Journal of Global Optimization **18:** 283–299, 2000.

A Branch and Bound Algorithm for Solving Low Rank Linear Multiplicative and Fractional Programming Problems

HIROSHI KONNO[1] and KENJI FUKAISHI[2]
[1]*Department of Industrial Engineering and Management, Tokyo Institute of Technology, 2-12-1 Oh-Okayama Meguro-Ku, Tokyo 152, Japan;* [2]*NTT Data Corporation.*

(Received 12 April 1999; accepted in revised form 10 May 2000)

Abstract. This paper is concerned with a practical algorithm for solving low rank linear multiplicative programming problems and low rank linear fractional programming problems. The former is the minimization of the sum of the product of two linear functions while the latter is the minimization of the sum of linear fractional functions over a polytope. Both of these problems are nonconvex minimization problems with a lot of practical applications. We will show that these problems can be solved in an efficient manner by adapting a branch and bound algorithm proposed by Androulakis–Maranas–Floudas for nonconvex problems containing products of two variables. Computational experiments show that this algorithm performs much better than other reported algorithms for these class of problems.

Key words: Linear multiplicative programming problem, Linear fractional programming problem, Global minimization, Branch and bound method, Linear underestimating function

1. Introduction

In this paper, we will discuss a branch and bound algorithm for solving rank-p linear multiplicative programming problems(LMP):

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p}(\mathbf{c}_j^t\mathbf{x}+c_{j0})(\mathbf{d}_j^t\mathbf{x}+d_{j0}) \\ \text{subject to} & \mathbf{x}\in X, \end{array}\right. \tag{1}$$

and rank-p linear fractional programming problems(LFP):

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p}\frac{\mathbf{c}_j^t\mathbf{x}+c_{j0}}{\mathbf{d}_j^t\mathbf{x}+d_{j0}} \\ \text{subject to} & \mathbf{x}\in X, \end{array}\right. \tag{2}$$

where $\mathbf{c}_j, \mathbf{d}_j \in \Re^n, c_{j0}, d_{j0} \in \Re^1$ $(j = 1, \ldots, p), \mathbf{x} \in \Re^n$ and $X \subset \Re^n$ is a polytope.

Rank-p LMP (2) is a special case of nonconvex quadratic programming problems, which is known to be NP-hard even when $p = 1$ [19]. Rank-1 LFP, on the other hand, is a quasi-convex minimization problem, whose local solution is a global solution. However, when $p \geqslant 2$, the problem (2) is a nonconvex miniminzation problem with multiple local minima.

The problems (1) and (2) have a lot of interesting applications in engineering and finance. Readers are referred to [1, 11, 12, 15] for such applications. These problems attracted attention of researchers in global optimization, where one of its promising research direction is to construct efficient algorithms for practical problems by exploiting their special structure (see [11]).

To avoid technical complication, we assume that the condition:

$$\mathbf{c}_j^t\mathbf{x} + c_{j0} > 0, \mathbf{d}_j^t\mathbf{x} + d_{j0} > 0, \quad \forall\, \mathbf{x} \in X, j = 1, \ldots, p, \tag{3}$$

holds throughout the paper. Let us note that this condition is valid for almost all practical applications.

Let us briefly review some of the earlier algorithmic studies on LMP's (1) and LFP's (2). For a recent survey on LFP's, readers are referred to [21].

There exists a number of very efficient algorithms for solving rank-1 LMP's. For example, variants of parametric simplex algorithms have been developed by Konno–Kuno [13], Swarup [22] for this problem. In particular, a parametric simplex algorithm proposed in [13] can solve rank-1 LMP's in no more than twice as much computation time than that of solving a linear program: minimize $\{\mathbf{c}_j^t\mathbf{x} + c_{j0} | \mathbf{x} \in X\}$. Further, it has been proved that the amount of computation time of this algorithm is average polynomial order under some assumption on the probability distribution of the problem data [11].

When $p \geqslant 2$, LMP's are much more difficult. A number of practical algorithms have been developed by Kuno-Konno [18], Phong-An-Tao [20] and others. Computation time grows exponentially as p increases. But it has been demonstrated in [20] that the careful implementation of a branch and bound algorithm using convex underestimating function of Phong et al. can solve the LMP's up to, say $p = 10$.

Computational studies of rank-p linear fractional programming problem (2) are less intensive than (1). When $p = 1$, Charnes and Cooper [5] developed an efficient algorithm using a simplex type procedure. When $p = 2$, Konno–Yajima [16] and Hirsche [8] proposed a parametric simplex algorithm and successive underestimation method, both of which can successfully solve (2) in an efficient manner. Also, Konno–Abe [12] proposed a heuristic algorithm for $p = 3$ by employing Konno–Yajima [16] algorithm for solving rank-2 problems.

When $p \geqslant 3$, Konno–Yamashita applied generalized convex multiplicative programming algorithm [17] and showed that it can solve the problem (2) up to $p = 5$. Also Falk-Polacsay [7] proposed yet another parametric approach, whose efficiency has not been tested.

2. Convex Relaxation

Let us consider a quadratically constrained linear programming problem:

$$(P_0)\quad \left|\begin{array}{ll} \text{minimize} & g(\mathbf{x}) = \displaystyle\sum_{j=1}^{n} c_{0j}x_j \\ \text{subject to} & \displaystyle\sum_{j=1}^{n} a_j^i x_j + \sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk}^i x_j x_k \leqslant b_i, \quad i = 1, \ldots, m, \\ & \alpha_j^0 \leqslant x_j \leqslant \beta_j^0, \quad j = 1, \ldots, n, \\ & \mathbf{x} \in X. \end{array}\right. \tag{4}$$

If, at least one $Q_i = (q_{jk}^i) \in \Re^{n\times n}$ is not positive semi-definite, then this is a non convex minimization problem whose local solution may not be a global solution.

Let us note that both LMP's and LFP's can be put in the form of (4). In fact, rank-p LMP (1) is equivalent to

$$\left|\begin{array}{ll} \text{minimize} & x_{n+1} \\ \text{subject to} & \displaystyle\sum_{j=1}^{p} (\mathbf{c}_j^t\mathbf{x} + c_{j0})(\mathbf{d}_j^t\mathbf{x} + d_{j0}) \leqslant x_{n+1}, \\ & \mathbf{x} \in X. \end{array}\right. \tag{5}$$

Also, rank-p LFP (2) can be converted to

$$\left|\begin{array}{ll} \text{minimize} & x_{n+1} + x_{n+2} + \cdots + x_{n+p} \\ \text{subject to} & (\mathbf{c}_j^t\mathbf{x} + c_{j0}) - x_{n+j}(\mathbf{d}_j^t\mathbf{x} + d_{j0}) \leqslant 0, \quad j = 1, \ldots, p, \\ & \mathbf{x} \in X, \end{array}\right. \tag{6}$$

under assumption (3).

The basic strategy for solving (4) is to introduce a simple convex underestimating function for each nonconvex term $q_{jk}^i x_j x_k$ in the rectangular region $D_{jk}^0 \equiv [\alpha_j^0, \beta_j^0] \times [\alpha_k^0, \beta_k^0]$ by employing the scheme proposed in Andourakis et al. [3].

When $j = k$, we obtain underestimating convex function(convex envelope)

$$f_{jj}^i(x_j, x_j) = \begin{cases} q_{jj}^i x_j^2 & \text{if } q_{jj}^i > 0, \\ q_{jj}^i(\alpha_j^0 + \beta_j^0)x_j - q_{jj}^i\alpha_j^0\beta_j^0 & \text{if } q_{jj}^i \leqslant 0, \end{cases} \tag{7}$$

as shown in Figure 1.

PROPOSITION 1. *Let*

$$f_{jk}^i(x_j, x_k) = \begin{cases} q_{jk}^i \max\{\alpha_j^0 x_k + \alpha_k^0 x_j - \alpha_j^0\alpha_k^0, \beta_j^0 x_k + \beta_k^0 x_j - \beta_j^0\beta_k^0\} & \text{if } q_{jk}^i > 0, \\ q_{jk}^i \min\{\alpha_j^0 x_k + \beta_k^0 x_j - \alpha_j^0\beta_k^0, \beta_j^0 x_k + \alpha_k^0 x_j - \beta_j^0\alpha_k^0\} & \text{if } q_{jk}^i \leqslant 0. \end{cases} \tag{8}$$

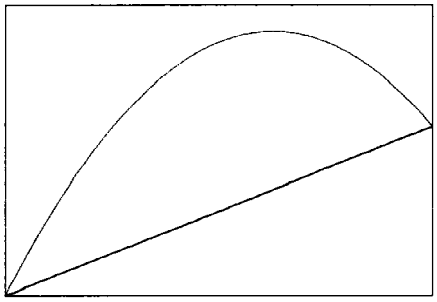

Figure 1. Function $-x^2$ and its convex envelope.

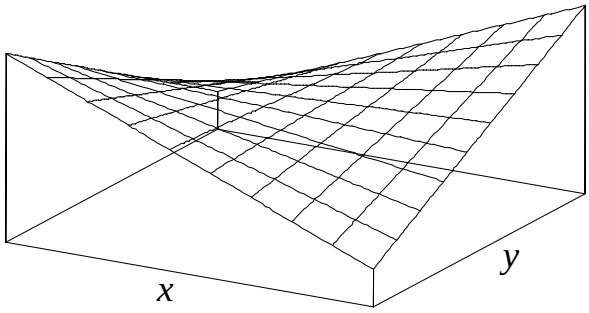

Figure 2. Graph of the function xy.

$f^i_{jk}(x_j, x_k)$ is a convex envelope of the quadratic function $q^i_{jk}x_jx_k$. In particular, the following relation holds:

$$f^i_{jk}(x_j, x_k) = q^i_{jk}x_jx_k, \quad \forall (x_j, x_k) \text{ on } D^0_{jk}, \tag{9}$$

$$|f^i_{jk}(x_j, x_k) - q^i_{jk}x_jx_k| \leqslant |q^i_{jk}|\frac{(\alpha_j - \beta_j)(\alpha_k - \beta_k)}{4}, \quad \forall (x_j, x_k) \text{ on } D^0_{jk}. \tag{10}$$

Proof. See [2,3] □

We see from (10) that the maximal difference of $q^i_{jk}x_jx_k$ and $f^i_{jk}(x_j, x_k)$ is proportional to the area of the rectangle D^0_{jk}. Figure 2 and 3 illustrates the relation of the function $q^i_{jk}x_jx_k$ and $f^i_{jk}(x_j, x_k)$.

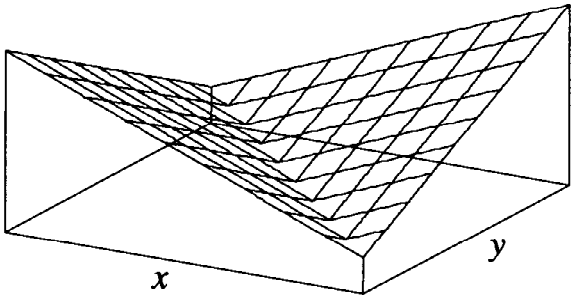

Figure 3. Convex envelope of the function xy.

Let us define a convex minimization problem:

$$\left|\begin{array}{ll} \text{minimize} & g(\mathbf{x}) = \displaystyle\sum_{j=1}^{n} c_{0j}x_j \\ \text{subject to} & \displaystyle\sum_{j=1}^{n} a_j^i x_j + \sum_{j=1}^{n}\sum_{k=1}^{n} f_{jk}^i(x_j, x_k) \leqslant b_i, \quad i = 1, \ldots, m, \\ & \alpha_j^0 \leqslant x_j \leqslant \beta_j^0, \quad j = 1, \ldots, n, \\ & \mathbf{x} \in X. \end{array}\right. \tag{11}$$

PROPOSITION 2.

(i) If (11) is infeasible, then (4) is also infeasible.

(ii) Let $\hat{\mathbf{x}}$ *be an optimal solution of (11).Then*

(a) $g(\hat{\mathbf{x}})$ *gives the lower bound of the minimal value of* $g(\mathbf{x})$ *of (4).*

(b) $\hat{\mathbf{x}}$ *is an optimal solution of the problem (4) if* $\hat{\mathbf{x}}$ *is a feasible solution of (4).*

3. A Branch and Bound Algorithm

We showed in the previous section how to calculate a lower bound of the optimal value of the problem (4). Let us note that the problem (11) is a convex programming problem, so that it can be solved in an efficient manner. Therefore we can develop a branch and bound method if we can provide:

(i) an efficient method to calculate a feasible solution of (4) which gives an upper bound of the optimal solution of (4),

(ii) a convergent (exhaustive) branching procedure.

It is generally not easy to calculate a feasible solution of nonconvex minimization problem (4). Fortunately, however any solution $\tilde{\mathbf{x}} \in X$ such that $\tilde{\mathbf{x}} \in [\boldsymbol{\alpha}, \boldsymbol{\beta}]$ gives a feasible solution of (5) and (6). To see this let

$$\tilde{x}_{n+1} = \sum_{j=1}^{p} (\mathbf{c}_j^t \tilde{\mathbf{x}} + c_{j0})(\mathbf{d}_j^t \tilde{\mathbf{x}} + d_{j0}).$$

Then $(\tilde{\mathbf{x}}, \tilde{x}_{n+1})$ is a feasible solution of (5). Also, let

$$\tilde{x}_{n+j} = (\mathbf{c}_j^t \tilde{\mathbf{x}} + c_{j0})/(\mathbf{d}_j^t \tilde{\mathbf{x}} + d_{j0}), \quad j = 1, \ldots, p.$$

Then $(\tilde{\mathbf{x}}, \tilde{x}_{n+1}, \ldots, \tilde{x}_{n+p})$ is a feasible solution of (6).

One standard method of branching is the following bisection of the hyper-rectangle D into two subrectangles by using the longest edge. Let

$$\beta_t^0 - \alpha_t^0 = \max[\beta_j^0 - \alpha_j^0 | j = 1, \ldots, n]$$

and let $\beta_t^1 = (\alpha_t^0 + \beta_t^0)/2$. We partition $D \equiv [\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0]$ into two rectangles $D_1 \equiv [\boldsymbol{\alpha}^1, \boldsymbol{\beta}^1]$ and $D_2 \equiv [\boldsymbol{\alpha}^2, \boldsymbol{\beta}^2]$ by defining

$$\alpha_j^1 = \alpha_j^0, \quad j = 1, \ldots, n,$$
$$\beta_j^1 = \begin{cases} \beta_j^0, & j \neq s, \\ (\alpha_t^0 + \beta_t^0)/2, & j = t, \end{cases}$$
$$\alpha_j^2 = \begin{cases} \alpha_j^1, & j \neq s, \\ (\alpha_t^0 + \beta_t^0)/2, & j = t, \end{cases}$$
$$\beta_j^2 = \beta_j^0, \quad j = 1, \ldots, n.$$

Branch and Bound Algorithm

Step 0. Let $\varepsilon_c > 0, \varepsilon_f > 0, L = -\infty, \mathcal{P} = \{(P_0)\}$. Calculate a locally optimal solution $\hat{\mathbf{x}}$ of (P_0) by applying appropriate local search algrithm. Let $L_0 = U = g(\hat{\mathbf{x}})$.

Step 1. If $\mathcal{P} = \phi$, then stop. Otherwise choose a problem $(P_s) \in \mathcal{P}$ such that $L_s = \min\{L_j | P_j \in \mathcal{P}\}$:

$$(P_s) \left| \begin{array}{ll} \text{minimize} & g(\mathbf{x}) = \displaystyle\sum_{j=1}^{n} c_{0j} x_j \\ \text{subject to} & \displaystyle\sum_{j=1}^{n} a_j^i x_j + \sum_{j=1}^{n} \sum_{k=1}^{n} q_{jk}^i x_j x_k \leqslant b_i, \quad i = 1, \ldots, m, \\ & \alpha_j^s \leqslant x_j \leqslant \beta_j^s, \quad j = 1, \ldots, n, \\ & \mathbf{x} \in X. \end{array} \right. \tag{12}$$

$$\mathcal{P} = \mathcal{P} \setminus (P_s).$$

Step 2. Generate two subproblems (P_{l+1}) and (P_{l+2}):

$$(P_{l+1})\left|\begin{array}{ll} \text{minimize} & g(\mathbf{x}) = \sum_{j=1}^{n} c_{0j}x_j \\ \text{subject to} & \sum_{j=1}^{n} a_j^i x_j + \sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk}^i x_j x_k \leqslant b_i, \quad i = 1, \ldots, m, \\ & \alpha_j^{l+1} \leqslant x_j \leqslant \beta_j^{l+1}, \quad j = 1, \ldots, n, \\ & \mathbf{x} \in X. \end{array}\right.$$

$$(P_{l+2})\left|\begin{array}{ll} \text{minimize} & g(\mathbf{x}) = \sum_{j=1}^{n} c_{0j}x_j \\ \text{subject to} & \sum_{j=1}^{n} a_j^i x_j + \sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk}^i x_j x_k \leqslant b_i, \quad i = 1, \ldots, m, \\ & \alpha_j^{l+2} \leqslant x_j \leqslant \beta_j^{l+2}, \quad j = 1, \ldots, n, \\ & \mathbf{x} \in X. \end{array}\right.$$

where

$$\begin{aligned} \alpha_j^{l+1} &= \alpha_j^s, \quad j = 1, \ldots, n, \\ \beta_j^{l+1} &= \begin{cases} \beta_j^s, & j \neq d, \\ (\alpha_j^s + \beta_j^s)/2, & j = d, \end{cases} \\ \alpha_j^{l+2} &= \begin{cases} \alpha_j^s, & j \neq d, \\ (\alpha_j^s + \beta_j^s)/2, & j = d, \end{cases} \\ \beta_j^{l+2} &= \beta_j^s, \quad j = 1, \ldots, n, \\ \mathcal{P} &= \mathcal{P} \cup \{(P_{l+1}), (P_{l+2})\} \backslash (P_s) \\ s &:= s + 1 \end{aligned}$$

$$\beta_d^s - \alpha_d^s = \max[\beta_j^s - \alpha_j^s | j = 1, \ldots, n]$$

Step 3. Generate $(\bar{P}_s)(s = l+1, l+2)$ by replacing nonconvex term $q_{jk}^i x_j x_k$ by convex underestimating function $f_{jk}^i(x_j, x_k)$.

Step 4. Solve $(\bar{P}_s)(s = l+1, l+2)$. If $(\bar{P}_s)$ is infeasible, delete $(\bar{P}_s)$ from $\mathcal{P}$. Otherwise let $\mathbf{x}^k$ be an optimal solution of $(\bar{P}_s)$. If

$$\sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk}^i x_j^s x_k^s - \sum_{j=1}^{n}\sum_{k=1}^{n} f_{jk}^i(x_j^s, x_k^s) \leqslant \varepsilon_f,$$

then $U_s = \sum_{j=1}^{n} c_{0j}x_j^s$ and go to Step 5. Otherwise $L_s = \sum_{j=1}^{n} c_{0j}x_j^s$ and go to Step 6.

Step 5. If $U_s < U$, then $\tilde{\mathbf{x}} = \mathbf{x}^s$, $U = \sum_{j=1}^{n} c_{0j} x_j^s$ and delete all subproblem (P_t) from $\mathcal{P}$ such that $L_t > U$.

Step 6. $L = \min L_s$. If $U - L > \varepsilon_c$, go to Step 1. Otherwise ε_c-convergence has been reached. The global minimum solution $\mathbf{x}^*$ is given by $\tilde{\mathbf{x}}$. □

THEOREM 1. *The branch and bound algorithm proposed above terminates in finitely many steps.*

Proof. The subdivision scheme is exhaustive as proved in Konno–Thach–Tuy [11] (Corollary 6.3) or Horst–Tuy [10]. □

4. Computational Experiments

4.1. GENERATION OF TEST PROBLEMS

We tested the algorithm proposed in the previous section on rank-p LMP's:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p} (\mathbf{c}_j^t \mathbf{y} + c_{j0})(\mathbf{d}_j^t \mathbf{y} + d_{j0}) \\ \text{subject to} & \mathbf{A}\mathbf{y} \leqslant \mathbf{b}, \mathbf{y} \geqslant \mathbf{0}, \end{array}\right. \tag{13}$$

and rank-p LFP's:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p} \frac{\mathbf{c}_j^t \mathbf{y} + c_{j0}}{\mathbf{d}_j^t \mathbf{y} + d_{j0}} \\ \text{subject to} & \mathbf{A}\mathbf{y} \leqslant \mathbf{b}, \mathbf{y} \geqslant \mathbf{0}, \end{array}\right. \tag{14}$$

where $\mathbf{c}_j, \mathbf{d}_j, \mathbf{y} \in \Re^n$, $c_{j0}, d_{j0} \in \Re^1$ $(j = 1, \ldots, p)$, $\mathbf{A} = (a_{ij}) \in \Re^{m \times n}$, $\mathbf{b} \in \Re^m$. All elements of the matrix $\mathbf{A}$ and vector $\mathbf{b}$, $\mathbf{c}_i$'s, $\mathbf{d}_i$'s are randomly generated from the uniform distribution in certain intervals: $\mathbf{A} \in [0.01, 1.0], \mathbf{b} \in [0.1, 10.0]$, so that the feasible region is always non-empty and bounded. Also, we chose $\mathbf{c}_i \in [-0.1, 0.1], \mathbf{d}_i \in [-0.1, 0.1], \ \forall i$. Also, c_{j0}, d_{j0} are chosen in such a way that the condition (3) is satisfied. Ten test problems are generated for each size of the parameter (p, n, m).

Performance of the algorithm can be different for other class of test problems. But the main purpose of the test is to compare the performance of the present algorithm with earlier algorithms, where the above class of problems are used as test problems.

4.2. RANK-p LINEAR MULTIPLICATIVE PROGRAMMING PROBLEMS

As mentioned in Section 2, we convert the problem (13) to the following form:

$$\begin{array}{ll} \text{minimize} & w \\ \text{subject to} & \mathbf{Ay} \leqslant \mathbf{b}, \mathbf{y} \geqslant \mathbf{0}, \\ & u_i = \mathbf{c}_j^t \mathbf{y} + c_{j0}, \quad i = 1, \ldots, p, \\ & v_i = \mathbf{d}_j^t \mathbf{y} + d_{j0}, \quad i = 1, \ldots, p, \\ & \sum_{i=1}^{p} u_i v_i \leqslant w, \\ & \underline{u}_i \leqslant u_i \leqslant \bar{u}_i, \underline{v}_i \leqslant v_i \leqslant \bar{v}_i, \quad i = 1, \ldots, p. \end{array} \tag{15}$$

Let

$$\mathbf{x} = (u_i, \ldots, u_p, v_i, \ldots, v_p, \mathbf{y}, w),$$

and let

$$X = \{\mathbf{x} | \mathbf{Ay} \leqslant \mathbf{b}, \mathbf{y} \geqslant \mathbf{0}, u_i = \mathbf{c}_j^t \mathbf{y} + c_{j0}, v_i = \mathbf{d}_j^t \mathbf{y} + d_{j0}, \quad i = 1, \ldots, p\}.$$

Then the problem can be put into the form (4).

Let (P_s) be the subproblem:

$$\begin{array}{ll} \text{minimize} & w \\ \text{subject to} & \sum_{i=1}^{p} u_i v_i \leqslant w, \\ & \underline{u}_i^s \leqslant u_i \leqslant \bar{u}_i^s, \underline{v}_i^s \leqslant v_i \leqslant \bar{v}_i^s, \quad i = 1, \ldots, p, \\ & \mathbf{x} \in X. \end{array} \tag{16}$$

We tested the following three subdivision schemes of the rectanguler region D_s represented as the product of p rectangles $\prod_{i=1}^{p} ([\underline{u}_i^s, \bar{u}_i^s] \times [\underline{v}_i^s, \bar{v}_i^s])$.

(1) Bisect using the variable associated with the longest edge of D_s.
(2) Let (u_{j^*}, v_{j^*}) be the pair of variables such that the difference of u_i, v_i and the underestimating function is maximal at the current solution $\mathbf{x}^s$.
   (a) Bisect D_s by using either u_{j^*} or v_{j^*} associated with the longer edge of D_s
   (b) Subdivide D_s into subrectangles at the point $(u_{j^*}^s, v_{j^*}^s)$ using either u_{j^*} or v_{j^*} as the subdividing variable

Subdivision scheme 2(a) and 2(b) are similar to the ω-subdivision scheme used in various global optimization algorithms to accelerate convergence. We may be able to prove finite convergence of these strategies by extending the result presented in Section 7.3 of [9]. However, it still remains an open question.

Second improvement is to calculate the upper bound and lower bound of variables u_i's and v_i's by solving linear programming problems

$$\text{minimize}\{u_i | \mathbf{x} \in X, \underline{u}_j^s \leqslant u_j \leqslant \bar{u}_j^s, \underline{v}_j^s \leqslant v_j \leqslant \bar{v}_j^s, j = 1, \ldots, p, j \neq i\}$$

Table 1. Strategies to be tested

Algorithm	Subdivision	Bounds
1	1	×
2	1	1
3	1	2
4	1	3
5	1	2
6	1	3
7	2(a)	3
8	2(b)	3

etc. Since the maximal discrepacy between the underestimating function and the bilinear function is proportional to the area of the rectangular region, this option may help improve convergence. We tested three alternative strategies:
(1) Calculate bounds of all variables u_i's and v_i's before solving $(\bar{P}_s)$.
(2) Calculate bounds of the variable say u_l used for subdivision in addition to (1).
(3) Calculate bounds of v_l in addition to (2).

The Table 1 shows the result of experiments. We generated 10 test problem by fixing $p = 4, n = 10, m = 30$, $\varepsilon_c = 10^{-5}$ (relative error), $\varepsilon_f = 10^{-6}$. Program was coded in C and the experiment were conducted on Dell Dimension XPS H450.

We see from Figure 4 that subdivision scheme 2 is better than scheme 1, while there is no significant difference between 2(a) and 2(b). Calculating upper and lower bounds of u_i's and v_i's has significant effects on the performance of the algorithm. Among the three strategies, version 3 appears to be the best. (Note that Algorithm 6 using bounding strategy 3 performs best when we fix subdivision strategy.)

Based upon these observations we conducted experiments for problems with larger p's using Algorithm 7 and 8 which use ω-subdivision strategy.

We see from Table 2 and Figure 5 that Algorithm 7 performs very well for LMP's. Computation time grow much slower as a function of p compared with algorithm presented in Kuno–Konno [18].

Figure 6 shows the computation time for Algorithm 7 for LMP's for different values of ϵ_c. We see from this that computation time increases very mildly as ϵ_c decrease.

We see from Tables 2, 3 and Figure 5 that Algorithm 7 performs better than Algorithm 8.

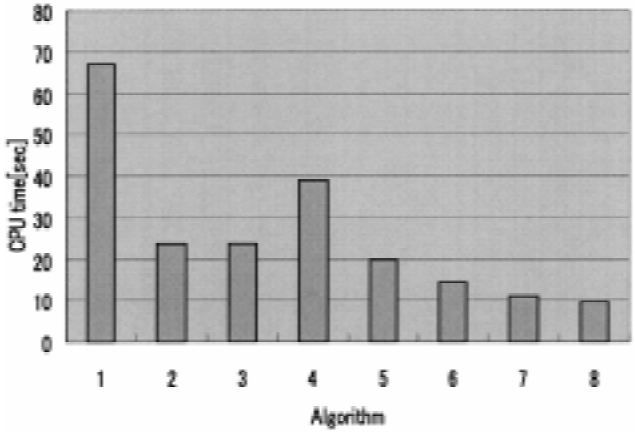

Figure 4. Performance of Algorithms for LMP's ($p = 4, n = 10, m = 30$)

Table 2. Computation time of Algorithm 7.

p	(n, m)	Average CPU time (s.d.)	Average # of iteration (s.d.)
	(10,30)	5.9(1.8)	5.4(4.0)
3	(30,10)	51.8(35.8)	34.4(27.7)
	(30,50)	59.2(52.1)	30.0(30.4)
	(50,30)	80.1(28.3)	22.6(11.7)
	(10,30)	9.7(3.0)	8.7(4.8)
4	(30,10)	92.6(84.3)	49.9(44.5)
	(30,50)	65.0(40.44)	34.5(29.5)
	(50,30)	146.7(167.0)	48.0(45.5)
5	(10,30)	53.3(61.6)	50.7(59.3)
	(30,10)	156.2(76.4)	108.2(56.1)
6	(10,30)	30.2(26.6)	27.1(30.0)
7	(10,30)	103.0(97.8)	94.2(92.4)
8	(10,30)	71.9(40.6)	47.0(34.1)

Table 3. Computation time of Algorithm 8.

p	(n, m)	Average CPU time (s.d.)	Average # of iteration (s.d.)
	(10,30)	6.3(2.0)	5.4(3.9)
3	(30,10)	40.0(21.0)	26.2(19.7)
	(30,50)	50.7(37.0)	25.0(23.2)
	(50,30)	70.2(29.2)	18.5(13.9)
	(10,30)	10.1(3.8)	9.5(5.9)
4	(30,10)	96.3(103.4)	49.3(49.7)
	(30,50)	74.7(40.0)	39.7(31.5)
	(50,30)	147.1(121.6)	48.4(49.8)
5	(10,30)	78.7(109.1)	67.7(89.4)
	(30,10)	177.9(79.1)	123.6(56.6)
6	(10,30)	48.6(39.5)	46.2(45.4)
7	(10,30)	196.6(197.2)	157.8(151.7)
8	(10,30)	222.8(127.5)	152.3(93.3)

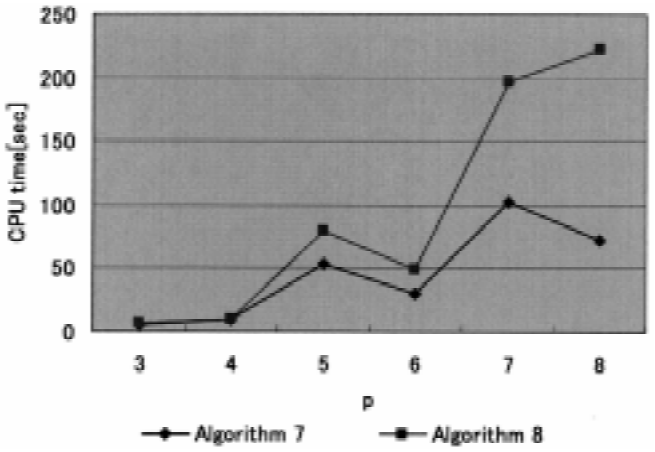

Figure 5. Computation time for LMP's ($n = 10, m = 30$)

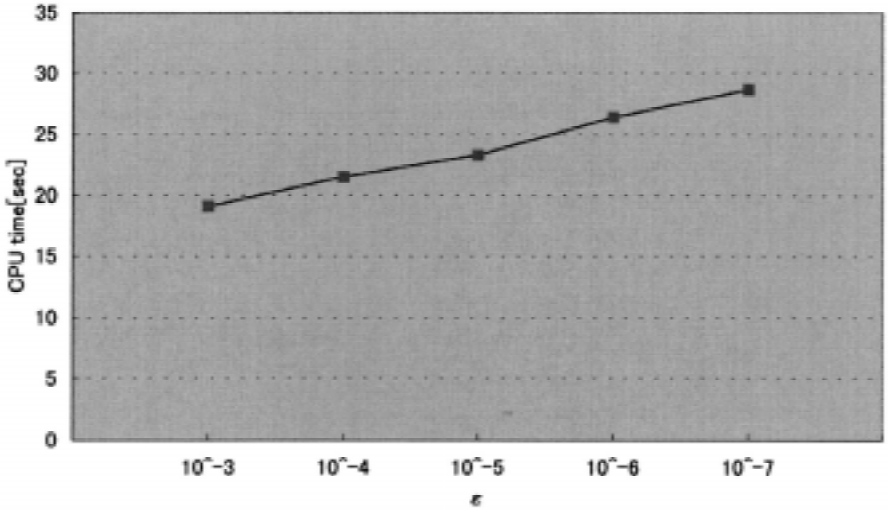

Figure 6. Computation time of Algorithm 7 for LMP's ($p = 4$, $n = 10$, $m = 30$) as a function of ε_c.

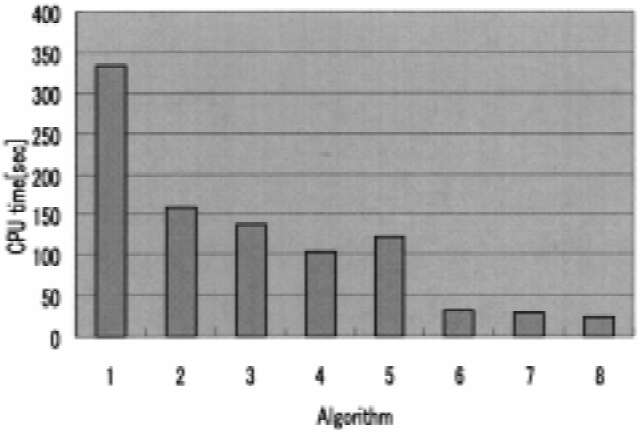

Figure 7. Performance of Algorithms for LFP's ($p = 4$, $n = 10$, $m = 30$).

4.3. RANK-p LINEAR FRACTIONAL PROGRAMMING PROBLEMS

We conducted similar experiments for rank-p LFP's:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p} \frac{\mathbf{c}_j^t \mathbf{y} + c_{j0}}{\mathbf{d}_j^t \mathbf{y} + d_{j0}} \\ \text{subject to} & \mathbf{A}\mathbf{y} \leqslant \mathbf{b}, \mathbf{y} \geqslant \mathbf{0}. \end{array}\right. \tag{17}$$

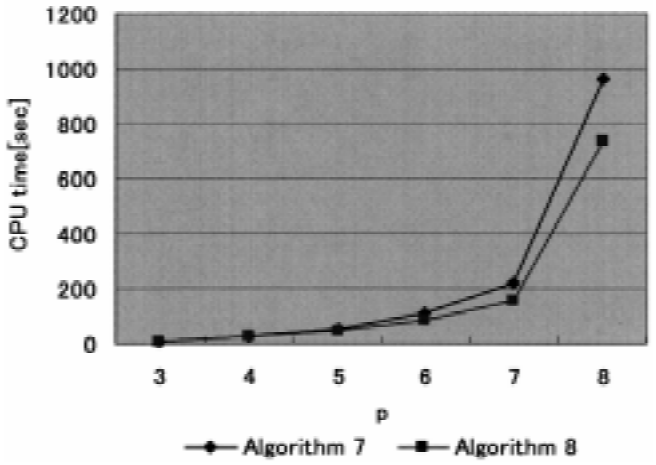

Figure 8. Computation Time for LFP's ($n = 10, m = 30$)

To solve this problem, we first apply Charnes-Cooper transformation

$$z_0 = 1/(\mathbf{d}_p^t \mathbf{y} + d_{p0}), \tag{18}$$

$$\mathbf{z} = \mathbf{y} z_0, \tag{19}$$

and reduce the problem to an equivalent problem with one less fractional terms.

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p-1} \frac{\mathbf{c}_j^t \mathbf{z} + c_{j0} z_0}{\mathbf{d}_j^t \mathbf{z} + d_{j0} z_0} + \mathbf{d}_p^t \mathbf{z} + d_{j0} z_0 \\ \text{subject to} & \mathbf{A}\mathbf{z} - \mathbf{b} z_0 \leqslant \mathbf{0},\ (\mathbf{z}, z_0) \geqslant \mathbf{0}, \end{array}\right. \tag{20}$$

which is equivalent to

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{p-1} w_i + \mathbf{d}_p^t \mathbf{z} + d_{j0} z_0 \\ \text{subject to} & u_i = \mathbf{c}_j^t \mathbf{z} + c_{j0} z_0, \quad i = 1, \ldots, p-1, \\ & v_i = \mathbf{d}_j^t \mathbf{z} + d_{j0} z_0, \quad i = 1, \ldots, p-1, \\ & u_i - v_i w_i \leqslant 0, \quad i = 1, \ldots, p-1, \\ & \mathbf{A}\mathbf{z} - \mathbf{b} z_0 \leqslant \mathbf{0},\ (\mathbf{z}, z_0) \geqslant \mathbf{0}, \end{array}\right. \tag{21}$$

For the proof of the equivalence of (17) and (20), readers are referred to Konno–Yamashita [17]:

Figure 7 shows the computational results for the case ($p = 4, n = 10, m = 30$). We see from this that Algorithm 7 and 8 perform best in this case. Tables 4 and 5 show the results of Algorithm 7 and 8 for larger problems.

We see from Figure 8 that both Algorithm 7 and 8 perform more or less equally. The computation time is almost the same as LMP's (Figure 5), but it jumps at $p = 8$ as opposed to the mild increase in the case of LMP's.

Table 4. Computation time of Algorithm 7.

p	(n, m)	Average CPU time (s.d.)	Average # of iteration (s.d.)
	(10,30)	8.1(1.2)	5.7(2.0)
3	(30,10)	34.4(7.2)	9.8(6.0)
	(30,50)	35.1(4.6)	6.9(2.9)
	(50,30)	77.8(9.3)	6.9(3.1)
	(10,30)	23.7(5.2)	16.1(7.2)
4	(30,10)	87.8(30.6)	26.5(16.7)
	(30,50)	78.2(24.7)	16.4(10.6)
	(50,30)	161.4(29.8)	16.8(7.4)
5	(10,30)	52.2(16.8)	32.2(16.3)
	(30,10)	257.7(172.2)	77.1(66.3)
6	(10,30)	111.5(33.4)	49.5(24.9)
7	(10,30)	217.2(70.4)	85.8(33.1)
8	(10,30)	958.4(845.2)	276.4(215.7)

5. Conclusion

We showed in this paper that the branch and bound algorithm can be used as a practical algorithm for solving rank-p linear fractional programming problems (LMP) and linear fractional programming problem (LFP), up to $p = 10$ in the case of LMP's and up to $p = 8$ in the case of LFP's.

These results show that our algorithm is superior to the earlier algorithms proposed in the literature. In the case of rank-p LMP's problem, our algorithm is much faster than the algorithm of Kuno–Konno [18]. Also, it is slightly faster than the algorithm of Phong-An-Tao [20]. In the case of rank-p LFP's, our algorithm is much faster than the algorithm proposed in Konno–Yamashita [17] and Falk–Polacsay [7]

We observe a large variance of computation time (see Tables 2–5), which is common to all branch and bound type algorithms. However, an optimal solution is generated at the earlier stage of computation for almost all test problems. Therefore, we can now solve problems (1) and (2) if p is less than 10.

Table 5. Computation time of Algorithm 8

p	(n, m)	Average CPU time (s.d)	Average # of iteration (s.d)
	(10,30)	8.1(0.9)	5.7(1.7)
3	(30,10)	37.8(7.0)	10.2(6.0)
	(30,50)	36.3(6.2)	7.7(4.3)
	(50,30)	77.7(9.9)	6.9(3.2)
	(10,30)	23.3(6.4)	16.8(8.4)
4	(30,10)	87.3(29.4)	26.4(15.9)
	(30,50)	79.8(30.0)	16.8(13.0)
	(50,30)	151.6(28.2)	14.6(7.4)
5	(10,30)	46.3(13.5)	26.5(13.5)
	(30,10)	238.71(165.9)	70.9(65.5)
6	(10,30)	84.9(27.3)	39.4(19.6)
7	(10,30)	154.9(52.9)	55.7(25.6)
8	(10,30)	736.9(917.9)	200.6(237.15)

6. Acknowledgements

This research was supported in part by Grant-in-Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan, B(2) 10450041.

References

1. Almogy, Y. and Levin, O. (1964), Parametric Analysis of a Multi-Stage Stochastic Shipping Problem, *Proc. of the Fifth IFORS Conference*, Venice, 359–370.
2. Al-Khayyol, F. and Falk, J.E. (1983), Jointly Constrained Bi-convex Programming, *Mathematics of Operations Research* 8: 273–286.
3. Androulakis, I.P., Maranas, C.D. and Floudas, C.A. (1995), αBB: A Global Optimization Method for General Constrained Nonconvex Problems, *J. of Global Optimization* 7: 337–363.
4. Cambini, A., Martein, L. and Schaible, S. (1989), On Maximizing the Sum of Ratios, *J. of Information and Optimization Science* 10: 141–151.
5. Charnes, A. and Cooper, W.W. (1962), Programming with Linear Fractional Functions, *Naval Research Logistics* 9: 181–186.
6. Craver, B.D. (1988), *Fractional Programming*, Heldermann-Verlag, Berlin.
7. Falk, J.E. and Polacsay, S.W. (1992), Optimizing the Sum of Linear Fractional Fuctions in (Floudas and Pardalos eds.), *Recent Advances in Global Optimization*, Princeton University Press, 221–258.

8. Hirsche, J. (1995), Optimizing of Sum and Products of Linear Fractional Function under Linear Constraints, Technical Report 3, Department of Computer Science and Scientific Computing, Martin-Luther-Universitat.
9. Horst, R., Pardalos, P.M. and Thoai, N.V. (1995), *Introduction to Global Optimization*, Kluwer Academic Publishers.
10. Horst, R. and Tuy, H. (1996), *Global Optimization: Deterministic Approach* (3rd edn.), Springer-Verlag.
11. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers.
12. Konno, H. and Abe, N. (1999), Minimization of the Sum of Three Linear Fractional Functions, *J. of Global Optimization* 4: 419–432.
13. Konno, H. and Kuno, T. (1992), Linear Multiplicative Programming, *Mathematical Programming* 56: 51–64.
14. Konno, H., Kuno, T. and Yajima, Y. (1994), Global Minimization of a Generalized Convex Multiplicative Function, *J. of Global Optimizaton* 4: 47–62.
15. Konno, H. and Watanabe, H. (1994), Bond Portfolio Optimization Problems and Their Applications to Index Tracking, *J. of the Operation Reseach Social of Japan* 39: 295–306.
16. Konno, H. and Yajima, Y. (1992), Minimizing and Maximizing the Product of Linear Fractional Functions, in Floudas and Pardalos (eds)., Recent Advances in Global Optimization, Princeton University Press, 259–273.
17. Konno, H. and Yamashita, H. (1999), Minimization of the Sum and the Product of Several Linear Fractional Functions, *Naval Research Logistics* 46: 583–596.
18. Kuno, T. and Konno, H. (1991), A Parametric Successive Underestimation Method for Convex Multiplicative Programming Problems, *J. of Global Optimization* 1: 267–286.
19. Matsui, T. (1996), NP-Hardness of Linear Multiplicative Programming and Related Problems, *J. of Global Optimization* 9: 113–119.
20. Phong, T.Q., An, L.T.H. and Tao, P.D. (1995), On Globally Solving Linearly Constrained Indefinite Quadratic Minimization Problems by Decomposition Branch and Bound Method, *Operations Research Letters* 17: 215–220.
21. Schaible, S. (1992), Some Recent Results in Fractional Programming, in P. Mazzolevi (ed.), *Generalized Convexity for Economic Application*, Proc. of the Workshop held in Pisa.
22. Swarup, H. (1966), Programming with Indefinite Quadratic Function with Linear Constraints, *Cahier du Coutre d'Etudes de Recherche Operationelle* 8: 223–234.
23. Tuy, H. (1998), *Convex Analysis and Global Optimization*, Kluwer Academic Publishers.