*Journal of Global Optimization* **26:** 229–259, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

# A Unified Monotonic Approach to Generalized Linear Fractional Programming

NGUYEN THI HOAI PHUONG and HOANG TUY
*Institute of Mathematics, PO Box 631, Bo Ho, Hanoi, Vietnam (e-mail: htuy@hn.vnn.vn)*

**Abstract.** We present an efficient unified method for solving a wide class of generalized linear fractional programming problems. This class includes such problems as: optimizing (minimizing or maximizing) a pointwise maximum or pointwise minimum of a finite number of ratios of linear functions, optimizing a sum or product of such ratios, etc. – over a polytope. Our approach is based on the recently developed theory of monotonic optimization.

**Key words:** Generalized fractional programming, Sum or product of ratios of linear functions, Monotonic optimization, Global optimization, Polyblock approximation approach

## 1. Introduction

In a variety of applications we encounter a class of nonconvex optimization problems which have either of the following formulations:

$$\max\left\{\Phi\left(\frac{f_1(x)}{g_1(x)},\dots,\frac{f_m(x)}{g_m(x)}\right) \mid x\in D\right\} \tag{P}$$

$$\min\left\{\Phi\left(\frac{f_1(x)}{g_1(x)},\dots,\frac{f_m(x)}{g_m(x)}\right) \mid x\in D\right\} \tag{Q}$$

where $D$ is a nonempty polytope in $R^n$, $f_1,\dots,f_m,g_1,\dots,g_m$ are linear affine functions on $R^n$ such that

$$-\infty < a_i := \min_{x\in D}\frac{f_i(x)}{g_i(x)} < +\infty \quad i=1,\dots,m, \tag{1}$$

while $\Phi : R^m \to R$ is a continuous function, *increasing* on $R^m_{a+} := \{y\in R^m \mid y_i \geqslant a_i\ (i=1,\dots,m)\}$, i.e. satisfying

$$a_i \leqslant y'_i \leqslant y_i\ (i=1,\dots,m) \quad\Rightarrow\quad \Phi(y') \leqslant \Phi(y). \tag{2}$$

Important special cases of these problems that have been previously studied in the literature include, aside from linear multiplicative programs [10], various generalized linear fractional programs, namely:

1. *Maxmin and Minmax*

$$\max\left\{\min\left(\frac{f_1(x)}{g_1(x)},\dots,\frac{f_m(x)}{g_m(x)}\right) \mid x\in D\right\} \quad (\Phi(y)=\min\{y_1,\dots,y_m\}) \tag{3}$$

$$\min\left\{\max\left(\frac{f_1(x)}{g_1(x)},\dots,\frac{f_m(x)}{g_m(x)}\right) \mid x\in D\right\} \quad (\Phi(y)=\max\{y_1,\dots,y_m\}) \tag{4}$$

2. *Maxsum and Minsum*

$$\max\left\{\sum_{i=1}^m \frac{f_i(x)}{g_i(x)} \mid x\in D\right\} \qquad \left(\Phi(y)=\sum_{i=1}^m y_i\right) \tag{5}$$

$$\min\left\{\sum_{i=1}^m \frac{f_i(x)}{g_i(x)} \mid x\in D\right\} \qquad \left(\Phi(y)=\sum_{i=1}^m y_i\right) \tag{6}$$

3. *Maxproduct and Minproduct*

$$\max\left\{\prod_{i=1}^m \frac{f_i(x)}{g_i(x)} \mid x\in D\right\} \qquad \left(\Phi(y)=\prod_{i=1}^m y_i\right) \tag{7}$$

$$\min\left\{\prod_{i=1}^m \frac{f_i(x)}{g_i(x)} \mid x\in D\right\} \qquad \left(\Phi(y)=\prod_{i=1}^m y_i\right) \tag{8}$$

(For the last two problems it is assumed that $a_i > 0,\ i=1,\dots,m$, in condition (1))

Aside from the above mentioned problems, the classes (P) and (Q) include a variety of other problems of interest which, to our knowledge, have been little studied in the literature so far, such as e.g.

$$\max\left\{\sum_{j=1}^r c_j \prod_{i=1}^m \left[\frac{f_i(x)}{g_i(x)}\right]^{\alpha_{ij}} \mid x\in D\right\} \qquad \left(\Phi(y)=\sum_{j=1}^r c_j\prod_{i=1}^m [y_i]^{\alpha_{ij}}\right) \tag{9}$$

where $r$ is a natural number and $c_j, \alpha_{ij}$ are real positive numbers and (1) holds with $a_i > 0, i=1,\dots,m$.

Standard linear fractional programs which are special cases of problems (P), (Q) when $m=1$ were first considered by Charnes and Cooper [5]. Problems (3), (5) have received much attention from researchers (see [18] and references therein), problems (5), (6) have been studied in [7, 9, 12], and problems (7), (8) in [9, 13] (see also [11]). Other results related to fractional programming can be found in [1, 2, 19]. For a review on fractional programming and extensions, up to 1993, we refer the reader to ref. [18], where various, actual and potential, applications of fractional programming are also described. In general, fractional programming arises in situations where one would like to optimize functions of such ratios as return/investment, return/risk, cost/time, output/input, etc. Let us also mention the book [17] which is a comprehensive presentation of the theory, methods and applications of fractional programming.

From a computational point of view, so far different ingenious methods have been proposed for solving different variants of problems (P) and (Q). A general unifying idea of parametrization underlies the methods developed by Konno and his associates ([12], [13], [14]; see also [11]). Each of these methods is devised for a specific class of problems, and most of them work quite well for problem instances with $m \leqslant 3$. For example, the parametric method in [12], valid under the assumption

$$\min_{x \in D} f_i(x) \geqslant 0 \quad \min_{x \in D} g_i(x) > 0, \quad i = 1, \ldots, m \tag{10}$$

is reported to have successfully solved problems (5) with $m \leqslant 4$. A phenomenon quite common for this class of nonconvex global optimization problems is that the required computational time sharply increases as $m$ increases. According to [9], the time needed for solving problems with $m = 3$ by the parametric algorithm in [12] is about 50 times more than for $m = 2$ by the parametric simplex algorithm. This motivates the development of heuristic methods to cope with the “curse of dimensionality”. One such method has been recently proposed in [9] for minimizing the sum of three linear fractional functions. Reported computational experiments show that this heuristic method gives almost the same results as exact methods but is ten times faster.

The aim of the present paper is to develop a unified approach to all variants of problems (P) and (Q), based on the recent theory of monotonic optimization [22]. By providing a general numerical method that seems to work equally well on different variants of these problems (at least for currently considered values of $m$), this approach once more illustrates the wide applicability of monotonic optimization as discussed in a number of previous studies ([15, 16, 23]).

The paper is organized as follows. In Section 2 we will first briefly review the concepts and properties of normal sets and polyblocks, as were introduced and investigated in [22]. These constitute the basic tools of our approach. Several modifications will be made, however, to the basic procedure in [22], so for the sake of completeness, we will provide a proof for the most essential facts.

In Section 3 we will reformulate problems (P) and (Q) as monotonic optimization problems, namely: maximizing an increasing function over a normal set or minimizing it over a reverse normal set. The proposed solution method will be presented in Section 4, and some implementation issues discussed in Section 5. Section 6 is devoted to some improved versions of the basic algorithm. Finally Section 7 presents numerical examples illustrating the practicability of the proposed approach (at least for problem instances with $m \leqslant 7$).

## 2. Normal Set and Polyblock

Throughout this paper, for any two vectors $y, y' \in R^m$ we shall write $y' \leqslant y$ (and say that $y$ *dominates* $y'$) to mean that $y'_i \leqslant y_i \ \forall i = 1, \dots, m$. If $z \in R^n_+$, the hyperrectangle $[0, z] = \{y \in R^m_+ | \ 0 \leqslant y \leqslant z\}$ is called a *box*. Given any finite set $T \subset R^m_+$ the union of all the boxes $[0, z], \ z \in T$, is called a *polyblock* with vertex set $T$. A vertex $z \in T$ is said to be *proper* if $z$ is not dominated by any other vertex, i.e. if there is no $z' \in T$ such that $z' \neq z$ and $z' \geqslant z$. Obviously a polyblock is fully determined by its proper vertices.

A set $G \subset R^m_+$ is called *normal* if $y \in G$ always implies that $[0, y] \subset G$. A polyblock, in particular a box, is normal. The orthant $R^m_+$ and the emptyset are also normal sets. The intersection of any family of normal sets is obviously a normal set. The intersection of finitely many polyblocks is a polyblock. More specifically:

PROPOSITION 1. *The intersection of two polyblocks $P_1, P_2$ with vertex sets $T_1, T_2$ is a polyblock $P$ with vertex set*

$$T = \{z \wedge z' | \ z \in T_1, z' \in T_2\}, \quad \textit{where } (z \wedge z')_i = \min\{z_i, z'_i\} \ i = 1, \dots, m. \tag{11}$$

*Proof.* $P_1 \cap P_2 = (\cup_{z \in T_1}[0, z]) \cap (\cup_{z' \in T_2}[0, z']) = \cup\{[0, z] \cap [0, z']\}$ where the union is taken over all pairs $z \in T_1, z' \in T_2$. Noting that $[0, z] \cap [0, z'] = [0, z \wedge z']$ we get the result. □

For an arbitrary subset $S$ of $R^m_+$ the intersection of all normal sets containing $S$ is called the *normal hull* of $S$: it is of course the smallest normal set containing $S$. Clearly, a polyblock is the normal hull of its vertex set. It is also easily seen that the normal hull of $S$ consists of all $y \in R^m_+$ for which there exists $z \in S$ such that $y \leqslant z$. If $D \subset R^n$ and $w : D \to R^m_+$ is any nonnegative-valued function on $D$ then the set $G = \{y \in R^m_+ | \ y \leqslant w(x), \ x \in D\}$ is normal. It can be verified that $G$ is the normal hull of the set $S = \{y = w(x) | \ x \in D\}$.

A point $y \in R^m_+$ is called an *upper boundary point* of a nonempty normal set $G \subset R^m_+$ if $\alpha y \in G \ \forall \alpha < 1$ but $\alpha y \notin G \quad \forall \alpha > 1$. The set of upper boundary points of $G$ is called the *upper boundary* of $G$ and is denoted by $\partial^+ G$.

Clearly, for every $z \in R^m_+ \setminus \{0\}$ the halfline from 0 through $z$ meets the upper boundary $\partial^+ G$ of $G$ at a unique point $\pi_G(z)$ defined by

$$\pi_G(z) = \lambda z, \quad \lambda = \sup\{\alpha \geqslant 0 | \ \alpha z \in G\}. \tag{12}$$

We have $\lambda \geqslant 1$, i.e. , $\pi_G(z) \geqslant z$ if and only if $z \in G$.

PROPOSITION 2. *Let $\Phi(y) : R^m_+ \to R$ be an increasing function, i.e. a function satisfying (2). The maximum of $\Phi(y)$ over a polyblock is attained at one proper vertex of this polyblock. The maximum of $\Phi(y)$ over a nonempty compact normal set G is attained on $\partial^+ G$.*

*Proof.* Let $\bar{z}$ be a maximizer of $\Phi(y)$ over a polyblock $P$. If $\bar{z}$ is not a proper vertex of $P$ then $\bar{z} \leqslant \tilde{z}$ for some proper vertex $\tilde{z}$ and since $\Phi(\tilde{z}) \geqslant \Phi(\bar{z})$ it follows that $\tilde{z}$ is also a maximizer. Likewise, if $\bar{z} \in G$ is a maximizer of $\Phi(y)$ over $G$ then so is $\pi_G(\bar{z})$, because $\pi_G(\bar{z}) \geqslant \bar{z}$. □

PROPOSITION 3. *Let $G$ be a nonempty compact normal set contained in a box $[0, b] \subset R^m_+$ and $P \subset [0, b]$ be a polyblock containing $G$, with proper vertex set $T$. For a given $\hat{z} \in T \setminus G$, with $\pi_G(\hat{z}) = \hat{y}$, let $T'$ be any subset of $T$ such that $\hat{z} \in T' \subset \{z \in T \mid z > \hat{y}\}$, and let $T^*$ be the set obtained from $T$ by replacing each $z \in T'$ with $\{z^{*1}, \ldots, z^{*m}\}$ where $z^{*i}$ are given by*

$$z^{*i} = z - (z_i - \hat{y}_i)e^i, \qquad e^i = i\text{-th unit vector of } R^m_+.$$

*Then the polyblock $P^*$ with vertex set $T^*$ satisfies $G \subset P^* \subset P \setminus \{\hat{z}\}$.*

*Proof.* Let $K$ be the cone formed by all $y > \hat{y}$. Clearly $G \cap K = \emptyset$ for if $y \in G \cap K$ then for $\varepsilon > 0$ small enough we would still have $(1+\varepsilon)\hat{y} < y$ (because $\hat{y} < y$) while $(1+\varepsilon)\hat{y} \in G$ (because $[0, y] \subset G$ by normality of $G$), contradicting the definition of $\hat{y} = \pi_G(\hat{z})$. Now for any finite set $E$ denote by $[E]$ the polyblock with vertex set $E$. For every $z \in T'$, since $G \cap K = \emptyset$ we have $G \cap [0, z] \subset [0, z] \setminus K$, while, as can be easily verified, $[0, z] \setminus K \subset [\{z^{*1}, \ldots, z^{*m}\}]$, hence $G \cap [0, z] \subset [\{z^{*1}, \ldots, z^{*m}\}]$. Since $P = [T'] \cup [T \setminus T']$, the conclusion follows. □

REMARK 1. In one extreme case when $T' = \{\hat{z}\}$ the polyblock $P^*$ is largest among all polyblocks constructed as above. In the other extreme case when $T'$ includes all $z \in T$ such that $z > \hat{y}$ the polyblock $P^*$ is smallest among all these polyblocks.

## 3. Preliminary Transformations

Note that condition (1) is in general weaker than (10) which is assumed by most authors when studying (3), (5) or (4), (6). In particular, we do not exclude the case of (5) or (6) when the objective function is the difference (rather than the sum) of two linear fractional functions:

$$\frac{f_1(x)}{g_1(x)} - \frac{f_2(x)}{g_2(x)}$$

where $g_i(x) > 0\ \forall x \in D$ but $f_i(x), i = 1, 2$, may be arbitrary (in this case $\Phi(y) = y_1 + y_2, y_1 = \frac{f_1(x)}{g_1(x)}, y_2 = \frac{-f_2(x)}{g_2(x)}$).

However, by simple manipulations we can always reduce the problem to the case when

$$\min\{g_i(x), f_i(x)\} > 0 \quad \forall x \in D, \quad i = 1, \ldots, m. \tag{13}$$

Indeed, under (1) $g_i(x)$ does not vanish on $D$, hence has a constant sign on $D$ and, by replacing $f_i, g_i$ by their negatives if necessary, we can always assume $g_i(x) > 0\ \forall x \in D,\ i = 1, \dots, m$. Then setting $\tilde{f}_i(x) := f_i(x) - a_i g_i(x)$ we have, by (1), $\tilde{f}_i(x) \geqslant 0\ \forall x \in D,\ i = 1, \dots, m$, and since $\frac{\tilde{f}_i(x)}{g_i(x)} + a_i = \frac{f_i(x)}{g_i(x)}$, the problem (P) is equivalent to

$$\max \left\{ \tilde{\Phi}\left( \frac{\tilde{f}_1(x)}{g_1(x)}, \dots, \frac{\tilde{f}_m(x)}{g_m(x)} \right) \mid x \in D \right\}$$

where $\tilde{f}_i(x) > 0\ \forall x \in D,\ i = 1, \dots, m$ and $\tilde{\Phi}(y_1, \dots, y_m) = \Phi(y_1 + a_1, \dots, y_m + a_m)$ is increasing on $R^m_+$ in view of (2). Analogously for the problem (Q). Therefore, without loss of generality we can assume (13) and $\Phi : R^m_+ \to R$ in both problems (P) and (Q). Also to simplify certain argument it is convenient to assume that $\Phi : R^m_+ \to R_{++}$ (by adding a positive constant to $\Phi$).

Under this assumption we now reformulate problems (P) and (Q) as monotonic optimization problems. Define

$$G = \left\{ y \in R^m_+ \mid\ y_i \leqslant \frac{f_i(x)}{g_i(x)}\ (i = 1, \dots, m),\ x \in D \right\} \tag{14}$$

$$H = \left\{ y \in R^m_+ \mid\ y_i \geqslant \frac{f_i(x)}{g_i(x)}\ (i = 1, \dots, m),\ x \in D \right\}. \tag{15}$$

Clearly $G$ and $[0, b] \setminus H$ are normal sets contained in the box $[0, b]$ with

$$b_i = \max_{x \in D} \frac{f_i(x)}{g_i(x)} \quad i = 1, \dots, m. \tag{16}$$

It is then easily seen that

THEOREM 1. *Problems (P) and (Q) are equivalent, respectively, to the following problems (MP), (MQ):*

$$\max\{\Phi(y) \mid\ y \in G\} \tag{MP}$$

$$\min\{\Phi(y) \mid\ y \in H\} \tag{MQ}$$

*More precisely, if $\bar{x}$ solves (P) ((Q), resp.) then $\bar{y}$ with $\bar{y}_i = \frac{f_i(\bar{x})}{g_i(\bar{x})}$ solves (MP) ((MQ), resp.). Conversely, if $\bar{y}$ solves (MP) ((MQ), resp.) and $\bar{x} \in D$ satisfies $\bar{y}_i \leqslant \frac{f_i(\bar{x})}{g_i(\bar{x})}$ ($\bar{y}_i \geqslant \frac{f_i(\bar{x})}{g_i(\bar{x})}$, resp.), $i = 1, \dots, m$, then $\bar{x}$ solves (P) ((Q), resp.)*

*Proof.* Suppose that $\bar{x}$ solves (P). Then $\bar{y} = \left( \frac{f_1(\bar{x})}{g_1(\bar{x})}, \dots, \frac{f_m(\bar{x})}{g_m(\bar{x})} \right)$ satisfies $\Phi(\bar{y}) \geqslant \Phi(y)$ for every $y$ such that $y_i = \frac{f_i(x)}{g_i(x)},\ i = 1, \dots, m$, with $x \in D$, and since $\Phi(.)$ is increasing, this implies $\Phi(\bar{y}) \geqslant \Phi(y)\ \forall y \in G$, i.e. $\bar{y}$ solves (MP). Conversely, if $\bar{y}$ solves (MP) then $\bar{y}_i \leqslant \frac{f_i(\bar{x})}{g_i(\bar{x})},\ i = 1, \dots, m$, for some $\bar{x} \in D$, and since $\Phi(.)$ is increasing, $\Phi\left( \frac{f_1(\bar{x})}{g_1(\bar{x})}, \dots, \frac{f_m(\bar{x})}{g_m(\bar{x})} \right) \geqslant \Phi(\bar{y}) \geqslant \Phi(y)$ for every $y \in G$, and hence, for every $y$ such that $y_i = \frac{f_i(x)}{g_i(x)}$, $i = 1, \dots, m$, with $x \in D$, implying that $\bar{x}$ solves (P). □

Thus the original problems (P), (Q) in $R^n$ can be reduced to the above problems (MP), (MQ) in $R^m_+$, with $m$ usually much smaller than $n$. By specializing then the 'polyblock approximation method' developed in [22] to (MP), we obtain a unified method for solving any problem of the form (P). Furthermore, as we shall show later, a problem (MQ) can be transformed into a problem (MP). Therefore, the same method applies also to any problem of the form (Q). This unified method turns out to be quite practical for all problems under consideration, independently of the specific form of $\Phi(.)$ in each of them.

## 4. Proposed Solution Method

### 4.1. PROBLEM (MP)

Consider first the problem (MP):

$$\max\{\Phi(y)|\ y \in G\} \tag{MP}$$

where $G = \{y \in R^m_+ | y_i \leqslant \frac{f_i(x)}{g_i(x)} \quad \forall i = 1, \dots, m,\ x \in D\}$ is contained in the box $[0, b]$ with $b$ satisfying (16), i.e.

$$b_i = \max_{x \in D} \frac{f_i(x)}{g_i(x)} \quad i = 1, \dots, m.$$

In view of the assumption (13), we can select a vector $a \in R^m_{++}$ such that

$$a_i = \min_{x \in D} \frac{f_i(x)}{g_i(x)} > 0 \quad i = 1, \dots, m, \tag{17}$$

so the search can be restricted to the set $G \cap L$, where

$$L = \{y \in R^m_+ |\ y \geqslant a\}. \tag{18}$$

(One can assume $a_i < b_i$, $i = 1, \dots, m$ because if $a_i = b_i$ for some $i$ then $f_i(x)/g_i(x) = a_i\ \forall x \in D$ and $\Phi(.)$ reduces to a function of $m - 1$ variables.)

Let $\gamma$ be the optimal value of (MP) (or (P)) ($\gamma > 0$ since $\Phi : R^m_+ \to R_{++}$). Given a tolerance $\varepsilon \geqslant 0$, we say that a vector $\bar{y} \in G \cap L$ is an $\varepsilon$-optimal solution of (MP) if $\gamma \leqslant (1 + \varepsilon)\Phi(\bar{y})$, or equivalently, if $\Phi(y) \leqslant (1 + \varepsilon)\Phi(\bar{y})\ \forall y \in G \cap L$. Then the vector $\bar{x} \in D$ satisfying $\bar{y}_i = f_i(\bar{x})/g_i(\bar{x})$, $i = 1, \dots, m$, is called an $\varepsilon$-optimal solution of (P). We now present a procedure for finding an $\varepsilon$-optimal solution.

Clearly if $z \notin L$ then the box $[0, z]$ contains no point of $L$, hence no point of $G \cap L$. Keeping this in mind, observe that by Proposition 2, if we take a polyblock

$P_1 \supset G$ with vertex set $T_1 \subset L$, then the maximum of $\Phi(y)$ over $P_1$ is achieved at some proper vertex $z^1$ of $P_1$. Since $z^1 \in L$ if this vertex $z^1$ happens to belong to $G$ then it solves (MP); on the other hand, if $z^1 \notin G$ then by Proposition 3 we can construct a smaller polyblock $P_2$ still containing $G \cap L$ but excluding $z^1$. Repeating this procedure for the new polyblock $P_2$, and continuing as long as needed, we will generate a nested sequence of polyblocks enclosing $G \cap L$ :

$$P_1 \supset P_2 \supset \ldots \supset G \cap L. \tag{19}$$

The problem will be solved if we can construct this sequence in such a manner that either $z^k \in G \cap L$ for some $k$, or

$$\max\{\Phi(y)|\ y \in P_k\} \searrow \max\{\Phi(y)|\ y \in G \cap L\}. \tag{20}$$

In the former case $z^k$ is an exact optimal solution of the problem, while in the latter case, by stopping the sequence at some sufficiently advanced iteration $k$ we will obtain an $\varepsilon$-optimal solution.

Let us examine how such a sequence can be constructed. Starting with the polyblock $P_1 := [0, b] \supset G \cap L$, whose proper vertex set is $T_1 = \{b\}$, suppose that $P_1, \ldots, P_k$ satisfying (19) have already been constructed. Let $T_k$ be the proper vertex of $P_k$. By deleting all $z \in T_k$ such that $z \notin L$ we can assume $T_k \subset L$ (see (18)). Then the point

$$z^k \in \text{argmax}\{\Phi(z)|z \in T_k\}$$

is a maximizer of $\Phi(y)$ over $P_k$. Let

$$y^k = \pi_G(z^k),$$

so that $\max\{y_i^k - f_i(x^k)/g_i(x^k)|\ i = 1, \ldots, m\} = 0$ for some $x^k \in D$. Define $\tilde{y}^k$ such that $\tilde{y}_i^k = f_i(x^k)/g_i(x^k),\ i = 1, \ldots, m$. This vector $\tilde{y}^k$ is a feasible solution of (MP) with objective function value $\Phi(\tilde{y}^k) \geqslant \Phi(y^k)$ and it can be compared with previously known feasible solutions to determine the current best solution CBS$= \bar{y}^k$ and the current best value $CBV = \Phi(\bar{y}^k)$. We then construct $P_{k+1}$ as follows.

Select a subset $T_k'$ of $T_k$ such that $z^k \in T_k' \subset \{z \in T_k|\ z > y^k\}$, and let $T_k^*$ be the set obtained from $T_k$ by replacing each $z \in T_k'$ with the points $\{z^{*1}, \ldots, z^{*m}\}$ defined by

$$z^{*i} = z - (z_i - y_i^k)e^i \quad i = 1, \ldots, m. \tag{21}$$

From $T_k^*$ remove all improper elements as well as all points not belonging to $L$, and let $\tilde{T}_k$ be the remaining set. By Proposition 3 the polyblock with vertex set $\tilde{T}_k$ still contains $G \cap L$ but is smaller than $P_k$ and no longer contains $z^k$.

To further reduce the enclosing polyblock, observe that if every $z \in \tilde{T}_k$ satisfies $\Phi(z) \leqslant (1+\varepsilon)CBV$ then $\Phi(y) \leqslant (1+\varepsilon)CBV\ \forall y \in G \cap L$ (by monotonicity of

$\Phi)$, and consequently, CBV is the $\varepsilon$-optimal value and CBS an $\varepsilon$-optimal solution. Therefore we check whether $\Phi(z) \leqslant (1+\varepsilon)CBV$ for every $z \in \tilde{T}_k$. Let $T_{k+1}$ be the set that remains from $\tilde{T}_k$ after dropping all $z$ that satisfy that condition and also all $z \in \tilde{T}_k \setminus L$.

If $T_{k+1} = \emptyset$, we stop: $\Phi(y) \leqslant (1+\varepsilon)CBV\ \forall y \in G \cap L$, so $\bar{y}^k$ with $\Phi(\bar{y}^k) = CBV$ is an $\varepsilon$-optimal solution of (MP). The vector $\bar{x}^k \in D$ such that $\bar{y}^k_i = f_i(\bar{x}^k)/g_i(\bar{x}^k),\ i = 1, \dots, m$, is then an $\varepsilon$-optimal solution of (P).

On the other hand, if $T_{k+1} \neq \emptyset$, then the polytope $P_{k+1}$ with proper vertex set $T_{k+1}$ will still contain at least an $\varepsilon$-optimal solution, i.e. will satisfy the condition (19) in which $G \cap L$ is replaced by a set containing at least an $\varepsilon$-optimal solution. So we can go to the next iteration and continue the procedure.

In a formal way we can thus state

**Algorithm 1** (tolerance $\varepsilon \geqslant 0$).

Step 0. (Initialization) Start with $T_1 = \tilde{T}_1 = \{b\}$. Set $k = 1$.

Step 1. Select

$$z^k \in \operatorname{argmax}\{\Phi(z) | z \in \tilde{T}_k\}$$

Compute $y^k = \pi_G(z^k)$ with $x^k \in D$ satisfying $y^k_i \leqslant f_i(x^k)/g_i(x^k)\ \forall i = 1, \dots, m$. Define $\tilde{y}^k \in G$ by $\tilde{y}^k_i = f_i(x^k)/g_i(x^k),\ i = 1, \dots, m$ (so $\tilde{y}^k \in G \cap L$, i.e. is feasible). Determine the new current best solution $\bar{y}^k$ (with corresponding $\bar{x}^k \in D$) and the new current best value $CBV = \Phi(\bar{y}^k)$, by comparing $\tilde{y}^k$ with $\bar{y}^{k-1}$ (for $k = 1$ set $\bar{y}^1 = \tilde{y}^1, \bar{x}^1 = x^1$).

Step 2. Select a set $T'_k \subset T_k$ such that $z^k \in T'_k \subset \{z |\ z > y^k\}$ and let $T^*_k$ be the set obtained from $T_k$ by replacing each $z \in T'_k$ with the points $\{z^{*1}, \dots, z^{*m}\}$ defined by (21).

Step 3. From $T^*_k$ remove all improper elements, all $z \in \tilde{T}_k$ such that $\Phi(z) \leqslant (1+\varepsilon)CBV$ and also all $z \notin L$. Let $T_{k+1}$ be the set of remaining elements. If $T_{k+1} = \emptyset$, terminate: $\bar{y}^k$ is an $\varepsilon$-optimal solution of (MP), and the corresponding $\bar{x}^k$ an $\varepsilon$-optimal solution of (P).

Step 4. If $T_{k+1} \neq \emptyset$, set $k \leftarrow k+1$ and return to Step 1.

THEOREM 2. *Algorithm 1 can be infinite only if* $\varepsilon = 0$ *and in this case it generates an infinite sequence* $\{\bar{x}^k\}$ *every cluster point of which is an optimal solution of (P).*

*Proof.* This follows from Theorem 1 in [22]. For completeness let us sketch the proof. If the algorithm is infinite it generates an infinite sequence $\{z^k\}$ such that

$$\begin{aligned} &z^1 = b,\ z^{k+1} = z^k - (z^k_{i_k} - y^k_{i_k})e^{i_k}; \\ &y^k = \pi_G(z^k),\ i_k \in \{1, \dots, m\}. \end{aligned} \tag{22}$$

Since $z^1 \geqslant z^2 \geqslant \cdots \geqslant z^k \geqslant \cdots \geqslant a > 0$ this sequence has a limit $\bar{z} = \lim_{k\to\infty} z^k$, and consequently, $\lim_{k\to\infty} \|z^{k+1} - z^k\| = 0$. From (22) we have $z^{k+1}_{i_k} = y^k_{i_k}$, hence,

$$z^k_{i_k} - y^k_{i_k} = z^k_{i_k} - z^{k+1}_{i_k} \leqslant \|z^k - z^{k+1}\| \to 0 \quad (k \to +\infty). \tag{23}$$

But by construction, $z^k_{i_k} - y^k_{i_k} = \lambda_k z^k_{i_k}$, while $z^k \geqslant a > 0$, therefore $z^k_{i_k} - y^k_{i_k} \geqslant \lambda_k a_{i_k}$. This, together with (23), implies $\lambda_k \to 0$, i.e.

$$z^k - y^k \to 0 \quad (k \to \infty). \tag{24}$$

Hence, any cluster point $\bar{z}$ of $\{z^k\}$ satisfies $\bar{z} \in G \cap L$. Since $\Phi(z^k) \geqslant \Phi(y)\ \forall y \in G \cap L$, it follows that $\Phi(\bar{z}) \geqslant \Phi(y)\ \forall y \in G \cap L$, i.e $\bar{z}$ is an optimal solution. Furthermore, since $\Phi(z^k) \geqslant \Phi(\bar{y}^k) \geqslant \Phi(\tilde{y}^k) \geqslant \Phi(y^k)$, it follows from (24) that any cluster point of the sequence $\{\bar{y}^k\}$ is also an optimal solution. Hence, any cluster point of the sequence $\{\bar{x}^k\}$ is an $\varepsilon$-optimal solution of (P). □

REMARK 2. In Step 2 of the Algorithm some freedom exists for the selection of $T'_k$. If $T'_k = \{z^k\}$ is taken as in the Basic Polyblock Algorithm in [22] there are at most $m$ new vertices but the polyblock $P_{k+1}$ may sometimes be too rough an approximation of $G$. On the other hand, if a larger set $T'_k$ is selected, the polyblock $P_{k+1}$ better approximates $G$, but too many new vertices may appear. This flexibility can be exploited for the efficiency of the procedure.

## 4.2. PROBLEM (MQ)

Consider now the problem (MQ):

$$\min\{\Phi(y)|\ y \in H\} \tag{MQ}$$

where $H = \{y \in R^m_+|\ y_i \geqslant \frac{f_i(x)}{g_i(x)} \quad \forall i = 1, \ldots, m,\ x \in D\}$.

Since an optimal solution must lie in the box $[0, b]$, with $b_i = \max_{x\in D} \frac{f_i(x)}{g_i(x)},\ i = 1, \ldots, m$, by setting $\tilde{\Phi}(y) = -\Psi(b - y)$, $H^\flat = b - H \cap [0, b]$ we can rewrite (MQ) as

$$\max\{\Psi(y)|\ y \in H^\flat\} \tag{MQ$^\flat$}$$

Clearly $\Psi(y)$ is an increasing function on $[0, b]$ and $H^\flat$ is a normal set in $[0, b]$, so the problem (MQ$^\flat$) has now the same form as (MP) and hence can be solved by the same method.

A set $H \subset [0, b]$ such that $0 \leqslant y' \leqslant y \notin H$ always implies $y' \notin H$ is called a *reverse normal set* in $[0, b]$. For any finite set $T \subset [0, b]$ the set $Q = \cup_{z\in T}[z, b]$ is called a *reverse polyblock* of vertex set $T$. A vertex $z \in T$ is called *proper* if there is no $z' \in T$ such that $z' \neq z$ and $z' \leqslant z$. For any $z \in [0, b]$ we define

$$\rho_H(z) = b - \mu(b - z), \quad \mu = \max\{\alpha > 0|\ b - \alpha(b - z) \in H\}. \tag{25}$$

Clearly, $z \notin H$ if and only if $\mu < 1$ and then $\rho_H(z) \in \partial^- H$, where $\partial^- H$ denotes the lower boundary of $H$ (set of all points $y \in H$ such that $y' \notin H$ for all $y' < y$). The counterpart to Proposition 3 is now:

PROPOSITION 4. *Let $H$ be a closed reverse normal set with $b \in \mathrm{int} H$, where $b \in R^m_{++}$ and let $Q \subset [0, b]$ be a reverse polyblock containing $H \cap [0, b]$ and with proper vertex set $T$. For a given $\hat{z} \in T \setminus H$ with $\rho_H(\hat{z}) = \hat{y}$, let $T'$ be any subset of $T$ such that $\hat{z} \in T' \subset \{y |\ y < \hat{y}\}$, and let $T^*$ be the set obtained from $T$ by replacing each $z \in T'$ with $\{z^{*1}, \dots, z^{*m}\}$ where $z^{*i}$ are given by*

$$z^{*i} = z + (y_i - z_i)e^i \quad i = 1, \dots, m.$$

*Then the reverse polyblock $Q^*$ with vertex set $T^*$ satisfies $(H \cap [0, b]) \subset Q^* \subset Q \setminus \{\hat{z}\}$.*

In terms of $H$ and $\Phi(y)$ the polyblock method for solving (MQ) consists in constructing a nested sequence of reverse polyblocks $Q_1 := [0, b] \supset Q_2 \supset \cdots \supset H \cap [0, b]$ such that

$$\min\{\Phi(y) |\ y \in Q_k\} \nearrow \min\{\Phi(y) |\ y \in H\}.$$

Noting that $H^\flat = \{y |\ y_i \leqslant b_i - f_i(x)/g_i(x)\}$, and $\max_{x \in D}\{b_i - f_i(x)/g_i(x)\} = b_i - \min_{x \in D}\{f_i(x)/g_i(x)\}$ we see that the search can be restricted to the set

$$H \cap K, \quad K = \{y \in R^m_+ |\ y \leqslant \bar{b}\} \tag{26}$$

where $\bar{b}$ is defined by

$$\bar{b}_i = b_i - \min_{x \in D} \frac{f_i(x)}{g_i(x)}, \ i = 1, \dots, m. \tag{27}$$

**Algorithm 2** (tolerance $\varepsilon \geqslant 0$)

Step 0. (Initialization) Start with $T_1 = \tilde{T}_1 = \{0\}$. Set $k = 1$.

Step 1. Select

$$z^k \in \mathrm{argmin}\{\Phi(z) | z \in \tilde{T}_k\}$$

Compute $y^k = \rho_H(z^k)$ with $x^k \in D$ satisfying $y^k_i \geqslant f_i(x^k)/g_i(x^k)\ \forall i = 1, \dots, m$. Define $\tilde{y}^k$ such that $\tilde{y}^k_i = f_i(x^k)/g_i(x^k),\ i = 1, \dots, m$. Determine the current best solution $\bar{y}^k$ and the current best value $CBV = \Phi(\bar{y}^k)$, by comparing $\tilde{y}^k$ with $\bar{y}^{k-1}$.

Step 2. Select a set $T'_k \subset T_k$ such that $z^k \in T'_k \subset \{y |\ y < y^k\}$ and let $T^*_k$ be the set obtained from $T_k$ by replacing each $z \in T'_k$ with the points $\{z^{*1}, \dots, z^{*m}\}$ where

$$z^{*i} = z + (y^k_i - z_i)e^i \quad i = 1, \dots, m.$$

Step 3. From $T_k^*$ remove all improper elements, all $z$ such that $\Phi(z) \geqslant (1-\varepsilon)CBV$ and all $z \notin K$. Let $T_{k+1}$ be the set of remaining elements. If $T_{k+1} = \emptyset$, terminate: $\bar{y}^k$ is an $\varepsilon$-optimal solution of (MQ) and the corresponding $\bar{x}^k$ is an $\varepsilon$-optimal solution of (Q).

Step 4. If $T_{k+1} \neq \emptyset$, set $k \leftarrow k+1$ and return to Step 1.

THEOREM 3. *Algorithm 2 can be infinite only if* $\varepsilon = 0$ *and in this case it generates an infinite sequence* $\{x^k\}$, *every cluster point of which is an optimal solution of (Q).*

*Proof.* This follows from Theorem 2. □

## 5. Implementation Issues

### 5.1. COMPUTING $\pi_g(.)$ OR $\rho_h(.)$

One fundamental operation involved in Step 1 of Algorithm 1 (Algorithm 2, resp.) is the computation of the point $\pi_G(z)$ ($\rho_H(z)$, resp.), for a given $z$.

Recall from (5) that $\pi_G(z) = \lambda z$ where $\lambda$ is defined as follows

$$\begin{aligned}\lambda &= \max\{\alpha \mid \alpha z \in G\}\\ &= \max\{\alpha \mid \alpha \leqslant \min_{i=1,\dots,m} \frac{f_i(x)}{z_i g_i(x)}, x \in D\}\\ &= \max_{x\in D} \min_{i=1,\dots,m} \frac{f_i(x)}{z_i g_i(x)}.\end{aligned} \tag{28}$$

Therefore, computing $\pi_G(z)$ amounts to solving a problem (3). This can be done, e.g., by applying the following method adapted from [8].

Consider the subproblem

$$R(\alpha) := \max_{x\in D} \min_{i=1,\dots,m} (f_i(x) - \alpha z_i g_i(x)) \qquad \text{LP}[\alpha]$$

equivalent to the linear program

$$\max\{t \mid t \leqslant f_i(x) - \alpha z_i g_i(x)\ (i = 1, \dots, m),\ x \in D\}.$$

PROPOSITION 5.

(i) *We have* $R(\alpha) = 0 \Leftrightarrow \lambda = \alpha$; $\quad R(\alpha) < 0 \Leftrightarrow \lambda < \alpha$; $\quad R(\alpha) > 0 \Leftrightarrow \alpha < \lambda$.

(ii) *Let* $\{\alpha_s, s = 1, 2, ...\}$ *be a sequence such that* $R(\alpha_1) > 0$, $\tilde{\alpha}_s = \min_i \frac{f_i(x^s)}{z_i g_i(x^s)}$, $\alpha_{s+1} = (1+\eta)\tilde{\alpha}_s$, *where* $x^s$ *is an optimal solution of* LP[$\alpha_s$] *and* $\eta \in (0, 1)$ *is a constant. Then* $\alpha_s < \alpha_{s+1}$, *while* $R(\alpha_s) > R(\alpha_{s+1})$ *and the smallest* $s$ *such that* $R(\alpha_{s+1}) < 0$, *satisfies* $\tilde{\alpha}_s \leqslant \lambda < (1+\eta)\tilde{\alpha}_s$.

*Proof.* (i) Clearly $R(\alpha) > 0 \Leftrightarrow \max_{x\in D}\min_i[f_i(x) - \alpha z_i g_i(x)] > 0 \Leftrightarrow \max_{x\in D}$ $\min_i \frac{f_i(x)}{z_i g_i(x)} > \alpha \Leftrightarrow \lambda > \alpha$. Similarly, $R(\alpha) < 0 \Leftrightarrow \lambda < \alpha$. Hence $R(\alpha) = 0 \Leftrightarrow$ $\alpha = \lambda$.

(ii) If $R(\alpha_s) > 0$ then $\max_{x\in D}\min_i(f_i(x) - \alpha_s z_i g_i(x)) > 0$, i.e. $\min_i \frac{f_i(x)}{z_i g_i(x)} >$ $\alpha_s\ \forall x \in D$, hence $\tilde{\alpha}_s := \min_i \frac{f_i(x^s)}{z_i g_i(x^s)} > \alpha_s$, which in turn implies that $\alpha_{s+1} >$ $\alpha_s$, $R(\alpha_{s+1}) < R(\alpha_s)$. Since $R(\alpha_1) > 0$ the construction of the sequence $\{\alpha_1, \alpha_2, \dots\}$ can continue as long as $R(\alpha_s) > 0$. But, since $\alpha_{s+1} > (1+\eta)\alpha_s$ the fact $0 < \alpha_s < \lambda$ by (i) as long as $R(\alpha_s) > 0$ implies that one must have $R(\tilde{\alpha}_s) \geqslant 0 > R(\alpha_{s+1})$ for some $s$. Then $\tilde{\alpha}_s \leqslant \lambda < \alpha_{s+1}$ by virtue of (i). □

For a given $z \in [0, b]$ with $\pi_G(z) = \lambda z$ and for a given tolerance $\eta > 0$, define $y \approx \pi_G(z)$ to be a vector $y$ satisfying

$$y = (1+\eta)\hat{\lambda}z, \qquad \hat{\lambda} \leqslant \lambda \leqslant (1+\eta)\hat{\lambda} \tag{29}$$

Proposition 5 justifies the following procedure for computing $y \approx \pi_G(z)$.

PROCEDURE $\pi_G(z)$

1. With $\alpha_1 = 0$ solve LP$[\alpha_1]$ to obtain an optimal solution $x^1$ of it. Define $\tilde{\alpha}_1 = \min_i \frac{f_i(x^0)}{z_i g_i(x^0)}$, $s = 1$.
2. With $\alpha_{s+1} = \tilde{\alpha}_s(1+\eta)$ solve LP$[\alpha_{s+1}]$ to obtain an optimal solution $x^{s+1}$ of it. If $R(\alpha_{s+1}) \leqslant 0$, terminate: $y = \alpha_{s+1}z$ and $\tilde{y}^s_i = f_i(x^{s+1})/g_i(x^{s+1})$, $i = 1, \dots, m$.
3. If $R(\alpha_{s+1}) > 0$, set $\tilde{\alpha}_{s+1} = \min_i \frac{f_i(x^{s+1})}{z_i g_i(x^{s+1})}$, increment $s$ and go back to Step 2.

To guarantee termination of Algorithm 1 the tolerance $\eta$ in computing $y \approx \pi_G(z)$ must be suitably chosen. In fact, at iteration $k$ of Algorithm 1, for $y^k = \pi_G(z^k)$ we have $z^k - y^k \to 0$ (see (24)), so the termination criterion $\Phi(z^k) - (1+\varepsilon)CBV \leqslant$ $\Phi(z^k) - (1+\varepsilon)\Phi(y^k) \leqslant 0$ is satisfied when $k$ is sufficiently large. This is true if $y^k = \pi_G(z^k)$ exactly. Since however, we only have $y^k \approx \pi_G(z^k)$, this termination criterion may never occur if the tolerance $\eta$ is taken too large.

To address this issue observe that if $\eta$ is such that $y \approx \pi_G(z)$ (see (29)) satisfies

$$\Phi((1+\eta)\hat{\lambda}z) < (1+\varepsilon/2)\Phi(\hat{\lambda}z). \tag{30}$$

then $\Phi(y) \leqslant (1+\varepsilon/2)\Phi(\pi_G(z))$ (because $\hat{\lambda}z \leqslant \pi_G(z) \leqslant y := (1+\eta)\hat{\lambda}z$) and for $k$ so large that $\Phi(z^k) - \Phi(y^k) \leqslant \frac{\varepsilon}{2}\Phi(\pi_G(z^k))$ (which occurs because $z^k - y^k \to 0$) we shall have

$$\begin{aligned}(1+\varepsilon)CBV &\geqslant (1+\varepsilon)\Phi(\tilde{y}^k) \geqslant (1+\varepsilon)\Phi(\pi_G(z^k))\\ &\geqslant (1+\varepsilon/2)\Phi(\pi_G(z^k)) + (\varepsilon/2)\Phi(\pi_G(z^k))\\ &\geqslant \Phi(y^k) + (\varepsilon/2)\Phi(\pi_G(z^k)) \geqslant \Phi(y^k) + [\Phi(z^k) - \Phi(y^k)]\\ &= \Phi(z^k),\end{aligned}$$

i.e., the termination criterion will hold (in the previous argument we have used the fact that $\Phi(\tilde{y}^k) \geqslant \Phi(\pi_G(z^k))$).

One way to ensure (30) is to modify Step 2 of the above procedure as follows

Step 2. With $\alpha_{s+1} = \tilde{\alpha}_s(1+\eta)$ solve LP$[\alpha_{s+1}]$ to obtain an optimal solution $x^{s+1}$ of it. If $R(\alpha_{s+1}) \leqslant 0$ and $\Phi(\alpha_{s+1}z) \leqslant (1+\varepsilon/2)\Phi(\tilde{\alpha}_s z)$, terminate: $y = \alpha_{s+1}z$ and $\tilde{y}_i^s = f_i(x^{s+1})/g_i(x^{s+1}), i = 1,\dots,m$. If $R(\alpha_{s+1}) \leqslant 0$ but $\Phi(\alpha_{s+1}z) > (1+\varepsilon/2)\Phi(\tilde{\alpha}_s z)$, set $\eta \leftarrow \eta/2$, $\tilde{\alpha}_{s+1} = \min_i \frac{f_i(x^{s+1})}{z_i g_i(x^{s+1})}$, increment $s$ and go back to Step 2.

In many cases it is also possible to select $\eta$ in terms of $\varepsilon$. For instance, suppose the function $\Phi(.)$ is positively homogenous of degree $\theta$, i.e., satisfies $\Phi(\alpha y) = \alpha^\theta \Phi(y)\ \forall y \in R^m_+$, $\forall \alpha > 0$ (this holds for $\Phi(z) = \sum_{i=1}^m z_i$ with $\theta = 1$, and for $\Phi(z) = \Pi_{i=1}^m z_i$ with $\theta = m$). In such cases if we take $\eta \leqslant (1+\varepsilon/2)^{\frac{1}{\theta}} - 1$, then $R(\alpha_k(1+\eta)) \leqslant 0$ will imply that $\Phi((1+\eta)\alpha z) = (1+\eta)^\theta \Phi(\alpha z) \leqslant [(1+\varepsilon/2)^{\frac{1}{\theta}}]^\theta \Phi(\alpha z) = (1+\varepsilon/2)\Phi(\alpha z)$. Practically, it suffices to take $\eta = \varepsilon/2m$.

The computation of $\rho_H(.)$ is analogous to that of $\pi_G(.)$. Given the reverse normal set $H$ defined by (15) and a vector $b \in H$ the computation of $\rho_H(z) = z + \mu(b-z)$ for any $z \in [0,b] \setminus H$ amounts to computing

$$\begin{aligned}\mu &= \max\{\alpha \mid b - \alpha(b-z) \in H\} \\ &= \max\{\alpha \mid b_i - \alpha(b_i - z_i) \geqslant \frac{f_i(x)}{g_i(x)}\ (i = 1,\dots,m),\ x \in D\} \\ &= \max_{x\in D} \min_{i=1,\dots,m} \frac{b_i g_i(x) - f_i(x)}{(b_i - z_i)g_i(x)}\end{aligned} \tag{31}$$

So computing $\rho_H(z)$ again amounts to solving a problem (3).

Define the subprogram

$$S(\alpha) := \max_{x\in D} \min_{i=1,\dots,m} [b_i g_i(x) - f_i(x) - \alpha(b_i - z_i)g_i(x)] \quad \text{LQ}[\alpha]$$

or, equivalently,

$$\max\{t \mid t \leqslant b_i g_i(x) - f_i(x) - \alpha(b_i - z_i)g_i(x)\ (i = 1,\dots,m),\ x \in D\}.$$

Analogously to Proposition 5 one can prove that:

(i) $S(\alpha) = 0 \Leftrightarrow \alpha = \mu$; $\quad S(\alpha) > 0 \Leftrightarrow \alpha < \mu$; $\quad S(\alpha) < 0 \Leftrightarrow \alpha > \mu$;

(ii) Let $\{\alpha_s\}$ be a sequence such that $S(\alpha_1) > 0$, $\tilde{\alpha}_s = \min_i \frac{b_i g_i(x^s) - f_i(x^s)}{(b_i - z_i)g_i(x^s)}$, $\alpha_{s+1} = (1+\eta)\tilde{\alpha}_s$, where $x^s$ is an optimal solution of LQ$[\alpha_s]$ and $\eta \in (0,1)$ is a constant. Then there exists $s$ such that $S(\alpha_s) \geqslant 0$ but $S(\alpha_{s+1}) < 0$, and then $\tilde{\alpha}_s \leqslant \mu < (1+\eta)\tilde{\alpha}_s$.

For any given $z \in [0,b]$ with $\rho_H(z) = b - \mu(b-z)$ write $y \approx \rho_H(z)$ to denote a point $y$ satisfying

$$y = b - (1+\eta)\hat{\mu}(b-z), \qquad \hat{\mu} \leqslant \mu \leqslant (1+\eta)\hat{\mu}, \tag{32}$$

The above stated properties justify the next procedure for computing $y \approx \rho_H(z)$.

PROCEDURE $y \approx \rho_H(z)$

1. With $\alpha_1 = 0$ solve LP[$\alpha_1$] to obtain an optimal solution $x^1$ of it. Define $\tilde{\alpha}_1 = \min_i \frac{b_i g_i(x^0) - f_i(x^0)}{(b_i - z_i) g_i(x^0)}$, $s = 1$.
2. With $\alpha_{s+1} = \tilde{\alpha}_s(1+\eta)$ solve LQ[$\alpha_{s+1}$] to obtain an optimal solution $x^{s+1}$ of it. If $S(\alpha_{s+1}) \leqslant 0$ and $\Phi(b - \alpha_{s+1}(b - z)) \leqslant (1 + \varepsilon/2)\Phi(b - \tilde{\alpha}_s(b - z))$, terminate: $y = b - \alpha_{s+1}(b - z)$ and $\tilde{y}_i^s = \frac{b_i g_i(x^{s+1} - f_i(x^{s+1})}{(b_i - z_i) g_i(x^{s+1})}, i = 1, \dots, m$. If $S(\alpha_{s+1}) \leqslant 0$ but $\Phi(b - \alpha_{s+1}(b - z)) > (1 + \varepsilon/2)\Phi(b - \tilde{\alpha}_s(b - z))$, set $\eta \leftarrow \eta/2$, $\tilde{\alpha}_{s+1} = \min_i \frac{b_i g_i(x^{s+1} - f_i(x^{s+1})}{(b_i - z_i) g_i(x^{s+1})}$, increment $s$ and go back to Step 2.
3. If $S(\alpha_s(1+\eta)) > 0$, set $\tilde{\alpha}_{s+1} = \min_i \frac{b_i g_i(x^{s+1}) - f_i(x^{s+1})}{(b_i - z_i) g_i(x^{s+1})}$, increment $s$ and go back to Step 2.

### 5.2. REMOVING IMPROPER ELEMENTS

In Step 3 of Algorithm 1, the improper elements of $T_k^*$ can be removed by using the following rule which can be derived from the rule indicated in [22]:

For every pair $(z, z')$ such that $z \in T_k, z' \in T_k'$ and $z \geqslant y^k$ do: if $z_j' \leqslant z_j$ for all $j \neq i$ then remove $z^{' .i}$. (Indeed, $z^{' .i} \leqslant z^{.i}$ because $z_i^{' .i} = y_i^k = z_i^{.i}$ while $z_j^{' .i} = z_j' \leqslant z_j = z_j^{.i}\ \forall j \neq i$).

An analogous rule (with obvious modifications) can be used for removing improper elements from $T_k^*$ in Step 3 of Algorithm 2.

### 5.3. ALLEVIATING STORAGE PROBLEMS

To avoid storage problems connected with the growth of the set $T_k$ as the algorithm proceeds, and also to preclude possible jams, it may be useful to *restart* the algorithm whenever $|T_k| > L$ or $\bar{x}^k = \bar{x}^{k-h}$ where $L, h$ are user supplied fixed numbers. For instance, Step 4 of Algorithm 1 should be modified as follows:

*Step 4.* If $T_{k+1} \neq \emptyset$ and $|T_{k+1}| < L$ or $\bar{x}^{k+1} \neq \bar{x}^{k-h}$ then set $k \leftarrow k+1$ and return to Step 1; otherwise go to Step 5.

*Step 5.* Redefine $y^k \approx \pi_G(z^k), T_{k+1} = \{b - (b_i - y_i^k)e^i, i = 1, \dots, m\}$ (i.e. $P_{k+1} = [0, b] \setminus (y^k, b]$ and return to Step 1.

Analogously, Step 4 of Algorithm 2 should be modified as follows:

*Step 4.* If $|T_{k+1}| < L$ or $\bar{x}^k \neq \bar{x}^{k-h}$ then set $k \leftarrow k+1$ and return to Step 1; otherwise go to Step 5.

*Step 5.* Redefine $y^k \approx \rho_H(z^k)$, $T_{k+1} = y_i^k.e^i, i = 1, \dots, m\}$ (i.e. $P_{k+1} = [0, b] \setminus [0, y^k)$ and return to Step 1.

## 6. Modified Variants

Several modified variants of the basic method can be proposed to speed up the convergence and better handle large scale problems.

## 6.1. FLEXIBLE SELECTION RULE

In some problems, the function $\Phi(y)$ may be such that $\Phi(y) > \Phi(y')$ implies $\lambda_G(y) < \lambda_G(y')$, where $\lambda_G(z) = \max\{\alpha | \; \alpha z \in G\}$ (see (28)). In such cases the criterion $z^k \in \operatorname{argmax}\{\Phi(z) | \; z \in \tilde{T}_k\}$ used in Step 1 of Algorithm 1 may lead to selecting the point $z \in \tilde{T}_k$ that achieves the smallest value of $\lambda_G(z)$, i.e. the point that lies the farthest from the set $G$. Since the convergence speed of the algorithm depends on how fast the point $z^k$ is brought close to $G$, this fact may result in a slow convergence. Furthermore, when $m$ is large, the rapid growth of the set $T_k$ may pose serious storage problems while slowing down the procedure. To get round these difficulties, a flexible selection criterion is suggested which still ensures convergence of the algorithm, provided $\varepsilon > 0$.

Observe that the standard selection criterion $z^k \in \operatorname{argmax}\{\Phi(z) | \; z \in \tilde{T}_k\}$ is only necessary to prove convergence for $\varepsilon = 0$, i.e., when we want the algorithm to generate an infinite sequence converging to an *exact* optimal solution. In fact, close scrutiny of the proof of Theorem 2 shows that this criterion is needed only to ensure that every point $z^k$ satisfies $\Phi(z^k) \geqslant \Phi(z) \; \forall z \in \tilde{T}_k$, hence $\Phi(z^k) \geqslant \Phi(y) \; \forall y \in G \cap L$, in order to conclude that any cluster point of the sequence $\{z^k\}$ is an exact optimal solution.

In practice, however, what we often need is only an $\varepsilon$-optimal solution with $\varepsilon > 0$. In that case, the selection criterion for $z^k$ may influence the convergence speed but not the convergence status of the algorithm. To see this, suppose the algorithm is infinite. Then it must generate at least one infinite sequence $\{z^k\}$ satisfying (22), hence, satisfying $z^k - y^k \to 0$ $(k \to +\infty)$ (see (24)). But in view of the deletion criterion in Step 3 every $z^k$ satisfies $\frac{\Phi(z^k) - CBV(k)}{CBV(k)} > \varepsilon$, where $CBV(k)$ denotes the current best value at iteration $k$. Furthermore, since $y^k \leqslant \tilde{y}^k$ and $\tilde{y}^k$ is feasible, it follows that $CBV(k) \geqslant \Phi(\tilde{y}^k) \geqslant \Phi(y^k)$, hence $\Phi(z^k) - \Phi(y^k) \geqslant \Phi(z^k) - \Phi(\tilde{y}^k) \geqslant \Phi(z^k) - CBV(k) > \varepsilon CBV(k)$. The fact $z^k - y^k \to 0$ then implies that $\Phi(z^k) - \Phi(y^k) \to 0$, and since $CBV(k) \geqslant CBV(1) > 0$, we arrive at a contradiction: $0 \geqslant \varepsilon CBV(1) > 0$ as $k \to \infty$. Therefore, finiteness of the algorithm is guaranteed.

Algorithm 1 can thus be applied, using any rule for selecting $M_k$ in Step 1. For instance, computational experiments have shown that in most cases with $\varepsilon > 0$ the criterion $M_k \in \operatorname{argmin}\{\sum_{i=1}^m z_i | \; z \in \tilde{T}_k\}$ performs better than the standard one. Also, the freedom in selecting $M_k$ can sometimes be exploited to enhance efficiency: when the algorithm seems to be jamming, one can change the search direction by a proper selection of $z^k$.

The same remark applies obviously to Algorithm 2.

## 6.2. SUCCESSIVE IMPROVEMENT PROCEDURE

As a step towards solving a global optimization problem like (MP) it is often useful to consider the following feasibility problem

FP($r$): *Given a value* $r \in [\Phi(a), \Phi(b)]$, *find a feasible solution* $z$ *of* (MP) *such that* $\Phi(z) > r + \varepsilon r$.

The infeasibility of this problem would mean that $\Phi(z) \leqslant r + \varepsilon r \ \forall z \in G \cap L$, hence $\frac{\gamma - r}{r} \leqslant \varepsilon$, i.e., that $r$ is an $\varepsilon$-optimal value. Therefore, if an efficient procedure is available for solving FP($r$) for any given $r \in [\Phi(a), \Phi(b)]$, we can proceed as follows to find an $\varepsilon$-optimal solution of (MP).

**Successive Improvement Procedure**

1. Set $r = \Phi(a)$.
2. Solve FP($r$). If FP($r$) is feasible, i.e. a feasible solution $z^{[r]}$ is found such that $\Phi(z^{[r]}) > r + \varepsilon r$ then set $CBS = z^{[r]}$ and go to Step 2. If FP($r$) is infeasible with $r = \Phi(a)$ the problem (CP) is infeasible. If FP($r$) is infeasible with $r > \Phi(a)$, then $CBS$ is an $\varepsilon$-optimal solution of (CP).
3. Reset $r \leftarrow \Phi(z^{[r]})$, $L \leftarrow L \cap \{\Phi(z) \geqslant r + \varepsilon r\}$ and go back to Step 0.

The rationale of this approach is that since the feasibility problem FP($r$) does not involve an objective function, we can solve it by a variant of Algorithm 1 where the criterion for selecting $z^k$ in Step 1 is taken so as to force $z^k$ to approach $G$ rapidly. For example the criterion $z^k \in \operatorname{argmin}\{\sum_{i=1}^m z_i |\ z \in \tilde{T}_k\}$ would be suitable for that purpose. Using this criterion to solve FP($r$) and incorporating FP($r$) in this successive improvement procedure leads to the following modifed version of Algorithm 1 which is a kind of depth-first search on the graph formed by the vertices of the polyblocks that might be generated.

Define $L_r = \{z |\ z \geqslant a, \Phi(z) \geqslant r + \varepsilon r\}$. For convenience let us agree to say that, for a given $z$ with $y \approx \pi_G(z)$, the point $z^{*i} = z - (z_i - y_i^k)e^i$, $i \in \{1, \ldots, m\})$, is the $i$-th child of $z$ (and $z$ is the "father" of every $z^{*i}$). At each stage of the algorithm, every $z \in [0, b]$ is assigned a set $J(z) \subset I := \{1, \ldots, m\}$ of indices $i \in I$ still "active" at this stage.

**Algorithm 1**$^*$

Step 0. Set $r = \Phi(a)$. Let $z^0 = b$, $T_0 = \{z^0\}$, $J(z^0) = I := \{1, \ldots, m\}$, $k = 1$.

Step 1. Let $z^*$ be the last element of $T_{k-1}$. Compute $y^* \approx \pi_G(z^*)$ and

$$z^{*i} = z^* - (z_i^* - y_i^*)e^i, \ i = 1, \ldots, m$$

If $V_k = \{z^{*i} |\ i \in J(z^*),\ z^{*i} \in L_r\} \neq \emptyset$, go to Step 2.

If $V_k = \emptyset$, go to Step 3.

Step 2. Compute $z^k \in \operatorname{argmin}\{\sum_{i=1}^m z_i |\ z \in V_k\}$ and $y^k \approx \pi_G(z^k)$ with $x^k \in D$ satisfying $\max_{i=1,\ldots,m}\{y_i^k - f_i(x^k)/g_i(x^k)\} = 0$. Define $\tilde{y}^k$ such that $\tilde{y}_i^k = f_i(x^k)/g_i(x^k)$, $i = 1, \ldots, m$. If $\Phi(\tilde{y}^k) > r + \varepsilon r$, then go to Step 4. Otherwise let $z^k$ be the $i_k$-th child of $z^*$ (i.e. $z^k = z^{*i_k}$). Remove $i_k$ from $J(z^*)$ and set $J(z^k) = I$. Define $T_k$ to be the ordered set obtained by appending $z^k$ to $T_{k-1}$ so that it becomes the last member of $T_k$. Increment $k$ and go back to Step 1.

Step 3. If $z^* = b$, terminate: $CBS$ is an $\varepsilon$-optimal solution of (MP) if it is defined; the problem is infeasible if $CBS$ is not defined.
If $z^* \neq b$, let $z^{**}$ be the father of $z^*$ and let $z^*$ be the $i^*$-th child of $z^{**}$. Remove $i^*$ from $J(z^{**})$, set $T_k = T_{k-1} \setminus \{z^*\}$, increment $k$ and go back to Step 1.

Step 4. Let $z^k$ be the $i_k$-th child of $z^*$ (i.e. $z^k = z^{*i_k}$). Two options:
(a) Define $CBS = \tilde{y}^k$, reset $r \leftarrow \Phi(\tilde{y}^k)$, remove $i_k$ from $J(z^*)$ and define $T_k$ to be the ordered set of all $z \in T_{k-1}$ such that $\Phi(z) \geqslant r + \varepsilon r$. Increment $k$ and go back to Step 1.
(b) Define $CBS = \tilde{y}^k$, reset $r \leftarrow \Phi(\tilde{y}^k)$, $L \leftarrow L \cap \{z | \Phi(z) \geqslant r + \varepsilon r\}$ and go back to Step 0.

PROPOSITION 6. *Algorithm 1* terminates after finitely many steps, yielding an $\varepsilon$-optimal solution or establishing the infeasibility of the problem.*

*Proof.* It suffices to show that if the Algorithm terminates at a Step 3 with $z^* = b$ while no $CBS$ has been defined yet then FP($r$) is infeasible. At any iteration $k$ of the procedure let, for every $z \in T_k$, $U(z)$ denote the set of all children of $z$ that have index in $J(z)$ and belong to $L_r$, and let $V = \cup_{z \in T_k} U(z)$. Then the set $\Omega_r = \{z \in L_r | \Phi(z) \leqslant (1+\varepsilon)\Phi(\pi_G(z))$ is contained in the polyblock with vertex set $V$. But the event $z^* = b$ in Step 3 means that $T_k = \{b\}, U(b) = \emptyset$, hence $V = \emptyset$. Therefore, $\Omega_r = \emptyset$, i.e. (CP) is infeasible. □

REMARK 3. An advantage of Algorithm 1* is that at each iteration the set $T_k$ either increases at most by 1, or decreases.

Analogously, an Algorithm 2* may be formulated for problem (Q) which proceeds by successive improvement of the incumbent solution through solving feasibility problems of the form FQ($r$): Find a feasible solution $z$ of (MQ) such that $\Phi(z) \leqslant r - \varepsilon r$. (the infeasibility of this problem would mean that $r$ is an $\varepsilon$-optimal value).

### 6.3. BRANCH AND BOUND VARIANT

The basic Algorithm 1 can be interpreted as a rectangular branch an bound procedure in which the root of the tree that describes the branching scheme is $T_1 = \{b\}$ (the box $[0, b]$ contains all feasible solutions) and the active nodes at iteration $k$ are represented by the elements of $T_k$ (each element $z \in T_k$ represents a box $[0, z]$ containing a portion of the feasible set that remains to be explored).

An attractive feature of this branch and bound procedure is that an upper bound is readily available for each node $z \in T_k$, namely $\Phi(z)$. However, since each node has many ($m$) successors and the boxes represented by the active nodes at each iteration form a covering rather than a partition of the initial box $[a, b]$, this may cause slow convergence when $m$ is not small. To alleviate these diffficulties, a branch and bound procedure in a more usual sense can be proposed, in which branching is performed by subdividing the current box into two subboxes, and an upper bound $\alpha(M)$ over a box $M = [p, q] \subset [a, b]$ is computed according to a subroutine described below.

PROCEDURE FOR COMPUTING $\alpha(M)$

The subroutine involves two stages: 1) reducing the box $M$, by cutting away useless portions; 2) applying a truncated version of Algorithm 1 to the reduced box.

**I. Box reduction (preprocessing):**

Let $\delta > 0$ be a chosen tolerance. Compute

$$\xi_i = \sup\{\xi > 0 |\ p + \xi e^i \in G,\ p_i + \xi \leqslant q_i\}, \quad q' = p + \sum_{i=1}^{s} \xi_i e^i \tag{33}$$

$$\zeta_i = \sup\{\zeta > 0 |\ q' - \zeta e^i \in L,\ p_i \leqslant q'_i - \zeta\}, \quad p' = q' - \sum_{i=1}^{s} \zeta_i e^i. \tag{34}$$

If $p' \notin G$ or $q' \notin L$, then terminate: $M$ contains no feasible point, so $\alpha(M) = -\infty$.
If $q' \in G$ and $q' \in L$, then terminate: $\alpha(M) = \Phi(q')$,
If $p' \in G$ and $q' \in L$, while $p' - p > \delta$ then reset $[p, q] \leftarrow [p', q']$ and repeat.
If $p' \in G$ and $q' \in L$, while $p' - p \leqslant \delta$ then reset $[p, q] \leftarrow [p', q']$ and stop.

**II. Bound Computation**

0. Let $[p, q]$ be the last box obtained after "preprocessing". Select $\nu_{\max}$ (maximal number of iterations to be executed).
If $p \notin G$ then $[p, q] \cap G \cap L = \emptyset$, so $\alpha(M) = -\infty$.
If $p \in G$, and $q \in G$ then $\alpha(M) = \Phi(q)$, $y(M) = q$ (actually $\Phi(q)$ is then the exact maximum of $\Phi(y)$ over the feasible solutions in $[p, q]$).
If $p \in G$ and $z^1 = q \notin G$, set $\nu = 1$ ($\nu$ is the iteration counter for the subroutine).

1. Compute $y^\nu \approx \pi_G(z^\nu)$ and let $\tilde{y}^\nu = (f_1(x^\nu)/g_1(x^\nu), \ldots, f_m(x^\nu)/g_m(x^\nu))$ where $x^\nu \in D$ is the point satisfying $\max_{i=1,\ldots,m}[y_i^\nu - f_i(x^\nu)/g_i(x^\nu)] = 0$. Use $\tilde{y}^\nu$ to update $CBV$. Compute

$$z^{\nu,i} = z^\nu - (z_i^\nu - y_i^\nu)e^i,\ i = 1, \ldots, n.$$

Let $T'_\nu = (T_\nu \setminus \{z^\nu\}) \cup \{z^{\nu,1}, \ldots, z^{\nu,n}\}$.

2. Let $T_{\nu+1}$ be the set that remains from $T'_\nu$ after removing all improper elements and all $z$ such that $\Phi(z) \leqslant CBV + \varepsilon$. If $T_{\nu+1} = \emptyset$, set $\alpha(M) = CBV + \varepsilon$. Otherwise, compute

$$z^{\nu+1} \in \text{argmax}\{\Phi(z) |\ z \in T_{\nu+1}\}$$

If $\nu = \nu_{\max}$ set $\alpha(M) = \Phi(z^{\nu+1})$, $z(M) = z^{\nu+1}$, $y(M) = \tilde{y}^{\nu+1}$. Otherwise, set $\nu \leftarrow \nu + 1$ and return to Step 1.

REMARK 4. The numbers $\delta$ (tolerance in box reduction) and $\nu_{\max}$ should be flexibly chosen so as to require only a relatively small computational cost.

**Algorithm 1**$^{**}$

Step 0. Start with $\mathscr{P}_1 = \mathscr{S}_1 = \{M_1 = [a, b]\}$. Set $k = 1$.

Step 1. For each box $M \in \mathscr{P}_k$ compute $\alpha(M)$ and $y(M)$ (if $\alpha(M) > -\infty$). Update the incumbent by setting $CBS = \bar{y}^k$ equal to the best among all $y(M), M \in \mathscr{P}_k$. Let $CBV = \Phi(\bar{y}^k)$.

Step 2. Delete every $M \in \mathscr{S}_k$ such that $\alpha(M) \leqslant CBV + \varepsilon$. Let $\mathscr{R}_k$ be the collection of remaining members of $\mathscr{S}_k$. If $\mathscr{R}_k = \emptyset$, then terminate: $\bar{y}^k$ is an $\varepsilon$-optimal solution.

Step 3. Select $M_k \in \text{argmax}\{\alpha(M)|\ M \in \mathscr{R}_k\}$. Let $y^k = y(M_k)$. Choose $j_k \in \text{argmax}_i\{q_i - p_i\}$ and divide $M_k$ into two subboxes via the hyperplane $y_{j_k} = (p^k_{j_k} + q^k_{j_k})/2$. Let $\mathscr{P}_{k+1}$ be the partition of $M_k$.

Step 4. Set $\mathscr{S}_{k+1} = (\mathscr{R}_k \setminus \{M_k\}) \cup \mathscr{P}_{k+1}$. Set $k \leftarrow k + 1$ and go back to Step 1.

The convergence of this algorithm is guaranteed because the subdivision process is exhaustive and the bounding is consistent (see, e.g., [20]). Just as with Algorithm 1, for $\varepsilon > 0$, we can use an arbitrary selection rule in Step 3, e.g. $M_k \in \text{argmin}\{\alpha(M)|\ M \in \mathscr{R}_k\}$. To see that the algorithm using an arbitrary selection rule is still finite, consider any infinite nested sequence of boxes $\{M_k\}$ generated by the algorithm. As $k \to +\infty$, by exhaustiveness $M_k$ must shrink to a single point $\bar{y}$, while by bound consistency, $\alpha(M_k) - CBV \to 0$, so that for sufficiently large $k$ : $\alpha(M_k) - CBV \leqslant \varepsilon$, contrary to the deletion rule in Step 2 which implies that $\alpha(M_k) - CBV > \varepsilon\ \forall k$. The selection rule can also vary during the procedure: for instance, it may be preferable to use a breadth-first rule (every node on a level is branched on before moving to the next level) at the initial levels to identify promising branches, then to switch to a depth-first rule (the last node created is selected for branching) to keep the set $\mathscr{R}_k$ within manageable size.

An analogous branch and bound Algorithm 2$^{**}$ can also be formulated for problem (Q).

For large scale problems whose dimension cannot easily be reduced, the branch and bound Algorithm 1$^{**}$ (or Algorithm 2$^{**}$) seems to be the best. Since each partition set is reduced by cuts before computing the bound, this algorithm can also be viewed as a kind of branch and cut procedure.

## 7. Numerical Examples

The basic Algorithm 1 as well as the two other variants have been coded in Pascal and run on a P.C. Pentium III 450 MHz to solve, among others, a number of problems taken from the literature. Although linear programs have been solved by our own code and reoptimization techniques have not been used to solve the connected sequences of linear subprograms involved, the computational results are quite encouraging and demonstrate the capability of these algorithms to efficiently handle in a unified manner a wide class of problems including many of those earlier treated differently depending on the form of the function $\Phi(.)$.

In this section we present some numerical examples. For each example we give the data of the problem in its original formulation, the optimal solution, the number of iterations needed to reach the optimal solution and the total number of iterations needed to solve the problem, as well as the number of restarts, if any, together with the iterations where restarts are made. Of course, the tolerance $\varepsilon$ and the computation time are indicated. Most problems have been solved by all three algorithms (Algorithms 1, 1*,1**, or Algorithms 2, 2*, 2**), for comparison. Example 5 (maximizing the sum of 5 linear fractional functions), Example 7 (minimizing the sum of 7 linear fractional functions) and Example 12 (minimizing the product of 6 linear fractional functions) suggest that the approach should be practical for problems with moderate values of $m$, although, as should be expected for all deterministic nonconvex global optimization methods, the computational time sharply increases as $m$ increases.

## 7.1. MAXMIN AND MINMAX

Solving a maxmin problem is equivalent to computing a vector $\pi_G(z)$ since

$$\begin{aligned}&\max\left\{\min_{i=1,\dots,m}\frac{f_i(x)}{g_i(x)}\middle|\ x\in D\right\}\\&\quad=\max\left\{\lambda|\ \lambda\leqslant\frac{f_i(x)}{g_i(x)}\ (i=1,\dots,m),\ x\in D\right\}.\end{aligned}$$

On the other hand, solving a minmax problem reduces to solving a maxmin problem, since the two problems

$$\min\left\{\max_{i=1,\dots,m}\frac{f_i(x)}{g_i(x)}|\ x\in D\right\},\qquad \max\left\{\min_{i=1,\dots,m}\frac{g_i(x)}{f_i(x)}|\ x\in D\right\}$$

have the same optimal solutions and inverse optimal values.

EXAMPLE 1.

$$\max\min\left\{\frac{37x_1+73x_2+13}{13x_1+13x_2+13},\frac{63x_1-18x_2+39}{13x_1+26x_2+13}\right\}: \tag{35}$$

$$5x_1-3x_2=3,\ 1.5\leqslant x_1\leqslant 3. \tag{36}$$

With tolerance $\varepsilon=0.00000005$ :
Optimal value 1.489510 .
Optimal solution $x=(1.5, 1.5)$ found and confirmed in one iteration.
Computation time: 0.00 s.

EXAMPLE 2.

$$\begin{aligned}
&\min\max\left\{\frac{3x_1+x_2-2x_3+0.8}{2x_1-x_2+x_3}, \frac{4x_1-2x_2+x_3}{7x_1+3x_2-x_3}\right\} :\\
&x_1+x_2-x_3 \leqslant 1, \quad -x_1+x_2-x_3 \leqslant -1,\\
&12x_1+5x_2+12x_3 \leqslant 34.8, \quad 12x_1+12x_2+7x_3 \leqslant 29.1,\\
&-6x_1+x_2+x_3 \leqslant -4.1
\end{aligned}$$

With tolerance $\varepsilon = 0.00000005$ :
Optimal value 0.573102 .
Optimal solution $x = (1.015695, \ 0.590494, \ 1.403675)$ found and confirmed in one iteration.
Computation time: 0.06 s.

## 7.2. MAXSUM

EXAMPLE 3. (taken from [7])

$$\begin{aligned}
\max \quad & \{\frac{37x_1+73x_2+13}{13x_1+13x_2+13} + \frac{63x_1-18x_2+39}{13x_1+26x_2+13}\}\\
\text{s.t.} \quad & 5x_1-3x_2 = 3\\
& 1.5 \leqslant x_1 \leqslant 3
\end{aligned}$$

• By Algorithm 1, with tolerance $\varepsilon = 0.01$ :
Optimal value 5.000000 .
Optimal solution $x = (3.000000, \ 4.000000)$.
Optimal solution found at iteration 1 and confirmed at iteration 11.
Computation time: 0.05 s.
One restart at iteration 5. Maximal number of vertices: 5.
• By Algorithm 1*, with tolerance $\varepsilon = 0.01$ :
Optimal value: 5.000000.
Optimal solution: $x = (3.000000, 4.000000)$.
Computation time: 0.11 s.
Optimal solution found at iteration 1 and confirmed at iteration 10.
• By Algorithm 1**, with tolerance $\varepsilon = 0.01$:
Optimal value: 5.000000.
Optimal solution: $x = (3.000000, 4.000000)$.
Computation time: 0.05 s.
Optimal solution found at iteration 1 and confirmed at iteration 7.

EXAMPLE 4. (taken from [7])

$$\begin{array}{ll}\max & \dfrac{3x_1+x_2-2x_3+0.8}{2x_1-x_2+x_3}+\dfrac{4x_1-2x_2+x_3}{7x_1+3x_2-x_3}\\ \text{s.t.} & \left| \begin{array}{l} x_1+x_2-x_3\leqslant 1, \quad -x_1+x_2-x_3\leqslant -1\\ 12x_1+5x_2+12x_3\leqslant 34.8, \quad 12x_1+12x_2+7x_3\leqslant 29.1\\ -6x_1+x_2+x_3\leqslant -4.1\end{array}\right.\end{array}$$

• By Algorithm 1, with tolerance $\varepsilon = 0.000001$ :
Optimal value 2.471423.
Optimal solution $x = (1.000003,\ 0.000000,\ 0.000003)$.
Optimal solution found at iteration 1 and confirmed at iteration 5.
Computation time: 0.05 s.
Maximal number of vertices: 1.

• By Algorithm 1*, with tolerance $\varepsilon = 0.01$ :
Optimal value: 2.471423.
Optimal solution: $x = (0.944685, 0.000000, 0.000000)$.
Optimal solution found at iteration 0 and confirmed at iteration 3.
Computation time: 0.05 s.

• By Algorithm 1**, with tolerance $\varepsilon = 0.01$ :
Optimal value: 2.471424.
Optimal solution: $x = (1.000002, 0.000000, 0.000002)$.
Optimal solution found at iteration 1 and confirmed at iteration 4.
Computation time: 0.06 s.
Maximal number of active nodes: 1.

EXAMPLE 5. Problem (P) (Maxsum) for $m = 5, n = 12,\ f_i(x) = \langle c^i, x\rangle + r_i,\ g_i(x) = \langle d^i, x\rangle + s_i,\ D = \{x|\ Ax \leqslant q, x \geqslant 0\}$, with

$$\begin{array}{ll}
c^1 = (\ 0.0, -0.1, -0.3,\ 0.3,\ 0.5,\ 0.5, -0.8,\ 0.4, -0.4,\ 0.2,\ 0.2, -0.1), & r_1 = 14.6\\
d^1 = (-0.3, -0.1, -0.1, -0.1,\ 0.1,\ 0.4,\ 0.2, -0.2,\ 0.4,\ 0.2, -0.4,\ 0.3), & s_1 = 14.2\\
c^2 = (\ 0.2,\ 0.5,\ 0.0,\ 0.4,\ 0.1, -0.6, -0.1, -0.2, -0.2,\ 0.1,\ 0.2,\ 0.3), & r_2 = 7.1\\
d^2 = (\ 0.0,\ 0.1, -0.1,\ 0.3,\ 0.3, -0.2,\ 0.3,\ 0.0, -0.4,\ 0.5, -0.3,\ 0.1), & s_2 = 1.7\\
c^3 = (-0.1,\ 0.3,\ 0.0,\ 0.1, -0.1,\ 0.0,\ 0.3, -0.2,\ 0.0,\ 0.3,\ 0.5,\ 0.3), & r_3 = 1.7\\
d^3 = (\ 0.8, -0.4,\ 0.7, -0.4, -0.4,\ 0.5, -0.2, -0.8,\ 0.5,\ 0.6, -0.2,\ 0.6), & s_3 = 8.1\\
c^4 = (-0.1,\ 0.5,\ 0.1,\ 0.1, -0.2, -0.5,\ 0.6,\ 0.7,\ 0.5,\ 0.7, -0.1,\ 0.1), & r_4 = 4.0\\
d^4 = (\ 0.0,\ 0.6, -0.3,\ 0.3,\ 0.0,\ 0.2,\ 0.3, -0.6, -0.2, -0.5,\ 0.8, -0.5), & s_4 = 26.9\\
c^5 = (\ 0.7, -0.5,\ 0.1,\ 0.2, -0.1, -0.3,\ 0.0, -0.1, -0.2,\ 0.6,\ 0.5, -0.2), & r_5 = 6.8\\
d^5 = (\ 0.4,\ 0.2, -0.2,\ 0.9,\ 0.5, -0.1,\ 0.3, -0.8, -0.2,\ 0.6, -0.2, -0.4), & s_5 = 3.7
\end{array}$$

$$A = \begin{bmatrix}
-1.8 & -2.2 & 0.8 & 4.1 & 3.8 & -2.3 & -0.8 & 2.5 & -1.6 & 0.2 & -4.5 & -1.8 \\
4.6 & -2.0 & 1.4 & 3.2 & -4.2 & -3.3 & 1.9 & 0.7 & 0.8 & -4.4 & 4.4 & 2.0 \\
3.7 & -2.8 & -3.2 & -2.0 & -3.7 & -3.3 & 3.5 & -0.7 & 1.5 & -3.1 & 4.5 & -1.1 \\
-0.6 & -0.6 & -2.5 & 4.1 & 0.6 & 3.3 & 2.8 & -0.1 & 4.1 & -3.2 & -1.2 & -4.3 \\
1.8 & -1.6 & -4.5 & -1.3 & 4.6 & 3.3 & 4.2 & -1.2 & 1.9 & 2.4 & 3.4 & -2.9 \\
-0.5 & -4.1 & 1.7 & 3.9 & -0.1 & -3.9 & -1.5 & 1.6 & 2.3 & -2.3 & -3.2 & 3.9 \\
0.3 & 1.7 & 1.3 & 4.7 & 0.9 & 3.9 & -0.5 & -1.2 & 3.8 & 0.6 & -0.2 & -1.5 \\
0.5 & -4.2 & 3.6 & -0.6 & -4.8 & 1.5 & -0.3 & 0.6 & -3.6 & 0.2 & 3.8 & -2.8 \\
-0.1 & 3.3 & -4.3 & 2.4 & 4.1 & 1.7 & 1.0 & -3.3 & 4.4 & -3.7 & -1.1 & -1.4 \\
-0.6 & 2.2 & 2.5 & 1.3 & -4.3 & -2.9 & -4.1 & 2.7 & -0.8 & -2.9 & 3.5 & 1.2 \\
4.3 & 1.9 & -4.0 & -2.6 & 1.8 & 2.5 & 0.6 & 1.3 & -4.3 & -2.3 & 4.1 & -1.1 \\
0.0 & 0.4 & -4.5 & -4.4 & 1.2 & -3.8 & -1.9 & 1.2 & 3.0 & -1.1 & -0.2 & 2.5 \\
-0.1 & -1.7 & 2.9 & 1.5 & 4.7 & -0.3 & 4.2 & -4.4 & -3.9 & 4.4 & 4.7 & -1.0 \\
-3.8 & 1.4 & -4.7 & 1.9 & 3.8 & 3.5 & 1.5 & 2.3 & -3.7 & -4.2 & 2.7 & -0.1 \\
0.2 & -0.1 & 4.9 & -0.9 & 0.1 & 4.3 & 1.6 & 2.6 & 1.5 & -1.0 & 0.8 & 1.6
\end{bmatrix}$$

$$q = (15.7,\ 31.8, -36.4,\ 38.5,\ 40.3,\ 10.0,\ 89.8,\ 5.8,\ 2.7,\\ -16.3, -14.6, -72.7,\ 57.7, -34.5,\ 69.1)^\top$$

• By Algorithm 1, with tolerance $\varepsilon = 0.01$ :
Optimal value: 16.077978.
Optimal solution:

$$x = (6.223689, 20.060317, 3.774684, 5.947841, 0.000000, 7.456686,\\ 0.000000, 23.312579, 0.000204, 41.031824, 0.000000, 3.171106).$$

Optimal solution found at iteration 1 and confirmed at iteration 620.
Computation time: 65.58 s.
Maximal number of vertices: 18.

• By Algorithm 1$^{**}$, with tolerance $\varepsilon = 0.01$ :
Optimal value: 16.077978.
Optimal solution:

$$x = (6.224297, 20.059738, 3.774868, 5.947937, 0.000000, 7.456478,\\ 0.000000, 23.312241, 0.000204, 41.031278, 0.000000, 3.171060).$$

Optimal solution found at iteration 8 and confirmed at iteration 11.
Computation time: 4.61 s.
Maximal number of active nodes: 4.

### 7.3. MINSUM

EXAMPLE 6. (taken from [7]).

$$\begin{aligned}&\min \quad \left\{\frac{-x_1+2x_2+2}{3x_1-4x_2+5}+\frac{4x_1-3x_2+4}{-2x_1+x_2+3}\right\}\\&\text{s.t.} \quad x_1+x_2\leqslant 1.5, \quad x_1\leqslant x_2, \quad 0\leqslant x_1\leqslant 1. \quad 0\leqslant x_2\leqslant 1\end{aligned}$$

• By Algorithm 2, with tolerance $\varepsilon = 0.01$,
Optimal value: 1.623188.
Optimal solution: $x = (0.000000, 0.282165)$.
Optimal solution found at iteration 21 and confirmed at iteration 36.
Computation time: 0.11 s.
Maximal number of vertices: 16.
• By Algorithm 2*, with tolerance $\varepsilon = 0.01$ :
Optimal value: 1.623184.
Optimal solution: $x = (0.000000, 0.284476)$.
Optimal solution found at iteration 21 and confirmed at iteration 36.
Computation time: 0.28 s.
• By Algorithm 2**, with tolerance $\varepsilon = 0.01$ :
Optimal value: 1.623294.
Optimal solution: $x = (0.000000, 0.275655)$.
Optimal solution found at iteration 8 and confirmed at iteration 12.
Computation time: 0.22 s.
Maximal number of active nodes: 4.

EXAMPLE 7. Problem (Q) (Minsum) for $m = 7, n = 10$, $f_i(x) = \langle c^i, x\rangle + r_i$, $g_i(x) = \langle d^i, x\rangle + s_i$, $D = \{x|\ Ax \leqslant q, x \geqslant 0\}$, with

| | |
|---|---|
| $c^1 = (-0.7,\ 0.0,\ 0.4,\ 0.7,\ 0.4, -0.1, -0.4,\ 0.9, -0.1,\ 0.0)$ | $r_1 = 19.3$ |
| $d^1 = (0.4, -0.6,\ 0.0,\ 0.4, -0.1, -0.3, -0.1,\ 0.6,\ 0.7,\ 0.1)$ | $s_1 = 5.4$ |
| $c^2 = (0.3,\ 0.3,\ 0.1, -0.3, -0.3,\ 0.1, -0.1,\ 0.0, -0.1, -0.5)$ | $r_2 = 25.0$ |
| $d^2 = (0.5, -0.4, -0.4, -0.1, -0.3,\ 0.4, -0.1,\ 0.2, -0.1, -0.4)$ | $s_2 = 34.7$ |
| $c^3 = (0.3,\ 0.3,\ 0.0, -0.1,\ 0.1, -0.1,\ 0.6,\ 0.1,\ 0.4, -0.1)$ | $r_3 = -1.0$ |
| $d^3 = (0.2, -0.2,\ 0.4,\ 0.6, -0.6, -0.5,\ 0.2,\ 0.5, -0.6, -0.5)$ | $s_3 = 43.4$ |
| $c^4 = (0.6,\ 0.3,\ 0.4, -0.3, -0.2,\ 0.8,\ 0.1,\ 0.2, -0.1, -0.1)$ | $r_4 = 12.8$ |
| $d^4 = (0.2, -0.4, -0.5,\ 0.1,\ 0.9,\ 0.0,\ 0.4, -0.9, -0.5, -0.2)$ | $s_4 = 19.3$ |
| $c^5 = (0.0,\ 0.5, -0.2, -0.1,\ 0.2,\ 0.0,\ 0.3,\ 0.7, -0.8,\ 0.5)$ | $r_5 = 6.2$ |

$$
\begin{array}{ll}
d^5 = (0.5, -0.1,\ 0.6,\ 0.8,\ 0.6,\ 0.0, -0.7, -0.5, -0.1, -0.5) & s_5 = 51.7 \\
c^6 = (0.2, -0.5, -0.4, -0.2, -0.2,\ 0.7, -0.9, -0.2,\ 0.3,\ 0.5) & r_6 = 26.6 \\
d^6 = (0.5, -0.2,\ 0.0,\ 0.1, -0.3,\ 0.2, -0.1,\ 0.3,\ 0.0,\ 0.3) & s_6 = 7.5 \\
c^7 = (-0.5,\ 0.0,\ 0.1,\ 0.4,\ 0.4,\ 0.9,\ 0.6,\ 0.0,\ 0.2,\ 0.2) & r_7 = -0.6 \\
d^7 = (-0.5,\ 0.4,\ 0.4, -0.3,\ 0.1,\ 0.3, -0.7, -0.2,\ 0.2,\ 0.1) & s_7 = 12.8
\end{array}
$$

$$
A = \begin{bmatrix}
4.9 & 3.5 & 3.7 & 0.7 & 2.1 & 1.9 & -2.3 & 2.4 & 3.1 & 3.3 \\
1.6 & 1.5 & -0.4 & 4.8 & -4.7 & -0.1 & -1.4 & -2.9 & -2.7 & -4.7 \\
-1.0 & -1.0 & -2.7 & -0.6 & 2.9 & -0.5 & -2.5 & 4.8 & 0.8 & 3.5 \\
-0.7 & -0.3 & -0.8 & -1.9 & 4.3 & 1.1 & -2.0 & -2.7 & 4.4 & -1.4 \\
4.4 & -4.9 & 2.7 & 1.4 & -0.3 & 2.0 & 3.4 & -0.2 & -0.4 & -3.0 \\
2.6 & 1.5 & 2.2 & -1.8 & -2.7 & 0.4 & 0.3 & -0.9 & 2.5 & -4.1 \\
-0.8 & -4.4 & -1.3 & -0.8 & 1.8 & 0.8 & 1.4 & -2.6 & 1.4 & -3.8 \\
1.6 & 2.1 & -1.0 & -2.2 & 2.50 & -3.0 & -1.5 & 2.7 & -2.2 & -4.1 \\
-4.5 & -2.2 & 3.1 & 2.8 & -2.1 & 4.5 & 3.1 & 2.6 & -4.4 & -4.0 \\
-4.0 & 2.6 & -2.3 & 0.6 & -4.8 & -2.5 & 3.2 & 0.5 & -3.0 & -3.6 \\
3.2 & -1.5 & -2.8 & 2.0 & 1.6 & 3.8 & -1.6 & 0.9 & 2.6 & -3.9 \\
0.6 & 0.1 & 1.1 & 3.2 & -3.8 & -3.2 & 1.7 & 2.6 & 4.0 & -4.4 \\
-4.8 & 1.8 & 3.7 & 2.5 & -3.4 & 0.2 & -3.5 & 3.0 & 1.1 & 4.4 \\
2.0 & 1.3 & -3.9 & -3.9 & -1.1 & -4.3 & 2.5 & -1.2 & -4.7 & -2.4 \\
1.9 & -3.6 & 4.3 & -0.9 & -0.1 & -1.0 & 0.1 & 0.1 & 0.0 & 4.1
\end{bmatrix}
$$

$$
\begin{aligned}
q = (&161.1, -65.6,\ 45.7,\ 35.1,\ 3.5, -18.8, -46.4, -37.7, -59.4, \\
&- 117.0,\ 24.8,\ 1.5,\ 55.6, -124.8, -27.2)^\top
\end{aligned}
$$

• By Algorithm 2**, with tolerance $\varepsilon = 0.01$:
Optimal value: 5.642336
Optimal solution:

$$
\begin{aligned}
x = (&8.078763,\ 1.949795,\ 4.532075,\ 8.677625,\ 5.769029,\ 0.000000, \\
&0.001348,\ 0.000000,\ 11.303767,\ 13.543402).
\end{aligned}
$$

Optimal solution found at iteration 341 and confirmed at iteration 926.
Computational time: 376.90 s.
Maximal number of active nodes: 9.

## 7.4. MAXPRODUCT

EXAMPLE 8. (data from [5], but with maxproduct instead of maxsum).

$$
\max \quad \left\{ \frac{37x_1 + 73x_2 + 13}{13x_1 + 13x_2 + 13} \times \frac{63x_1 - 18x_2 + 39}{13x_1 + 26x_2 + 13} \right\}
$$

$$
\text{s.t.} \quad 5x_1 - 3x_2 = 3, \quad 1.5 \leqslant x_1 \leqslant 3
$$

• By Algorithm 1, with tolerance: $\varepsilon = 0.01$:
Optimal value: 5.098709.
Optimal solution: $x = (1.500000,\ 1.500000)$.
Optimal solution found at iteration 3 and confirmed at iteration 6.
Computation time: 0.00 s.
Maximal number of vertices: 4.

• By Algorithm 1**, with tolerance $\varepsilon = 0.01$:
Optimal value: 5.098709.
Optimal solution: $x = (1.500000,\ 1.500000)$.
Optimal solution found at iteration 1 and confirmed at iteration 5.
Computation time: 0.05 s.
Maximal number of boxes: 3

EXAMPLE 9.

$$\begin{aligned}
&\max \quad \left(\frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} \times \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3}\right)\\
&\text{s.t.} \quad x_1 + x_2 - x_3 \leqslant 1, \quad -x_1 + x_2 - x_3 \leqslant -1, \quad 12x_1 + 5x_2 + 12x_3 \leqslant 34.8,\\
&\qquad 12x_1 + 12x_2 + 7x_3 \leqslant 29.1, \quad -6x_1 + x_2 + x_3 \leqslant -4.1
\end{aligned}$$

• By Algorithm 1, with tolerance $\varepsilon = 0.01$:
Optimal value: 1.085714.
Optimal solution: $x = (1.000000,\ 0.000000,\ 0.000000)$.
Optimal solution found at iteration 1 and confirmed at iteration 10.
Computation time: 0.00 s.
Maximal number of vertices: 5.

• By Algorithm 1**, with tolerance $\varepsilon = 0.01$ :
Optimal value: 1.085714.
Optimal solution: $x = (1.000002,\ 0.000000,\ 0.000002)$.
Optimal solution found at iteration 1 and confirmed at iteration 9.
Computation time: 0.06 sec.
Maximal number of active nodes: 3.

## 7.5. MINPRODUCT

EXAMPLE 10.

$$\begin{aligned}
&\min \quad \frac{-x_1 + 2x_2 + 2}{3x_1 - 4x_2 + 5} \times \frac{4x_1 - 3x_2 + 4}{-2x_1 + x_2 + 3}\\
&\text{s.t.} \quad x_1 + x_2 = 1.5, \quad x_1 \leqslant x_2, \quad 0 \leqslant x_1 \leqslant 1, \quad 0 \leqslant x_2 \leqslant 1
\end{aligned}$$

• By Algorithm 1, with tolerance $\varepsilon = 0.01$:
Optimal value: 0.533335.

Optimal solution: $x = (0.000000,\ 0.000003)$.
Optimal solution found at iteration 1 and confirmed at iteration 37.
Computation time : 0.11 s.
Maximal number of vertices: 3.

• By Algorithm 1*, with tolerance $\varepsilon = 0.01$:
Optimal value: 0.533335.
Optimal solution: $x = (0.000000,\ 0.000004)$.
Optimal solution found at iteration 1 and confirmed at iteration 30.
Computation time: 0.22 s.

• By Algorithm 1**, with tolerance $\varepsilon = 0.001$:
Optimal value: 0.533333.
Optimal solution: $x = (0.000000,\ 0.000000)$.
Optimal solution found at iteration 1 and confirmed at iteration 16.
Computation time: 0.05 s.
Maximal number of active nodes: 4.

EXAMPLE 11. (taken from [3]). Problem (Q) (Minproduct) for $m = 2,\ n = 3,\ f_i(x) = \langle c^i, x\rangle + r_i,\ g_i(x) = 1,\ D = \{x|\ Ax \leqslant q, x \geqslant 0\}$, with

$$c^1 = (1, 0, 1/9) \quad r_1 = 0$$
$$c^2 = (0, 1, 1/9) \quad r_2 = 0$$

$$A = \begin{bmatrix} 9 & 9 & 2 \\ 8 & 1 & 8 \\ 1 & 8 & 8 \\ -7 & -1 & -1 \\ -1 & -7 & -1 \\ -1 & -1 & -7 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$q = (81,\ 72,\ 72,\ -9,\ -9,\ -9,\ 8,\ 8)^\top$$

• By Algorithm 2**, with tolerance $\varepsilon = 0.01$:
Optimal value: 0.901234.
Optimal solution: $x = (0.000000,\ 8.000000,\ 1.000000)$.
Optimal solution found at iteration 1 and confirmed at iteration 32.
Computation time: 0.38 s.
Maximal number of active nodes: 4.

EXAMPLE 12. Problem (Q) (Minproduct) for $m = 6,\ n = 12,\ f_i(x) = \langle c^i, x\rangle + r_i,\ g_i(x) = \langle d^i, x\rangle + s_i,\ D = \{x|\ Ax \leqslant q, x \geqslant 0\}$, with

$$\begin{array}{ll}
c^1 = (-0.2, -0.7, -0.1,\ 0.4,\ 0.0,\ 0.8,\ 0.1, -0.8, -0.2,\ 0.0,\ 0.1,\ 0.4) & r_1 = 21\\
d^1 = (0.2,\ 0.5, -0.6,\ 0.1,\ 0.6,\ 0.4, -0.4, -0.3,\ 0.7,\ 0.5,\ 0.4, -0.1) & s_1 = 13.3\\
c^2 = (-0.1,\ 0.1, -0.4, -0.1, -0.1,\ 0.4,\ 0.2,\ 0.5,\ 0.3, -0.4, -0.3,\ 0.3) & r_2 = 16.3\\
d^2 = (-0.3, -0.2, -0.7,\ 0.1,\ 0.2, -0.2, -0.5,\ 0.4,\ 0.3,\ 0.0,\ 0.6, -0.5) & s_2 = 16\\
c^3 = (0.8,\ 0.0, -0.1,\ 0.4,\ 0.2,\ 0.1, -0.5,\ 0.0,\ 0.5,\ 0.6, -0.3, -0.4) & r_3 = 3.7\\
d^3 = (0.1,\ 0.0,\ 0.0,\ 0.3,\ 0.2,\ 0.7,\ 0.4,\ 0.2, -0.1, -0.5,\ 0.6, -0.1) & s_3 = 16.7\\
c^4 = (0.6,\ 0.2,\ 0.2, -0.3,\ 0.5,\ 0.4,\ 0.1,\ 0.6, -0.3,\ 0.3,\ 0.4,\ 0.3) & r_4 = -1.8\\
d^4 = (-0.3,\ 0.0,\ 0.0, -0.5, -0.1,\ 0.2,\ 0.6, -0.6,\ 0.1, -0.2,\ 0.8, -0.3) & s_4 = 21.5\\
c^5 = (-0.3, -0.3,\ 0.5,\ 0.1,\ 0.2, -0.5,\ 0.1,\ 0.2,\ 0.0,\ 0.6,\ 0.3, -0.2) & r_5 = 5\\
d^5 = (0.3,\ 0.0, 0.3, 0.0, -0.8, -0.3, 0.3, -0.9, -0.1, -0.6, -0.1, 0.2) & s_5 = 18.7\\
c^6 = (0.2, -0.1,\ 0.0,\ 0.0, -0.2, -0.4,\ 0.0, -0.6,\ 0.8, -0.2,\ 0.0, -0.1) & r_6 = 12.7\\
d^6 = (0.0,\ 0.6,\ 0.0,\ 0.1,\ 0.0, -0.2,\ 0.0, -0.5,\ 0.2, -0.3,\ 0.3,\ 0.1) & s_6 = 19.2
\end{array}$$

$$A = \begin{bmatrix}
1.9 & 0.0 & -0.2 & -1.5 & 1.8 & 0.9 & -1.0 & 4.5 & 4.5 & -3.5 & -1.8 & -4.8\\
2.9 & 3.7 & -4.8 & -1.9 & 1.8 & -3.7 & 1.8 & 2.5 & -2.9 & 1.9 & -3 & 3.2\\
3.3 & 2.4 & 3.3 & 4.8 & -0.3 & 3.9 & 0.8 & -1.7 & 2.0 & -0.3 & -1.8 & 2.2\\
-4.3 & 1.8 & 2.1 & -4.5 & -0.5 & 2.4 & 1.4 & -0.3 & -2.0 & -2.8 & 0.4 & 4.5\\
1.5 & -0.3 & 0.4 & 1.2 & 1.1 & 1.9 & 1.5 & -1.2 & -3.3 & 4.4 & 3.2 & -4.3\\
-3.2 & 2.4 & -4.5 & -1 & -2.7 & 3.7 & -0.1 & 3.9 & -1.9 & 3.2 & 2.1 & 1.3\\
0.9 & 0.5 & 4.0 & -1.5 & 1.2 & -1.5 & 1.2 & -3.7 & -0.1 & 0.0 & -2.4 & -4.1\\
-4.1 & -4.5 & 2.2 & -3.1 & 4.4 & 4.8 & -3.4 & 2.2 & -2.1 & 2.3 & 2.6 & -1.4\\
2.4 & 2.3 & 4.7 & -1.7 & -1.6 & 3.8 & -4.0 & 1.3 & -0.4 & -0.4 & 2.9 & 1.2\\
0.0 & -3.2 & -0.2 & 2.0 & -2.9 & 2.7 & 3.1 & 2.9 & -2.6 & -4.3 & 0.2 & 4.6\\
-1.3 & -0.9 & 3.4 & 3.9 & 4.9 & 2.3 & -3.0 & -1.5 & 2.5 & -1.7 & 1.7 & -2.9\\
3.5 & 3.4 & 2.5 & -0.4 & -4.5 & 2.8 & -1.7 & 2.1 & -2.9 & -4.7 & 1.3 & 4.5\\
1.9 & -0.9 & -3.3 & -2.3 & 1.6 & -0.5 & -4.9 & 3.0 & -4.9 & 3.6 & -3.7 & 2.2\\
-1.4 & 3.5 & -2.8 & -1.2 & -4.7 & -3.2 & 2.2 & -4.0 & 2.8 & 3.3 & 4.4 & -3.1\\
-2.1 & 2.6 & -3.9 & 1.0 & 2.3 & 1.8 & 4.2 & 1.8 & 2.7 & 0.9 & 3.3 & 1.7
\end{bmatrix}$$

$$q = (-20.1,\ -1.0,\ 82.6,\ 14.6,\ 37.7,\ 40.7, -23,\ 47.4,\ 83.0,\ 9.9,\ 33.7,\\ 49.1,\ 14.0, -45.6,\ 30.4)^\top$$

• Since the problem can be converted into an equivalent maxproduct problem, we use Algorithm 1** to solve the latter. With tolerance $\varepsilon = 0.1$ the results are as follows:
Optimal value: 0.051237.
Optimal solution:

$$x = (0.000000,\ 0.000000,\ 1.698618,\ 4.345077,\ 4.300919,\ 4.020829,\ 0.000000,\\ 1.432845,\ 0.791345,\ 4.193991,\ 0.000000, 4.152697).$$

Optimal solution found at iteration 38 and confirmed at iteration 604.
Computation time: 135.01 s.
Maximal number of active nodes: 9.

**Acknowledgment**

The authors are grateful to the anonymous referees for useful remarks and suggestions which have helped to improve a first version of the paper.

## References

1. Barros, A.I. (1998), *Discrete and Fractional Programming Techniques for Loaction Models*, Kluwer.
2. Barros, A.I. and Frenk, J.B.G. (1995), Generalized fractional programming and cutting plane algorithms', *Journal of Optimization Theory and Applications* 87: 103–120.
3. Benson, H.P. and Boger, G.M. (2000), Outcome-space cutting plane algorithm for linear multiplicative programming, *Journal of Optimization Theory and Applications* 104: 301–322.
4. Cambini, A., Martein, L. and Schaible, S. (1989), On maximizing a sum of ratios, *Journal of Information and Optimization Science* 10: 141–151.
5. Charnes, A. and Cooper, W.W. (1962), Programming with Linear Fractional Functionals, *Naval Research Logistics Quaterly* 9: 181–186.
6. Craven, B.D. (1988), *Fractional Programming*, Heldermann Verlag, Berlin.
7. Falk, J.E. and Palocsay, S.W. (1992), Optimizing the sum of linear fractional functions. In: Floudas, C. and Pardalos, P. (eds.), *Global Optimization*. Princeton University Press, pp. 221–258.
8. Crouzeix, J.P., Ferland, J.A. and Schaible, S. (1985), An algorithm for generalized fractional programming, *Journal of Optimization Theory and Applications* 47: 35–49.
9. Konno, H. and Abe, N. (1999), Minimization of the sum of three linear fractional functions, *Journal of Global Optimization* 15: 419–432.
10. Konno, H. and Kuno, T. (1992), Linear multiplicative programming, *Mathematical Programming* 56: 51–64.
11. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers, Dordrecht/Boston/London.
12. Konno, H., Yajima, Y. and Matsui, T. (1991), Parametric simplex algorithms for solving a special class of nonconvex minimization problems, *Journal of Global Optimization* 1: 65–81.
13. Konno, H. and Yajima, Y. (1992), Minimizing and maximizing the product of linear fractional functions. In: Floudas, C. and Pardalos, P. (eds.), *Recent Advances in Global Optimization* eds. Princeton University Press, pp. 259–273.
14. Konno, H. and Yamashita, Y. (1997), Minimization of the sum and the product of several linear fractional functions, Tokyo Institute of Technology, Technical Report, Department of IE & Management, to appear in *Naval Research Logistics*.
15. Luc, L.T. (2001), Reverse polyblock approximation for optimization over the weakly efficient set and efficient set, *Acta Mathematica Vietnamica* 26: 65–80.
16. Rubinov, A., Tuy, H. and Mays, H. (2001), Algorithm for a monotonic global optimization problem, *Optimization* 49: 205–221.

17. Stancu-Minasian, I.M. (1997), *Fractional Programming: Theory, Methods and Applications*, Kluwer, Dordrecht.
18. Schaible, S. (1992), Fractional Programming. In: Horst, R. and Pardalos, P. (eds.), *Handbook of Global Optimization*, Kluwer, Dordrecht, pp. 495–608.
19. Tawarmalani, M. and Sahinidis, N.V. (2001), Semiddefinite relaxations of fractional programs via novel convexification techniques, *Journal of Global Optimization* 20: 137–158.
20. Tuy, H. (1998), *Convex Analysis and Global Optimization*, Kluwer, Dordrecht.
21. Tuy, H. (1999), Normal sets, polyblocks and monotonic optimization, *Vietnam Journal of Mathematics* 27(4): 277–300.
22. Tuy, H. (2000), Monotonic optimization: problems and solution approaches, *SIAM Journal of Optimization* 11(2): 464–494.
23. Tuy, H. and Luc, L.T. (2000), A new approach to optimization under monotonic constraint, *Journal of Global Optimization* 18: 1–15.