Vietnam Journal of Mathematics **31**:4 (2003) 391–402

Vietnam Journal
of
MATHEMATICS

Reverse Polyblock Approximation for Generalized Multiplicative/Fractional Programming

Hoang Tuy[1] **and Nguyen Duc Nghia**[2]

[1] *Institute of Mathematics, 18 Hoang Quoc Viet Road, 10307 Hanoi, Vietnam*
[2] *Faculty of Information Technology, Hanoi University of Technology, Hanoi, Vietnam*

Received June 11, 2002
Revised October 1, 2002

Abstract. We present a new efficient approach to a wide class of global optimization problems that includes as special cases generalized linear multiplicative and linear fractional programming problems (i.e. problems of minimizing the sum of a linear function and the product, or the ratio, of two linear functions over a polytope). Our approach is based on the recently developed theory of monotonic optimization.

1. Introduction

Consider a class of nonconvex global optimization problems which have the following general formulation

$$\min\{g_0(x) + \Phi(g_1(x), g_2(x))|\ x \in D\}, \tag{GM}$$

where D is a polytope in $\mathbb{R}^n$, $g_0(x), g_1(x), g_2(x)$ are affine functions satisfying $g_i(x) > 0 \ \ \forall x \in D (i = 0, 1, 2)$ and $\Phi(y_1, y_2)$ is an increasing function, i.e. a function satisfying $\Phi(y_1', y_2') \geq \Phi(y_1, y_2)$ whenever $(y_1', y_2') \geq (y_1, y_2)$. Examples of increasing functions $\Phi(.,.)$ include:

$\Phi(g_1(x), g_2(x)) = [g_1(x)]^{p_1}[g_2(x)]^{p_2}$ with $p_1, p_2 > 0$ $(\Phi(y_1, y_2) = y_1^{p_1}.y_2^{p_2})$;

$\Phi(g_1(x), g_2(x)) = p_1 e^{g_1(x)} + p_2 e^{g_2(x)}$ with $p_1, p_2 > 0$ $(\Phi(y_1, y_2) = p_1 e^{y_1} + p_2 e^{y_2})$;

$\Phi(g_1(x), g_2(x)) = \dfrac{g_1(x)^{p_1}}{r - g_2(x)^{p_2}}$ with $p_1, p_2 > 0, r > \max\limits_{x \in D}[g_2(x)]^{p_2}$
$(\Phi(y_1, y_2) = y_1^{p_1}/(r - y_2^{p_2}))$;

$\Phi(g_1(x), g_2(x)) = p_1 \log g_1(x) + p_2 \log g_2(x)$ with $p_1, p_2 > 0$
$(\Phi(y_1, y_2) = p_1 \log y_1 + p_2 \log y_2)$.

Several special cases of this problem have been studied earlier when $\Phi(y_1, y_2)$ is quasiconcave [7], or $\Phi(y_1, y_2) = y_1 y_2$ [4, 6]. See also related results in [3, 8]. However, the algorithms discussed in all these papers cannot be extended to solve (GM) in the general case considered here.

The aim of the present paper is to propose a method for solving (GM) based on the monotonic approach developed in [9] (with modifications and improvements in [10]). In Sec. 2 we shall reformulate the problem as a monotonic optimization problem as defined in [9] and study some basic properties which serve as the foundation of the solution method to be proposed. In Sec. 3 we shall describe this method, called the reverse polyblock approximation method. Sec. 4 is devoted to some implementation issues, while Secs. 5 and 6 present illustrative examples and computation experience with this method. We close the paper with some remarks in the last Sec. 7.

As a matter of notation we write $a \leq b$ to mean that $a_i \leq b_i, \ \forall i$, and $a < b$ to mean that $a_i < b_i, \ \forall i$. For $a \leq b$ we denote by $[a, b]$ the set of all $y \in \mathbb{R}^3$ such that $a \leq y \leq b$. Analogously, $[a, b) = \{y \in \mathbb{R}^3 : \ a \leq y < b\}$, $(a, b] = \{y \in \mathbb{R}^3 : a < y \leq b\}$. As usual, e^i denotes the vector with the i-th coordinate equal 1, the other coordinates equal 0 $(i = 0, 1, 2)$, and $e = (1, 1, 1)$, i.e. $e = e^0 + e^1 + e^2$.

2. Basic Properties

Before describing the method it is expedient to discuss some basic properties that motivate the method. For completeness we shall give a short proof for each of these properties, although most of them can be derived from the general theory of monotonic optimization [9, 10].

Let $b \in \mathbb{R}^3_{++}$ be a vector such that $[0, b] \supset g(D) = \{y \in \mathbb{R}^3 : \ y_i = g_i(x), x \in D\}$ (e.g. $b_i > \max_{x \in D} g_i(x)$ for $i = 0, 1, 2$). Define

$$H = \{y \in [0, b] : \ y_0 \geq g_0(x), \ y_1 \geq g_1(x), \ y_2 \geq g_2(x), \ x \in D\}, \tag{1}$$

$$\varphi(y) = y_0 + \Phi(y_1, y_2), \tag{2}$$

and consider the problem

$$\min\{\varphi(y) |\ y \in H\}. \tag{MOP}$$

Proposition 1. *If $\bar{x} \in D$ solves problem* (GM) *then $\bar{y} = (g_0(\bar{x}), g_1(\bar{x}), g_2(\bar{x}))$ solves* (MOP). *Conversely, if $\bar{y}$ solves* (MOP) *then any $\bar{x} \in D$ satisfying $\bar{y}_i \geq g_i(\bar{x}), i = 0, 1, 2$ solves* (GM).

Proof. If $\bar{x} \in D$ solves (GM) and $\bar{y} = (g_0(\bar{x}), g_1(\bar{x}), g_2(\bar{x}))$ then $\bar{y}$ is feasible to (MOP) and for any feasible solution y of (MOP), i.e. such that $y_i \geq g_i(x), i = 0, 1, 2$ for some $x \in D$ we will have $y_0 + \Phi(y_1, y_2) \geq g_0(x) + \Phi(g_1(x), g_2(x)) \geq g_0(\bar{x}) + \Phi(g_1(\bar{x}), g_2(\bar{x})) = \bar{y}_0 + \Phi(\bar{y}_1, \bar{y}_2)$, so that $\bar{y}$ solves (MOP). Conversely, if $\bar{y}$ solves (MOP) and $\bar{y}_i \geq g_i(\bar{x}), i = 0, 1, 3$ for $\bar{x} \in D$ then for any $x \in D$ we will have $g_0(x) + \Phi(g_1(x), g_2(x)) \geq \bar{y}_0 + \Phi(\bar{y}_1, \bar{y}_2) \geq g_0(\bar{x}) + \Phi(g_1(\bar{x}), g_2(\bar{x}))$, i.e. $\bar{x}$ solves (GM). ∎

Proposition 2. *The set H is a reverse normal set in $[0, b]$, i.e. satisfies*

$$y \le y' \le b,\ y \in H \Rightarrow y' \in H$$

Proof. Immediate. ∎

Since $\varphi(y)$ is increasing, while H is a reverse normal set, (MOP) is a monotonic optimization problem of the type studied in [9].

For each point $y \in [0,b] \setminus H$ define

$$\rho(y) = b + \lambda(y-b), \quad \text{where} \quad \lambda = \max\{\theta :\ b + \theta(y-b) \in H\}. \tag{3}$$

Proposition 3. *For $y \in [0,b] \setminus H$ the number $\lambda = \lambda(y)$ satisfying (3) is the optimal value of the linear program*

$$\max\{\theta|\ b_i + \theta(y_i - b_i) \ge g_i(x)\ (i = 0,1,2),\ x \in D\} \qquad \mathrm{LP}(y)$$

Proof. This follows from the definition of H. ∎

For any finite set $T \subset [0,b]$ the set $P = \cup_{z\in T}[z,b]$ is called a *reverse polyblock* in $[0,b]$ generated by T. Every $z \in T$ is then called a *vertex* of the reverse polyblock P. A vertex z is said to be *proper* if no other vertex z' is dominated by z, i.e. satisfies $z' \le z$, *improper* otherwise. Clearly, a reverse polyblock is fully determined by its proper vertex set. The simplest reverse polyblock in $[0,b]$ is obviously $[0,b]$ (generated by $\{0\}$).

Proposition 4. *The intersection of finitely many reverse polyblocks is a reverse polyblock.*

Proof. For any two $z, z' \in [0,b]$ we have $[z,b] \cap [z',b] = [z \vee z', b]$ where $y = z \vee z'$ means that $y_i = \max\{z_i, z'_i\}, i = 0,1,2$. Therefore, $(\cup_{z\in T}[z,b]) \cap (\cup_{z'\in T'}[z',b]) = \cup_{z,z'}([z,b] \cap [z',b]) = \cup_{z,z'}[z \vee z', b]$. ∎

Proposition 5. *Let P be a reverse polyblock with proper vertex set $T \subset [0,b]$, let $v \in T, \tilde{v} \in (v,b]$, and for every $z \in T_* := \{z \in T|\ z < \tilde{v}\}$ define $z^i = z + (\tilde{v}_i - z_i)e^i,\ i = 0,1,2$. Then the reverse polyblock $P \setminus [0,\tilde{v})$ has vertex set*

$$T' = (T \setminus T_*) \cup \{z^i|\ z \in T_*,\ i = 0,1,2\} \tag{4}$$

and its proper vertex set is obtained from T' by removing improper elements according to the rule:

$(*)$ For every pair z, y such that $z \in T_*$ and $y \in T_*^- := \{y \in T|\ y \le \tilde{v}\}$ compute $J(z,y) := \{j|\ z_j < y_j\}$ and remove z^i if $J(z,y) = \{i\}$.

Proof. Since $[z,b] \cap [0,\tilde{v}) = \emptyset$ for every $z \in T \setminus T_*$, it follows that $P \setminus [0,\tilde{v}) = P_1 \cup P_2$, where P_1 is the reverse polyblock generated by $T \setminus T_*$ and $P_2 = (\cup_{z\in T_*}[z,b]) \setminus [0,\tilde{v}) = \cup_{z\in T_*}([z,b] \setminus [0,\tilde{v}))$. Noting that $[0,b] \setminus [0,\tilde{v})$ is a polyblock with vertices $u^i = \tilde{v}_i e^i,\ i = 0,1,2$, we can then write $[z,b] \setminus [0,\tilde{v}) = [z,b] \cap ([0,b] \setminus [0,\tilde{v})) = [z,b] \cap (\cup_{i=0,1,2}[u^i,b]) = \cup_{i=0,1,2}[z,b] \cap [u^i,b] = \cup_{i=0,1,2}[z \vee u^i, b]$, hence $P_2 = \cup_{z\in T_*}([z,b] \setminus [0,\tilde{v})) = \cup\{[z^i,b]|\ z \in T_*, i = 0,1,2\}$, which shows that the vertex set of $P \setminus [0,\tilde{v})$ is the set T' given by (1).

It remains to show that every $z \in T \setminus T_*$ is proper, while a z^i with $z \in T_*$ is improper if and only if $J(z,y) = \{i\}$ for some $y \in T_*^-$.

Since every $y \in T \setminus T_*$ is proper in T, if $z' \leq y$ for some $z' \in T'$, then z' must be some z^i with $z \in T_*,\ i \in \{0,1,2\}$. But then $z \leq z^i \leq y$, conflicting with y being proper in T. Therefore, every $y \in T \setminus T_*$ is proper (in T'), and an improper element of T' must be some z^i for $z \in T_*$. Then either $z^i \geq y$ for some $y \in T$, or $z^i \geq y^l$ for some $y \in T_*$ and $l \in \{0,1,2\}$. In the first case, i.e. when $z^i \geq y \in T$, we have $\tilde{v} \geq z^i \geq y$, so $y \in T_*^-$, while $z_j = z_j^i \geq y_j\ \forall j \neq i$; the latter implies $z_i < y_i$ (for otherwise $z \geq y$, conflicting with T being proper), and consequently $J(z,y) = \{i\}$. In the second case, i.e. when $z^i \geq y^l$ for some $y \in T_*$, then for $i \neq l$ we have $z_l = z_l^i \geq y_l^l = \tilde{v}_l$, conflicting with $z < \tilde{v}$. Therefore, $i = l$, i.e., $z^i \geq y^i$, hence $z_i^i = \tilde{v}_i = y_i^i$, while for $j \neq i:\ z_j = z_j^i \geq y_j^i = y_j$, so that $z \geq y$, conflicting with z being proper in T. Consequently, if z^i is improper then there exists $y \in T_*^-$ such that $J(z,y) = \{i\}$. Conversely, if $J(z,y) = \{i\}$ for some $y \in T_*$ then $z_j \geq y_j\ \forall j \neq i$, hence $z^i \geq y^i$, i.e. z^i is improper. This completes the proof of the Proposition. ∎

Remark 1. When $T_* = \{v\}$, T' is exactly the proper vertex set of $P \setminus [0,x)$.

Proposition 6. *The minimum of $\varphi(y)$ over a reverse polyblock P is attained at a proper vertex. Any global optimal solution $\bar{y}$ of* (MOP) *must lie on the lower boundary of H, i.e. must satisfy $b + \theta(\bar{y} - b) \notin H\ \forall \theta > 1$.*

Proof. Indeed, the minimum of $\varphi(y)$ over any box $[z,b] \subset [0,b]$ is attained at z.

3. The Reverse Polyblock Algorithm

The properties discussed above suggest the following procedure for finding the minimum of $\varphi(y)$ over H.

Start with an initial set T_1 generating a reverse polyblock containing H. (e.g. $T_1 = \{0\}$). At iteration k a finite set T_k has been already defined such that the reverse polyblock P_k generated by T_k contains H. Select a proper vertex $v^k \in T_k$ with $\varphi(v^k) = \min\{\varphi(z) |\ z \in T_k\}$. If $v^k \in H$, then v^k solves (MOP). Otherwise, $\tilde{v}^k = \rho(v^k)$ satisfies $\tilde{v}^k \in (v^k, b]$, so using Proposition 5 construct a smaller reverse polyblock P_{k+1} still containing H but excluding v^k. Go to iteration $k+1$.

In a formal way we can state

RPA Algorithm. [1]

Initialization. Select $\varepsilon \geq 0$ (tolerance). Let T_1 be the vertex set of an initial reverse polyblock P_1 containing H (for example $T_1 = \{0\}$). Let $y^1 \in H$ be the best feasible solution available of (MOP) (the current best feasible solution), CBV$= \varphi(y^1)$. If no feasible solution is available, set CBV$= +\infty$. Set $k = 1$.

Step 1. From T_k remove all $z \in T_k$ such that $\varphi(z) \geq$ CBV $- \varepsilon$. Let $\widetilde{T}_k$ be the set of remaining elements of T_k.

Step 2. If $\widetilde{T}_k = \emptyset$, terminate: if CBV$= +\infty$, the problem is infeasible; if CBV $< +\infty, y^k$ is an ε-optimal solution.

Step 3. If $\widetilde{T}_k \neq \emptyset$, select $v^k \in \operatorname{argmin}\{\varphi(z) |\ z \in \tilde{T}_k\}$.

Solve LP(v^k) to obtain its optimal value λ_k. If $\lambda_k = 1$, i.e. $v^k \in H$, terminate: v^k is an optimal solution of (MOP), within prescribed tolerance. Otherwise, let $\widetilde{v}^k = b + \lambda_k(v^k - b) = \rho(v^k)$. Determine the new CBV and the new current best solution y^{k+1}.

Step 4. Let $\tilde{T}_{k,*} = \{z \in \tilde{T}_k |\ z < \tilde{v}^k\}$. Compute

$$T'_k = (\widetilde{T}_k \setminus \widetilde{T}_{k,*}) \cup \{z^i |\ z \in T_{k,*}, i = 0, 1, 2\}, \tag{5}$$

where $z^i = z + (x_i^k - z_i)e^i$. Let T_{k+1} be the set obtained from T'_k by removing every z^i such that $J(z, y) = \{i\}$ for some $y \in \widetilde{T}^-_{k,*} := \{y \in \widetilde{T}_k \,|y \leq \widetilde{v}^k\}$.

Step 5. Increment k and go back to Step 1.

Without loss of generality we can always assume that b has been chosen so that, for some $\alpha > 0$:

$$\min(b_i - y_i) \geq \alpha \|b - y\|, \quad \forall y \notin H. \tag{6}$$

Under this innocuous assumption

Proposition 7. *The RPA Algorithm can be infinite only if $\varepsilon = 0$ and in that case it generates an infinite sequence $\{y^k\}$ every cluster point of which is a global optimal solution.*

Proof. We first show that $v^k - \tilde{v}^k \to 0$ as $k \to \infty$. Indeed if it were not so, there would exist $\eta > 0$ and an infinite sequence k_ν such that $\|v^{k_\nu} - \tilde{v}^{k_\nu}\| \geq \eta > 0\ \forall \nu$. But for all $\mu > \nu$ we have $v^{k_\mu} \notin [0, \tilde{v}^{k_\nu})$ because $P_{k_\mu} \subset P_{k_\nu} \setminus [0, \tilde{v}^{k_\nu})$ by Proposition 5. Hence, noting that $v^{k_\mu} \notin H$ and taking account of (6), $\|v^{k_\mu} - \tilde{v}^{k_\nu}\| \geq \min_{i=0,1,2} \|\tilde{v}_i^{k_\nu} - v_i^{k_\nu}\| \geq \alpha \|\tilde{v}^{k_\nu} - v^{k_\nu}\| \geq \alpha\eta > 0$, conflicting with the fact that $\{v^{k_\nu}\}$ is bounded. Therefore, $v^k - \tilde{v}^k \to 0$ and since $\{v^k\}$ is bounded, we may assume, by passing to subsequences if necessary, that $v^{k_\nu} \to \bar{y},\ \tilde{v}^{k_\nu} \to \bar{y}$. Then, since $\tilde{v}^k \in H\ \forall k$, it follows that $\bar{y} \in H$, i.e. $\bar{y}$ is feasible. Furthermore, $\varphi(v^k) \leq \varphi(z)\ \forall z \in P_k \supset H$, hence by letting $k \to +\infty : \varphi(\bar{y}) \leq \varphi(y)\ \forall y \in H$, i.e. $\bar{y}$ is an optimal solution. Since $\varphi(y^{k+1}) \leq \varphi(\tilde{v}^k)\ \forall k$, any cluster point of the sequence $\{y^k\}$ will also be an optimal solution. ∎

Remark 2. The condition that D is a polytope can be weakened by assuming only that D is a polyhedron such that every affine function $g_i(x),\ \ i = 0, 1, 2$ is bounded on D.

4. Implementation Issues

4.1. Preprocessing

Suppose that a problem (GM) is given where all conditions specified in the Introduction are satisfied, except the condition $g_i(x) > 0,\ \ \forall x \in D\ (i = 0, 1, 2)$.

In this case, if $\alpha_i = \min\{g_i(x)|\ x \in D\} - 1$, then $\tilde{g}_i(x) = g_i(x) - \alpha_i > 0,\ \ \forall x \in D$ while the problem is equivalent to

$$\alpha_0 + \min \tilde{g}_0(x) + \tilde{\Phi}(\tilde{g}_1(x), \tilde{g}_2(x)),$$

where $\widetilde{\Phi}(y_1, y_2) = \Phi(y_1 + \alpha_1, y_2 + \alpha_2)$. Now the functions $\tilde{g}_i(x), i = 0, 1, 2$ are positive on D and the function $\tilde{\Phi}(y_1, y_2)$ is increasing on $(y_1, y_2) \geq 0$, so that all required conditions are satisfied. Thus the positivity assumption for $g_0(x), g_1(x), g_2(x)$ is innocuous. However, before applying RPA Algorithm it is necessary to make sure that this assumption holds.

Also, before proceeding to the algorithm it is useful to tighten the box $[0, b]$ when possible. For this let

$$\gamma_i = \sup\{\gamma > 0 :\ b - \gamma e^i \in H\}, \quad a = b - \sum_{i=1}^{n} \gamma_i e^i.$$

Then clearly $H \subset [a, b]$, so by resetting $H \leftarrow H \cap [a, b]$ and shifting the origin to a we have a box $[0, b]$ which is a tight approximation of H.

4.2. Initial Set T_1

With the initial set taken to be $T_1 = \{0\}$ the search direction in the next iteration is only one of the edges of the box $[0, b]$ emanating from 0. If none of these directions is promising (as it happens in problems where the function $\varphi(y)$ does not depend actually on y_i for some i) then many wasteful iterations will have to be performed before finding a good feasible solution. Therefore, instead of the set $\{0\}$ (vertex set of the reverse polyblock $P_1 = [0, b]$) it is often advisable to take an initial set T_1 which could provide a larger choice of initial search directions. One such set may be constructed as follows.

Consider a grid $U = \{c^0, c^1, c^2\} \subset \{z \in \mathbb{R}^3_+|\ z_0 + z_1 + z_2 = 1\}$. For example, let U consist of the following points

$$c^3 = (1/3, 1/3, 1/3) \quad \text{(barycentre of the unit simplex)},$$

$$c^0 = (1/9, 4/9, 4/9), c^1 = (4/9, 1/9, 4/9), c^2 = (4/9, 4/9, 1/9)$$

(barycentres of subsimplices $[c^3, e^1, e^2], [e^0, c^3, e^2], [e^0, e^1, c^3]$, resp.).

For each $k = 0, 1, 2$, let $\tilde{c}^k = \rho(c^k)$ (by Proposition 3, $\rho(c^k)$ is determined by solving LP(c^k)). Construct a set T_1 by proceeding as follows:
1. Let $T_1 = \{u^0, u^1, u^2\}$ with $u^i = b + (\tilde{c}^0_i - b_i)e^i \quad i = 0, 1, 2$. Set $k = 1$.
2. Let $T_{k,*} = \{z \in T_1|\ z \geq \tilde{c}^k\}$, and compute

$$T'_k = (T_k \setminus T_{k,*}) \cup \{z^i|\ z \in T_{k,*}, i = 0, 1, 2\} \tag{7}$$

where $z^i = z + (x^k_i - z_i)e^i$. Let T_{k+1} be the set obtained from T'_k by removing improper elements accroding to rule $(*)$. If $k < n$, let $k \leftarrow k + 1$ and go back to Step k. If $k = n$, stop.
The last set T_1 returned by the above procedure is then taken to be the initial polyblock for Algorithm RPA.

4.3. Alleviating Storage Problems

A potential disadvantage of the method is that the set T_k increases in size as the algorithm proceeds. To avoid storage problems which may arise, and also to preclude possible jams because of the rapid growth of T_k, it may be useful to *restart* the algorithm whenever $|T_k|$ exceeds a prescribed limit L or when the current best value remains unchanged during a prescribed number h of iterations. To allow for restart, Steps 4 and 5 should be modified as follows:

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

Step 5. Redefine $T_{k+1} = y_i^{k+1}.e^i, i = 0, 1, 2$ (i.e. $P_{k+1} = [0, b] \setminus [0, y^{k+1})$) and return to Step 1.

Another method for reducing the size of T_k is to use a simplified rule for computing T_{k+1} in Step 4 of the algorithm. In fact, the set T_{k+1} computed in Step 4 according to Proposition 5 may be too numerous, if $T_{k,*}$ involves many more elements other than $\tilde{v}^k$. To alleviate this difficulty one can use from time to time the following simplified rule for computing T_{k+1}.

Let $\tilde{T}_{k,*} = \{z \in \tilde{T}_k |\ z < \tilde{v}^k\}$. Compute

$$w = (w_0, w_1, w_2), \quad w_j = \min\{z_j |\ z \in \tilde{T}_{k,*}\}, \quad j = 0, 1, 2,$$
$$z^{*i} = w + (\tilde{v}_i - w_i)e^i, \quad i = 0, 1, 2.$$
$$T_k^\circ = (\tilde{T}_k \setminus \tilde{T}_{k,*}) \cup \{z^{*0}, z^{*1}, z^{*2}\}.$$

Let T_{k+1} be the set obtained from T_k^0 by removing improper elements accroding to rule $(*)$.

It can be proved that, if $\tilde{P}_k$ denotes the reverse polyblock with vertex set $\tilde{T}_k$ then the reverse polyblock P_k° with vertex set T_k° satisfies

$$H \subset \tilde{P}_k \setminus [0, \tilde{v}) \subset P^\circ \subset [0, b] \setminus [0, \tilde{v})$$

while T_{k+1} is the proper vertex set of P_k°.

Clearly the set T_{k+1} computed in that way is the same as the one computed as in Algorithm RPA when $\tilde{T}_{k,*}$ is the singleton $\{v^k\}$. However, while this set always includes at most n new elements, which is very convenient for our purpose, it may generate a larger reverse polyblock than $\tilde{P}_k \setminus [0, \tilde{v}^k)$, which may cause problems for convergence. Therefore, the simplified rule should be used only to alleviate storage problems in a limited number of iterations.

5. Illustrative Examples

Example 1.

$$\begin{aligned} \min\ & x_1 + 2x_2 + 5 + \frac{-5x_1 + 3x_2 + 15}{30 - 3x_1 - 7x_2} \\ \text{s.t. } & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 2 \\ & -3x_1 - x_2 \leq -3 \\ & -x_1 - 6x_2 \leq -6 \\ & x_1 \leq 3 \end{aligned}$$

Denote the constraint polytope by D. Then the equivalent problem (MOP) is

$$\begin{aligned} \min\ & y_0 + \frac{y_1}{30 - y_2} \\ \text{s.t. } & y_0 \geq x_1 + 2x_2 + 5 \\ & y_1 \geq -5x_1 + 3x_2 - 15 \\ & y_2 \geq 3x_1 + 7x_2. \\ & x \in D \end{aligned}$$

Solution

Initialization. Since $\max_{x\in D}(x_1 + 2x_2 + 5) = 13.5,\ \max_{x\in D}(-5x_1 + 3x_2 - 15) = 20.5,\ \max_{x\in D}(3x_1 + 7x_2) = 29$, we take $b = (14.85,\ 22.55,\ 31.90)$. The optimal solutions of these linear programs are, respectively $(1.5, 3.5),\ (0.25, 2.25), (1.5, 3.5)$. So the best feasible solution available is CBS $= y^0 = (1.5, 3.5)$ with objective function value CBV $= \varphi(y^0) = 9.773595$. With tolerance $\varepsilon = 0.05$ the computation results are as follows:

Iteration 1.

* T_1= {(0.000000, 0.000000, 0.000000)}
* Vector v^1: (0.000000, 0.000000, 0.000000)
* Vector $\widetilde{v}^1$: (7.754502, 11.775355, 15.749212)
* Lower Bound = 0.000000
* CBV: 10.085385
* CBS: (0.250000 , 2.250000)

Iteration 2.

* T_2 = {(0.000000, 0.000000, 15.749212) with $\varphi(.) = 0.000000$; (7.754502, 0.000000, 0.000000) with $\varphi(.) = 7.754502$ (0.000000, 11.775355, 0.000000) with $\varphi(.) = 0.114384$};
* Vector v^2: (0.000000, 0.000000, 15.749212)
* Vector $\widetilde{v}^2$: (7.754502, 11.775355, 24.617114)
* Lower Bound = 0.000000
* CBV: 7.918493
* CBS: (1.131754, 0.811374)

Iteration 3.

* T_3 = {(0.000000,11.775355 0.00000) with $\varphi(.) = 0.114384$; (7.754502, 0.000000, 0.000000) with $\varphi(.) = 7.754502$; (0.000000 0.000000, 23.274363) with $\varphi(.) = 0.00000$}
* Vector v^3: (0.000000 , 0.000000, 23.274363)
* Vector $\widetilde{v}^3$: (7.754502, 11.775355, 26.869965)
* Lower Bound = 0.000000
* CBV: 7.918493
* CBS: (1.131754, 0.811374)

Iteration 4.

* T_4 = {(7.754502, 0.000000, 0.000000) with $\varphi(.) = 7.754502$; (0.000000, 11.775355, 0.000000) with $\varphi(.) = 0.114384$; (0.00000, 0.0000, 26.869965) with $\varphi(.) = 0.00000$}.
* Vector v^4: (0.000000, 0.000000, 26.869965)

* Vector $\widetilde{v}^4$: (7.754502, 11.775355, 28.587984)
* Lower Bound = 0.000000
* CBV: 7.918493
* CBS: (1.131754, 0.811374)

Iteration 5.

* T_5 = {(0.000000 11.775355 0.000000) with $\varphi(.) = 0.114384$; (7.754502 0.000000 0.000000) with $\varphi(.)= 7.754502$; (0.00000, 0.00000, 28.587984) with $\varphi(.) = 0.00000$}.
* Vector v^5: (0.00000, 0.00000, 28.587984)
* Vector $\widetilde{v}^5$: (0.000000, 11.775355, 29.408873)
* Lower Bound = 0.00000
* CBV: 7.918493
* CBS: (1.131754, 0.811374)

. . . . . .

The algorithm terminates after 10 more iterations, with the following results:
* Optimal value: 7.643691
* Optimal solution $x^* = (0.705882,\ 0.882353)$
* Number of solved LP's: 11
* Number of restarts: 0
* Maximal number of vertices: 4

Example 2.

$$\min\{\langle c^0, x\rangle + (\langle c^1, x\rangle + \beta_1) \cdot (\langle c^2, x\rangle + \beta_2) |\ x \in D \subset \mathbb{R}^{10}\}$$

where $D = \{x \in \mathbb{R}^{10} |\ Ax \leq q,\ x \geq 0\}$ with

$c^0 = (6.664, 7.208, 10.912, 9.026, 7.046, 3.846, 10.090, 5.081, 2.621, 7.623)^T$

$c^1 = (–0.243, –0.708, 0.399, –0.638, 0.182, –0.181, –0.104, –0.355, –0.690, 0.080)^T$;

$c^2 = (1.339, 1.590, 1.619, 1.813, 1.109, 0.876, 1.665, 1.025, 0.851, 1.266)$;

$\beta_1 = 6.000;\ \ \beta_2 = 3.000$;

$q = (3.500, 2.280, 11.930, 12.270, 50.000, -12.840, -6.960)^T$;

$$A = \begin{pmatrix} 3.3 & -2.2 & 1.4 & 1.2 & 2.6 & -4.3 & 2.9 & 2.6 & -3.0 & -0.4 \\ 4.7 & 0.2 & 4.3 & -3.2 & 2.9 & 2.1 & -0.5 & -3.5 & 2.8 & -3.2 \\ -1.0 & 2.9 & -0.4 & -1.6 & 3.5 & 0.6 & -4.8 & 3.8 & 3.8 & 4.6 \\ 2.4 & 3.6 & 2.3 & 1.3 & -2.4 & 1.0 & 1.3 & 0.4 & 4.6 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.3 & -4.6 & -1.9 & -3.0 & -3.5 & 2.7 & -2.2 & -0.5 & -1.6 & -4.9 \\ 2.4 & -4.1 & 3.9 & 4.8 & -2.3 & 1.6 & -3.9 & -0.4 & -4.2 & -2.9 \end{pmatrix}$$

Solving this problem by Algorithm RPA [1] gives the following results

Number of iterations: 271
Optimal value: 48.743985
Optimal solution: x^* = (0, 2.572, 0, 0, 0, 0, 0, 0, 0.631, 0)
Number of solved LP's : 277
Number of restarts: 0
Maximal number of vertices : 52

6. Computational Experiments

The algorithm was coded in PASCAL and run on IBM PC Intel Pentium 166 MHz to solve problems of the form

$$\min\{f(x) \equiv g_0(x) + g_1(x) \cdot g_2(x) |\ x \in D\},$$

where $g_i(x)$ are affine functions and D is a polytope in $\mathbb{R}^n$ defined by m constraints, with n ranging from 2 to 100 and m from 4 to 40. The particular choice of $\Phi(y_1, y_2) = y_1 y_2$ in the test problems is motivated by the fact that the method is not very sensitive to the form of $\Phi(y_1, y_2)$. For simplicity we used two termination criteria: either (CBV-LB)$\leq \varepsilon$ LB (criterion 1) or $1 - \lambda_k \leq \bar{\delta}$ (criterion 2).

The computational results are reported in Table 1, where the following notations are used:

n: number of variables;

m: number of constraints;

Iter : number of iterations;

Iopt : index of iteration in which the optimal solution is found;

Vmax: maximal number of vertices in one iteration;

Time: computational time (in sec.);

Crit: 0 for termination by criterion 1, 1 for termination by criterion 2.

Table 1

n	m	Iter	Iopt	Vmax	Time	Crit	ε	δ
2	4	457	287	40	5.88	1	0.69341	0.009988
2	4	3	3	5	0.05	1	1.04753	0.015474
7	6	1849	1173	39	29.99	1	0.19891	0.000969
10	7	1150	754	21	19.38	1	0.09154	0.000828
10	7	238	238	40	3.90	1	2.48452	0.002633
15	5	1751	1671	224	41.91	1	0.21166	0.000952
15	10	26	25	19	1.04	0	0.04333	0.000000
15	15	26	26	16	1.04	1	0.26838	0.000807
15	20	3071	3071	47	164.61	1	1.36999	0.000973
20	5	6	5	7	0.16	0	0.03593	0.000000
20	10	25	25	18	0.93	1	0.14283	0.000919
20	10	4786	2303	22	251.89	1	0.05043	0.000861
20	15	3834	3635	61	320.27	1	0.07848	0.000613
20	20	18	17	10	2.25	0	0.04221	0.000000
25	5	9	8	10	0.27	0	0.03571	0.000000
25	10	472	472	24	38.34	1	0.22859	0.006027

n	m	Iter	Iopt	Vmax	Time	Crit	ε	δ
25	15	604	306	42	68.66	1	0.13340	0.006677
25	20	36	36	71	5.22	1	0.65860	0.007738
30	5	18	18	34	0.55	1	0.35696	0.007382
30	10	21	21	40	1.54	1	0.78589	0.007853
30	15	7	6	8	1.32	0	0.04639	0.000000
30	20	850	850	37	730.73	1	0.33307	0.007123
30	23	25	25	13	5.54	1	1.70088	0.007134
30	30	80	80	24	30.87	1	0.25064	0.002889
35	5	19	19	10	0.66	1	0.15321	0.006172
35	10	500	7	20	45.58	1	0.15312	0.007396
35	15	13	12	14	1.93	0	0.04944	0.000000
35	20	12	12	19	4.06	1	0.40514	0.007208
35	25	525	270	40	186.96	1	0.13056	0.007837
35	30	18	17	9	9.23	0	0.04849	0.000000
35	35	30	30	57	20.05	1	0.27371	0.007622
40	5	9	8	4	0.39	0	0.04431	0.000000
40	10	52	52	91	3.46	1	0.16537	0.007982
40	15	16	13	31	3.51	1	0.39435	0.007602
40	20	576	364	48	156.27	1	0.16009	0.007638
40	25	14	14	27	7.42	1	1.84710	0.007977
40	30	13	13	24	10.38	1	0.28699	0.007307
40	35	6	6	7	5.00	1	0.17816	0.007547
40	40	16	16	29	17.91	1	0.18153	0.007061
45	5	484	12	17	22.30	1	0.06889	0.007465
45	10	13	12	14	1.15	0	0.04943	0.000000
45	20	7	6	8	3.84	0	0.04288	0.000000
45	25	19	19	35	11.48	1	0.30697	0.007380
45	30	30	30	17	21.81	1	1.44745	0.005789
50	10	233	165	11	16.32	0	0.04673	0.000000
50	25	92	69	16	50.36	0	0.10000	0.000000
60	20	16	16	30	8.57	1	0.46732	0.007224
70	15	19	19	37	5.93	1	0.50081	0.007221
80	15	20	20	13	6.75	1	0.00469	0.000000
90	20	21	21	12	17.19	1	0.05932	0.005123
100	30	8	7	9	21.2	0	0.00475	0.000000
100	30	1114	1109	42	4107.5	1	0.17417	0.056390

7. Final Remarks

We have presented in this paper a monotonic approach to a wide class of nonconvex optimization problems including so called generalized linear multiplicative and linear fractional programming problems. Of course the same approach works for maximization rather than minimization problems (but then polyblocks, rather than reverse polyblocks should be used).

It should also be noted that our approach could be extended to more general problems of the form

$$\min\{g_0(x) + \Phi(g_1(x), \dots, g_k(x)| \quad x \in D^n\},$$

where D is a compact convex subset of $\mathbb{R}^n$, $g_i(x),\ i = 0, 1, \dots, k$ are convex positive-valued on D while $\Phi(y_1, \dots, y_k)$ is an increasing function. In that general case, the approach would involve solving convex minimization rather than linear subproblems; furthermore, for $n > 5$ a branch and bound approach generalizing the one described above would be more efficient. This will constitute a subject for future research.

References

1. N. T. Hoai Phuong and H. Tuy, *A Monotonicity Based Approach to Nonconvex Quadratic Minimization*, Preprint, Institute of Mathematics, Hanoi, 2001.
2. H. Konno and T. Kuno, *Generalized Linear Multiplicative and Fractional Programming*, IHSS Report 89-13, Institute of Human and Social Sciences, Tokyo Institute of Techology, 1989.
3. H. Konno, P. T. Thach, and H. Tuy, *Optimization on Low Rank Nonconvex Structures*, Kluwer, 1997.
4. H. Konno, Y. Yajima, and T. Matsui, Parametric simplex algorithms for solving a special class of nonconvex minimization problems, *J. Global Optim.* **1** (1991) 65–81.
5. L. D. Muu and B. T. Tam, Minimizing the sum of a convex function and the product of two affine functions over a convex set, *Optimization* **24** (1992) 567–620.
6. S. Schaible and C. Sodini, Finite algorithm for generalized linear multiplicative programming, *J. Optim. Theory and Appl.* **87** (1995) 441–455.
7. H. Tuy, B. T. Tam, and N. D. Dan, Minimizing the sum of a convex function and a specially structured nonconvex function, *Optim.* **28** (1994) 237–248.
8. H. Tuy, *Convex Analysis and Global Optim.*, Kluwer, 1998.
9. H. Tuy, Monotonic optimization: Problems and solution approaches, *SIAM J. Optim.* **11** (2000) 464–494.
10. H. Tuy, *Polyblock Algorithms Revisited*, Preprint, Institute of Mathematics, Hanoi, 2001.