Journal of Global Optimization **29:** 19–44, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Optimization of Polynomial Fractional Functions

HOANG TUY, PHAN THIEN THACH[1] and HIROSHI KONNO[2]*
[1]*Institute of Mathematics, 18 Hoang Quoc Viet Road, 10307, Hanoi, Vietnam*
(*e-mail: htuy@math.ac.vn*)
[2]*Department of Industrial Engineering and Management, Tokyo Institute of Technology, Tokyo*

(Received 27 June 2003; accepted 12 October 2003)

Abstract. A new approach is proposed for optimizing a polynomial fractional function under polynomial constraints, or more generally, a synomial fractional function under synomial constraints. The approach is based on reformulating the problem as the optimization of an increasing function under monotonic constraints.

Key words: fractional programming, global optimization, minimizing a sum of many linear fractions, monotonic optimization, polyblock approximation method, polynomial, synomial.

1. Introduction

We shall be concerned with the following polynomial fractional programming problem

$$\max\left\{\frac{h(x)}{g(x)}\middle|\ c_i(x)\leqslant 0\ (i=1,\dots,m),\ x\in[0,r]\right\} \tag{FP}$$

where $r=(r_1,\dots,r_n)$ with $r_i>0\ i=1,\dots,n,\quad [0,r]=\{x\in\mathbb{R}^n|\ 0\leqslant x_i\leqslant r_i\ (i=1,\dots,n)\}$ and $h,g,c_i\colon \mathbb{R}^n\to\mathbb{R}$ are polynomials of any degree such that $g(x)\geqslant\alpha>0$ for all $x\in[0,r]$. Often $c_i(x)$ are affine functions, so that the constraint set is a polytope. Since the problem remains essentially unchanged when we replace $h(x)/g(x)$ with $[h(x)+Mg(x)]/g(x)=[h(x)/g(x)]+M$ where M is a constant so large that $h(x)+Mg(x)>0\ \forall x\in[0,r]$, without loss of generality we can assume that

$$g(x)\geqslant\alpha>0,\quad h(x)>0\quad \forall x\in[0,r]. \tag{1}$$

Also, the seemingly more general case when some of the functions $f(x)=h(x)/g(x)$, $c_i(x)$ are sums of fractions with positive denominators for $0\leqslant x\leqslant r$ can in an obvious way be cast into the above formulation. In particular, by writing a sum of linear fractions as a single polynomial fraction, the problem of maximizing (or minimizing) a sum of linear fractions [5–8], etc., which is encountered in various applications such as: multi-stage stochastic shipping, cluster analysis, multi-objective bond portfolio, etc., can be considered and studied as a special

* This work was completed while Hoang Tuy and Phan Thien Thach were visiting Tokyo Institute of Technology (October 2000-January 2001)

polynomial fractional programming problem. Thus (FP) includes in fact a very broad class of problems of theoretical and practical interest.

In the special case when g, h are both convex quadratic functions, or both d.c. functions, the problem was studied by Gotoh and Konno [7] and J. Shi [20], respectively. If $h(x)$ is convex, $g(x)$ is concave the problem is a quasiconvex maximization; if $h(x)$ is concave, $g(x)$ is convex, the problem is a quasiconcave maximization. Several algorithms are available for solving the problem in these cases. However, when both $h(x)$ and $g(x)$ are convex as considered in [7], the problem is neither convex nor concave, hence is much more difficult. Gotoh and Konno proposed in [7] an efficient method for solving the problem under these assumptions. For a review of fractional programming and its applications up to 1995, see [16] and also the recent book [22].

When $g(x) \equiv 1$ (FP) reduces to the general polynomial programming problem earlier investigated by Shor [21], Sherali and Tuncbilek [17–19] and some other authors [2, 28]. Shor reduces general polynomial programs to quadratic ones, whereas Sherali and Tuncbilek use a technique called Reformulation-Convexification (R-C) to solve the problem by branch and bound, where bounds are computed through reformulation and convex relaxation. Most recently, Lasserre [13, 14] developed a class of positive semidefinite relaxations for polynomial programming with the property that any polynomial program can be approximated as closely as desired by a semidefinite program of this class. A common feature of all these methods is that they require introducing a huge number of additional variables even for problems of small size involving polynomials of high degree.

Since any polynomial is a d.c. function, i.e. a function that can be represented as a difference of two convex functions, a polynomial programming problem, or more generally, a polynomial fractional programming problem is a d.c. optimization problem, and hence, can in principle be solved by d.c. optimization methods (see e.g. [23]). So far, though, very little experience has been gathered on solving polynomial fractional programming problems by d.c. optimization methods.

The purpose of this paper is to present a new approach to polynomial fractional programming (FP) based on the recently developed theory of monotonic optimization [25]. An important feature of this approach is its weak sensitivity with respect to the highest degree of the polynomials involved, which is in contrast with most existing methods for polynomial programming. Furthermore, without any additional manipulation, this approach applies as well to synomial fractional programming, i.e. to problems (FP) when h, g, c_i are synomials (a synomial is a polynomial with positive rational, rather than simply integral, exponents).

In Section 2 we will first show that any fractional programming problem (FP) in $\mathbb{R}^n$ can be converted to a polynomial programming problem in $\mathbb{R}^{n+1}$. Furthermore, as is well known from global optimization theory, the key subproblem for a global optimization method is to transcend a given incumbent value t, i.e. to find, for this value t, a feasible solution to (FP) such that $h(x)/g(x) > t$, if there is one. It turns out that this subproblem is a polynomial program in $\mathbb{R}^n$, so that solving (FP) can also be reduced to solving an adaptively generated sequence of polynomial programs in $\mathbb{R}^n$. Since in global optimization the cost of an extra dimension quickly increases with n, it can be expected and it has been indeed confirmed by computational experience that, as n increases, the latter approach may outperform the former.

In Section 3, after introducing some basic concepts of monotonic optimization we will show how a polynomial programming problem can in turn be written as a monotonic optimization problem, i.e. an optimization problem involving only increasing functions in the objective and the constraints.

Next, in Section 4 we discuss an outer approximation method called polyblock approximation for solving the transformed polynomial optimization problem. The name comes from the fact that the algorithm is similar to the classical outer approximation method for convex maximization but uses ‘polyblocks’ instead of polyhedrons to approximate the feasible set. Although this polyblock outer approximation has several advantages over the classical polyhedral outer approximation, it cannot overcome all drawbacks inherent to outer approximation and, consequently, may not be efficient for solving large scale problems. Therefore, in Section 5 we propose a branch and bound procedure which usually performs better than outer approximation methods especially in high dimension. Moreover, by using a new efficient method for computing bounds, this branch and bound algorithm substantially improves upon the one originally proposed in [25].

Finally Section 7 is devoted to some numerical illustrative examples. Despite their relative simplicity these examples are by no means trivial. The weak sensitivity of the method with respect to the highest degree of the polynomials involved allows it to be applied to advantage to generalized multiplicative programs, such as the optimization of sums or products of linear fractions, especially when the number of variables is small compared to the number of sums or products. This fact will be illustrated by a numerical example (Example 5) featuring the maximization of a sum of many ratios. Also two examples (Examples 6 and 7) will illustrate the application of the method to synomial fractional programming. Hopefully, these examples could help to convince the reader of the viability and the interest of this approach which, to our knowledge, is the first attempt of its kind to attack this class of difficult problems.

2. Reduction to Polynomial Programming

A polynomial programming problem is obviously a special case of (FP) in which $g(x) \equiv 1$. It turns out that conversely, any fractional program can be reduced to a polynomial program according to the following proposition.

PROPOSITION 1. *The problem* (FP) *is equivalent to the polynomial optimization problem*

$$\begin{array}{lll} \text{maximize} & yh(x) & \\ \text{s.t.} & yg(x) \leqslant 1,\ c_i(x) \leqslant 0 \quad (i=1,\dots,m), & \text{(PFP)} \\ & 0 \leqslant x \leqslant r,\ 0 \leqslant y \leqslant 1/\alpha. & \end{array}$$

Proof. If $(\bar{x}, \bar{y})$ solve (PFP) then one must have $\bar{y} = 1/g(\bar{x})$ so $\bar{y}h(\bar{x}) = h(\bar{x})/g(\bar{x})$. For any feasible solution x of (FP) the constraints of (PFP) are satisfied by setting $y = 1/g(x)$, so $yh(x) \leqslant \bar{y}h(\bar{x})$, i.e. $h(x)/g(x) \leqslant h(\bar{x})/g(\bar{x})$. Conversely, if $(\bar{x})$ solves (FP) then for any feasible solution (x, y) of (PFP) we have $yh(x) \leqslant h(x)/g(x) \leqslant h(\bar{x})/g(\bar{x}) = \bar{y}h(\bar{x})$ with $\bar{y} = 1/g(\bar{x})$. □

Thus, a fractional program in $\mathbb{R}^n$ can be solved through solving an equivalent polynomial program in $\mathbb{R}^{n+1}$.

On the other hand, it is well known (see [9, 11] or [23]) that the key for solving a global optimization problem is to transcend any incumbent value t of the objective function. In the case of (FP) this subproblem consists in the following:

$(*)$ *Given a value t, find a feasible solution x to (FP) such that $h(x)/g(x) > t$ or else establish that no such solution exists.*

If $\bar{t} = h(\bar{x})/g(\bar{x})$ for some feasible solution $\bar{x}$ then solving $(*)$ for $t = \bar{t}$ amounts to checking whether $\bar{x}$ is optimal and, if it is not, finding a better feasible solution.

In general we must be content with an approximate optimal value. A number $\bar{t}$ is said to be an *ε-optimal value* of (FP) if $|t^* - \bar{t}| \leqslant \varepsilon\bar{t}$ (so that the relative error made by accepting $\bar{t}$ as optimal is at most ε). A vector $\bar{x} \in X$ is said to be an *ε-optimal solution* of (FP) if $\bar{t} = h(\bar{x})/g(\bar{x})$ is ε-optimal, i.e. if $|t^* - h(\bar{x})/g(\bar{x})| \leqslant \varepsilon h(\bar{x})/g(\bar{x})$.

The next proposition shows that solving $(*)$ reduces to solving a polynomial program in $\mathbb{R}^n$. For any fixed t let $F(t)$ denote the optimal value of the problem

$$\max\{h(x) - tg(x) \mid c_i(x) \leqslant 0\ (i=1,\dots,m),\ 0 \leqslant x \leqslant r\}. \qquad \text{(PP}(t))$$

(with the usual convention $F(t) = -\infty$ in case of infeasibility). Let X be the feasible set of this problem, i.e. also the feasible set of (FP).

PROPOSITION 2. (i) *$F(t)$ is a convex decreasing function.*

(ii) *Let x^t be an optimal solution of problem $PP(t)$. Then $h(x^t)/g(x^t) > t$ if and only if $F(t) > 0$ and x^t is an optimal solution of (FP) if and only if $F(t) = 0$.*

(iii) *If $|F(t)| \leqslant \varepsilon\alpha t$ then t is an ε-optimal value and any $\bar{x} \in X$ such that $|h(\bar{x}) - tg(\bar{x})| \leqslant \varepsilon\alpha t$ is an 2ε-optimal solution of (FP).*

Proof. (i) and (ii) are well known results [1, 10]. Let us prove (iii). If $F(t) \leqslant \varepsilon\alpha t$ then $h(x) - tg(x) \leqslant F(t) \leqslant \varepsilon\alpha t\ \forall x \in X$, hence, $h(x)/g(x) - t \leqslant \varepsilon\alpha t/g(x) \leqslant \varepsilon t\ \forall x \in X$ (because $0 < \alpha \leqslant g(x)$). This implies that $t^* - t \leqslant \varepsilon t$. On the other hand, since $F(t) \geqslant -\varepsilon\alpha t$ there exists $x \in X$ such that $h(x) - tg(x) \geqslant -\varepsilon\alpha t$, hence $h(x)/g(x) - t \geqslant -\varepsilon\alpha t/g(x) \geqslant -\varepsilon t$, and consequently, $t^* - t \geqslant -\varepsilon t$. Thus, $-\varepsilon t$

$\leqslant t^*-t\leqslant \varepsilon t$, proving that t is an ε-optimal value. Furthermore, this implies the existence of a $\bar{x}\in X$ such that $|h(\bar{x})/g(\bar{x})-t|\leqslant \varepsilon$, hence $|t^*-h(\bar{x})/g(\bar{x})| \leqslant |t^*-t|+|t-h(\bar{x})/tg(\bar{x})|\leqslant 2\varepsilon$. □

Thus, the optimal value t^* of (FP) is merely the zero of a decreasing convex function, $F(t)$. If an algorithm is available for solving PP(t), i.e. for finding $F(t)$, for every given t, then the value t^* can be computed by either of the following methods:

2.1. METHOD A

Let t_0 be a lower bound for t^*. Solve PP(t_0) until a solution x^1 is obtained such that $h(x^1)/g(x^1)>t_0$. Reset $t_0\leftarrow t_1=h(x^1/g(x^1)$ and repeat. Terminate when for some k we have $F(t_k)\leqslant \varepsilon\alpha t_k$.

Theoretically, this generated sequence $t_0, t_1, \dots$ may be infinite. To preclude such event it suffices, after every finitely many iterations, to reset t only when a solution x is obtained such that $h(x)-tg(x)>\varepsilon\alpha t$, i.e. such that $h(x)/g(x)> \left(1+\varepsilon\frac{\alpha}{g(x)}\right)t$.

2.2. METHOD B

Suppose an interval $[t_0,\Gamma]$ can easily be determined that contains t^* (i.e. such that $F(t_0)>0>F(\Gamma)$). Then a popular method for computing t^* is to start from this interval and iteratively reduce it by a Bolzano type procedure. Specifically, assuming $X\neq\emptyset$ we can proceed as follows:

0. Set $t_1=0,\ t_2=\Gamma$.
1. If $t_2-t_1\leqslant\varepsilon t_2$ terminate. Otherwise, solve PP(t) for $t=(t_1+t_2)/2$.
2. If $F(t)>0$ and $h(x^t)-tg(x^t)>0$ for some $x^t\in X$ then reset $t_1=h(x^t)/g(x^t)$, and go back to Step 1.
3. If $F(t)<0$ then reset $t_2=t$ and go back to Step 1.
4. If $|F(t)|\leqslant\varepsilon\alpha t$ then terminate.

PROPOSITION 3. *If the above scheme terminates at a step* 1 *then* t_1 *is an* ε*-optimal value, with* x^{t_1} *as* ε*-optimal solution. If it terminates at a step* 4 *then* t *is an* ε*-optimal value and* $x^t\in X$ *such that* $|h(x^t)-tg(x^t)|\leqslant\varepsilon\alpha t$ *is an* 2ε*-optimal solution.*

Proof. By induction on the iteration counter it can easily be shown that $F(t_1)\geqslant 0>F(t_2)$ at every iteration. Since by Proposition 2 the optimal value t^* of (FP) is a zero of the decreasing convex function $F(t)$, one must have $t_1\leqslant t^*\leqslant t_2$, hence, if the scheme terminates at step 1, t^* is an ε-optimal solution of (FP). If the scheme terminates at a step 4, i.e. $|F(t)|\leqslant\varepsilon\alpha t$, then the conclusion follows from Proposition 2. □

To make the above conceptual scheme implementable, we must develop an iterative algorithm for solving PP(t) such that:

(i) Each iteration k of the algorithm produces an interval $[L_k(t), U_k(t)]$ enclosing $F(t)$ and satisfying $U_k(t) - L_k(t) \to 0$ as $k \to +\infty$.

(ii) At each iteration an $x^{k,t} \in X$ is available for which $h(x^{k,t}) - tg(x^{k,t}) = L_k(t)$. When incorporated into the above scheme for solving (FP) such an algorithm will necessarily terminate after finitely many iterations by one of the following situations which correspond to steps 2, 3, 4, respectively:

(a) $L_k(t) > 0 : S$ then $F(t) > 0$.
(b) $U_k(t) < 0$: then $F(t) < 0$.
(c) $U_k(t) - L_k(t) \leqslant \varepsilon\alpha t$: then $|F(t)| \leqslant \varepsilon\alpha t$.

(Indeed, if neither a) nor b) occurs then $U_k(t) \geqslant 0 \geqslant L_k(t)$ and since $L_k(t) \leqslant F(t) \leqslant U_k(t)$ and $0 \in [L_k(t), U_k(t)]$ it follows that $|F(t)| \leqslant \varepsilon\alpha t$.)

However, especially when X is nonconvex, an algorithm for PP(t) with the mentioned properties is not always easy to construct. More often an algorithm only produces at each iteration k an upper bound $U_k(t)$ for $F(t)$ such that $U_k(t) - F(t) \to 0$ as $k \to \infty$. In such cases the scheme for solving (FP) should be modified as follows.

0. Set $t_1 = 0$, $t_2 = \Gamma$.
1. If $t_2 - t_1 \leqslant \varepsilon t_2$ terminate.
   Otherwise, solve PP(t) for $t = (t_1 + t_2)/2$.
2. If $h(x^t) - tg(x^t) > 0$ for some $x^t \in X$ then reset $t_1 = h(x^t)/g(x^t)$, and go back to Step 1.
3. If $U_k(t) < \varepsilon\alpha t$ then reset $t_2 = t$ and go back to Step 1.
   (In the latter case $g(x) - th(x) \leqslant U_k(t) \leqslant \varepsilon\alpha t \quad \forall x \in X$, hence $t^* \leqslant (1+\varepsilon)t$).

Remark 1. Method B is practical only when the sign of $F(t)$ for every t can be determined at relatively low cost. For instance, in a problem studied by Gotoh and Konno [7], where both functions $h(x), g(x)$ are quadratic convex, the computational cost of determining the sign of $F(t)$ very much depends on the nonconvexity rank of $h(x) - tg(x)$, which in turn depends on the value of t. The efficiency of the method is then very sensitive to the choice of the sequence $\{t_k\}$ approaching t^*. By exploiting the quadratic structure of $h(x)$ and $g(x)$ of [7] the authors have been able to devise a procedure for determining an efficient sequence $\{t_k\}$.

Remark 2. The two above approaches for solving the fractional programming problem (FP): via the polynomial program (PFP) in $\mathbb{R}^{n+1}$ (the direct approach) and via a connected sequence of polynomial programs PP(t) in $\mathbb{R}^n$ (the parametric approach) are closely related to each other. In fact, an alternative polynomial program equivalent to (FP) is the following

$$\max\{t \,|\, tg(x) \leqslant h(x), x \in X\}. \tag{2}$$

Since the constraint of this problem can be expressed as

$$\max\{h(x)-tg(x)\,|\,x\in X\}\geqslant 0$$

we see that the second approach merely amounts to solving the polynomial program (2) equivalent to (FP). Upon the change of variables $t=yh(x)$ (PFP) becomes identical to (2).

3. Monotonic Reformulation

It was shown in [25] that any polynomial program can be converted into a monotonic optimization problem. In this section we follow this method to reformulate (PFP) or (PP(t) as a problem of maximizing a monotonic function under monotonic constraints. Let us first introduce some notations and definitions.

For any two vectors $x', x\in\mathbb{R}^n$ write $x'\geqslant x$ and say that x' *dominates* x if $x'_i\geqslant x_i$ $\forall i=1,\dots,n$. Write $x'>x$ and say that x' *strictly dominates* x if $x'_i>x_i$ $\forall i=1,\dots,n$. Let $\mathbb{R}^n_+=\{x\in\mathbb{R}^n\,|\,x\geqslant 0\}$ and $\mathbb{R}^n_{++}=\{x\in\mathbb{R}^n\,|\,x>0\}$. For $x\in\mathbb{R}^n_+$ let $I(x)=\{i\,|\,x_i=0\}$ and denote

$$K_x=\{x'\in\mathbb{R}^n_+|\ x'_i>x_i\ \forall i\notin I(x)\},\quad \mathrm{cl}K_x=\{x'\in\mathbb{R}^n_+\,|\,x'\geqslant x\}. \tag{3}$$

If $a\leqslant b$ we define the box $[a,b]$ to be the set of all x such that $a\leqslant x\leqslant b$. Also write $(a,b]:=\{x\,|\,a<x\leqslant b\},[a,b):=\{x\,|\,a\leqslant x<b\}$. As usual e is the vector of all ones and e^i the ith unit vector of the space under consideration, i.e. the vector whose ith component equals 1, and all other components equal zero.

A function f: $\mathbb{R}^n\to\mathbb{R}$ is said to be *increasing* on a a box $[a,b]\subset\mathbb{R}^n$ if $f(x)\leqslant f(x')$ whenever $a\leqslant x\leqslant x'\leqslant b$. It is said to be *d.m.* (*d*ifference *m*onotonic) on $[a,b]\subset\mathbb{R}^n$ if it can be represented as the difference of two increasing functions on $[a,b]$.

PROPOSITION 4. *Any polynomial $Q(x_1,\dots,x_n)$ (in particular any affine or quadratic function) is a d.m. function on any box $[a,b]\subset\mathbb{R}^n_+$.*

Proof. By grouping separately the terms with positive coefficients and those with negative coefficients, one can write $Q(x)=Q^+(x)-Q^-(x)$, where each Q^+,Q^- is a polynomial with nonnegative coefficients, hence an increasing function, on $[a,b]\subset\mathbb{R}^n_+$. In the case of a linear function $Q(x)=\langle c,x\rangle$, we can write $c=c^+-c^-$ with $c_i^+=\max\{0,c_i\},c_i^-=\max\{0,-c_i\}$, so that $Q^+(x)=\langle c^+,x\rangle,Q^-(x)=\langle c^-,x\rangle$. □

Consider now the general polynomial program

$$\max\{Q(x)\,|\,c_i(x)\leqslant 0,\ i=1,\dots,m,\ 0\leqslant x\leqslant r\} \tag{PP}$$

where $Q(x), c_1(x), \ldots, c_m(x)$ are polynomials in $\mathbb{R}^n$. Let $Q(x) = Q^+(x) - Q^-(x)$, $c_i(x) = c_i^+(x) - c_i^-(x)$ be the d.m. representations of $Q(x)$, $c_i(x)$ as described in the proof of Proposition 4. Then the problem can be rewritten as

$$\max \quad Q^+(x) + w_1 \tag{4}$$

$$\text{s.t.} \quad w_1 + Q^-(x) \leqslant 0 \tag{5}$$

$$c_i(x) \leqslant 0, \quad i = 1, \ldots, m, \tag{6}$$

$$0 \leqslant x \leqslant r, \quad -Q^-(r) \leqslant w_1 \leqslant -Q^-(0) \tag{7}$$

To see the equivalence between (PP) and the above problem, just observe that any optimal solution (x, w_1) of the latter must satisfy (5) as equality, hence must solve (PP). Note that, since $Q^-(x)$ is increasing, $-Q^-(r) \leqslant -Q^-(0) \leqslant 0$.

It remains to convert the inequalities (6) into monotonic ones. For that we replace the system (6) by the equivalent single inequality

$$\max_{i=1,\ldots,m} [c_i^+(x) - c_i^-(x)] \leqslant 0,$$

then using the formula $\max_{i \in I}(u_i - v_i) = \max \sum_i u_i - \min(\sum_{j \neq i} u_j + v_i)$, we rewrite this inequality as

$$u(x) - v(x) \leqslant 0 \tag{8}$$

where $u(x), v(x)$ are increasing functions defined by

$$u(x) = \sum_{i=1}^{m} c_i^+(x), \quad v(x) = \min_{i=1,\ldots,m} v_i(x),$$

$$v_i(x) = \sum_{k \neq i} c_k^+(x) + c_i^-(x) \ (i = 1, \ldots, m). \tag{9}$$

In turn the d.m. inequality $u(x) - v(x) \leqslant 0$ is equivalent to the monotonic system:

$$u(x) + w_2 \leqslant 0, \quad v(x) + w_2 \geqslant 0, \quad -u(r) \leqslant w_2 \leqslant -u(0).$$

Now, setting $z = (x, z_{n+1}, z_{n+2}) \in \mathbb{R}^n \times \mathbb{R}^2$, with $z_i = x_i \ (i = 1, \ldots, n), z_{n+1} = w_1$, $z_{n+2} = w_2$ let us define the functions

$$f(z) = Q^+(x) + z_{n+1} \tag{10}$$

$$\varphi(z) = \max\{z_{n+1} + Q^-(x), \ u(x) + z_{n+2}\} \tag{11}$$

$$\psi(z) = v(x) + z_{n+2} \tag{12}$$

These functions are increasing in the interval $[a, b] \subset \mathbb{R}^{n+2}$ with

$$a = (0, \ldots, -Q^-(r), -u(r)), \quad b = (r, -Q^-(0), -u(0)). \tag{13}$$

Furthermore, $f(z)$ is a polynomial with nonnegative coefficients, $\varphi(z)$ is the pointwise maximum of two such polynomials while $\psi(z)$ is the pointwise minimum of at most m such polynomials. We have thus proved the following

PROPOSITION 5. *Any polynomial program* (PP) *can be rewritten as a monotonic optimization problem of the form*

$$\max\{f(z)\,|\,\varphi(z)\leqslant 0\leqslant\psi(z),\ z=(x,z_{n+1},z_{n+2})\in[a,b]\} \tag{MP}$$

where $f(z),\varphi(z),\psi(z)$ are increasing. More precisely, if $\bar{x}$ solves (PP) *then $\bar{z}=(\bar{x},\bar{z}_{n+1},\bar{z}_{n+1})$ with $\bar{z}_{n+1}=-Q^-(\bar{x})$, $\bar{z}_{n+2}=-u(\bar{x})$, solves* (MP). *Conversely, if $\bar{z}=(\bar{x},\bar{z}_{n+1},\bar{z}_{n+2})$ solves* (MP), *then $\bar{x}$ solves* (PP).

As immediate consequence of this Proposition it follows that

(i) If $\varphi(b)\leqslant 0\leqslant\psi(b)$ then b is an optimal solution of (PP);
(ii) If $\varphi(a)>0$ or $\psi(b)<0$ then (PP) is infeasible.

Therefore it only remains to consider the case when

$$\varphi(a)\leqslant 0<\varphi(b);\quad \psi(b)\geqslant 0. \tag{14}$$

Remark 3. Since $\varphi(b)=\max\{-Q^-(0)+Q^-(r),\ u(r)-u(0)\}$, $\psi(b)=v(r)-u(0)$, case (i) means that $Q^-(r)=Q^-(0), u(r)=u(0), v(r)\geqslant u(0)=u(r)$, i.e. $Q(x)$ is increasing on $[0,r]$ while r is feasible $(u(r)\leqslant v(r))$, which obviously implies that r is an optimal solution. On the other hand, since $\varphi(a)=\max\{-Q^-(r)+Q^-(0),\ u(0)-u(r)\}, \psi(b)=v(r)-u(0)$, case (ii) means that $u(x)-v(x)\geqslant u(0)-v(r)>0\ \forall x\in[0,r]$, i.e. no point $x\in[0,r]$ is feasible. The assumption (14) simply means that these trivial cases are excluded.

Also note that if $Q(x)$ is increasing ($Q^-(x)\equiv 0$) then the variable $z_{n+1}=w_1$ is not needed. Likewise, if $u(x)-v(x)$ is monotonic ($v(x)\equiv 0$ or $u(x)\equiv 0$) then the variable z_{n+2} is not needed.

4. Polyblock Approximation Method

In this and the next sections we apply the polyblock approximation method as developed in [25] (and revised in [26]) to solve the problem (MP), equivalent to (PP). For notational convenience we set $\ell=n+2$.

It should be noted that this method, in some of its basic components, was put forward first in some works of Kuno-Yajima-Konno and especially in [12] for solving certain classes of problems related to multiplicative programming. Subsequently, though independently, it was developed in [15] for more general classes of problems, then streamlined and extended to the whole range of monotonic optimization in [25] (see also [8, 27] for the application to some special problems).

4.1. NORMAL SETS AND POLYBLOCKS

Define the sets

$$G=\{z\in\mathbb{R}^{\ell}\,|\,\varphi(z)\leqslant 0,\ a\leqslant z\leqslant b\},$$
$$H=\{z\in\mathbb{R}^{\ell}\,|\,\psi(z)\geqslant 0\}.$$

These sets are characterized by the following important property

$$[a\leqslant z'\leqslant z\leqslant b,\ z\in G]\Rightarrow z'\in G, \tag{15}$$
$$[a\leqslant z'\leqslant z\leqslant b,\ z\notin H]\Rightarrow z'\notin H. \tag{16}$$

A set G with property (15) is called a *normal set* in $[a,b]$, while a set H with property (16) is called a *reverse normal set* in $[a,b]$. Thus the problem (MP) is to find

$$\max\{f(z)\,|\,z\in G\cap H\}, \tag{17}$$

where G is a compact normal set, H a compact reverse normal set in $[a,b]$. In view of the assumption (14) we have

$$a\in\operatorname{int}G\setminus H,\quad b\in\operatorname{int}H\setminus G. \tag{18}$$

As a result, for any point $z=(x,x_{n+1},w)\in[a,b]$ there is a unique point, $\pi_G(z)$ on the halfline from a through z defined by

$$\pi_G(z)=a+\lambda_G(z-a),\quad \lambda_G=\max\{\lambda>0\,|\,a+\lambda(z-a)\in G\} \tag{19}$$

This point $\pi_G(z)$ can be determined without difficulty, provided the value of $\varphi(z)$ at every given z is given or can be computed easily. Indeed, since $\varphi(a)<0<\varphi(z)$, and the function $\lambda\mapsto\varphi(a+\lambda(z-a))$ is increasing, using a Bolzano binary search on the segment $[0,1]$ one can locate the $\lambda\in(0,1)$ satisfying $\varphi(a+\lambda(z-a))=0$. Specifically, from (11) we have $\varphi(z)=\max\{\varphi_1(z),\varphi_2(z)\}$, with

$$\varphi_1(z)=z_{n+1}+Q^-(x)-Q^-(r), \tag{20}$$
$$\varphi_2(z)=u(x)+z_{n+2}-u(r). \tag{21}$$

PROPOSITION 6. *Let λ_1, λ_2, be the roots (in the interval $(0,1)$) of the polynomial equations of one variable*

$$\varphi_1(a+\lambda(z-a))=0,\quad \varphi_2(a+\lambda(z-a))=0,$$

respectively. Then

$$\lambda_G=\min\{\lambda_1,\lambda_2\}.$$

Proof. Immediate. In particular if $Q^-(x)$ and $c_1^+(x),\ldots,c_m^+(x)$ are quadratic then λ_1,λ_2 are obtained by solving quadratic equations of one variable. □

4.2. POLYBLOCKS AND APPROXIMATION OF NORMAL SETS

A set $P \subset \mathbb{R}^\ell$ is called a *polyblock* in $[a,b]$ if it is the union of a finite number of boxes $[a,z]$, $z \in T \subset [a,b]$ $(|T| < +\infty)$. The set T is called the *vertex set* of the polyblock. We write $T = \text{vert}\,P$ and also say that the polyblock P is generated by T. A vertex $z \in T$ is said to be *proper* if it is not dominated by any other $z' \in T$, i.e. if $z \notin [a,z']$ $\forall z' \in T \setminus \{z\}$. A vertex which is not proper is said to be *improper*. Of course a polyblock is fully determined by its proper vertices. We denote the set of proper vertices of P by pvert(P). The proof of the next Proposition 7 can be found in [25] (or [24]) and that of Proposition 8 in [26].

PROPOSITION 7. *Any polyblock is closed and normal. The intersection of finitely many polyblocks is a polyblock. An increasing function $f(z)$ achieves its maximum over a polyblock at a proper vertex.*

PROPOSITION 8. *Let G be a compact normal set in $[a,b] \subset \mathbb{R}^\ell$, let P be a polyblock in $[a,b]$ containing G, with* pvert$\,P = V$. *Let $z^* \in V \setminus G$, $\bar{z} = \pi_G(z^*)$ and $V_* = \{z \in V \,|\, z \geqslant \bar{z}\}$. For every $z \in V$ define*

$$z^i = z - (z_i - \bar{z}_i)\mathrm{e}^i, \quad i = 1, \ldots, \ell. \tag{22}$$

Then the vertex set of the polyblock $P \setminus (\bar{z}, b]$ is

$$V' = (V \setminus V_*) \cup \{z^i \,|\, z \in V_*,\ z_i > \bar{z}_i,\ i = 1, \ldots, \ell\},$$

while its proper vertex is obtained from V' by removing every z^i for which there exists $z' \in V_*$ such that $z_i > z'_i$ but $z_j \not> z'_j$ $\forall j \neq i$ (i.e. i is the unique index satisfying $z_i > z'_i$).

4.3. POLYBLOCK ALGORITHM

Based on the above properties an outer approximation procedure can be developed for solving the problem (MP), or equivalently, the problem (PP).

Let γ be the optimal value of (MP). An ε-optimal solution of (MP) is a feasible solution $\bar{z}$ of (MP) such that $\gamma \leqslant (1+\varepsilon) f(\bar{z})$. To find an ε-optimal solution of (MP) by the polyblock outer approximation method we construct inductively a sequence of polyblocks $P_1 \supset P_2 \supset \cdots \supset G \cap H$ together with a sequence of vectors $\bar{z}^k \in G \cap H$ $(k = 1, 2, \ldots)$, such that $f(\bar{z}^1) \leqslant f(\bar{z}^2) \leqslant \cdots$ and

$$\max\{f(z) \,|\, z \in P_k\} - f(\bar{z}^k) \searrow 0 \quad (k \to \infty).$$

The procedure will be stopped when

$$\max\{f(z) \,|\, z \in P_k\} \leqslant (1+\varepsilon) f(\bar{z}^k), \tag{23}$$

because this will imply that $\gamma \leqslant (1+\varepsilon) f(\bar{z}^k)$, hence $\bar{z}^k$ is ε-optimal.

As initial polyblock we take $P_1=[a,b]$ with proper vertex set $T_1=\{b\}$.

At iteration k we have a polyblock P_k with proper vertex set T_k and a vector $\bar{z}^k$ which is the best feasible solution known so far (it may happen, however, that no feasible solution has been known yet, in this case we set $f(\bar{z}^k)=-\infty$). Then we check the condition (23). Observe first that every vertex $z\in T_k\setminus H$ can be discarded, since for these z we have $[a,z]\cap H=\emptyset$, and hence, $[a,z]\cap(G\cap H)=\emptyset$. Also if $f(\bar{z}^k)=-\infty$ then condition (23) is not fulfilled. If, on the other hand, $\bar{z}^k$ exists so that $f(\bar{z}^k)>-\infty$, then we have to check whether $f(z)\leqslant(1+\varepsilon)f(\bar{z}^k)$ for every $z\in T_k$, or rather, for every $z\in T_k\cap H$.

Let $\tilde{T}_k$ be the set of all $z\in T_k\cap H$ such that $f(z)>(1+\varepsilon)f(\bar{z}^k)$ (i.e. the set that remains from T_k after removing all $z\in T_k\setminus H$ and all $z\in T_k$ such that $f(z)\leqslant(1+\varepsilon)f(\bar{z}^k)$). Then the polyblock $\tilde{P}_k$ of vertex set $\tilde{T}_k$ contains all feasible points z still of interest. If $\tilde{T}_k=\emptyset$, this means that $f(z)\leqslant(1+\varepsilon)f(\bar{z}^k)$ for every feasible solution z, hence condition (23) holds, so the procedure can be stopped: if a $\bar{z}^k$ exists it is an ε-optimal solution, otherwise the problem is infeasible.

If $\tilde{T}_k\neq\emptyset$, let

$$z^k\in\operatorname{argmax}\{f(z)\,|\,z\in\tilde{T}_k\}=\operatorname{argmax}\{f(z)\,|\,z\in\tilde{P}_k\}.$$

(see Proposition 7) and let $\tilde{z}^k=\pi_G(z^k)$ be the point computed according to (19) and Proposition 6. At this point we note the particular role of the variable $z_{n+1}=w_1$ which, for any given choice of x, should be maximized while satisfying the inequality $z_{n+1}+P^-(x)\leqslant 0$ (see (5)). Therefore, if $\tilde{z}^k$ is such that $\tilde{z}^k_{n+1}+P^-(\tilde{x}^k)<0$ then by increasing $\tilde{z}^k_{n+1}$ to $\hat{z}^k_{n+1}=-P^-(\tilde{x}^k)$ we improve the objective function value without violating any constraint. Thus, in any case we can replace $\tilde{z}^k$ by $\hat{z}^k=(\hat{x}^k,\hat{z}^k_{n+1},\hat{z}^k_{n+2})$ where $\hat{x}^k=\tilde{x}^k,\hat{z}^k_{n+1}=-P^-(\tilde{x}^k)$, and $\hat{z}^k_{n+2}=\tilde{z}^k_{n+2}$. In the sequel it will be convenient to denote this vector $\hat{z}^k$ by $\hat{\pi}(z^k)$. Since $\hat{z}^k\in G$, if $\hat{z}^k\in H$ then $\hat{z}^k$ is feasible and can be used to update CBV. If, in addition, $(1+\varepsilon)f(\hat{z}^k)\geqslant f(z^k)$ then, since $f(z^k)\geqslant\gamma$, we have $(1+\varepsilon)f(\hat{z}^k)\geqslant\gamma$, so $\hat{z}^k$ is an ε-optimal solution of the problem and the procedure terminates (this occurs in particular if $\hat{z}^k=z^k$). On the other hand, if $\hat{z}^k\neq z^k$, then $z^k\notin G$ and using Proposition 8 we can compute the proper vertex set T_{k+1} of the polyblock $P_{k+1}=\tilde{P}_k\setminus(\hat{z}^k,b]$. Specifically, letting $T_{k*}=\{z\in\tilde{T}_k\,|\,z>\hat{z}^k\}$, $z^i=z-(z_i-\hat{z}_i)e^i$ $(i=1,\ldots,\ell)$ define the set

$$V_{k+1}=(\tilde{T}_k\setminus T_{k*})\cup\{z^i\,|\,z\in T_{k*},z_i>\hat{z}_i\}. \tag{24}$$

and let T_{k+1} be the set that remains from V_{k+1} after removing every z^i for which there exists $z'\in T_{k*}$ satisfying $\{i\}=\{j\,|\,z_j>z'_j\}$.

We can thus state the following *Polyblock Algorithm* for problem (MP).

ALGORITHM 1 (for (MP) with tolerance $\varepsilon>0$).

4.3.1. *Initialization*

Set $T_1=\{b\}$. Let $\bar{z}^1$ be the best feasible solution available, $CBV\ =f(\bar{z}^1)$. If no feasible solution is available, let $CBV=-\infty$. Set $k=1$.

Step 1. From T_k remove all $z\in T_k$ such that $z\notin H$ and all z such that $f(z)\leqslant(1+\varepsilon)CBV$. Let $\tilde{T}_k$ be the set of remaining elements of T_k.

Step 2. If $\tilde{T}_k=\emptyset$, terminate: if $CBV=-\infty$, the problem is infeasible; if $CBV>-\infty$, the current best feasible solution $\bar{z}^k$ is an *ε-optimal solution* of (MP).

Step 3. If $\tilde{T}_k\neq\emptyset$, select $z^k\in$ argmax$\{f(z)|z\in\tilde{T}_k\}$. Compute $\tilde{z}^k=\pi_G(z^k)=\lambda_k z^k$, and $\hat{z}^k=\hat{\pi}(z^k)$.

(3a) If $\psi(\hat{z}^k)\geqslant 0$ (so that $\hat{z}^k$) is feasible and $f(z^k\leqslant(1+\varepsilon)f(\hat{z}^k)$ then terminate: $\hat{z}^k$ is an ε-optimal solution.

(3b) If $\psi(\hat{z}^k)\geqslant 0$ and $f(\hat{z}^k)>CBV$ then reset $CBS=\hat{z}^k, CBV=f(\hat{z}^k)$.

Step 4. Compute V_{k+1} according to (24) and let T_{k+1} be the set that remains from V_{k+1} after removing all improper elements as indicated in Proposition 4.

Step 5. Set $k\leftarrow k+1$ and return to Step 1.

THEOREM 1. *The above Algorithm is finite provided for some* $\delta>0$

$$\min_{i=1,\ldots,\ell}(z_i-a_i)>\delta\quad\forall z\in H. \tag{25}$$

Proof. This follows from a general convergence theorem established in [25]. Although Algorithm 1 involves a modification of the standard Polyblock Outer Approximation Algorithm in [25] to take account of the particular role of y (replacement of $\tilde{y}^k$ by $\hat{y}^k$ in Step 3), the convergence proof given in [25] is still valid with some minor and obvious modifications. □

Condition (25) can easily be made to hold by slightly moving the origin if necessary.

5. Discussion

1. To avoid storage problems in connection 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 $|T_k|>L$, where L is a user supplied fixed number. Specifically, Step 5 of Algorithm 1 should be modified as follows. Let $\tilde{z}$ be the point where we would like to restart the Algorithm (usually, $\tilde{z}=z^k$ or current best solution).

Step 5. If $|T_{k+1}|\leqslant L$ then set $k\leftarrow k+1$ and return to Step 1. Otherwise go to Step 6.

Step 6. Redefine $z^k=\hat{\pi}_G(\tilde{z})$, $T_{k+1}=\{b-(b_i-z_i^k)e^i,\ i=1,\ldots,\ell\}$, (i.e. $P_{k+1}=[a,b]\setminus(z^k,b)$, then set $k\leftarrow k+1$ and return to Step 1.

with this modification, an occurence of Step 6 means a restart, i.e. the beginning of a new cycle of iterations.

2. The performance of the algorithm is sensitive to the size of the box $[a,b]$, or rather the size of the set $[a,b]\setminus G\cap H$ because the search procedure is carried out in this region. Moreover, it has been observed that the performance depends also on how fast the quantity $(1-\lambda_k)\|z^k\|$ tends to zero, where $z^k\in H$, and $\min_i z_i^k\geqslant\delta$ (see (25)). Therefore, before starting the algorithm, it may be useful to try to reduce the box $[a,b]$ if possible and to select a suitable value of δ. This can be achieved by the following box reduction operation. Denote the box that results from the reduction of the box $[a,b]$ by Red $[a,b]$.

Box Reduction Procedure. Compute

$$\sigma_i=\sup\{\sigma>0\,|\,a+\sigma e^i\in G,\ a_i+\sigma\leqslant b_i\},\quad b'=a+\sum_{i=1}^{\ell}\sigma_i e^i \tag{26}$$

$$\theta_i=\sup\{\theta>0\,|\,b'-\theta e^i\in H,\ a_i\leqslant b_i'-\theta\},\quad a'=b'-\sum_{i=1}^{\ell}\theta_i e^i. \tag{27}$$

Then Red $[a,b]=[a',b']$.

If $\varphi(a')>0$ or $\psi(b')<0$, then (MP) is infeasible.

If $\varphi(b')\leqslant 0\leqslant\psi(b')$, then b' solves (MP).

If $\varphi(a')\leqslant 0\leqslant\psi(b')$, we can either further reduce $[a',b']$ (if $a'>a$) or stop the reduction process and reset $b\leftarrow b'$, $a\leftarrow a'-\delta_0(b'-a')$ where $\delta_0>0$ is chosen so that $\varphi(a'-\delta_0(b'-a')<0$.

3. In some problems it may happen that the initial rectangle $[a,b]$ is very thin, so that the selection rule for z^k in Step 3 favours certain search directions which may not be promising. One way to avoid such unpleasant situations is to rescale the variables in order to obtain a more balanced initial box.

Also it may happen that the objective function does not depend upon certain variables $x_i, i\notin J$. In that case, with T_k defined as in Algorithm 1, the value $\max\{f(z)\,|\,z\in\tilde{T}_k\}$ in Step 3 will remain constant through a large number of iterations, causing a possible stall of the algorithm. To avoid such kind of difficulty, one should consider the problem in $\mathbb{R}^{\ell_1}$ (space of $(z_1,\dots,z_{\ell_1})$), rather than in R^{ℓ}. More precisely, the feasible set over which one should maximize the objective function is the set of all $(z_1,\dots,z_{\ell_1})$ for which there exists $z_{\ell_1+1},\dots,z_\ell$ such that $\varphi(z_1,\dots,z_\ell)\leqslant 0\leqslant\psi(z_1,\dots,z_\ell)$, $z=(z_1,\dots,z_\ell)\in[a,b]$.

4. Polynomial programs involving only polynomials with nonnegative coefficients are in general much easier to handle than those with polynomials having both positive and negative coefficients. Quite often the convergence of Algorithm 1 for problems of the latter category is very slow due to the peculiar role of the variable z_{n+1} in the expression $f(z)=P^+(x)+z_{n+1}$. In fact every fixed value of z_{n+1} gives rise to a monotonic problem whose optimal value is a highly non-linear non-monotone function of z_{n+1}. To overcome difficulties which may arise from this, in the next section we will propose a branch and bound algorithm

(Algorithm 2) using an efficient bounding method based on a truncated version of Algorithm 1.

5. An equality constraint can in principle be handled by replacing it with two opposite inequality constraints, as usual. However it is often more efficient to use the equality constraints to eliminate one or more variables, so as to reduce the problem to one with less variables and only inequality constraints. Also note that sometimes an equality constraint can actually be replaced by one inequality constraint without changing the set of optimal solutions.

6. A common feature of many deterministic global optimization algorithms is that the optimal solution may be found rather fast, while most of the computation time (sometimes more than 80%) is spent on verifying (confirming) *global* optimality. This is one aspect of the inherent difficulty of global optimization. In particular, solving PP(t) to improve a current best value t of (FP) when t is not yet optimal usually takes much less time than solving it to check optimality when t is already optimal or nearly optimal. Fortunately, at each stage of the algorithm an upper bound of the optimal value is available as well as a current best value, so one knows how far the current best value is from the optimum. This is a very useful information rarely provided by local or nondeterministic methods.

6. Branch and Bound Method

The polyblock approximation algorithm has proved to work well on a number of difficult nonconvex problems [8, 15, 27]. However, since it is essentially an outer approximation procedure, it is expected to encounter storage problems and other difficulties inherent to procedures of this kind when solving large scale problems. To alleviate these difficulties, one way is to use the polyblock approximation not as an independent outer approximation algorithm, but, instead, as a bounding technique in a well suited branch and bound scheme. In fact, given any box $M := [p, q] \subset [0, r] \subset \mathbb{R}^n_+$, an upper bound for

$$\gamma(M) := \max\{P(x) \mid c_i(x) \leqslant 0 \ (i = 1, \ldots, m), \ p \leqslant x \leqslant q\} \qquad (\text{PP}[p, q])$$

can be obtained by means of one or a few iterations of Algorithm 1 applied to the monotonic problem

$$\max\{f(z) \mid z \in G \cap H, \ \tilde{p} \leqslant z \leqslant \tilde{q}\} \qquad (\text{MP}[\tilde{p}, \tilde{q}])$$

equivalent to PP($[p, q]$). In many cases, even tighter bounds can be derived by *exploiting the monotonic structure combined with the d.c. structure* of the subproblem (MP$[\tilde{p}, \tilde{q}]$). Incorporating this bounding technique into a rectangular partitioning scheme will produce a branch and bound method for solving (PP) which is often much more efficient than Algorithm 1.

Let us first describe the procedure for computing an *upper bound* for (PP$[p, q]$), i.e. a number $\beta(M) \geqslant \gamma(M) := \max\{P(x) \mid c_i(x) \leqslant 0 \ \ (i = 1, \ldots, m), p \leqslant x \leqslant q\}$, where $M := [p, q]$ is any subbox of $[0, r]$.

6.1. BOUNDING SUBROUTINE

Let us rewrite the polynomial program (PP($[p,q]$) in the form:

$$\max\{P^+(x)-P^-(x)\,|\,u(x)-v(x)\leqslant 0,\ p\leqslant x\leqslant q\}$$

where $P(x)=P^+(x)-P^-(x)$ and $u(x),v(x)$ are increasing functions defined by (9). As shown in Section 3, this problem can be reformulated as

$$\max\{f(z)\,|\,\varphi(z)\leqslant 0\leqslant\psi(z),\ z=(x,z_{n+1},z_{n+2})\in[\tilde p,\tilde q]\} \qquad (\text{MP}[p,q])$$

where

$$f(z)=P^+(x)+z_{n+1}; \qquad (28)$$

$$\varphi(z)=\max\{z_{n+1}+P^-(x),\ u(x)+z_{n+2}\};\quad \psi(z)=v(x)+z_{n+2}; \qquad (29)$$

$$\tilde p_i=p_i,\quad i=1,\dots,n,\quad \tilde p_{n+1}=-P^-(q),\quad \tilde p_{n+2}=-u(q); \qquad (30)$$

$$\tilde q_i=q_i,\quad i=1,\dots,n,\quad \tilde q_{n+1}=-P^-(p),\quad \tilde q_{n+2}=-u(p). \qquad (31)$$

PROPOSITION 9. (1) *If* $\varphi(\tilde q)\leqslant 0\leqslant\psi(\tilde q)$, *then* q *is an optimal solution of* PP$[p,q]$, *i.e.* $\beta(M)=\gamma(M)=P(q)$;
(2) *If* $\varphi(\tilde p)>0$ *or* $\psi(\tilde q)<0$ *then* $\gamma(M)=-\infty$ (*no feasible point in* M).

Proof. This follows from the equivalence between PP$[p,q]$ and MP$[p,q]$ (cases (i) and (ii) considered at the end of Section 3). Note that $\varphi(\tilde q)\leqslant 0\Rightarrow -P^-(p)+P^-(q)\leqslant 0\Rightarrow P^-(p)=P^-(q)$, and since $\tilde q$ solves MP$[p,q]$ it follows that $\gamma(M)=f(\tilde q)=P^+(q)-P^-(p)=P(q)$. □

Let us now agree to call *bounding function* any function $F(p,q)$ defined for every box $[p,q]\subset[0,r]$ such that $F(p,q)\in\mathbb{R}\cup\{-\infty\}$, $F(p,q)\geqslant\gamma(M)$ and

$$F(p^\nu,q^\nu)-\gamma(M_\nu)\to 0\quad\text{as}\quad \nu\to+\infty. \qquad (32)$$

for any nested sequence $\{M_\nu=[p^\nu,q^\nu]\}$ satisfying $q^\nu-p^\nu\to 0$ $(\nu\to\infty)$.

PROPOSITION 10. *The function*

$$F(p,q)=\begin{cases} P^+(q)-P^-(p) & \textit{if } \varphi(\tilde p)\leqslant 0\leqslant\psi(\tilde q)\\ -\infty & \textit{otherwise}\end{cases}$$

is a bounding function.

Proof. That $F(p,q)\geqslant\gamma(M)$ is obvious by Proposition 9. Consider any infinite nested sequence $\{M_\nu=[p^\nu,q^\nu]\}$ such that $q^\nu-p^\nu\to 0$. Then $q^\nu_{n+1}-p^\nu_{n+1}=P^-(q^\nu)-P^-(p^\nu)\to 0$ and $q^\nu_{n+2}-p^\nu_{n+2}=u(q^\nu)-u(p^\nu)\to 0$ as $\nu\to\infty$. Hence $\tilde q^\nu-\tilde p^\nu\to 0$, so $\tilde p^\nu$ and $\tilde q^\nu$ tend to a common limit $\bar z\in\mathbb{R}^{n+2}$ as $\nu\to\infty$. Let $\bar x=(\bar z_1,\dots,\bar z_n)\in\mathbb{R}^n$. If for some ν we have $\varphi(\tilde p^\nu)>0$ (or $\psi(\tilde q^\nu)<0$, respectively) then for all $\mu\geqslant\nu$, since $[p^\mu, q^\mu] \subset [p^\nu, q^\nu]$, it follows that $\varphi(\tilde{p}^\mu) \geqslant \varphi(\tilde{p}^\nu) > 0$ (or $\psi(\tilde{q}^\mu) \leqslant \psi(\tilde{q}^\nu) < 0$, respectively), hence $F(p^\mu, q^\mu) = -\infty$, whereas by Proposition 9 $\gamma(M_\mu) = -\infty$, so that (32) holds. On the other hand, if $\varphi(\tilde{p}^\nu) \leqslant 0 \leqslant \psi(\tilde{q}^\nu)$ $\forall \nu$ then $\varphi(\bar{z}) \leqslant 0 \leqslant \psi(\bar{z})$, hence $\max\{f(z) \,|\, \varphi(z) \leqslant 0 \leqslant \psi(z), z \in [\tilde{p}^\nu, \tilde{q}^\nu]\} = \gamma(M_\nu) \to f(\bar{z}) = P(\bar{x})$, whereas $F(p^\nu, q^\nu) = P^+(q^\nu) - P^-(p^\nu) \to P(\bar{x})$. Therefore, (32) holds in any case. □

Clearly, if $\{F_i(p,q), i \in I\}$ is a finite family of bounding functions then their lower envelope $\tilde{F}(p,q) = \min\{F_i(p,q) \,|\, i \in I\}$ is also a bounding function. We can now state the following bounding subroutine.

Select a bounding function $F(p,q)$ which may be obtained via any simple bounding method available. For instance, when MP$[p,q]$ happens to be also a d.c. optimization problem then $F(p,q)$ may be computed by solving a convex relaxation of the latter. In any case one can take $F(p,q) = P^+(q) - P^-(p)$. Select also an integer L (the maximal number of iterations of Algorithm 1 we are willing to execute in the subroutine).

Bounding Subroutine (for PP$[p,q]$)

Step 0. Compute $\tilde{p}, \tilde{q}$ according to (30), (31). If $\varphi(\tilde{p}) > 0$ or $\psi(\tilde{q}) < 0$ then set $\beta(M) = -\infty$, $y(M) = \emptyset$ and terminate. If $\varphi(\tilde{q}) \leqslant 0 \leqslant \psi(\tilde{q})$ then set $\beta(M) = P(q)$, $y(M) = q$ and terminate $(P(q) = \gamma(M))$. Otherwise, go to Step 1.

Step 1. (Box reducing). Compute $\beta_i = \sup\{\beta > 0 \,|\, \varphi(\tilde{p} + \beta e^i) \leqslant 0,\ \tilde{p}_i + \beta \leqslant \tilde{q}_i\}$ for every $i = 1, \ldots, n+2$, and let $\tilde{q}' = \tilde{p} + \sum_{i=1}^{n+2} \beta_i e^i$. Then compute $\alpha_i = \sup\{\alpha > 0 \,|\, \psi(\tilde{q}' - \alpha e^i) \geqslant 0,\ \tilde{p}_i \leqslant \tilde{q}'_i - \alpha\}$ for every $i = 1, \ldots, n+2$, and let $\tilde{p}' = \tilde{q}' - \sum_{i=1}^{n+2} \alpha_i e^i$.
Reset $p \leftarrow (\tilde{p}'_1, \ldots, \tilde{p}'_n), q \leftarrow (\tilde{q}'_1, \ldots, \tilde{q}'_n)$.

Step 2. Compute $\tilde{p}, \tilde{q}$ according to (30), (31). If $\varphi(\tilde{p}) > 0$ or $\psi(\tilde{q}) < 0$ then set $\beta(M) = -\infty$, $y(M) = \emptyset$ and terminate. If $\varphi(\tilde{q}) \leqslant 0 \leqslant \psi(\tilde{q})$ then set $\beta(M) = P(\tilde{q})$, $y(M) = q$ and terminate. Otherwise, let $z^1 = \tilde{q}$, $T_1 = W_1 = \{z^1\}$, $\pi^1 = \pi_G(z^1)$ (intersection of the surface $\varphi(z) = 0$ with the segment joining $\tilde{p}$ and $z^1 = \tilde{q}$; this intersection exists because $\varphi(\tilde{p}) \leqslant 0 < \varphi(\tilde{q})$). Set $\nu = 1$ (ν: iteration counter for subroutine) and go to Step 3.

Step 3. Compute

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

Let $T_{\nu+1} = (W_\nu \setminus \{z^\nu\}) \cup \{z^{\nu,1}, \ldots, z^{\nu,n}\}$.

Step 4. Let $W_{\nu+1}$ be the collection of all $z \in T_{\nu+1} \cap H$ such that there is no $z' \in T_{\nu+1}$ satisfying $z' \neq z, z' \geqslant z$ (i.e. z is a proper element of $T_{\nu+1}$). If $W_{\nu+1} = \emptyset$, set $\beta(M) = -\infty$ and terminate. Otherwise, compute

$$z^{\nu+1} \in \operatorname{argmax}\{\hat{F}(p,z) \,|\, z \in W_{\nu+1}\}.$$

where $\hat{F}(p,z) = \min\{F(p,z), f(z)\}$, and let $\pi^{\nu+1} = \pi_G(z^{\nu+1})$.

Step 5. If $\nu = L$, set $\beta(M) = \hat{F}(p, z^{\nu+1})$, $z(M) = z^{\nu+1}, y(M) = \pi^{\nu+1}$ and terminate. Otherwise, set $\nu \leftarrow \nu + 1$ and return to Step 3.

A subdivision procedure in $\mathbb{R}^n$ is said to be exhaustive (see [9]) if $q^\nu - p^\nu \to 0$ for any infinite filter of rectangles $[p^\nu, q^\nu] \subset \mathbb{R}^n$ generated by this subdivision procedure.

PROPOSITION 11. *The bound computed by the above subroutine is consistent with any exhaustive subdivision procedure.*

Proof. This is a straightforward consequence of the properties of the bounding function. □

6.2. BRANCH AND BOUND ALGORITHM

Given a box $M = [p, q] \subset \mathbb{R}^n$ and a couple (ω, j) where $\omega \in M$, $j \in \{1, \ldots, n\}$, the collection $\{M_1, M_2\}$ is called a partition of M via (ω, j) (see [6, 23]) if

$$M_1 = \{z \in M \,|\, p_j \leqslant z_j \leqslant \omega_j\}, \quad M_2 = \{z \in M \,|\, \omega_j \leqslant z_j \leqslant q_j\}.$$

When $\omega = \frac{p+q}{2}$, and $j \in \operatorname{argmax}_{i=1,\ldots,n}(q_i - p_i)$ the partition via (ω, j) is called a *bisection.* As is well known, the subdivision procedure by bisection is exhaustive (see e.g. [23]).

ALGORITHM 2 (for (PP) with tolerance $\varepsilon > 0$).

6.2.1. *Initialization*

Let $\bar{x}^0$ be the best feasible solution available of (PP), $CBV = P(\bar{x}^0)$. If no feasible solution is available, let $CBV = -\infty$. Set $\mathcal{P}_1 = \mathcal{S}_1 = \{M_0 := [0, r]\}$, $k = 1$.

Step 1. (*Bounding*) For each rectangle $M = [p, q] \in \mathcal{P}_k$ compute $\beta(M)$, $z(M)$ and $y(M)$ using Bounding Subroutine. If $y(M)$ is feasible to (PP), then let $\bar{x}^k \in \operatorname{argmax}\{P(\bar{x}^{(k-1)}), P(y(M))\ (M \in \mathcal{P}_k)\}$ and reset $CBV = P(\bar{x}^k)$.

Step 2. (*Pruning*) Delete every $M \in \mathcal{S}_k$ such that $\beta(M) \geqslant (1+\varepsilon)CBV$. Let $\mathcal{R}_k$ be the collection of remaining members of $\mathcal{S}_k$.

Step 3. (*Termination Criterion*) If $\mathcal{R}_k = \emptyset$ then terminate: $\bar{x}^k$ (current incumbent) is a global ε-optimal solution of Problem (PP).

Step 4. (*Branching*) Select $M_k \in \operatorname{argmax}\{\beta(M) | M \in \mathcal{R}_k\}$. Bisect M_k and let $\mathcal{P}_{k+1}$ be its partition.

Step 5. (*New Net*) Set $\mathcal{S}_{k+1} = (\mathcal{R}_k \setminus \{M_k\}) \cup \mathcal{P}_{k+1}$.

Set $k \leftarrow k+1$ and go back to Step 1.

PROPOSITION 12. *Algorithm* 2 *terminates after finitely many iterations, yielding an ε-optimal solution.*

Proof. Since the subdivision is exhaustive while the bounding method is consistent by Proposition 11, finiteness of the algorithm follows from the general theory of branch and bound methods for nonconvex optimization problems (see e.g. [23]). □

Remark 4. In many cases the convergence of Algorithm 2 can be sped up by using in Step 4 a more flexible subdivision than bisection. Specifically, we subdivide M_k via $(\omega(M_k), j_k)$ where

$$j_k \in \operatorname{argmax}\{z_i(M_k) - y_i(M_k)\}, \quad \omega(M_k) = \frac{z(M_k)+y(M_k)}{2}.$$

This subdivision is often referred to as an *adaptive subdivision*. Finiteness of the algorithm in that case can be proved as follows. Suppose the algorithm is infinite and let $\{M_\nu := [p^\nu, q^\nu], \nu = 1, 2, \dots\}$ be any filter of boxes generated by the algorithm, with $z^\nu = z(M_\nu)$, $y^\nu = y(M_\nu)$, $\omega^\nu = \omega(M_\nu)$. Without loss of generality we can assume that $j_\nu = 1\ \forall \nu$ and $q^\nu \to \bar{q}$, $p^\nu \to \bar{p}$, while $z^\nu \to \bar{z}$, $y^\nu \to \bar{y}$, $\omega^\nu \to \bar{\omega} = (\bar{z}+\bar{y})/2$. Then, by a known property of rectangular subdivision [23] (Lemma 5.4, page 160) $\bar{\omega}_1 \in \{\bar{p}_1, \bar{q}_1\}$, hence $\bar{z}_1 = \bar{y}_1$, and so $\bar{z} = \bar{y}$ because $1 = j_\nu$. Since $\varphi(y^\nu) \leqslant 0$, $\psi(z^\nu) \geqslant 0$ it follows that $\varphi(\bar{y}) \leqslant 0 \leqslant \psi(\bar{y})$. On the other hand, $\hat{F}(p^\nu, z^\nu) = \beta(M_\nu) \geqslant \gamma := \max\{P(x) | u(x) - v(x) \leqslant 0, 0 \leqslant x \leqslant r\}$, hence $F(p, \bar{z}) = P(\bar{y}) = \gamma$. This implies that for ν large enough one must have $\beta(M_\nu) \geqslant (1+\varepsilon)$ CBV, so $\mathcal{R}_\nu = \emptyset$ and the termination criterion in Step 3 would have been met at this iteration, contradicting the assumption. □

Remark 5. There is an obvious trade-off between the maximal iteration number L allowed for the subroutine and the convergence speed of the BB algorithm. It seems reasonable to take L sufficienttly large at the first stage of the BB (within a few iterations) and subsequently take $L \leqslant 5$. For many problems even the standard bisection, along with $L = 1$ for bound computation perform quite satisfactorily.

7. Illustrative Examples

To illustrate how the above presented method works we solved a number of simple, but nontrivial, numerical examples. The experiments confirmed that Algorithms 2 usually outperforms Algorithm 1 on polynomial programs even with a small number of variables. Therefore, for fractional programs (FP), we always used Algorithm 2 for solving (PFP) in the direct approach, or the subproblems PP(t) in the parametric approach. The algorithms corresponding to the two approaches were coded in C^{++} and run on a PC Pentium IV 2.53 GHz.

Examples 1 and 2 below are test problems for univariate polynomial programming taken from [3]. They have been solved very fast, as well as the other test problems in Chapter 4 of [3] which are not reported here. Examples 3 and 4 are multivariate polynomial fractional programs. Example 5 features the maximization of a sum of many ratios. Example 6 is a synomial fractional program with linear constraints, while Example 7 is a synomial fractional program with synomial constraints.

EXAMPLE 1 (Test problem 1, Chapter 4, [3]).

$$\min_{-2\leqslant x\leqslant 11}\left\{\frac{1}{6}x^6-\frac{52}{25}x^5+\frac{39}{80}x^4+\frac{71}{10}x^3-\frac{79}{20}x^2-x+\frac{1}{10}\right\}$$

By the change of variable $x+2=z$ this problem becomes

$$\min_{0\leqslant z\leqslant 13}\{0.166667z^6-4.08z^5+31.2875z^4-106.666667z^3+171.55z^2- \\ -114z+14.526667\}$$

Algorithm 2, with tolerance 0.01, found the optimal solution $z=11.999847$, i.e. $x=9.999847$, with objective function value: -29763.232229 at iteration 1106 and confirmed its optimality at iteration 1918 (note that each iteration in Algorithm 2 is executed very fast). Computation time: 0.015 sec, maximal number of nodes in each iteration: 12.

EXAMPLE 2 (Test problem 5, Chapter 4, [3]).

$$\min_{-5\leqslant x_1,x_2\leqslant 5}\left\{2x_1^2-1.05x_1^4+\frac{1}{6}x_1^6-x_1x_2+x_2^2\right\}$$

(the three-hump camel-back function). Observe that if (x_1, x_2) is a feasible solution then $(|x_1|, |x_2|)$ is also a feasible solution, no worse than (x_1, x_2). Therefore, the problem is the same as

$$\min_{0\leqslant x_1,x_2\leqslant 5}\{2x_1^2-1.05x_1^4+\frac{1}{6}x_1^6-x_1x_2+x_1^2\}$$

Algorithm 2, with tolerance 0.01, found the optimal solution: $x=(0,0)$ with objective function value 0, right at the beginning (iteration 0) and confirmed its optimality at iteration 3131. Computational time: 0.219 sec, maximal number of nodes in each iteration: 697.

EXAMPLE 3.

$$\max\left\{\frac{h(x)}{g(x)}\middle| c(x)\leqslant 0,\ x\geqslant 0\right\}. \tag{33}$$

where

$$\begin{aligned}
h(x) &= 3.525x_1^2+1.95x_1x_2+1.95x_1x_3+3.25x_1x_4+0.825x_2^2+6.5x_2x_3+ \\
&\quad +1.0834x_2x_4+0.625x_3^2+1.0834x_3x_4+1.4027x_4^2+19.7- \\
&\quad -11.7x_1-3.9x_2-3.9x_3-6.5x_4, \\
g(x) &= 1.5x_1^2+0.3x_2^2+0.5x_3^2+0.2x_4^2+5, \\
c(x) &= 1.5x_1+0.5x_2+0.5x_3+5/6x_4-3.
\end{aligned}$$

The nonzero vertices of the constraint polytope X of (33) are

$$(2,0,0,0),\ (0,6,0,0),\ (0,0,6,0),\ (0,0,0,3.6)$$

so this polytope is contained in the box $0 \leqslant x \leqslant r := (2,6,6,3.6)$.
For fixed t the problem PP(t) is

$$\max\{h(x) - tg(x) \mid c(x) \leqslant 0,\ 0 \leqslant x \leqslant r\}. \tag{34}$$

The bounding subproblem for any box $[p,q] \subset [0,r]$ is

$$\begin{array}{ll} \max & h^+(x) + s \\ \text{s.t.} & s + h^-(x) + tg(x) \leqslant 0 \\ & c(x) \leqslant 0 \\ & p \leqslant x \leqslant q,\quad -h^-(q) - tg(q) \leqslant s \leqslant -h^-(p) - tg(p). \end{array}$$

With tolerance 0.01, the parametric approach found the optimal solution

$$x = (0.000,\ 3.421875,\ 2.578125,\ 0.000)$$

with objective function value $h(x)/g(x) = 5.699282$, in 1.625 sec, generating 4 subproblems PP(t) (note that, each PP(t) is usually not solved to optimality); the direct method found the same optimal solution in 1.313 sec, i.e. is only slightly faster.

EXAMPLE 4.

$$\min\left\{ \frac{h(x)}{g(x)} \,\middle|\, c(x) \leqslant 0,\ x \geqslant 0 \right\}$$

where

$$\begin{aligned} h(x) &= h^+(x) - h^-(x), \\ h^+(x) &= 8x_1x_2x_4^4x_5^2 + 2x_1x_2^2x_3^2x_5^2 + 7x_1^4x_2^2x_3^3x_4x_5^4 + 4x_1x_2^4x_3^2 + 3x_1^2x_2^2x_3x_4^3x_5 + \\ &\quad + 6x_1^3x_3x_5 + 9x_1^3x_2^4x_3x_5 + 10x_1^2x_2x_4^2x_5^2 + 9x_1^3x_2^2x_4^3 + 5x_1^4x_3x_4^3x_5 \\ h^-(x) &= 8x_2^3x_3^3x_5^2 + 5x_3^2x_5^4 + 4x_1^2x_2^3x_3^4x_4^4 + 7x_1^4x_2x_3^2x_4^3x_5 + 4 \\ g(x) &= 3x_1x_2^3x_3^3x_4^4x_5 + 2x_2x_3x_4^2 + 3x_1^4x_2x_3^4x_4^2x_5^3 + 3x_1^2x_2x_3^3x_4^4x_5^4 + 10 \\ c(x) &= x_1 + 4x_2 + 8x_3 + 10x_4 + 8x_5 - 16 \end{aligned}$$

Initial box $[0,r]$, $r = (16,4,2,1.6,2)$.
With tolerance 0.01, the parametric approach found the optimal solution:

$$x = (9.5,\ 0,\ 0.40625,\ 0,\ 0.40625)$$

with objective function value 84.498009, in 9.875 sec, generating 10 subproblems PP(t) (maximal number of nodes in each iteration: 1158); while the direct method found the optimal solution $x=(9.601563,\ 0,\ 0.399902,\ 0,\ 0.399902)$ with objective function value 84.532589 in 46.578 sec, i.e. is much slower.

EXAMPLE 5.

$$\max \quad \frac{x_1+x_2}{x_1+x_2+1}+\sum_{k=2}^{N}(-1)^k\frac{kx_1+x_2}{x_1+x_2+k}, \tag{35}$$

$$\text{s.t.} \quad x_1^2+x_2/2\leqslant 1; \qquad x_1/2+x_2^2\leqslant 1,\ x_1,x_2\geqslant 0, \tag{36}$$

where $N\geqslant 2$ is a natural number to be specified. Setting

$$g(x)=\prod_{k=1}^{N}[x_1+x_2+k], \quad g_i(x)=\prod_{k=1}^{N}\{x_1+x_2+k,\ k\neq i\}$$

$$h^+(x)=[x_1+x_2]g_1(x)+\sum_{i=1}^{[N/2]}[2ix_1+x_2]g_{2i}(x)$$

$$h^-(x)=\sum_{i=1}^{[(N-1)/2]}[(2i+1)x_1+x_2]g_{2i+1}(x),$$

we see that these are polynomials of degree N in x_1,x_2, with nonnegative coefficients. The equivalent problem (PFP) is

$$\begin{aligned}
\max \quad & yh^+(x)+s\\
\text{s.t.} \quad & s+yh^-(x)\leqslant 0\\
& yg(x)-1\leqslant 0\\
& x_1^2+x_2/2-1\leqslant 0; \quad x_1/2+x_2^2-1\leqslant 0\\
& 0\leqslant x\leqslant r, \quad 1/g(r)\leqslant y\leqslant 1/g(0), \quad -h^-(r)/g(0)\leqslant s\leqslant -h^-(0)/g(r),
\end{aligned}$$

where $r=(1,1)$. Since $h(x)\leqslant h^+(q)-h^-(p)$, and $g(x)\geqslant g(p)$ $\forall x\in[p,q]$, an upper bound for $\frac{h(x)}{g(x)}$ over a box $[p,q]$ is $[h^+(q)-h^-(p)]/g(p)$.

- For $N=7$, with tolerance 0.01 the direct approach found the optimal solution

  $$x_1=0,\ x_2=1$$

  with objective function value $h(x)/g(x)=0.634521$ at iteration 0 and confirmed it at iteration 11824, in 6.11 sec (maximal number of nodes in each iteration: 3578).
  With tolerance 0.01 the parametric approach found the optimal solution

  $$x_1=0, \quad x_2=1$$

with objective function value $h(x)/g(x)=0.634521$ in 22.438 sec, generating 2 subproblems PP(t) (maximal number of nodes in each iteration: 3446).

- For $N=10$, with tolerance 0.01 the direct approach found the optimal solution

  $$x_1=0.794813, \quad x_2=0.736547$$

  with $h(x)/g(x)=1.302486$, at iteration 5330 and confirmed it at iteration 26917, in 65.359 sec (maximal number of nodes in each iteration: 9027).
  With tolerance 0.01 the parametric approach found the optimal solution

  $$x_1=0.794813, \quad x_2=0.736547$$

  with $h(x)/g(x)=1.302486$, in 130.281 sec, generating 6 subproblems PP(t) (maximal number of nodes in each iteration: 9815).

The above example demonstrates the efficiency of our method for the optimization of sums or products of linear functions when the number of variables is small compared to the number of sums or products. Also it suggests that for problems of small dimension (typically, $n\leqslant 4$) the direct method often outperforms the parametric method, though the situation may be reversed as n increases (as shown by Example 4).

EXAMPLE 6 (Minimize a synomial fraction under linear constraints).

$$\min\left\{\frac{h(x)}{g(x)}\middle|\, c(x)\leqslant 0, x\geqslant 0\right\}$$

where

$$\begin{aligned}
h(x) &= h^+(x)-h^-(x)\\
h^+(x) &= 3x_1^2x_3^4x_4^{4/5}+9x_1^{2/3}x_2x_3^{4/3}x_4^{3/4}x_5^{4/3}+7x_1^{1/2}x_2^{1/4}x_3^{1/2}+x_1^{1/2}x_2^2x_3^2x_5^{1/2}+\\
&\quad +2x_1^{2/5}x_2x_3^{1/2}x_4^{1/2}x_5^{3/5}+7x_2x_3^4x_5^{2/3}+2x_2^{1/2}x_3^2x_4^{3/4}x_5^{4/5}+6x_1x_3^{3/2}x_4x_5^{2/5}+\\
&\quad +x_1^{3/4}x_2^2x_3^{3/4}x_4^{4/5}x_5^2+3x_1^{1/4}x_2^{3/4}x_3^{1/3}x_4^{2/5}x_5^{1/4}+7\\
h^-(x) &= 8x_1^{1/4}x_4x_5+8x_1x_2^{3/5}x_5^{1/2}+x_1^{3/5}x_2^{4/3}x_3^{3/5}x_4x_5^3+9x_1^2x_3^{3/2}x_4x_5+1\\
g(x) &= x_3^4x_4^2+8x_2^{1/2}x_3^3x_4^{1/2}x_5^{1/2}+\\
&\quad +4x_1^{3/2}x_2^{4/3}x_3^{1/4}x_4^{4/5}+x_1^{1/5}x_2^{1/4}x_3^{1/5}x_4^{4/3}x_5^2+9\\
c(x) &= 10x_1+10x_2+x_3+2x_4+7x_5-10.
\end{aligned}$$

Initial box $[0,r]$, $r=(1,1,10,5,1.428571)$.

With tolerance 0.01 the parametric method found the optimal solution:

$$x=(0,\ 0.176514,\ 7.058105,\ 0,\ 0.168108)$$

with objective function value 104.446287, in 26.063 sec, generating 12 subproblems PP(t) (maximal number of nodes in each iteration: 1841).

EXAMPLE 7 (Maximize a synomial fraction under synomial constraints).

$$\text{Maximize } \frac{3x_3^{3/4}x_4^{3/2} - 4x_1^{1/4}x_2^2x_4^{2/3} - 2x_1^{1/2}x_4^{1/3} - x_1^{1/3}x_3^{1/3}x_4^{2/3}}{4x_1^{1/2}x_3x_4^{1/2} + 2x_1x_2^{3/2}x_3^{3/4}x_4 + 5x_1^3x_2^3x_3^{1/2}x_4^{1/4} + 2}$$

subject to

$$\begin{aligned}
&5x_2^{3/4}x_3^{3/4}x_4 + 4x_2^{1/2}x_3^{2/3} \leqslant 24.474783\\
&5x_1x_3^{3/2} + 5x_2^{1/2}x_4 + 5x_3^{1/4}x_4 \geqslant 21.142136\\
&4x_1^{1/4}x_3^{1/3}x_4^{1/2} + 2x_2^{1/2}x_3x_4 + 3x_1x_4^{1/2} + 5x_3x_4^3 + 5x_1^{1/2}x_2^{1/2}x_4^{1/2} \geqslant 43.656854\\
&4x_1^2x_2^{1/3}x_3^{3/2}x_4^{2/3} + x_2^2x_3x_4^{3/2} - 4x_1x_2x_3^{1/4}x_4 -\\
&\quad -2x_2x_3^{3/4}x_4 - x_3^{1/3} - 4x_1^{1/2}x_2^3x_3^2 \leqslant 5.313708\\
&3x_1^3x_3^{3/4}x_4^{3/4} + 3x_1^{1/2}x_2x_3^{3/2}x_4^3 + 4x_1x_2^2x_4^{1/2} - 4x_1^2x_2^{1/2}x_3^{1/2}x_4 -\\
&\quad -2x_1x_4^{3/2} - 2x_1x_2x_3 - 4x_1^{1/3}x_2x_3^{1/2}x_4^{1/2} \leqslant 0\\
&3x_2^{3/2}x_3^{1/3}x_4^3 + 3x_1^{1/2}x_3^{1/2}x_4^{3/2} + 5x_1x_2^{1/2}x_3 - 3x_1^{3/4}x_2^{3/4}x_4^{1/2} \leqslant 71.882251\\
&0 \leqslant x_1 \leqslant 5; \quad 1.2 \leqslant x_2 \leqslant 5; \; 0 \leqslant x_3 \leqslant 5; \; 0 \leqslant x_4 \leqslant 5.
\end{aligned}$$

With tolerance: 0.01, the parametric approach found the optimal solution:

$$x = (0,\ 1.2,\ 1.419764,\ 2.528262)$$

with objective function value 7.843075, in 2.219 sec, generating 4 subproblems PP(t) (maximal number of nodes in each iteration: 69).

8. Concluding Remarks

Polynomial (or more generally, synomial) fractional programs are among the hardest global optimization problems. At the present state of knowledge any deterministic algorithm can hardly solve large scale non specially structured problems of this class. The method proposed in this paper is certainly not an exception, even though it is quite practical for problems of reasonably small size. However, as the first general approach it leaves much room for further improvements and specialized adaptations to specific problems arising from practical applications (for example problems of the form mentioned in the Introduction: $\min\{\Phi(y) \,|\, y_i = \frac{h_i(x)}{g_i(x)}, x \in \mathbb{R}^n,\ i = 1, \ldots, m\}$ where $\Phi(y)$ is an increasing function in $\mathbb{R}^n_+$ with n small and m fairly large). Especially, it seems to work quite well when each of the functions $h(x), g(x), c_i(x)$ is a polynomial with coefficients of the same sign (either all positive, or all negative), so that additional variables are not needed. Also a positive feature worth noticing of the proposed method is that it is easy to implement and often gives a good feasible solution rather fast.

Acknowlegements

Research of the first author was supported in part by the Collaborative Research Fund endowed to the Center for Research in Advanced Financial Technology, Tokyo Institute of Technology, by Hitachi Research Institute, Co. Research of the second and third authors was supported by Grant-in-Aid for Scientific Research B(2)12480105 of the Japan Ministry of Education, Science and Culture. We are indebted to N.T. Hoai Phuong for most of the illustrative examples in Section 7.

References

1. Dinkelbach, W. (1967), On nonlinear fractional programming, *Management Science*, 9, 16–24.
2. Duong, P.C. (1987), Finding the global extremum of a polynomial function. In: *Essays on Nonlinear Analysis and Optimization Problems*, Institute of Mathematics, Hanoi, pp. 111–120.
3. Floudas, C.A. et al. (1999), *Handbook of Test Problems in Local and Global Optimization*, Kluwer.
4. Ibaraki, T. (1981), Solving mathematical programming problems with fractional objective functions, In: Schaible, S. and Ziemba, W.T. (eds.), *Generalized Concavity in Optimization and Economics*, Academic Press, pp. 441–472.
5. Konno, H. and Fukaisi, K. (2000), A branch and bound algorithm for solving low rank linear multiplicative and fractional programming problems, *Journal of Global Optimization*, 18, 283–299.
6. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on Low Rank Nonconvex Structures*, Kluwer.
7. Gotoh, J.Y. and Konno, H. (1999), Maximization of the ratio of two convex quadratic functions over a polytope, Preprint, Dept of Industrial Engineering and Management, Tokyo Institute of Technology, Tokyo.
8. Hoai-Phuong, Ng.T. and Tuy, H. (2000), A unified monotonic approach to generalized linear fractional programming, Preprint, Hanoi Institute of Mathematics, Hanoi.
9. Horst, R. and Tuy, H. (1996), *Global Optimization, Deterministic Approaches*, third edition, Springer.
10. Ibaraki, T. (1983), Parametric approaches to fractional programs, *Mathematical Programming*, 26, 345–362.
11. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on Low Rank Nonconvex Structures*, Kluwer.
12. Kuno, T. (1997), A variant of the outer approximation method for globally minimizing a class of composite functions, *Journal of the Operations Research Society of Japan*, 40, 245–260.
13. Lasserre, J. (2001), Global optimization with polynomials and the problem of moments, *SIAM J. on Optimization*, 11, 796–817.
14. Lasserre, J. (2002), Semidefinite programming vs. LP relaxations for polynomial programming, *Mathematics of Operations Research*, 27, 347–360.
15. Rubinov, A., Tuy, H. and Mays, H. (2001), An algorithm for monotonic global optimization problems, *Optimization*, 49, 205–222.
16. Schaible, S. (1995), Fractional programming. In: Horst, R. and Pardalos, P. (eds.), *Handbook of Global Optimization*, Kluwer, pp. 495–608.
17. Sherali, H. and Tuncbilek, C.H. (1992), A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, *Journal of Global Optimization*, 2, 101–112.
18. Sherali, H. and Tuncbilek, C.H. (1997), New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems, *Operations Research Letters*, 21, 1–9.

19. Sherali, H. (1998), Global optimization of nonconvex polynomial programming problems having rational exponents, *Journal of Global Optim.*
20. Shi, J. (1999), Global Optimization for Fractional Programming, Preprint, School of Management, Science University of Tokyo.
21. Shor, N.Z. (1998), *Nondifferentiable Optimization and Polynomial Problems*, Kluwer.
22. Stancu-Minasian, I.M. (1997), *Fractional Programming: Theory, Methods and Applications* (Mathematics and Its Applications), Kluwer.
23. Tuy, H. (1998), *Convex Analysis and Global Optimization*, Kluwer.
24. Tuy, H. (1997), Normal sets, polyblocks and monotonic optimization, *Vietnam Journal of Mathematics*, Springer Verlag, 27(4), 277–300.
25. Tuy, H. (2000), Monotonic Optimization: Problems and Solution Approaches, *SIAM Journal on Optimization*, 11(2), 464–494.
26. Tuy, H. (2001), Polyblock Algorithms Revisited, preprint 2001/40, Institute of Mathematics, Hanoi.
27. Tuy, H. and Luc, L.T. (2000), A new approach to optimization under monotonic constraint, *Journal of Global Optimization,* in press.
28. Wingo, D.R. (1985), Globally minimizing polynomials without evaluating derivatives, *Intl J. Comput. Math.*, 17(287).