A Branch-and-Bound Algorithm for Maximizing the Sum of Several Linear Ratios

Takahito Kuno*

September 8, 2000

ISE-TR-00-175

Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
Phone: +81-298-53-5540, Fax: +81-298-53-5206, E-mail: takahito@is.tsukuba.ac.jp

* The author was partly supported by Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture, Grant No. (C2)11650064.

A Branch-and-Bound Algorithm for Maximizing the Sum of Several Linear Ratios

Takahito Kuno*

Institute of Information Sciences and Electronics

University of Tsukuba

September 2000

Abstract. In this paper, we develop a branch-and-bound algorithm for maximizing a sum of p (≥ 2) linear ratios on a polytope. The problem is embedded into a $2p$-dimensional space, in which a concave polyhedral function overestimating the optimal value is constructed for the bounding operation. The branching operation is carried out in a p-dimensional space, in a way similar to the usual rectangular branch-and-bound method. We discuss the convergence properties and report some computational results, which indicate the algorithm is promising.

Key words: Global optimization, nonconvex optimization, fractional programming, sum of linear ratios, branch-and-bound algorithm.

1. Introduction

Since the classical paper [3] by Charnes and Cooper in 1962, intensive research has been done on fractional programs [18]. Fractional programming is one of the most successful fields today in nonlinear optimization. In fact, the linear fractional program which optimizes a single linear ratio has been proved equivalent to a linear program by Charnes and Cooper [3]; and hence it can be solved in polynomial time now using interior-point algorithms [10]. Even in the multi-ratio case, the problem of maximizing the minimum value of linear ratios can be solved quite efficiently using a local search algorithm similar to Newton's method [6]. Unfortunately, however, there is still no decisive method for optimizing a sum of linear ratios on a polyhedron though it is also a multi-ratio problem.

The optimization of a sum of linear ratios arises in various areas: multi-stage stochastic shipping [1], cluster analysis [19] and multi-objective bond portfolio [14], to name but a few. While there is much demand for solution to this problem, all the theoretical results reported so far make us pessimistic about the existence of efficient algorithms [5, 17]. The only thing known about the optimality is that a globally optimal solution lies on the boundary of the feasible set if it exists [5]. For the last decade, however, some promising

*The author was partly supported by Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture, Grant No. (C2) 11650064.

algorithms that use the low-rank nonconvexity [13] have been proposed for the problem with a few ratios [15, 11, 16]. As for the problem where the number of ratios is not limited, Falk and Palocsay suggested an interesting approach in an "image space" [8]. They associated a new variable with each of the ratios and defined the image space, in which optimization is easy in certain directions. Sequentially optimizing in these directions, they yielded a globally optimal solution. In their recent paper [12], Konno and Fukaishi also associated a new variable with each of the ratios, thereby moving nonlinearities into the constraints. They further transformed the ratio constraints to multiplicative ones. To solve the resulting problem, they applied a branch-and-bound algorithm. In a special case that the dimensionality of the problem is fixed at two, Chen et al. very recently developed a remarkably efficient algorithm using computational geometry [4].

In this paper, we will develop a branch-and-bound algorithm for maximizing a sum of p (≥ 2) linear ratios on a polytope. We associate a new variable with each of the denominators and numerators and define a $2p$-dimensional space. We construct a concave polyhedral function overestimating the value of the sum of ratios in this space and compute a lower bound on the optimal value. Therefore, the bounding operation is carried out mainly in the $2p$-dimensional space. In contrast to this, the stage of the branching operation is substantially the p-dimensional image space, i.e., we subdivide the range of each ratio successively in the algorithm. The organization of the paper is as follows. In Section 2, after giving the formal definition of the problem, we embed it into the $2p$-dimensional space. We also explain the outline of the branch-and-bound algorithm there. In Section 3, we construct the function overestimating the value of the sum of ratios. We then show that the lower bound can be computed by solving a linear program. Section 4 is devoted to the branching operation. We discuss the convergence of the algorithm and show that it is guaranteed if we subdivide the range of each ratio according to the the same rules as adopted in the usual rectangular branch-and-bound method for separable concave minimization problems [9, 20]. In Section 5, we summarize the algorithm and prove that it generates a globally ϵ-optimal solution in finite time. Lastly, we report computational results in Section 6.

2. Reduction to $2p$-dimensional problem

Let us consider a problem of maximizing a sum of p linear ratios

$$\left|\begin{array}{ll} \text{maximize} & z = \displaystyle\sum_{i=1}^{p} \frac{\mathbf{d}^i\mathbf{x} + \delta_i}{\mathbf{c}^i\mathbf{x} + \gamma_i} \\ \text{subject to} & A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \end{array}\right. \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c}^i, \mathbf{d}^i \in \mathbb{R}^n$ and $\gamma_i, \delta_i \in \mathbb{R}$ for $i = 1, \ldots, p$. We assume that the feasible set

$$X = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geq \mathbf{0}\}$$

is nonempty and bounded, and that

$$\mathbf{c}^i\mathbf{x} + \gamma_i > 0, \quad \mathbf{d}^i\mathbf{x} + \delta_i > 0, \quad \forall \mathbf{x} \in X, \quad i = 1, \ldots, p. \tag{2.2}$$

As is well known, under condition (2.2) each ratio $(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i)$ is continuous and quasimonotonic on X (i.e., both quasiconvex and quasiconcave; see [2] for details). The sum of quasimonotonic functions is, however, nether quasiconvex nor quasiconcave in general. Therefore, (2.1) can have multiple locally optimal solutions, many of which fail to be globally optimal though at least one exists by compactness of X. What is even worse, no vertex of X might provide a globally optimal solution. This means that vertex enumeration often used in multiextremal global optimization [9] does not work on this problem (2.1).

For convenience, let us introduce two vectors $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, each of p auxiliary variables, and define a $2p$-dimensional set:

$$\Omega = \{(\boldsymbol{\xi}, \boldsymbol{\eta}) \in \mathbb{R}^{2p} \mid \boldsymbol{\xi} = \mathbf{C}\mathbf{x} + \boldsymbol{\gamma},\ \boldsymbol{\eta} = \mathbf{D}\mathbf{x} + \boldsymbol{\delta},\ \mathbf{x} \in X\},$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}^1 \\ \vdots \\ \mathbf{c}^p \end{bmatrix}, \quad \boldsymbol{\gamma} = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_p \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \mathbf{d}^1 \\ \vdots \\ \mathbf{d}^p \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_p \end{bmatrix}.$$

For each $i = 1, \ldots, p$, we also introduce four numbers s_i^1, t_i^1, u_i and v_i satisfying

$$\left.\begin{aligned} 0 < s_i^1 &\le \min\{(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i) \mid \mathbf{x} \in X\} \\ \infty > t_i^1 &\ge \max\{(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i) \mid \mathbf{x} \in X\} \\ 0 < u_i &\le \min\{(\mathbf{c}^i + \mathbf{d}^i)\mathbf{x} \mid \mathbf{x} \in X\} + \gamma_i + \delta_i \\ \infty > v_i &\ge \max\{(\mathbf{c}^i + \mathbf{d}^i)\mathbf{x} \mid \mathbf{x} \in X\} + \gamma_i + \delta_i. \end{aligned}\right\} \tag{2.3}$$

Notice that we can easily obtain each of these numbers by solving a linear programming problem [3]. Let

$$\begin{aligned} \Gamma_i &= \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid u_i \le \xi_i + \eta_i \le v_i\} \\ \Delta_i^1 &= \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i^1\xi_i \le \eta_i \le t_i^1\xi_i\}, \end{aligned}$$

where $\mathbb{R}_+^{\cdot}$ denotes the nonnegative orthant of $\mathbb{R}^{\cdot}$; and let

$$\Gamma = \Gamma_1 \times \cdots \times \Gamma_p, \quad \Delta^1 = \Delta_1^1 \times \cdots \times \Delta_p^1.$$

Since Ω is a subset of $\Gamma \cap \Delta^1$, problem (2.1) reduces to a $2p$-dimensional master problem

$$\text{MP} \quad \left| \begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{p} \eta_i/\xi_i \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega \cap \Gamma \cap \Delta^1. \end{array} \right.$$

We apply a branch-and-bound method to this problem, instead of the original problem (2.1) of dimensionality n.

In our algorithm, while partitioning the cone Δ^1 successively into

$$\Delta^j = \Delta_1^j \times \cdots \times \Delta_p^j, \quad j \in \mathcal{J}, \tag{2.4}$$

we solve each subproblem of MP with a feasible set $\Omega \cap \Gamma \cap \Delta^j$, where

$$\left.\begin{array}{l} \Delta_i^j = \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i^j \xi_i \leq \eta_i \leq t_i^j \xi_j\} \\ \Delta_i^1 = \bigcup_{j \in \mathcal{J}} \Delta_i^j; \quad \Delta_i^j \cap \Delta_i^k = \emptyset \text{ if } j \neq k. \end{array}\right\} \tag{2.5}$$

The outline is as follows:

Let $\mathcal{J} := \{1\}$ and $k := 1$. Repeat Steps 1 – 3 while $\mathcal{J} \neq \emptyset$.

Step 1. Take an appropriate index j from $\mathcal{J}$ and let $\Delta := \Delta^j$. Define a subproblem

$$\mathrm{P}(\Delta) \quad \left| \begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{p} \eta_i / \xi_i \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega \cap \Gamma \cap \Delta. \end{array} \right.$$

Step 2 (bounding operation). Compute an upper bound $\overline{z}(\Delta)$ on the value of $\mathrm{P}(\Delta)$. If $\overline{z}(\Delta)$ is less than or equal to the value of the best feasible solution obtained so far, discard Δ and return to Step 1.

Step 3 (branching operation). Otherwise, divide Δ into two cones Δ^{2k} and Δ^{2k+1}. Add $\{2k, 2k+1\}$ to $\mathcal{J}$ and $k := k+1$.

Needless to say, the efficiency of this algorithm is most influenced by Steps 2 and 3. We will show how to carry them out in order. Throughout the paper, we identify $\mathcal{J}$ with the set of cones Δ^j, $j \in \mathcal{J}$.

3. Bounding operation (Step 2)

The cone Δ defining problem $\mathrm{P}(\Delta)$ is a direct product of p cones, each in a two-dimensional plane:

$$\Delta_i = \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i \xi_i \leq \eta_i \leq t_i \xi_i\}, \quad i = 1, \ldots, p.$$

Therefore, $\Gamma_i \cap \Delta_i$ for each i constitutes a trapezoid with four vertices:

$$\begin{array}{ll} S = (u_i, s_i u_i)/(s_i + 1), & T = (v_i, s_i v_i)/(s_i + 1) \\ U = (v_i, t_i v_i)/(t_i + 1), & V = (u_i, t_i u_i)/(t_i + 1) \end{array}$$

(see Figure 3.1). Let

$$\left.\begin{array}{l} f_i(\xi_i, \eta_i) = (t_i + 1)(\eta_i - s_i \xi_i)/u_i + s_i \\ g_i(\xi_i, \eta_i) = (s_i + 1)(\eta_i - t_i \xi_i)/v_i + t_i. \end{array}\right\} \tag{3.1}$$

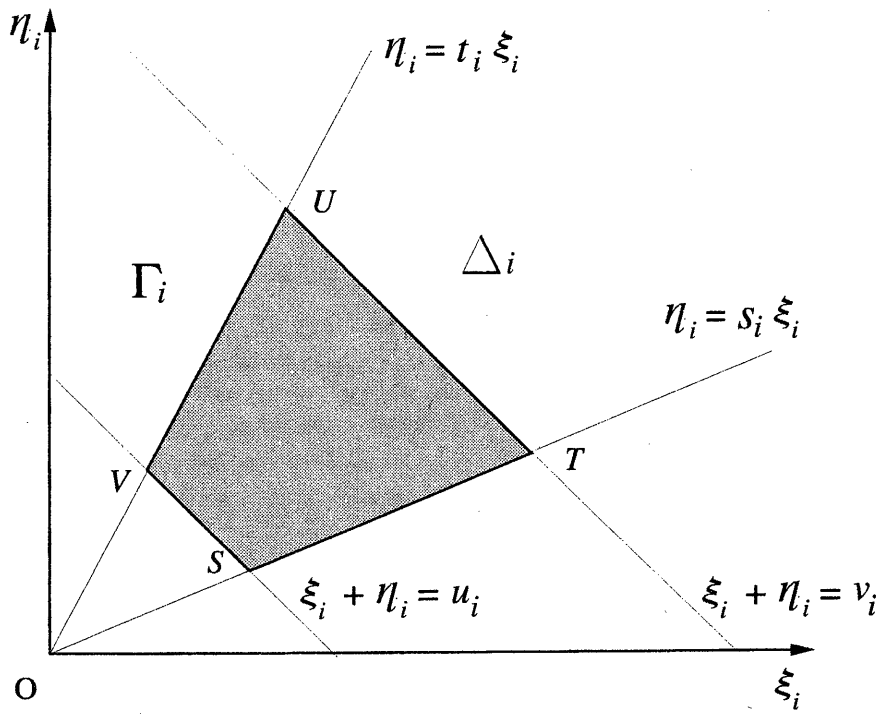


Figure 3.1. Trapezoid $\Gamma_i \cap \Delta_i$

As is shown in Figure 3.2, f_i is an affine function passing η_i/ξ_i at vertex V and side S–T of the trapezoid; and g_i passes it at vertex T and side U–V. Using these two functions, we define

$$\phi_i(\xi_i, \eta_i) = \min\{f_i(\xi_i, \eta_i), g_i(\xi_i, \eta_i)\}. \tag{3.2}$$

Lemma 3.1. *The function ϕ_i is concave, polyhedral and satisfies the following for any $(\xi_i, \eta_i) \in \Gamma_i$:*

$$\left.\begin{array}{ll} \phi_i(\xi_i, \eta_i) \geq \eta_i/\xi_i & \textit{if } (\xi_i, \eta_i) \in \Delta_i \\ \phi_i(\xi_i, \eta_i) < \eta_i/\xi_i & \textit{if } (\xi_i, \eta_i) \notin \Delta_i. \end{array}\right\} \tag{3.3}$$

In particular, the value of ϕ_i agrees with η_i/ξ_i at two sides S–T $(\eta_i/\xi_i = s_i)$ and U–V $(\eta_i/\xi_i = t_i)$ of trapezoid $\Gamma_i \cap \Delta_i$.

Proof: Let us divide $\Gamma_i = \{(\xi_i, \eta_i) \mid u_i \leq \xi_i + \eta_i \leq v_i\}$ by line T–V into

$$\Gamma_i^f = \Gamma_i \cap \{(\xi_i, \eta_i) \mid f_i(\xi_i, \eta_i) \leq g_i(\xi_i, \eta_i)\}$$
$$\Gamma_i^g = \Gamma_i \cap \{(\xi_i, \eta_i) \mid f_i(\xi_i, \eta_i) \geq g_i(\xi_i, \eta_i)\}$$

(see Figure 3.2), and take an arbitrary point (ξ_i', η_i') from Γ_i^f. We then have $\phi_i(\xi_i', \eta_i') = f_i(\xi_i', \eta_i')$. If $\eta_i' = 0$, then (ξ_i', η_i') cannot be a point in Δ_i; and hence the second of (3.3)

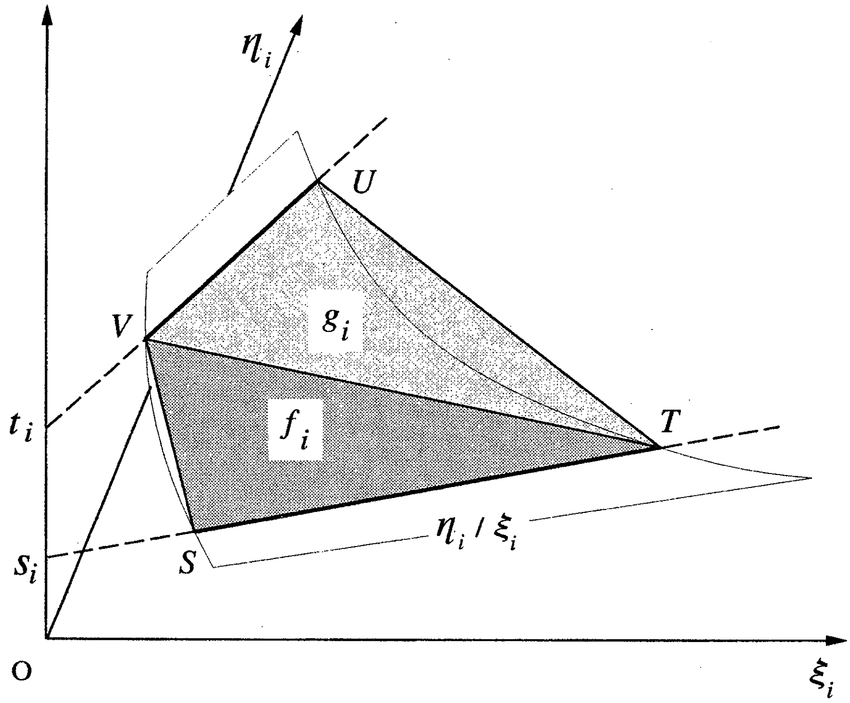


Figure 3.2. Overestimator ϕ_i of η_i/ξ_i

holds. Assuming $\eta_i' \neq 0$, we have

$$\begin{aligned}\phi_i(\xi_i', \eta_i') - \eta_i'/\xi_i' &= (t_i + 1)(\eta_i' - s_i\xi_i')/u_i - (\eta_i' - s_i\xi_i')/\xi_i' \\ &= (\eta_i' - s_i\xi_i')(t_i\xi_i' + \xi_i' - u_i)/(u_i\xi_i').\end{aligned}$$

If $(\xi_i', \eta_i') \in \Delta_i$, then $s_i\xi_i' \leq \eta_i' \leq t_i\xi_i'$ and

$$\phi_i(\xi_i', \eta_i') - \eta_i'/\xi_i' \geq (\eta_i' - s_i\xi_i')(t_i\xi_i' - \eta_i')/(u_i\xi_i') \geq 0.$$

Otherwise, $\eta_i' < s_i\xi_i' \leq t_i\xi_i'$ and

$$\phi_i(\xi_i', \eta_i') - \eta_i'/\xi_i' \leq (\eta_i' - s_i\xi_i')(t_i\xi_i' - \eta_i')/(u_i\xi_i') < 0.$$

Similarly, (3.3) holds for any point in Γ_i^g. The rest follows from definition. ■

We refer to ϕ_i as the *overestimator* of η_i/ξ_i on $\Gamma_i \cap \Delta_i$. Replacing η_i/ξ_i by its overestimator for each i in $\mathrm{P}(\Delta)$, we have a concave maximization problem

$$\overline{\mathrm{P}}(\Delta) \quad \left| \begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{p} \phi_i(\xi_i, \eta_i) \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega \cap \Gamma \cap \Delta. \end{array} \right.$$

Let $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$ be an optimal solution to $\overline{\mathrm{P}}(\Delta)$ and $\overline{z}(\Delta)$ the value of $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$. Also let $z(\Delta)$ denote the optimal value of $\mathrm{P}(\Delta)$. If $\Omega \cap \Gamma \cap \Delta = \emptyset$, we interpret both $z(\Delta)$ and $\overline{z}(\Delta)$ as $-\infty$. The following is an immediate consequence of Lemma 3.1.

Lemma 3.2. *If $\overline{z}(\Delta) > -\infty$, then*

$$\sum_{i=1}^{p} \overline{\eta}_i/\overline{\xi}_i \leq z(\Delta) \leq \overline{z}(\Delta). \tag{3.4}$$

SOLUTION TO $\overline{\mathrm{P}}(\Delta)$

We should remark that $\overline{\mathrm{P}}(\Delta)$ is equivalent to a linear programming problem

$$\left|\begin{array}{lll} \text{maximize} & z = \sum_{i=1}^{p} \zeta_i & \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} & \\ & \left.\begin{array}{l} (t_i + 1)(\mathbf{d}^i - s_i\mathbf{c}^i)\mathbf{x} - u_i\zeta_i \geq \alpha_i \\ (s_i + 1)(\mathbf{d}^i - t_i\mathbf{c}^i)\mathbf{x} - v_i\zeta_i \geq \beta_i \\ s_i \leq \zeta_i \leq t_i \end{array}\right\} & i = 1, \ldots, p, \end{array}\right. \tag{3.5}$$

where

$$\alpha_i = (t_i + 1)(s_i\gamma_i - \delta_i) - s_iu_i, \quad \beta_i = (s_i + 1)(t_i\gamma_i - \delta_i) - t_iv_i.$$

To see this, we need first to prove the following:

Lemma 3.3. *Let (ξ_i, η_i) be a point in Γ_i. Then*

$$(\xi_i, \eta_i) \in \Delta_i \quad \textit{iff} \quad s_i \leq \phi_i(\xi_i, \eta_i) \leq t_i. \tag{3.6}$$

Proof: Let Γ_i^f and Γ_i^g denote the subsets of Γ_i defined in the proof of Lemma 3.1. Then $\Gamma_i^f \cap \Delta_i$ is a triangle with vertices S, T and V (see Figure 3.2). It is easy to check that side S–T and vertex V provide the minimum value s_i and the maximum value t_i, respectively, for $\phi_i = f_i$ on $\Gamma_i^f \cap \Delta_i$. If $(\xi_i, \eta_i) \in \Gamma_i^f \setminus \Delta_i$, then $\xi_i < s_i\eta_i$ and hence $\phi_i(\xi_i, \eta_i) = (t_i + 1)(\eta_i - s_i\xi_i)/u_i + s_i < s_i$. Therefore, (3.6) holds for any $(\xi_i, \eta_i) \in \Gamma_i^f$. Similarly, we can show (3.6) for $(\xi_i, \eta_i) \in \Gamma_i^g$. ∎

Proposition 3.4. *If (3.5) is infeasible, then $\overline{z}(\Delta) = -\infty$. Otherwise, for any optimal solution $(\overline{\mathbf{x}}, \overline{\boldsymbol{\zeta}})$ to (3.5) we have*

$$\overline{\boldsymbol{\xi}} = \mathbf{C}\overline{\mathbf{x}} + \boldsymbol{\gamma}, \quad \overline{\boldsymbol{\eta}} = \mathbf{D}\overline{\mathbf{x}} + \boldsymbol{\delta}, \quad \overline{z}(\Delta) = \sum_{i=1}^{p} \overline{\zeta}_i.$$

Proof: We see from (3.6) that $\overline{\mathrm{P}}(\Delta)$ is equivalent to

$$\left|\begin{array}{lll} \text{maximize} & z = \sum_{i=1}^{p} \zeta_i & \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega & \\ & \left.\begin{array}{l} \zeta_i = \phi_i(\xi_i, \eta_i) \\ s_i \leq \zeta_i \leq t_i \end{array}\right\} & i = 1, \ldots, p. \end{array}\right.$$

This problem reduces to (3.5); and hence the assertion follows. ■

Problem $\overline{\mathrm{P}}(\Delta)$ itself is actually a linear programming problem even if we do not replace the constraint $(\xi_i, \eta_i) \in \Delta_i$ by $s_i \leq \phi_i(\xi_i, \eta_i) \leq t_i$ for each i. This transformation, however, enables one to compute $\overline{z}(\Delta)$ using the upper bounding simplex method and to reduce the total computational time considerably (see [7] for details). Anyway, by solving a linear programming problem, we obtain an upper bound $\overline{z}(\Delta)$ on the value of $\mathrm{P}(\Delta)$ and its feasible solution $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$, both needed in Step 2 of the algorithm.

4. Branching operation (Step 3)

In Step 3, we have to divide Δ in such a way that the resulting sets Δ^{2k} and Δ^{2k+1} satisfy (2.4) and (2.5). This can be done by giving an index $i \in \{1, \ldots, p\}$ and a number $w_i \in [s_i, t_i]$. Namely,

$$\Delta^j = \Delta_1 \times \cdots \times \Delta_{i-1} \times \Delta_i^j \times \Delta_{i+1} \times \cdots \times \Delta_p, \quad j = 2k, 2k+1, \tag{4.1}$$

where

$$\left.\begin{aligned} \Delta_i^{2k} &= \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i\xi_i \leq \eta_i \leq w_i\xi_i\} \\ \Delta_i^{2k+1} &= \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid w_i\xi_i \leq \eta_i \leq t_i\xi_i\}. \end{aligned}\right\} \tag{4.2}$$

In general, no matter how we select i and w_i, the finiteness of the algorithm cannot be guaranteed without a tolerance for the optimal value of problem (2.1). In that case, the algorithm generates an infinite sequence of cones Δ^{j_ℓ}, $\ell = 1, 2, \ldots$ such that

$$\Delta^{j_1} \supset \Delta^{j_2} \supset \cdots, \quad \Omega \cap \Gamma \cap \left(\bigcap_{\ell=1}^{\infty} \Delta^{j_\ell} \right) \neq \emptyset. \tag{4.3}$$

Let us denote Δ^{j_ℓ} simply by Δ^ℓ and the sequence by the index set $\mathcal{L} = \{1, 2, \ldots, \ell, \ldots\}$. We assume that for each $\ell \in \mathcal{L}$, cone $\Delta^{\ell+1}$ is generated from Δ^ℓ via (4.1) and (4.2) for some $i_\ell \in \{1, \ldots, p\}$ and $w_{i_\ell}^\ell \in [s_{i_\ell}^\ell, t_{i_\ell}^\ell]$, where $\Delta_{i_\ell}^\ell = \{(\xi_{i_\ell}, \eta_{i_\ell}) \in \mathbb{R}_+^2 \mid s_{i_\ell}^\ell \eta_{i_\ell} \leq \xi_{i_\ell} \leq t_{i_\ell}^\ell \eta_{i_\ell}\}$. The following lemma shows that $\mathcal{L}$ possesses a property similar to nested rectangles generated by the rectangular branch-and-bound method for separable concave minimization problems (see Lemma 5.4 in [20]):

Lemma 4.1. *There exists an infinite subsequence $\mathcal{L}_q \subset \mathcal{L}$ such that $i_\ell = q$ for all $\ell \in \mathcal{L}_q$. Also, $\{s_q^\ell \mid \ell \in \mathcal{L}_q\}$ and $\{t_q^\ell \mid \ell \in \mathcal{L}_q\}$ have limits s_q^* and t_q^* such that $s_q^* \leq t_q^*$; and $\{w_q^\ell \mid \ell \in \mathcal{L}_q\}$ converges to $w_q^* \in \{s_q^*, t_q^*\}$.*

Proof: Since i_ℓ is an element of the finite set $\{1, \ldots, p\}$, we can take an infinite subsequence $\mathcal{L}_q$ such that $i_\ell = q$ for all $\ell \in \mathcal{L}_q$. Assuming $\mathcal{L}_q = \{1, 2, \ldots\}$ without loss generality, we have

$$s_q^1 \leq s_q^\ell \leq s_q^{\ell+1} \leq t_q^{\ell+1} \leq t_q^\ell \leq t_q^1, \quad \forall \ell \in \mathcal{L}_q.$$

Hence, for some s_q^* and t_q^* such that $s_q^1 \le s_q^* \le t_q^* \le t_q^1$ we have

$$\lim_{\ell\to\infty} s_q^\ell = \lim_{\ell\to\infty} s_q^{\ell+1} = s_q^*, \quad \lim_{\ell\to\infty} t_q^\ell = \lim_{\ell\to\infty} t_q^{\ell+1} = t_q^*.$$

These also imply that $\lim_{\ell\to\infty} w_q^\ell = w_q^* \in \{s_q^*, t_q^*\}$ because w_q^ℓ coincides with either $s_q^{\ell+1}$ or $t_q^{\ell+1}$ for each $\ell \in \mathcal{L}_q$. ■

In the rest of this section, we will give two different rules of selecting $(i_\ell, w_{i_\ell}^\ell)$ to divide Δ^ℓ for each $\ell \in \mathcal{L}$. The sequence $\mathcal{L}$ generated by each of these rules satisfies

$$\lim_{\ell\to\infty} \left[\overline{z}(\Delta^\ell) - \sum_{i=1}^{p} \overline{\eta}_i^\ell / \overline{\xi}_i^\ell \right] = 0, \tag{4.4}$$

where $(\overline{\boldsymbol{\xi}}^\ell, \overline{\boldsymbol{\eta}}^\ell)$ is an optimal solution to $\overline{\mathrm{P}}(\Delta^\ell)$ and $\overline{z}(\Delta^\ell)$ the optimal value. The condition (4.4) is the key to guarantee the finiteness of the algorithm when a positive tolerance is allowed for the optimal value of problem (2.1).

BISECTION

On the analogy of the rectangular branch-and-bound method, the easiest way to divide Δ^ℓ is bisection. For each $\ell \in \mathcal{L}$, let us select

$$i_\ell \in \arg\max\{t_i^\ell - s_i^\ell \mid i = 1, \ldots, p\}; \tag{4.5}$$

and divide $\Delta_{i_\ell}^\ell$ by the line $\eta_{i_\ell} = w_{i_\ell}^\ell \xi_{i_\ell}$ for

$$w_{i_\ell}^\ell = (1-\lambda) s_{i_\ell}^\ell + \lambda t_{i_\ell}^\ell, \tag{4.6}$$

where $\lambda \in (0,1)$ is a constant. We refer to this selection rule of $(i_\ell, w_{i_\ell}^\ell)$ as *bisection* of ratio λ.

Lemma 4.2. *If $\mathcal{L}$ is generated according to the bisection rule of ratio $\lambda \in (0,1)$, then (4.4) holds.*

Proof: As in Lemma 4.1, let $\mathcal{L}_q \subset \mathcal{L}$ denote the infinite sequence where $i_\ell = q$ for all ℓ. Then we have $s_q^\ell \to s_q^*$, $t_q^\ell \to t_q^*$ and $w_q^\ell \to w_q^* \in \{s_q^*, t_q^*\}$ as $\ell \to \infty$ in $\mathcal{L}_q$. From (4.6), however, we have

$$(1-\lambda) s_q^* + \lambda t_q^* = w_q^* \in \{s_q^*, t_q^*\},$$

which holds only if $w_q^* = s_q^* = t_q^*$. This, together with (4.5), implies that if $k \to \infty$ in $\mathcal{L}$, cone Δ^ℓ shrinks to a half-line:

$$\Delta^* = \{(\boldsymbol{\xi}, \boldsymbol{\eta}) \in \mathbb{R}_+^{2p} \mid \xi_i = w_i^* \eta_i,\ i = 1, \ldots, p\}, \tag{4.7}$$

where w_i^* is some pint in $[s_i^1, t_i^1]$.

For each i, the sequence $\{(\overline{\xi}_i^\ell, \overline{\eta}_i^\ell) \mid \ell \in \mathcal{L}\}$ is generated in the compact set $\Gamma_i \cap \Delta_i^1$, and hence has at least one limit point (ξ_i^*, η_i^*), which satisfies $\xi_i^* = w_i^* \eta_i^*$ by (4.7). Therefore,

$$\lim_{\ell\to\infty} \overline{\eta}_i^\ell/\overline{\xi}_i^\ell = w_i^*. \tag{4.8}$$

On the other hand, we have $s_i^\ell \leq \phi_i^\ell(\overline{\xi}_i^\ell, \overline{\eta}_i^\ell) \leq t_i^\ell$ from Lemma 3.3, where ϕ_i^ℓ denotes the overestimator of η_i/ξ_i on $\Gamma_i \cap \Delta_i^\ell$. Hence,

$$\lim_{\ell\to\infty} \phi_i^\ell(\overline{\xi}_i^\ell, \overline{\eta}_i^\ell) = w_i^*. \tag{4.9}$$

Since $\overline{z}(\Delta^\ell) = \sum_{i=1}^p \phi_i^\ell(\overline{\xi}_i^\ell, \overline{\eta}_i^\ell)$, the condition (4.4) follows from (4.8) and (4.9). ■

ω-DIVISION

The bisection rule is simple but does not entirely exploit the characteristics of problem $\overline{\mathrm{P}}(\Delta^\ell)$. As stated in Lemma 3.1, the overestimator ϕ_i^ℓ composing the objective function agrees with η_i/ξ_i on two sides of trapezoid $\Gamma_i \cap \Delta_i^\ell$. The next selection rule of $(i_\ell, w_{i_\ell}^\ell)$ uses this property of ϕ_i^ℓ to fulfill the condition (4.4).

For each $\ell \in \mathcal{L}$, let us select

$$i_\ell \in \arg\max \left\{ \phi_i^\ell(\overline{\xi}_i^\ell, \overline{\eta}_i^\ell) - \overline{\eta}_i^\ell/\overline{\xi}_i^\ell \mid i = 1, \ldots, p \right\}; \tag{4.10}$$

and let

$$w_{i_\ell}^\ell = \overline{\eta}_i^\ell/\overline{\xi}_i^\ell. \tag{4.11}$$

This kind of selection rules is often called *ω-division* in global optimization branch-and-bound methods (see [9]); and we follows the custom.

Lemma 4.3. *If $\mathcal{L}$ is generated according to the ω-division rule, then (4.4) holds.*

Proof: Suppose that $w_q^\ell \to w_q^* = s_q^*$ as $k \to \infty$ in $\mathcal{L}_q \subset \mathcal{L}$, where $\mathcal{L}_q$ is an infinite sequence with $i_\ell = q$ for all $\ell \in \mathcal{L}_q$. Let $\mathcal{L}_q'$ be a subsequence of $\mathcal{L}_q$ such that $w_q^\ell = s_q^{\ell+1}$ for all $\ell \in \mathcal{L}_q'$. Then we have $\overline{\eta}_q^\ell/\overline{\xi}_q^\ell = s_q^{\ell+1}$ from (4.11), and $\phi_q^{\ell+1}(\overline{\xi}_q^\ell, \overline{\eta}_q^\ell) = s_q^{\ell+1}$ from Lemma 3.1. If $\ell \to \infty$ in $\mathcal{L}_q'$, then $\overline{\eta}_q^\ell/\overline{\xi}_q^\ell \to s_q^*$ and $\phi_q^{\ell+1}(\overline{\xi}_q^\ell, \overline{\eta}_q^\ell) \to s_q^*$. Hence, we have

$$\lim_{\ell\to\infty} \left[\phi_q^\ell(\overline{\xi}_q^\ell, \overline{\eta}_q^\ell) - \overline{\eta}_q^\ell/\overline{\xi}_q^\ell \right] = 0. \tag{4.12}$$

Even when $w_q^* = t_q^*$, we have the same result. The condition (4.4) follows from (4.12). ■

5. Description of the algorithm

The last thing to be discussed is how to select a cone Δ^j from the set $\mathcal{J}$ in Step 1. In the usual branch-and-bound methods, either of the following rules is adopted:

Depth first. The set $\mathcal{J}$ is maintained as a list of *stack*. A cone Δ^j is taken from the top of $\mathcal{J}$; and cones Δ^{2k} and Δ^{2k+1} are added in this order to the top.

Best bound. The set $\mathcal{J}$ is maintained as a list of *priority queue*. A cone Δ^j of largest $\overline{z}(\Delta^j)$ is taken out of $\mathcal{J}$.

We can naturally use the either in our algorithm. In addition to this, if we incorporate the bisection or ω-division rule into Step 2, our algorithm is completed.

Let $\epsilon \geq 0$ be a given tolerance for the optimal value of problem (2.1). Then the algorithm is summarized as follows:

algorithm SUMRATIO.
begin
  for $i = 1, \dots, p$ **do begin**
    compute s_i^1, t_i^1, u_i and v_i;
    $\Gamma_i := \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid u_i \leq \xi_i + \eta_i \leq v_i\}$; $\Delta_i := \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i^1 \xi_i \leq \eta_i \leq t_i^1 \xi_i\}$;
  end;
  $\Gamma := \Gamma_1 \times \cdots \times \Gamma_p$; $\Delta^1 := \Delta_1^1 \times \cdots \times \Delta_p^1$; $\mathcal{J} := \{1\}$; $z^\epsilon := 0$; $k := 1$;
  while $\mathcal{J} \neq \emptyset$ **do begin**
    select $j \in \mathcal{J}$ by a fixed rule (*depth first or best bound*); /* *Step 1* */
    $\mathcal{J} := \mathcal{J} \setminus \{j\}$; set $\Delta := \Delta^j$ and define a subproblem $\mathrm{P}(\Delta)$;
    for $i = 1, \dots, p$ **do** /* *Step 2* */
      determine the overestimator ϕ_i of η_i/ξ_i on $\Gamma_i \cap \Delta_i$;
    construct the concave maximization problem $\overline{\mathrm{P}}(\Delta)$ using ϕ_i's;
    solve $\overline{\mathrm{P}}(\Delta)$ to obtain an upper bound $\overline{z}(\Delta)$ on the value of $\mathrm{P}(\Delta)$;
    if $\overline{z}(\Delta) - z^\epsilon > \epsilon$ **then begin** /* *Step 3* */
      let $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$ be a solution of value $\overline{z}(\Delta)$ to $\overline{\mathrm{P}}(\Delta)$;
      if $\sum_{i=1}^p \overline{\eta}_i/\overline{\xi}_i > z^\epsilon$ **then**
        update $z^\epsilon := \sum_{i=1}^p \overline{\eta}_i/\overline{\xi}_i$ and $(\boldsymbol{\xi}^\epsilon, \boldsymbol{\eta}^\epsilon) := (\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$;
      select $i \in \{1, \dots, p\}$ and $w_i \in [s_i, t_i]$ by a fixed rule (*bisection or ω-division*);
      $\Delta_i^{2k} := \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid s_i \xi_i \leq \eta_i \leq w_i \xi_i\}$;
      $\Delta_i^{2k+1} := \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 \mid w_i \xi_i \leq \eta_i \leq t_i \xi_i\}$;
      $\Delta^j := \Delta_1 \times \cdots \Delta_{i-1} \times \Delta_i^j \times \Delta_{i+1} \times \cdots \times \Delta_p$ for $j = 2k, 2k+1$;
      $\mathcal{J} := \mathcal{J} \cup \{2k, 2k+1\}$; $k := k + 1$
    end
  end;
  let $\mathbf{x}^\epsilon$ be a feasible solution to (2.1) such that $\boldsymbol{\xi}^\epsilon = \mathbf{C}\mathbf{x}^\epsilon + \boldsymbol{\gamma}$ and $\boldsymbol{\eta}^\epsilon = \mathbf{D}\mathbf{x}^\epsilon + \boldsymbol{\delta}$
end;

Theorem 5.1. *When $\epsilon > 0$, the algorithm SUMRATIO terminates in finite time and yields a globally ϵ-optimal solution $\mathbf{x}^\epsilon$ to problem (2.1).*

Proof: Let us assume the contrary: the algorithm SUMRATIO is infinite. Then it generates an infinite sequence $\mathcal{L}$ of Δ^{j_ℓ}'s satisfying (4.3). The back tracking criterion $\overline{z}(\Delta) - z^\epsilon > \epsilon$ implies that the following inequalities hold at the end of each iteration in which j^ℓ for $\ell \in \mathcal{L}$ is taken out of $\mathcal{J}$:

$$\sum_{i=1}^{p} \overline{\eta}_i^\ell / \overline{\xi}_i^\ell \leq z^\epsilon < \overline{z}(\Delta^\ell) - \epsilon,$$

where $\Delta^\ell = \Delta^{j_\ell}$. From Lemmas 4.2 and 4.3, however, $\lim_{\ell \to \infty} \overline{z}(\Delta^\ell) - \sum_{i=1}^{p} \overline{\eta}_i^\ell / \overline{\xi}_i^\ell = 0$ whichever rule we adopt for selecting (i, w_i) in Step 3. Therefore, $\epsilon \leq 0$, which is a contradiction. The ϵ-optimality of $\mathbf{x}^\epsilon$ follows from the back tracking criterion. ■

Corollary 5.2. *Suppose $\epsilon = 0$. If the best bound rule is adopted in Step 1, the sequence of $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$'s generated by the algorithm SUMRATIO has limit points, each of which is a globally optimal solution to problem MP.*

Proof: If the algorithm happens to be finite, the assertion is obvious from the back tracking criterion. Assume that it is infinite and generates an infinite sequence $\mathcal{L}$ just stated in the proof of the previous theorem. The best bound rule then implies the following at the beginning of each iteration in which j_ℓ for $\ell \in \mathcal{L}$ is taken out of $\mathcal{J}$:

$$\overline{z}(\Delta^\ell) \geq \overline{z}(\Delta^j) \geq z(\Delta^j), \quad \forall j \in \mathcal{J},$$

where $\Delta^\ell = \Delta^{j_\ell}$. However, $\lim_{\ell \to \infty} \overline{z}(\Delta^\ell) - \sum_{i=1}^{p} \overline{\eta}_i^\ell / \overline{\xi}_i^\ell = 0$; and besides $\max\{z(\Delta^j) \mid j \in \mathcal{J}\}$ is nothing but the optimal value of MP. Hence, every limit point $\{(\overline{\boldsymbol{\xi}}^\ell, \overline{\boldsymbol{\eta}}^\ell) \mid \ell \in \mathcal{L}\}$ is a globally optimal solution to MP. ■

6. Experiment with the algorithm

In this section, we will report computational results of testing the algorithm SUMRATIO on randomly generated problems, which were of the form:

$$\left| \begin{array}{lll} \text{maximize} & z = \displaystyle\sum_{i=1}^{p} \frac{\sum_{j=1}^{n'} d_{ij} x_{ij} + c}{\sum_{j=1}^{n'} c_{ij} x_{ij} + c} & \\ \text{subject to} & \displaystyle\sum_{j=1}^{n'} a_{kj} x_j \leq 1.0, & k = 1, \ldots, m \\ & \qquad x_j \geq 0.0, & j = 1, \ldots, n'. \end{array} \right. \tag{6.1}$$

Data c_{ij}, $d_{ij} \in [0.0, 0.5]$ and $a_{kj} \in [0.0, 1.0]$ were uniformly random numbers. All constant terms of denominators and numerators were the same number c, which ranged between 4.0 and 100.0.

The algorithm was coded in double precision C language according to the description in Section 5. The tolerance ϵ was fixed at 10^{-5}. As to the numbers s_i^1, t_i^1, u_i and v_i, which are not specified in the description, we exploited the structure of (6.1) and determined them by solving a single linear programming problem for each i. First, u_i was set to $2c$ because both $\sum_{j=1}^{n'} c_{ij} x_{ij} + c$ and $\sum_{j=1}^{n'} d_{ij} x_{ij} + c$ have the same minimum value c in (6.1). Then, a linear programming problem was solved to determine the maximum value v_i of $\sum_{i=1}^{n} (c_{ij} + d_{ij}) x_{ij} + 2c$. Finely, s_i^1 and t_i^1 were set to $c/(v_i - c)$ and $(v_i - c)/c$, respectively.

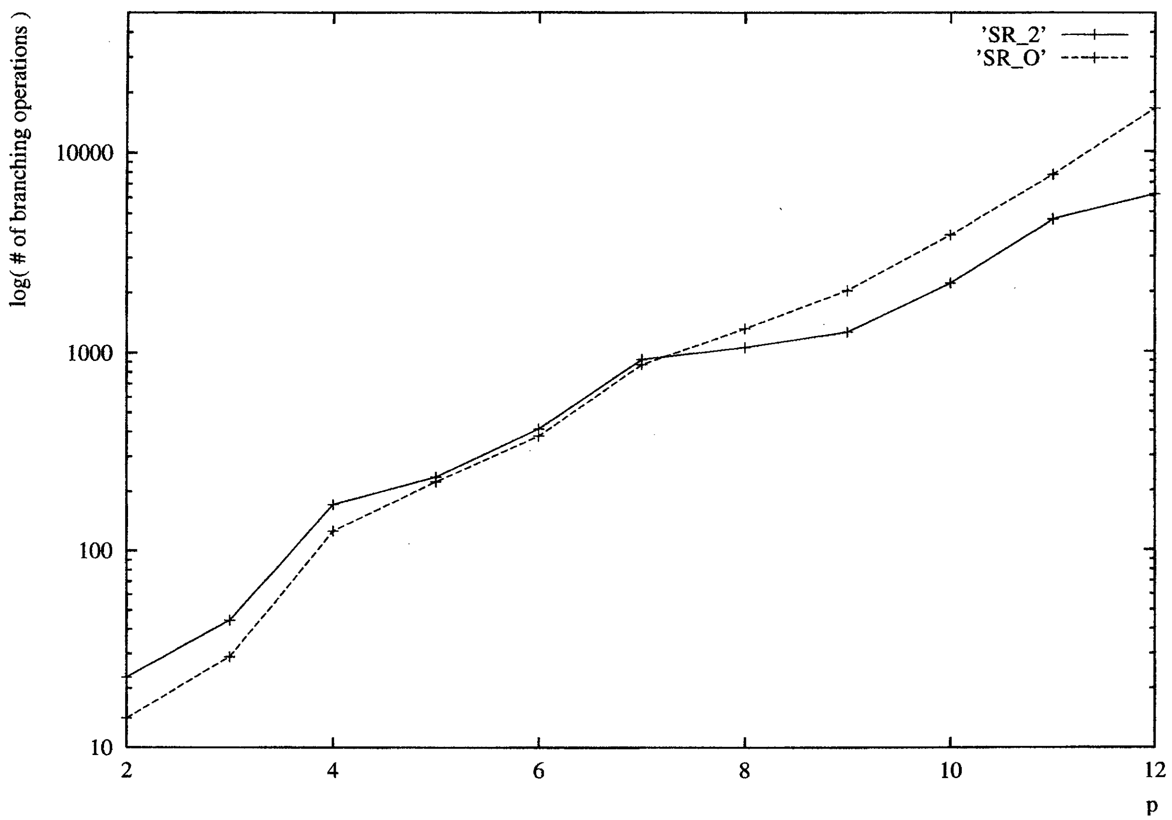
It is easily seen that these s_i^1, t_i^1, u_i and v_i satisfy (2.3), though the resulting $\Gamma \cap \Delta^1$ is somewhat baggy to wrap up Ω. In Step 1, depth first was adopted as the rule for selecting j from $\mathcal{J}$ in order to save on memory. In Step 2, the linear programming problem (3.5) was solved to compute $(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\eta}})$ and $\bar{z}(\Delta)$. Starting from the preceding solution, we restored the primal feasibility of (3.5) by applying some dual simplex pivoting operations. As the rule for dividing Δ in Step 3, we tried both bisection and ω-division. The code adopting the former was named SR_2 and the latter SR_O. Both were tested on a Unix workstation (UltraSPARC-IIi, 440MHz).

COMPUTATIONAL RESULTS

Figure 6.1 depicts the average performance of the algorithm SUMRATIO on ten instances of size $(m, n') = (60, 40)$ for each p when the value of c was fixed at 10.0. The size of p was made to change by 2 each from 2 to 12. We see from the line graph at the top that SR_O requires more branching operations than SR_2 for p greater than 7. This is an unexpected result in comparison with the usual rectangular branch-and-bound method for separable concave minimization problems (see e.g., Remark 5.6 in [20]). As is shown by the graph at the bottom, however, SR_O requires less CPU time than SR_2 for all p except $p = 12$. In the ω-division rule, $(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\eta}})$ always belongs to both Δ^{2k} and Δ^{2k+1}. Therefore, the feasibility of (3.5) can recover quickly whichever cone is chosen from $\mathcal{J}$ in the next iteration. In other words, the less CPU time of SR_O is due to its fewer simplex pivoting operations.

Figure 6.2 gives the results on instances of size $(m, n', p) = (60, 40, 5)$ for ten different values of c from 4.0 to 100.0. These two line graphs show that the algorithm SUMRATIO is very sensitive to the magnitude of c, whether it uses bisection or ω-division. For a small c, each trapezoid $\Gamma_i \cap \Delta_i$ is defined near the origin in the ξ_i-η_i plane. In that case, η_i/ξ_i is quite different from linear in shape; and hence ϕ_i defined only by two affine functions is too simple to estimate η_i/ξ_i precisely. In contrast to this, ϕ_i can make a fine estimate of η_i/ξ_i if $\Gamma_i \cap \Delta_i$ is far away from the origin, i.e., c is a large number. We can recognize from the figure that such a fine estimate is given when c is greater than 20.0.

Based upon the above observations, we tried to solve larger-size problems with c fixed at 10.0 using the ω-division code SR_O. The results are listed in Table 6.1. The columns labeled *branch* and *time* contain the average number of branching operations and the average CPU time in seconds, respectively, required to solve ten instances of size up to $(m, n', p) = (120, 100, 6)$. We see from this table that SR_O is rather insensitive to the size (m, n') and can solve fairly large-size problems as long as p is less than 7. Since we have not yet compared our algorithm with other existing ones, we cannot make a final conclusion. At least for the randomly generated class (6.1), however, these computational results will support our claim that SUMRATIO can serve as a practical deterministic algorithm.



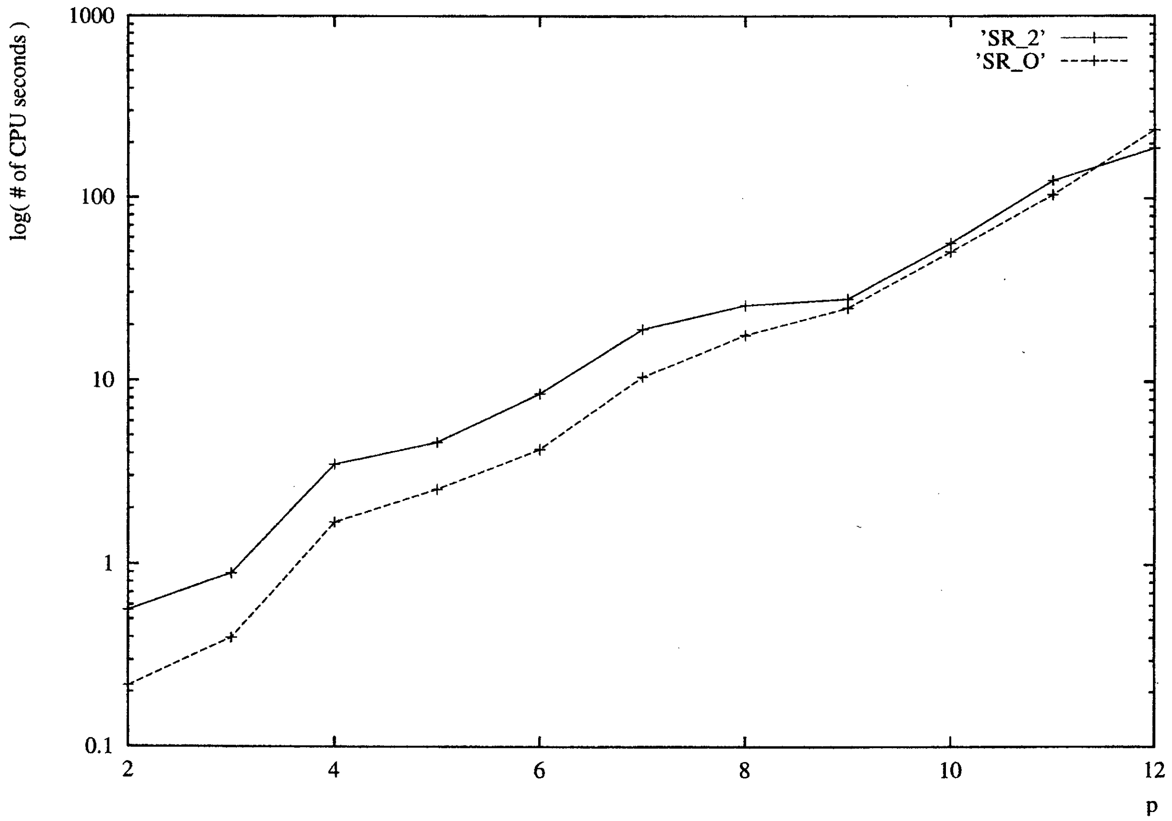


Figure 6.1. Behavior of SUMRATIO when $(m, n') = (60, 40)$ and $c = 10.0$.

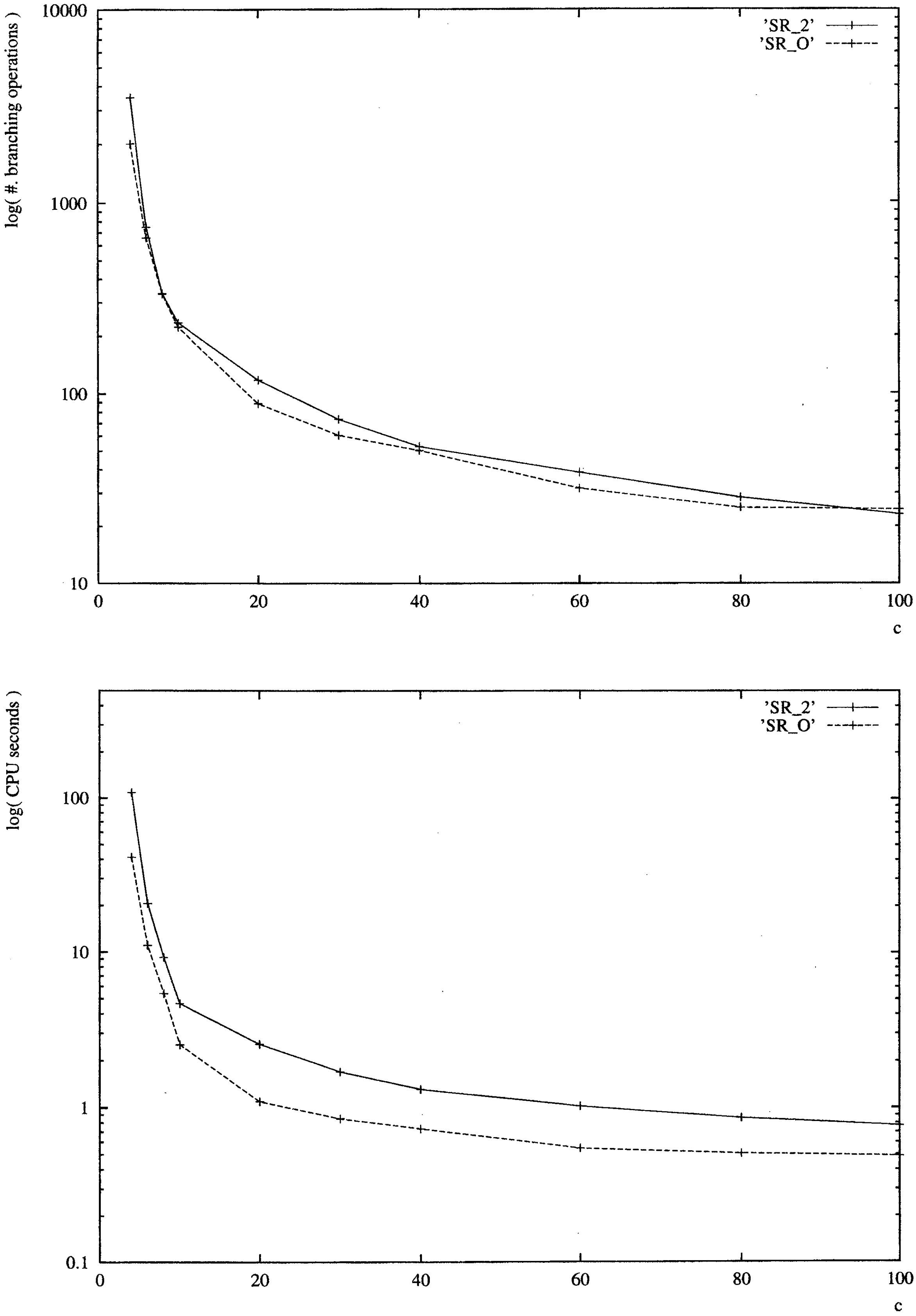


Figure 6.2. Behavior of SUMRATIO when $(m, n', p) = (60, 40, 5)$.

Table 6.1. Computational results of SR_O when $c = 10.0$.

	$p=3$		$p=4$		$p=5$		$p=6$	
$m\times n'$	*branch*	*time*	*branch*	*time*	*branch*	*time*	*branch*	*time*
40×60	38.9	.712	83.4	1.42	232	3.97	291	4.62
80×60	47.9	1.41	76.6	2.15	145	4.05	441	11.3
60×80	56.0	2.27	120	4.58	469	16.3	396	13.1
100×80	76.7	4.47	135	7.52	506	28.2	916	43.5
80×100	73.1	6.67	141	11.3	397	23.0	1,303	92.2
120×100	45.3	5.37	180	16.1	416	33.4	1,389	103

References

[1] Almogy, Y. and O. Levin, "Parametric analysis of a multi-stage stochastic shipping problem", *Proc. of the fifth IFORS Conference* (1964), 359 – 370.

[2] Avriel, M., W.E. Diewert, S. Schaible and I. Zang, *Generalized Convexity*, Plenum Press (N.Y., 1988).

[3] Charnes, A. and W.W. Cooper, "Programming with linear fractional functionals", *Naval Research Logistics Quarterly* **9** (1962), 181 – 186.

[4] Chen, D.Z., O. Daescu, Y. Dai, N. Katoh and W. Xiaodong, "Optimizing the sum of linear fractional functions and applications", *Proc. of the 11th ACM/SIAM Symposium on Discrete Algorithms* (2000), 707 – 716.

[5] Craven, B.D., *Fractional Programming*, Heldermann (Berlin, 1988).

[6] Crouzeix, J.P., J.A. Ferland and S. Schaible, "An algorithm for generalized fractional programs", *Journal of Optimization Theory and Applications* **47** (1985), 35 – 49.

[7] Dantzig, G.B. and M.N. Thapa, *Linear Programming 1: Introduction*, Springer-Verlag (Berlin, 1997).

[8] Falk, J.E. and S.W. Palocsay, "Image space analysis of generalized fractional programs", *Journal of Global Optimization* **4** (1994), 63 – 88.

[9] Horst, R. and H. Tuy, *Global Optimization: Deterministic Approaches*, 2nd ed. Springer-Verlag (Berlin, 1993).

[10] Karmarkar, N., "A new polynomial-time algorithm for linear programming", *Combinatorica* **4** (1984), 373 – 395.

[11] Konno, H. and N. Abe, "Minimization of the sum of three linear fractional functions", *Journal of Global Optimization* **15** (1999), 419 – 432.

[12] Konno, H. and K. Fukaishi, "A branch-and-bound algorithm for solving low rank linear multiplicative and fractional programming problems", Research Report, Tokyo

Institute of Technology (1999).

[13] Konno, H., P.T. Thach and H. Tuy, *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers (Dordrecht, 1997).

[14] Konno, H. and H. Watanabe, "Bond portfolio optimization problems and their applications to index tracking", *Journal of the Operations Research Society of Japan* **39** (1996), 295 – 306.

[15] Konno, H., Y. Yajima and T. Matsui, "Parametric simplex algorithms for solving a special class of nonconvex minimization problems", *Journal of Global Optimization* **1** (1991), 65 – 81.

[16] Konno, H. and H. Yamashita, "Minimization of the sum and the product of several linear fractional functions", *Naval Research Logistics* **46** (1999), 583 – 596.

[17] Schaible, S., "A note on the sum of a linear and linear-fractional function", *Naval Research Logistics Quarterly* **24** (1977), 691 – 693.

[18] Schaible, S., "Fractional programming", in R. Horst and P.M. Pardalos (eds.), *Handbook of Global Optimization*, Kluwer Academic Publishers (Dordrecht, 1995), 495 – 608.

[19] Rao, M.R., "Cluster analysis and mathematical programming", *Journal of the American Statistical Association* **66** (1971), 622 – 626.

[20] Tuy, H., *Convex Analysis and Global Optimization*, Kluwer Academic Publishers (Dordrecht, 1998).