JOURNAL OF APPLIED MATHEMATICS AND DECISION SCIENCES, 8(3), 155–174

Linear Programs With an Additional Separable Concave Constraint

TAKAHITO KUNO† takahito@is.tsukuba.ac.jp
Institute of Information Sciences and Electronics
University of Tsukuba, Ibaraki 305-8573, Japan

JIANMING SHI shi@csse.muroran-it.ac.jp
Department of Computer Science and Systems Engineering
Muroran Institute of Technology, Muroran, Hokkaido 050-8585, Japan

Abstract. In this paper, 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.

1. Introduction

In this paper, 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 $G \subset R^n$ 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:

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

† Requests for reprints should be sent to Takahito Kuno, Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan.

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

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

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 portion 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 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 have been developed: Soland [16], Horst-Thoai [9], Kuno [11], Ryoo-Sahinidis [15] to name but a few. The purpose of this paper is to develop practical algorithms by making full use of 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 also explained in detail. Sections 3 and 4 are devoted to the algorithms for LPASC. In Section 3, 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. Basic Properties and Folk-Soland's Algorithm

The problem we consider in this paper is of the form

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

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

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

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.

2.1. Dual problem of LPASC

Let $\overline{M}$ denote the closure, and ∂M the boundary of a set M. We assume the following hereafter:

Assumptions. Problem (2.1) satisfies

(a) $C \cap D \setminus G \neq \emptyset$.

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

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

These assumptions are not specific to our problem but often imposed on d.c. optimization problems. In fact, if (b) fails, then (2.1) is equivalent to an ordinary linear program

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

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 (a) and bounded. Assumption (c) means that there is a point $\mathbf{y} \in C \cap D$ in any neighborhood of each feasible solution such that $g(\mathbf{y}) < 0$. Under Assumptions (a)–(c), we can obtain two important theorems, even when g is inseparable (see e.g., [8], [10] for their proofs).

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

THEOREM 2.2 *Let* $\mathbf{x}^* \in C \cap D \setminus G$. *Then* $\mathbf{x}^*$ *is an optimal solution to (2.1) if and only if* $\min\{g(\mathbf{x}) \mid \mathbf{x} \in C \cap D,\ \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^*\} = 0$.

The problem given in Theorem 2.2, i.e.,

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

plays a crucial role in solution to (2.1) when we try to exploit the separability of the side constraint $g(\mathbf{x}) = \sum_{j=1}^{n} g_j(x_j) \leq 0$. This problem is called the *dual problem* of (2.1) because it has the same optimal solution $\mathbf{x}^*$ as the original problem (2.1) if $\alpha = \mathbf{c}^\mathsf{T}\mathbf{x}^*$, though the objective and side constraint are reversed.

2.2. Falk-Soland's algorithm for PD(α)

Since PD(α) is a separable concave minimization, we can solve it using the rectangular branch-and-bound algorithm by Falk-Soland [3], which is well known as the most powerful tool in global optimization. We will run through the mechanism of Falk-Soland's algorithm. Let

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ -\mathbf{c}^\mathsf{T} \end{bmatrix}, \quad \mathbf{b}' = \begin{bmatrix} \mathbf{b} \\ -\alpha \end{bmatrix}, \quad C' = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}'\mathbf{x} \leq \mathbf{b}'\}. \tag{2.3}$$

and rewrite PD(α) as follows

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

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

$$D^k = [l_1^k, u_1^k] \times [l_2^k, u_2^k] \times \cdots \times [l_n^k, u_n^k], \quad k \in \mathcal{K}, \tag{2.4}$$

we solve each subproblem P(D^k). 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

$$D = \bigcup_{k \in \mathcal{K}} D^k, \quad \mathrm{int}D^i \cap \mathrm{int}D^k = \emptyset \ \text{ if } \ i \neq k, \tag{2.5}$$

where int$M$ denotes the interior of a set M.

The outline of the algorithm consists of three basic steps. Let $\epsilon \geq 0$ be a given tolerance for the optimal value of PD(α).

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 $z^k + \epsilon \geq g(\mathbf{x}^*)$ for the best feasible solution $\mathbf{x}^*$ to $\mathrm{PD}(\alpha)$ 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 tight lower bound z^k by solving a linear program.

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 $j = 1, \ldots, n$:

$$h_j(x_j) = \frac{g_j(u_j) - g_j(l_j)}{u_j - l_j} x_j + \frac{u_j g_j(l_j) - l_j g_j(u_j)}{u_j - l_j}. \tag{2.6}$$

From the concavity of g_j, we see that

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

where we should remark that $h_j(x_j) = g_j(x_j)$ if $x_j \in \{l_j, u_j\}$. Since g is the sum of g_j's, this property is inherited to

$$h(\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j). \tag{2.7}$$

LEMMA 2.1 *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 $\mathrm{P}(D)$:

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

Let $\mathbf{x}^k$ denote an optimal solution to 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 2.2 *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 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 pivoting operations.

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 (2.4) and (2.5). We can carry out this as follows, given an index $j \in \{1, \ldots, n\}$ and a number $m_j \in [l_j, u_j]$:

$$\left.\begin{aligned} D^{2k} &= [l_1, u_1] \times \cdots \times [l_{j-1}, u_{j-1}] \times [l_j, m_j] \\ &\qquad \times [l_{j+1}, u_{j+1}] \times \cdots \times [l_n, u_n] \\ D^{2k+1} &= [l_1, u_1] \times \cdots \times [l_{j-1}, u_{j-1}] \times [m_j, u_j] \\ &\qquad \times [l_{j+1}, u_{j+1}] \times \cdots \times [l_n, u_n] \end{aligned}\right\} \tag{2.8}$$

In general, no matter how we select j and m_j, the algorithm is not ensured to be finite if $\epsilon = 0$. In that case, it generates an infinite sequence of rectangles D^{k_i}, $i = 1, 2, \ldots$, such that

$$D^{k_1} \supset D^{k_2} \supset \cdots, \quad C' \cap \left(\bigcap_{i=1}^{\infty} D^{k_i} \right) \neq \emptyset. \tag{2.9}$$

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 $\mathcal{L} = \{1, 2, \ldots, i, \ldots\}$. We assume that for each $i \in \mathcal{L}$, rectangle D^{i+1} is generated from D^i via (2.8) for some pair $(j_i, m_{j_i}^i)$.

LEMMA 2.3 *There is an infinite subsequence $\mathcal{L}_q \subset \mathcal{L}$ such that $j_i = q$ for all $i \in \mathcal{L}_q$, and we have $i_q^i \to l_q^*$, $u_q^i \to u_q^*$ and $m_q^i \to m_q^* \in \{l_q^*, u_q^*\}$ as $i \to +\infty$ in $\mathcal{L}_q$.*

When $\epsilon > 0$, the rules below of selecting (j, w_j) guarantee the finiteness of the algorithm by the next lemma.

Bisection. For each $i \in \mathcal{L}$, letting

$$j_i \in \arg\max\{u_j^i - l_j^i \mid j = 1, \ldots, n\},$$

then the interval $[l_{j_i}^i, u_{j_i}^i]$ is divided at $m_{j_i}^i = (1-\lambda)l_{j_i}^i + \lambda u_{j_i}^i$, where $\lambda \in (0,1)$ is a constant.

ω-division. For each $i \in \mathcal{L}$, letting

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

then the interval $[l_{j_i}^i, u_{j_i}^i]$ is divided at $m_{j_i}^i = x_{j_i}^i$.

LEMMA 2.4 *If $\mathcal{L}$ is generated according to either bisection of ratio $\lambda \in (0,1)$ or ω-division, 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.

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 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 $\mathcal{K}$.

If we adopt the latter, even when $\epsilon = 0$, the sequence of $\mathbf{x}^k$'s generated by the rectangular branch-and-bound algorithm has accumulation points, each of which is a globally optimal solution to PD(α).

3. Direct Application of Falk-Soland's Algorithm

A straightforward way to solve our problem (2.1) by exploiting its separability is to apply Falk-Soland's algorithm to the dual problem PD(α) for an appropriate α. To fix the value of α, Tuy [18], Tuy-Thuong [20] and Pferschy-Tuy [14] have suggested 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 PD(α) with $\alpha = \mathbf{c}^\mathsf{T}\mathbf{x}^*$. If the optimal value of 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.

3.1. 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($-\infty$), using Falk-Soland's algorithm. Since the objective function g of PD($-\infty$) is concave, it achieves the minimum at some vertex $\mathbf{v}^1 \in C \cap D$. By Assumption (c), we must have $g(\mathbf{v}^1) < 0$. To be precise, Falk-Soland's algorithm might not yield a vertex of $C \cap D$ when its tolerance $\epsilon > 0$; 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

$$\lambda^* = \min\{\lambda \in [0, 1] \mid g[(1 - \lambda)\mathbf{v}^{\ell-1} + \lambda\mathbf{v}^\ell] \geq 0\}. \tag{3.1}$$

Since (3.1) 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 the target (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).

3.2. 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 Falk-Soland's algorithm 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, Falk-Soland's algorithm 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 Falk-Soland's algorithm with this ϵ as tolerance. Then the algorithm must terminate in finite time and yield 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 $\mathbf{x}''$ 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 Falk-Soland's algorithm 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.

3.3. Description of the algorithm

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

Algorithm 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 ∂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 $\mathbf{c}^{\mathsf{T}}\mathbf{x}' = \alpha$;
    $C' := C \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \alpha\}$;
    solve PD(α) using Falk-Soland's algorithm with tolerance $\epsilon := g(\mathbf{x}')$ 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 3.1 *When $\delta > 0$, Algorithm 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 1 follows this. □

4. 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 rather 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, in the bounding operation. 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 constitutes a partition of D. However, the problem to be solved for each partition set D^k is not $\mathrm{P}(D^k)$ but a subproblem 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 $\epsilon \geq 0$. The outline of the algorithm is as follows:

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 $w^k - \epsilon \leq \mathbf{c}^{\mathsf{T}}\mathbf{x}^*$ 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.

4.1. Linearization and its solution

As shown in Lemma 2.1, the convex envelope h of g defined in (2.6) and (2.7) satisfies $h(\mathbf{x}) \leq g(\mathbf{x})$ for all $\mathbf{x} \in D$. Hence, by letting

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

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

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

LEMMA 4.1 *If $w^k=-\infty$, then $Q(D)$ is infeasible. Otherwise, the following relationship holds:*

$$w^k\geq w(D). \tag{4.1}$$

Since h is an affine function, $\mathrm{QL}(D)$ 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

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

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

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 $\mathrm{QL}(D)$ in two stages starting from an optimal solution $\mathbf{x}^{k-1}$ of the proceeding linear program. In both stages, the matrix we mainly deal with is not $\mathbf{A}''$ but $\mathbf{A}'$ defined in (2.3).

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 4.2 *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:

$$\left| \begin{array}{ll} \text{minimize} & h(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C \cap D, \ \mathbf{c}^\mathsf{T}\mathbf{x} \geq \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}. \end{array} \right. \tag{4.2}$$

Three cases can occur:

Case 1: Problem (4.2) is infeasible. In this case, $\phi(\mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}) = +\infty$ and 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 (4.2) 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 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 (4.2) 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 $h(\mathbf{x}')$ 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

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

using the parametric right-hand-side simplex algorithm [2]. Then it generates a sequence of intervals $[\alpha_1, \alpha_2]$, $[\alpha_2, \alpha_3]$, ..., and a sequence of bases $\mathbf{B}^1$, $\mathbf{B}^2$, ... of $\mathbf{A}'$ such that $\mathbf{B}^i$ is optimal for PQ(α) when $\alpha \in [\alpha_i, \alpha_{i+1}]$.

The optimal value $\phi(\alpha)$ of PQ(α) is affine on each interval $[\alpha_i, \alpha_{i+1}]$. If we find a break point α_ℓ satisfying

$$\phi(\alpha_i) < 0, \quad i = 1, \ldots, \ell - 1, \quad \phi(\alpha_\ell) \geq 0, \tag{4.3}$$

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 PQ(α'). If no break point satisfies (4.3), 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 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 $[\alpha_2, \alpha_1]$, $[\alpha_3, \alpha_2]$, ..., and the piecewise affine function $\phi(\alpha)$ for $\alpha \leq \alpha_1$. If we can find a break point α_ℓ satisfying

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

the rest of the procedure is the same as before. Otherwise, α reaches its minimum $\alpha_r = \min\{\mathbf{c}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in C \cap D\}$ 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 (4.2) 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, (4.2) 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 (4.2) 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 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 2
begin
/* *Stage 1* */
construct problem (4.2) 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 (4.2) is infeasible then return $w^k := -\infty$
else begin
let $\mathbf{x}'$ be an optimal solution to (4.2);
if $\mathbf{c}^\mathsf{T}\mathbf{x}' = \mathbf{c}^\mathsf{T}\mathbf{x}^{k-1}$ then

```
            if h(x') ≥ 0 then return w^k := −∞
            else α_1 := c^T x^{k−1}
        else begin
            if h(x') = 0 then return x^k := x' and w^k := c^T x'
            else α_1 := c^T x'
        end;
                                                    /* Stage 2 */
        if φ(α_1) < 0 then begin
            solve PQ(α) parametrically as increasing α from α_1;
            let α_2, ..., α_q be break points of the optimal value
            function φ of PQ(α);
            if φ(α_i) < 0 for i = 1, ..., ℓ − 1 and (α_ℓ) ≥ 0
            then begin
                compute w^k ∈ [α_{ℓ−1}, α_ℓ] such that φ(w^k) = 0;
                let x^k be an optimal solution to PQ(w^k) and
                return x^k and w^k
            end
            else let x^k be an optimal solution to PQ(α_q) and
            return x^k and w^k := α_q
        end
        else begin
            solve PQ(α) parametrically as decreasing α from α_1;
            let α_2, ..., α_r be break points of φ;
            if φ(α_i) > 0 for i = 1, ..., ℓ − 1 and φ(α_ℓ) ≤ 0
            then begin
            compute w^k ∈ [α_{ℓ−1}, α_ℓ] such that φ(w^k) = 0;
            let x^k be an optimal solution to PQ(w^k) and return x^k
            and w^k
        end
        else return w^k := −∞
    end
end;
```

4.2. 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 (4.1) holds with equality. Unfortunately,

in general, $\mathbf{x}^k$ is not even a feasible solution to $\mathrm{Q}(D)$. 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 $\mathrm{PQ}(\alpha)$ lies on G or not, in the second stage of solution to $\mathrm{QL}(D)$. Here, we will give a more handy approach.

Suppose that a feasible solution $\widetilde{\mathbf{x}}$ to the original problem (2.1) is given and satisfies $g(\widetilde{\mathbf{x}}) < 0$. If $g(\mathbf{x}^k) \geq 0$ holds, the line segment connecting $\mathbf{x}^k$ and $\widetilde{\mathbf{x}}$ intersects ∂G at $\mathbf{x}' = (1-\lambda)\mathbf{x}^k + \lambda\widetilde{\mathbf{x}}$ for some $\lambda \in [0,1]$. Such a number λ can be computed in a way similar to (3.1). This boundary point $\mathbf{x}'$ of G is a feasible solution to (2.1) because the segment $\mathbf{x}^k$–$\widetilde{\mathbf{x}}$ 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 $\widetilde{\mathbf{x}}$ 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's algorithm, i.e., we can use both bisection and ω-division for selecting (j, m_j) in (2.8). 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}, $i = 1, 2, \ldots$, as defined in (2.9). If we generate the sequence $\mathcal{L}$ according to the ω-division rule, for each $i \in \mathcal{L}$ we select

$$j_i \in \arg\max\{g_j(x_j^i) - h_j(x_j^i) \mid j = 1, \ldots, n\}, \tag{4.4}$$

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

$$m_{j_i}^i = x_{j_i}^i, \tag{4.5}$$

where $\mathbf{x}^i$ is an optimal solution to $\mathrm{QL}(D^i)$. From (4.4) and Lemma 2.3, 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 (4.5), 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 2.1, however, the convex envelope h agrees with g at each of these vertices; and hence the optimal values of $\mathrm{Q}(D^*)$ and $\mathrm{QL}(D^*)$ coincide. Thus, we have the following:

LEMMA 4.3 *If $\mathcal{L}$ is generated according to either of the bisection and ω-division rules, then there is a sequence $\mathcal{L}' \subset \mathcal{L}$ such that*

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

4.3. 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 Falk-Soland's algorithm, 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 3
begin
construct the linearized problem QL(D) of (2.1) and solve it to obtain $\mathbf{x}^1$;
select $j \in \{1, \ldots, m\}$ and $m_j \in [l_j, u_j]$ by a fixed rule (*bisection or ω-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 $\widetilde{\mathbf{x}} := \mathbf{v}$; $w^* := \mathbf{c}^{\mathsf{T}}\widetilde{\mathbf{x}}$
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 2;
if $w^k - \epsilon > w^*$ then begin
/* *Step 3* */
if $w^* > -\infty$ then begin
let $\mathbf{x}'$ be an intersection point of $\mathbf{x}^k$–$\widetilde{\mathbf{x}}$ and ∂G;
if $\mathbf{c}^{\mathsf{T}}\mathbf{x}' > w^*$ then update $w^* := \mathbf{c}^{\mathsf{T}}\mathbf{x}'$ and $\mathbf{x}^* := \mathbf{x}'$

```
                end;
                select j ∈ {1, ..., n} and m_j ∈ [l_j, u_j] by a fixed rule;
                D^{2k} := [l_1, u_1] × ··· × [l_j, m_j] × ··· × [l_n, u_n];
                D^{2k+1} := [l_1, u_1] × ··· × [m_j, u_j] × ··· × [l_n, u_n];
                K := K ∪ {2k, 2k + 1}; k := k + 1
            end
        end
    end;
```

The following results are analogous to those of Falk-Soland's algorithm.

THEOREM 4.1 *When $\epsilon > 0$, Algorithm 3 terminates in a finite number of iterations and yields a globally ϵ-optimal solution $\mathbf{x}^*$ to problem (2.1).*

Corollary 4.1 *Suppose $\epsilon = 0$. If the best bound rule is adopted in Step 1, the sequence of $\mathbf{x}^k$'s generated by Algorithm 3 has accumulation points, each of which is a globally optimal solution to problem (2.1).*

5. 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 Algorithm 1 and 3 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 1 and 3 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 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 1 considerably. In addition, we could incorporate devices of [11], [15] into Algorithm 3 because its structure is basically the same as Falk-Soland's branch-and-bound algorithm.

Acknowledgments

Supported by Grant-in-Aid for Scientific Research of Japan Society for the Promotion of Sciences, C(2) 13680505 and C(2) 14550405.

References

1. Baumol, W.L. and P. Wolfe. A warehouse location problem. *Operations Research*, 6:252–263, 1958.
2. Chvátal, V. *Linear Programming*. Freeman and Company, N.Y., 1983.
3. Falk, J.E. and R.M. Soland. An algorithm for separable nonconvex programming roblems. *Management Science*, 15: 550–569, 1969.
4. Fülöp, J. A finite cutting plane method for solving linear programs with an additional reverse convex constraint. *European Journal of Operations Research*, 44:395–409, 1990.
5. Hillestad, R.J. Optimization problems subject to a budget constraint with economies of scale. *Operations Research*, 23:1091–1098, 1975.
6. Hillestad, R.J. and S.E. Jacobsen. Reverse convex programming. *Applied Mathematics and Optimization*, 6:63–78, 1980.
7. Hillestad, R.J. and S.E. Jacobsen. Linear programs with an additional reverse convex constraint. *Applied Mathematics and Optimization*, 6:257–269, 1980.
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, 35-45, 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. *Computational Optimization and Applications*, 20:119–136, 2001.
12. Moshirvaziri, K. and M. A. Amouzegar. A subdivision scheme for linear programs with an additional reverse convex constraint. *Asia-Pacific Journal of Operations Research*, 15:179–192, 1998.
13. Muu, L.D. A convergent algorithm for solving linear programs with an additional reverse convex constraint. *Kybernetika*, 21:428–435, 1985.
14. Pferschy, U. and H. Tuy, Linear programs with an additional rank two reverse convex constraint. *Journal of Global Optimization*, 4:441–454, 1994.
15. Ryoo, H.S. and N.V. Sahinidis. A branch-and reduce approach to global optimization. *Journal of Global Optimization*, 8:107–138, 1996.
16. Soland, R.M. Optimal facility location with concave costs. *Operations Research*, 22:373–382, 1974.
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 255*. Springer-Verlag, Berlin, 291-302, 1984.
18. Tuy, H. Convex programs with an additional reverse convex constraint. *Journal of Optimization Theory and Applications*, 52:463–486, 1987.
19. Tuy, H. D.c. optimization: theory, methods and algorithms. *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, 149–216, 1995.
20. Tuy, H. and T.V. Thuong. On the global minimization of a convex function under general nonconvex constraints. *Applied Mathematics and Optimization*, 18:119–142, 1988.