}') = 0$, then $\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$ is the maximum of α; but $\mathrm{QL}(D)$ can provide no better solution than $\mathbf{x}^{k-1}$ as long as $h(\mathbf{x}') \geq 0$. If $h(\mathbf{x}') < 0$, let $\alpha_1 = \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$.

Case 3: Problem (5.2) has an optimal solution $\mathbf{x}'$ such that $\mathbf{c}^\mathsf{T}\mathbf{x}' > \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$. Let $\alpha_1 = \mathbf{c}^\mathsf{T}\mathbf{x}'$. Then we have $\phi(\alpha_1) = h(\mathbf{x}')$. If $h(\mathbf{x}')$ is vanishes, then α_1 is the maximum of α; hence, $\mathbf{x}^k$ and w^k can be set to $\mathbf{x}'$ and α_1, respectively.

Second stage of solution to QL(D). If $\phi(\alpha_1) \neq 0$, then we adjust the value of α to restore $\phi(\alpha) = 0$. Suppose that $\phi(\alpha_1) < 0$. In this case, as increasing the value of α from α_1, we solve

$$\mathrm{PQ}(\alpha)\left|\begin{array}{ll} \text{minimize} & h(\mathbf{x}) \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{c}^\mathsf{T}\mathbf{x} = \alpha, \end{array}\right.$$

using the parametric right-hand-side simplex algorithm [2]. Then it generates a sequence of intervals $[\alpha_1, \alpha_2], [\alpha_2, \alpha_3], \ldots$, and a sequence of bases $\mathbf{B}^1, \mathbf{B}^2, \ldots$ of $\mathbf{A}'$ such that $\mathbf{B}^i$ is optimal for $\mathrm{PQ}(\alpha)$ when $\alpha \in [\alpha_i, \alpha_{i+1}]$. The optimal value $\phi(\alpha)$ of $\mathrm{PQ}(\alpha)$ is affine on each interval $[\alpha_i, \alpha_{i+1}]$. If we find a break point α_ℓ satisfying

$$\phi(\alpha_i) < 0, \quad i = 1, \ldots, \ell - 1, \quad \phi(\alpha_\ell) \geq 0, \tag{5.3}$$

we can easily compute a point $\alpha' \in [\alpha_{\ell-1}, \alpha_\ell]$ such that $\phi(\alpha') = 0$. Then we have $w^k = \alpha'$ and $\mathbf{x}^k$ as an optimal solution to $\mathrm{PQ}(\alpha')$. If no break point satisfies (5.3), then α reaches its maximum $\alpha_q = \max\{\mathbf{c}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in C \cap D\}$ and still satisfies $\phi(\alpha_q) < 0$. In that case, we have $C \cap D \subset H$; since $h(\mathbf{x}) \leq 0$ is redundant to $\mathrm{QL}(D)$, we may set α_q to w^k.

In the case that $\phi(\alpha_1) > 0$, we may solve PQ(α) as decreasing the value of α from α_1. Again, we have a sequence of intervals $[\alpha_2, \alpha_1], [\alpha_3, \alpha_2], \dots,$ and the piecewise affine function $\phi(\alpha)$ for $\alpha \leq \alpha_1$. If we can find a break point α_ℓ satisfying

$$\phi(\alpha_i) > 0, \quad i = 1, \dots, \ell - 1, \quad \phi(\alpha_\ell) \leq 0,$$

the rest of the procedure is the same as before. Otherwise, α reaches its minimum $\alpha_r = \min\{\mathbf{c}^\mathsf{T} \mid \mathbf{x} \in C \cap D\}$ and still satisfies $\phi(\alpha_r) > 0$. This implies $C \cap D \cap H = \emptyset$, and hence $w^k = -\infty$.

In these stages, we should remark that (5.2) is of the same form as PL(D), the linearized subproblem in Falk-Soland's algorithm. While α in PL(D) is treated as just a constant, (5.2) is solved via PQ(α) parametrically as changing the value of α from $\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$. In this connection, we also note that (5.2) is dual for the linearization QL(D) in the sense of Theorem 2.2 if $\mathbf{x}^{k-1}$ is optimal for the latter. This branch-and-bound algorithm, therefore, can be thought of as a method for solving the dual problem of each subproblem while Algorithm 4.1 tries to solve the dual problem of the original problem.

Let us summarize the above procedure, which receives h, $\mathbf{x}^{k-1}$, and returns the optimal value w^k of the linearized subproblem QL(D):

Procedure 5.1.

```
begin
                                                        /* Stage 1 */
  construct problem (5.2) of minimizing $h(\mathbf{x})$ on $C \cap D \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}\}$;
  if (5.2) is infeasible then return $w^k := -\infty$
  else begin
    let $\mathbf{x}'$ be an optimal solution to (5.2);
    if $\mathbf{c}^\mathsf{T}\mathbf{x}' = \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$ then
      if $h(\mathbf{x}') \geq 0$ then return $w^k := -\infty$
      else $\alpha_1 := \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$
    else begin
      if $h(\mathbf{x}') = 0$ then return $\mathbf{x}^k := \mathbf{x}'$ and $w^k := \mathbf{c}^\mathsf{T}\mathbf{x}'$
      else $\alpha_1 := \mathbf{c}^\mathsf{T}\mathbf{x}'$
    end;
                                                        /* Stage 2 */
    if $\phi(\alpha_1) < 0$ then begin
      solve PQ($\alpha$) parametrically as increasing $\alpha$ from $\alpha_1$;
      let $\alpha_2, \dots, \alpha_q$ be break points of the optimal value function $\phi$ of
      PQ($\alpha$);
      if $\phi(\alpha_i) < 0$ for $i = 1, \dots, \ell - 1$ and $\phi(\alpha_\ell) \geq 0$ then begin
```

```
            compute $w^k \in [\alpha_{\ell-1}, \alpha_\ell]$ such that $\phi(w^k) = 0$;
            let $\mathbf{x}^k$ be an optimal solution to PQ($w^k$) and return $\mathbf{x}^k$ and
            $w^k$
         **end**
         **else** let $\mathbf{x}^k$ be an optimal solution to PQ($\alpha_q$) and return $\mathbf{x}^k$ and
         $w^k := \alpha_q$
      **end**
      **else begin**
         solve PQ($\alpha$) parametrically as decreasing $\alpha$ from $\alpha_1$;
         let $\alpha_2, \ldots, \alpha_r$ be break points of $\phi$;
         **if** $\phi(\alpha_i) > 0$ for $i = \ldots, \ell - 1$ **and** $\phi(\alpha_\ell) \leq 0$ **then begin**
            compute $w^k \in [\alpha_{\ell-1}, \alpha_\ell]$ such that $\phi(w^k) = 0$;
            let $\mathbf{x}^k$ be an optimal solution to PQ($w^k$) and return $\mathbf{x}^k$ and
            $w^k$
         **end**
         **else** return $w^k := -\infty$
      **end**
   **end**
**end**;
```

5.2 Bounding and Branching

If an optimal solution $\mathbf{x}^k$ to QL(D) is not a point in G, then $\mathbf{x}^k$ is also an optimal solution to Q(D) and (5.1) holds with equality. Unfortunately, in general, $\mathbf{x}^k$ is not even a feasible solution to Q(D). To perform the bounding operation efficiently, however, we need a feasible solution giving a lower bound on (2.1) and have to update it timely. One way to find a feasible solution to (2.1) is to check if each solution to PQ(α) lies on G or not, in the second stage of solution to QL(D). Here, we will give a more handy approach.

Suppose that a feasible solution $\tilde{\mathbf{x}}$ to the original problem (2.1) is given and satisfies $g(\tilde{\mathbf{x}}) < 0$. If $g(\mathbf{x}^k) \geq 0$ holds, the line segment connecting $\mathbf{x}^k$ and $\tilde{\mathbf{x}}$ intersects ∂G at $\mathbf{x}' = (1 - \lambda)\mathbf{x}^k + \lambda\tilde{\mathbf{x}}$ for some $\lambda \in [0, 1]$. Such a point λ can be computed in a way similar to (4.1). This boundary point $\mathbf{x}'$ of G is a feasible solution to (2.1) because the segment $\mathbf{x}^k$–$\tilde{\mathbf{x}}$ is entirely included in the convex set $C \cap D^1$. We compute $\mathbf{x}'$ in each iteration if possible, and then update the incumbent and lower bound with $\mathbf{x}'$. The point $\tilde{\mathbf{x}}$ need not be determined beforehand. As the rectangular subdivision advances, some vertex of D^k, $k \in \mathcal{K}$, becomes feasible for (2.1). Hence, we may check if each vertex of D is feasible for (2.1) in each iteration. Since g is concave, checking requires $O(2^n)$ time. In the usual application, however, each g_j constituting g represents a cost of x_j and is nondecreasing. In that case, we need only to check the feasibility of vertex $\mathbf{l}$.

The branching operation can be performed in the same way as in Falk-Soland' algorithm, i.e., we can use both bisection and ω-division for selecting (j, m_j) in (3.7). Since the convergence by the bisection rule is obvious, we briefly discuss the case of ω-division. Let $\mathcal{L}$ denote the nested sequence of D^{k_i}, $i = 1, 2, \dots$, as defined in (3.8). If we generate the sequence $\mathcal{L}$ according to the ω-division rule, for each $i \in \mathcal{L}$ we select

$$j_i \in \arg\max\{g_j(x_j^i) - h_j(x_j^i) \mid j = 1, \dots, n\}, \tag{5.4}$$

and divide the interval $[l_{j_i}^i, u_{j_i}^i]$ at

$$m_{j_i}^i = x_{j_i}^i, \tag{5.5}$$

where $\mathbf{x}^i$ is an optimal solution to QL(D^i). From (5.4) and Lemma 3.3, there is a subsequence $\mathcal{L}' \subset \mathcal{L}$ such that $l_j^i \to l_j^*$ and $u_j^i \to u_j^*$ for each j as $i \to +\infty$ in $\mathcal{L}'$; and besides, l_j^* and u_j^* are accumulation points of m_j^i. This, together with (5.5), implies that $\mathbf{x}^i$ has an accumulation point among the vertices of $D^* = [l_1^*, u_1^*] \times \cdots \times [l_n^*, u_n^*]$. From Lemma 3.1, however, the convex envelope h agrees with g at each of these vertices; and hence the optimal values of Q(D^*) and QL(D^*) coincide. Thus, we have the following:

Lemma 5.3. *If $\mathcal{L}$ generated according to either of the bisection and ω-division rules, then there is a subsequence $\mathcal{L}' \subset \mathcal{L}$ such that*

$$w^i - w(D^i) \to 0 \quad \textit{as } i \to +\infty \textit{ in } \mathcal{L}'.$$

■

5.3 Description of the algorithm

Lastly, we need to decide the rule of selecting an index i_k from the set $\mathcal{K}$ in Step 1. As in Algorithm 3.1, we can adopt either of the depth first and best bound rules. We are now ready to describe the algorithm, where $\epsilon > 0$ is a given tolerance.

Algorithm 5.2.

```
begin
  construct the linearized problem QL($D$) of (2.1) and solve it to obtain
  $\mathbf{x}^1$;
  select $j \in \{1, \dots, n\}$ and $m_j \in [l_j, u_j]$ by a fixed rule (bisection or
  $\omega$-division);
  $D^2 := [l_1, u_1] \times \cdots \times [l_j, m_j] \times \cdots \times [l_n, u_n]$;
```

```
$D^3 := [l_1, u_1] \times \cdots \times [m_j, u_j] \times \cdots \times [l_n, u_n]$;
$D^1 := D$; $\mathcal{K} := \{2, 3\}$; $k := 2$; $w^* := -\infty$;
while $\mathcal{K} \neq \emptyset$ do
                                                            /* Step 1 */
  select an index $i_k$ from $\mathcal{K}$ by a fixed rule (depth first or best bound);
  $\mathcal{K} := \mathcal{K} \setminus \{i_k\}$; $D := D^{i_k}$;
  if $w^* = -\infty$ and some vertex $\mathbf{v}$ of $D$ lies on $C \cap D^1 \setminus \overline{G}$ then begin
    $\tilde{\mathbf{x}} := \mathbf{v}$; $w^* := \mathbf{c}^\mathsf{T}\tilde{\mathbf{x}}$
  end;
                                                            /* Step 2' */
  construct a subproblem Q($D$) and its linearization QL($D$);
  compute an optimal solution $\mathbf{x}^k$ and the value $w^k$ of QL($D$) using
  Procedure 5.1;
  if $w^k - w^* > \epsilon$ then begin
                                                            /* Step 3 */
      if $w^* > -\infty$ then begin
        let $\mathbf{x}'$ be an intersection point of $\mathbf{x}^k$–$\tilde{\mathbf{x}}$ and $\partial G$;
        if $\mathbf{c}^\mathsf{T}\mathbf{x}' > w^*$ then update $w^* := c^\mathsf{T}\mathbf{x}'$ and $\mathbf{x}^* := \mathbf{x}'$;
      end;
      select $j \in \{1, \ldots, n\}$ and $m_j \in [l_j, u_j]$ by a fixed rule;
      $D^{2k} := [l_1, u_1] \times \cdots \times [l_j, m_j] \times \cdots \times [l_n, u_n]$;
      $D^{2k+1} := [l_1, u_1] \times \cdots \times [m_j, u_j] \times \cdots \times [l_n, u_n]$;
      $\mathcal{K} := \mathcal{K} \cup \{2k, 2k+1\}$; $k := k + 1$
  end
 end
end;
```

The following results are analogous to Theorem 3.6 and Corollary 3.7:

Theorem 5.4. *When $\epsilon > 0$, Algorithm 5.2 terminates in a finite number of iterations and yields a globally ϵ-optimal solution $\mathbf{x}^*$ to problem (2.1).* ■

Corollary 5.5. *Suppose $\epsilon = 0$. If the best bound rule is adopted in Step 1, the sequence of $\mathbf{x}^k$'s generated by Algorithm 5.2 has accumulation points, each of which is a globally optimal solution to problem (2.1).* ■

6. Concluding remarks

We have seen that Falk-Soland's rectangular branch-and-bound algorithm can serve as a useful procedure in solving linear programs with an additional separable reverse convex constraint (LPASC). Since we have not yet compared Algorithms 4.1 and 5.2 with other algorithms, we can make no final conclusions about their computational properties. However, if we think of the success of Falk-Soland's algorithm in concave minimization, we can strongly expect Algorithms 4.1 and 5.2 using it in a direct or indirect manner to be reasonably practical.

As stated in Section 1, the rectangular branch-and-bound algorithm has made great progress since Falk-Soland [3]. Although we have used Falk-Soland's classical branch-and-bound in Algorithm 4.1, we can instead employ modern algorithms of this kind such as [11, 15]. These are reported to be more efficient than Falk-Soland's original algorithm. Therefore, such modification will improve the efficiency of Algorithm 4.1 considerably. In addition, we could incorporate devices of [15, 11] into Algorithm 5.2 because its structure is basically the same as Falk-Soland's branch-and-bound algorithm. We will report these improvements on Algorithms 4.1 and 5.2 elsewhere, together with their computational results.

References

[1] Baumol, W.J. and P. Wolfe, "A warehouse location problem", *Operations Research* **6** (1958), 252–263.

[2] Chvătal, V., *Linear Programming* (Freeman and Company, N.Y., 1983).

[3] Falk, J.E. and R.M. Soland, "An algorithm for separable nonconvex programming problems", *Management Science* **15** (1969), 550–569.

[4] fülöp, J., "A finite cutting plane method for solving linear programs with an additional reverse convex constraint", *European Journal of Operations Research* **44** (1990), 395–409.

[5] Hillestad, R.J., "Optimization problems subject to a budget constraint with economies of scale", *Operations Research* **23** (1975), 1091–1098.

[6] Hillestad, R.J. and S.E. Jacobsen, "Reverse convex programming", *Applied Mathematics and Optimization* **6** (1980), 63–78.

[7] Hillestad, R.J. and S.E. Jacobsen, "Linear programs with an additional reverse convex constraint", *Applied Mathematics and Optimization* **6** (1980), 257–269.

[8] Horst, R., P.M. Pardalos and N.V. Thoai, *Introduction to Global Optimization*, Kluwer Academic Publishers (Dordrecht, 1995).

[9] Horst, R. and N.V. Thoai, "Global minimization of separable concave functions under linear constraints with totally unimodular matrices", *State of the Art in Global Optimization*, Kluwer Academic Publishers (Dordrecht, 1996).

[10] Horst, R. and H. Tuy, *Global Optimization: Deterministic Approaches*, 2nd ed,, Springer-Verlag (Berlin, 1993).

[11] Kuno, T., "A finite branch-and-bound algorithm for linear multiplicative programming", to appear in *Computational Optimization and Applications.*

[12] Moshirvaziri, K. and M. Amousegar, "A subdivision scheme for linear programs with an additional reverse convex constraint", *Asia-Pacific Journal of Operations Research* **15** (1998), 179–192.

[13] Muu, L.D., "A convergent algorithm for solving linear programs with an additional reverse convex constraint", *Kybernetika* **21** (1985), 428–435.

[14] Pferschy, U. and H. Tuy, "Linear programs with an additional rank two reverse convex constraint", *Journal of Global Optimization* **4** (1994), 441–454.

[15] Ryoo, H.S. and N.V. Sahinidis, "A branch-and-reduce approach to global optimization", *Journal of Global Optimization* **8** (1996), 107–138.

[16] Soland, R.M., "Optimal facility location with concave costs", *Operations Research* **22** (1974), 373–382.

[17] Thuong, T.V. and H. Tuy, "A finite algorithm for solving linear programs with an additional reverse convex constraint", *Lecture Notes in Economics and Mathematical Systems* **225**, Springer-Verlag (Berlin, 1984).

[18] Tuy, H., "Convex programs with an additional reverse convex constraint", *Journal of Optimization Theory and Applications* **52** (1987), 463–486.

[19] Tuy, H., "D.c. optimization: theory, methods and algorithms", *Handbook of Global Optimization*, Kluwer Academic Publishers (Dordrecht, 1995), 149–216.

[20] Tuy, H. and T.V. Thuong, "On the global minimization of a convex function under general nonconvex constraints", *Applied Mathematics and Optimization* **18** (1988), 119–142.