Journal of Global Optimization (2005) 33: 215–234
DOI 10.1007/s10898-004-1952-z

A Revision of the Trapezoidal Branch-and-Bound Algorithm for Linear Sum-of-Ratios Problems

TAKAHITO KUNO*
Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennoh-dai, Tsukuba, 305-8573 Ibaraki, Japan (e-mail: takahito@cs.tsukuba.ac.jp)

(Received: 9 December 2003; accepted: 28 July 2004)

Abstract. In this paper, we point out a theoretical flaw in Kuno [(2002) *Journal of Global Optimization* 22, 155–174] which deals with the linear sum-of-ratios problem, and show that the proposed branch-and-bound algorithm works correctly despite the flaw. We also note a relationship between a single ratio and the overestimator used in the bounding operation, and develop a procedure for tightening the upper bound on the optimal value. The procedure is not expensive, but the revised algorithms incorporating it improve significantly in efficiency. This is confirmed by numerical comparisons between the original and revised algorithms.

Key words: Branch-and-bound algorithm, Fractional programming, Global optimization, Nonconvex optimization, Sum-of-ratios problem

1. Introduction

The sum-of-ratios problem is a class of fractional programming problems and optimizes a sum of multiple ratios of functions over a convex set. During the past 20 years, the interest of researchers and practitioners has gradually shifted to multi-ratios problems of this kind, from the single-ratio problem on which adequate results have already been achieved for both theories and algorithms [21]. Included among those is the problem of maximizing the minimum of ratios, which can be solved rather efficiently using a local search algorithm similar to Newton's method [5]. In contrast to this, the sum-of-ratios problem has not yet been solved rigorously even when the number of ratios is only 15, though various methods have been tested for 20 years.

One of the reasons why the sum-of-ratios problem has attracted attention is that it has a broad range of applications. In light of applications of the single-ratio problem, ratios may be representing profit/capital, profit/cost, return/risk, and so on. The sum-of-ratios problem is the

* The author was partially supported by the Grand-in-Aid for Scientific Research (C)(2) 15560048 from the Japan Society for the Promotion of Science.

handiest approach for optimizing these simultaneously. Therefore, it can easily be imagined that the range of applications is as wide as that of the single-ratio problem. In fact, transportation problems [1], layered manufacturing Problems [18, 23], portfolio problems [14], named only a few, have been formulated into this problem. Another reason is that the sum-of-ratios problem is challenging and intriguing, especially to researchers. Even in the simplest case where the ratios are all linear, their sum is neither quasiconvex nor quasiconcave, though each of them has both properties. As a result, the problem has multiple local maxima, many of which fail to be globally optimal. Since the difficulty of the problem strongly depends on the number of ratios, some of the algorithms proposed so far [8, 11, 15, 16, 19] assume it to be a few and solve this multiextremal optimization problem by exploiting the low-rank nonconcavity [13]. When the number of ratios is not limited, we have to rely on branch-and-bound algorithms [3, 6, 12, 17] at this stage to solve the problem within a practical amount of time. Among others, promising are the algorithms by Kuno [17] and Benson [3], both of which use concave envelopes of ratios on quadrangles to compute upper bounds on the optimal value in the bounding process. In the case of convex functions, it is rather easy to compute tight upper bounds on rectangles or simplices generated by subdividing the feasible set in the branching process. However, the sum of ratios is not convex, as mentioned above. Then Kuno subdivided the projection of the feasible set on each denominator–numerator space into trapezoids and defined a concave envelope over each of them using two affine functions. Benson showed that the similar concave envelope can be defined on a rectangle, as in the usual rectangular branch-and-bound algorithms [10]. Other than branch-and-bound, an interesting approach is the image space analysis proposed by Falk and Palocsay [7]. They associated a new variable with each ratio and defined an ''image space'', in which optimization is easy along the coordinate axes. In their recent paper [9], HoaiPhuong and Tuy elaborated this approach using theory of monotonic optimization [24] and solved a wider class of fractional programming problems. Readers are referred to a recent survey [22] for more details of applications and algorithms.

Kuno reported [17] that his trapezoidal branch-and-bound algorithm can solve the problem with linear ratios even when the number of ratios exceeds ten. However, there is a little theoretical flaw in [17]. In this paper, we will correct it and show that his algorithm works correctly and generates globally optimal solutions even though it was designed based on an incorrect observation. Moreover, we will develop an inexpensive procedure for tightening the upper bound of the trapezoidal algorithm significantly. The organization of the paper is as follows. In Section 2, we present an

outline of the trapezoidal algorithm and show the flaw in [17]. We also re-prove the correctness of the algorithm. In Section 3, we show that the upper bound of the trapezoidal algorithm can be more tightened using a characteristic of ratios, and then propose two revised branch-and-bound algorithms. Section 4 is devoted to a report of computational comparison between the revised algorithms and the original one. In Section 5, we give some concluding remarks.

2. Linear Sum-of-Ratios Problem and the Trapezoidal Algorithm

The problem we consider in this paper is a linear sum-of-ratios problem:

$$\begin{aligned} &\text{maximize} \quad z = \sum_{i=1}^{p} \frac{\mathbf{d}^i\mathbf{x} + \delta_i}{\mathbf{c}^i\mathbf{x} + \gamma_i} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}, \end{aligned} \tag{2.1}$$

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

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

and assume that X is bounded and has a nonempty relative interior. We also assume throughout the paper that for any $\mathbf{x} \in X$,

$$\mathbf{c}^i\mathbf{x} + \gamma_i > 0, \qquad \mathbf{d}^i\mathbf{x} + \delta_i \geqslant 0, \qquad i = 1, \ldots, p. \tag{2.2}$$

Under this condition, each ratio $(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i)$ is pseudomonotonic on X (i.e., pseudoconcave and pseudoconvex; see [2] for detail). The sum of pseudomonotonic functions is, however, neither pseudoconcave nor pseudoconvex, even nor quasiconvex in general. Therefore, (2.1) can have multiple locally optimal solutions, many of which fail to be globally optimal; and besides, no vertex of polytope X might provide a globally optimal solution for (2.1), unlike the usual global optimization problems of maximizing a convex function.

We first run over basic workings of the trapezoidal branch-and-bound algorithm [17] for globally solving this multiextremal global optimization problem (2.1).

2.1. OVERVIEW OF THE TRAPEZOIDAL ALGORITHM

For convenience, let us introduce two vectors $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, each of p auxiliary variables, and define

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

where

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

From (2.2) and the compactness of X, we can select four appropriate numbers s_i, t_i, u_i and v_i for each $i = 1, \ldots, p$ such that

$$0 \leqslant s_i \leqslant \min\{(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i) \mid \mathbf{x} \in X\}$$
$$\infty > t_i \geqslant \max\{(\mathbf{d}^i\mathbf{x} + \delta_i)/(\mathbf{c}^i\mathbf{x} + \gamma_i) \mid \mathbf{x} \in X\}$$
$$0 < u_i \leqslant \min\{(\mathbf{c}^i + \mathbf{d}^i)\mathbf{x} \mid \mathbf{x} \in X\} + \gamma_i + \delta_i$$
$$\infty > v_i \geqslant \max\{(\mathbf{c}^i + \mathbf{d}^i)\mathbf{x} \mid \mathbf{x} \in X\} + \gamma_i + \delta_i.$$

Using these numbers, let us define

$$\Gamma_i = \{(\xi_i, \eta_i) \in \mathbb{R}^2_+ \mid u_i \leqslant \xi_i + \eta_i \leqslant v_i\}$$
$$\Delta_i = \{(\xi_i, \eta_i) \in \mathbb{R}^2_+ \mid s_i\xi_i \leqslant \eta_i \leqslant t_i\xi_i\},$$

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

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

Then (2.1) is reduced to an equivalent $2p$-dimensional problem:

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

The branch-and-bound algorithm proposed in [17] solves (P) recursively while replacing Δ_k by

$$\Delta'_k = \{(\xi_k, \eta_k) \in \mathbb{R}^2_+ \mid s_k\xi_k \leqslant \eta \leqslant w_k\xi_k\}$$
$$\Delta''_k = \{(\xi_k, \eta_k) \in \mathbb{R}^2_+ \mid w_k\xi_k \leqslant \eta \leqslant t_k\xi_k\}$$

for some k and $w_k \in (s_k, t_k)$. The key to efficiency of this kind of algorithms is held by the bounding operation. In [17], it is carried out by solving a relaxed problem of (P).

The objective function of the relaxed problem is a sum of overestimators for η_i/ξ_i's, each of which is defined by two affine functions on the trapezoid $\Gamma_i \cap \Delta_i$ (see Figure 1). Let us denote the four vertices of $\Gamma_i \cap \Delta_i$ by

$$S = (u_i, s_iu_i)/(s_i + 1), \quad T = (v_i, s_iv_i)/(s_i + 1)$$
$$U = (v_i, t_iv_i)/(t_i + 1), \quad V = (u_i, t_iu_i)/(t_i + 1).$$

One of the affine functions, say f_i, is determined to agree with η_i/ξ_i at S, T and V; and the other, say g_i, passes it at T, U and V. Then we have

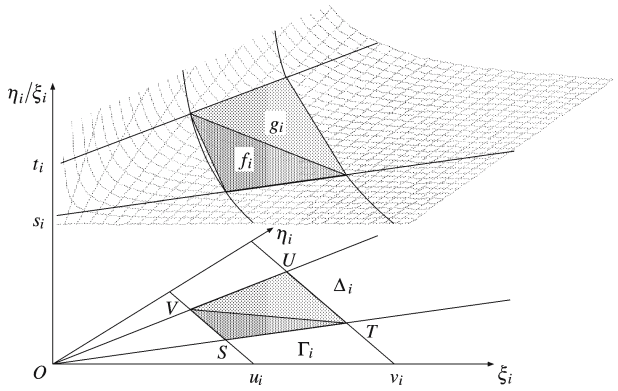

Figure 1. Overestimator ϕ_i of η_i/ξ_i.

$$\begin{aligned} f_i(\xi_i, \eta_i) &= (t_i + 1)(\eta_i - s_i\xi_i)/u_i + s_i \\ g_i(\xi_i, \eta_i) &= (s_i + 1)(\eta_i - t_i\xi_i)/v_i + t_i. \end{aligned} \tag{2.3}$$

The overestimator for η_i/ξ_i is given by the pointwise minimum of these as follows:

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

PROPOSITION 2.1. *Function ϕ_i is concave, polyhedral and satisfies the following for any $(\xi_i, \eta_i) \in \Gamma_i$:*

$$\phi_i(\xi_i, \eta_i) \geqslant \eta_i/\xi_i \quad \textit{if } (\xi_i, \eta_i) \in \Delta_i; \quad \phi_i(\xi_i, \eta_i) < \eta_i/\xi_i \quad \textit{otherwise}.$$

For the proof of Proposition 2.1, see Lemma 3.1 in [17]. In [3], Benson pointed out that ϕ_i is a concave envelope of η_i/ξ_i, i.e., a minimal concave function overestimating the value of η_i/ξ_i, on $\Gamma_i \cap \Delta_i$. Anyway, the sum of concave functions is concave; and hence we have a concave maximization problem, which gives an upper bound of (P):

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

Let $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$ be an optimal solution to $(\overline{\mathrm{P}})$ and let $= \sum_{i=1}^{p} \phi_i(\overline{\xi}_i, \overline{\eta}_i)$, which is regarded as $-\infty$ when $(\overline{\mathrm{P}})$ is infeasible, If $\overline{z}$ is less than or equal to the incumbent value of (2.1), we can discard Δ from further consideration in the branch-and-bound algorithm.

2.2. FLAW IN [17] AND ITS CORRECTION

In [17], it is asserted that $(\overline{\mathrm{P}})$ is equivalent to a linear programming problem:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{p} y_i \\ \text{subject to} & \mathbf{Ax} = \mathbf{b}, \quad \mathbf{x} \geqslant \mathbf{0} \\ & \left.\begin{array}{l} (t_i+1)(s_i\mathbf{c}^i - \mathbf{d}^i)\mathbf{x} + u_i y_i \leqslant \alpha_i \\ (s_i+1)(t_i\mathbf{c}^i - \mathbf{d}^i)\mathbf{x} + v_i y_i \leqslant \beta_i \\ s_i \leqslant y_i \leqslant t_i \end{array}\right\} \quad i = 1, \ldots, p, \end{array} \tag{2.4}$$

where

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

Although $(\overline{\mathrm{P}})$ is certainly a linear programming problem, this assertion is incorrect as is shown by a simple example below,

EXAMPLE 2.1. Consider a three-dimensional problem:

$$\begin{array}{ll} \text{maximize} & \dfrac{x_2+1}{x_1+1} + \dfrac{x_1+1}{x_2+1} \\ \text{subject to} & x_1 + x_2 + x_3 = 1, \quad \mathbf{x} \geqslant 0. \end{array} \tag{2.5}$$

Problem (P) associated with (2.5) is of the form:

$$\begin{array}{ll} \text{maximize} & \eta_1/\xi_1 + \eta_2/\xi_2 \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega, \quad \boldsymbol{\xi}, \boldsymbol{\eta} \geqslant \mathbf{0} \\ & 2 \leqslant \xi_1 + \eta_1 \leqslant 4, \quad 2 \leqslant \xi_2 + \eta_2 \leqslant 4 \\ & (1/3)\xi_1 \leqslant \eta_1 \leqslant 3\xi_1, \quad (1/3)\xi_2 \leqslant \eta_2 \leqslant 3\xi_2, \end{array} \tag{2.6}$$

where

$$\Omega = \left\{ (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \mathbb{R}^4 \,\middle|\, \begin{array}{lll} \xi_1 = x_1 + 1, & \eta_1 = x_2 + 1, & x_1 + x_2 + x_3 = 1 \\ \xi_2 = x_2 + 1, & \eta_2 = x_1 + 1, & \mathbf{x} \geqslant \mathbf{0} \end{array} \right\}.$$

If we apply the trapezoidal algorithm with bisection of ratio 1/2 (the details will be shown later), it solves the following subproblem of (2.6) after seven iterations:

$$\begin{array}{ll} \text{maximize} & \eta_1/\xi_1 + \eta_2/\xi_1 \\ \text{subject to} & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega, \quad \boldsymbol{\xi}, \boldsymbol{\eta} \geqslant \mathbf{0} \\ & 2 \leqslant \xi_1 + \eta_1 \leqslant 4, \quad 2 \leqslant \xi_2 + \eta_2 \leqslant 4 \\ & (1/3)\xi_1 \leqslant \eta_1 \leqslant (2/3)\xi_1, \quad \xi_2 \leqslant \eta_2 \leqslant (4/3)\xi_2. \end{array} \tag{2.7}$$

The relaxed problem of (2.7) is

$$\begin{aligned}
\text{maximize} \quad & \min\left\{ \begin{array}{l} -(5/18)\xi_1 + (5/6)\eta_1 + 1/3 \\ -(2/9)\xi_1 + (1/3)\eta_1 + 2/3 \end{array} \right\} \\
& + \min\left\{ \begin{array}{l} -(7/6)\xi_2 + (7/6)\eta_2 + 1 \\ -(2/3)\xi_2 + (1/2)\eta_2 + 4/3 \end{array} \right\} \\
\text{subject to} \quad & (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega, \quad \boldsymbol{\xi}, \boldsymbol{\eta} \geqslant \mathbf{0} \\
& 2 \leqslant \xi_1 + \eta_1 \leqslant 4, \quad 2 \leqslant \xi_2 + \eta_2 \leqslant 4 \\
& (1/3)\xi_1 \leqslant \eta_1 \leqslant (2/3)\xi_1, \quad \xi_2 \leqslant \eta_2 \leqslant (4/3)\xi_2.
\end{aligned} \tag{2.8}$$

Since $\xi_1 = \eta_2$ and $\xi_2 = \eta_1$ in this particular example, the last two constraints in both (2.7) and (2.8) are inconsistent. However, if we transform (2.8) into the form (2.4), we have

$$\begin{aligned}
\text{maximize} \quad & y_1 + y_2 \\
\text{subject to} \quad & x_1 + x_2 + x_3 = 1, \quad \mathbf{x} \geqslant \mathbf{0} \\
& (5/9)x_1 - (5/3)x_2 + 2y_1 \leqslant 16/9 \\
& (8/9)x_1 - (4/3)x_2 + 4y_1 \leqslant 28/9 \\
& -(7/3)x_1 + (7/3)x_2 + 2y_2 \leqslant 2 \\
& -2x_1 + (8/3)x_2 + 4y_2 \leqslant 14/3 \\
& 1/3 \leqslant y_1 \leqslant 2/3, \quad 1 \leqslant y_2 \leqslant 4/3,
\end{aligned}$$

which is feasible and has an optimal solution $\mathbf{x} = (1/3, 0, 2/3)$ and $\mathbf{y} = (2/3, 4/3)$.

To obtain a linear programming problem actually equivalent to $(\overline{\mathrm{P}})$, we have to replace the constraint $s_i \leqslant y_i \leqslant t_i$ for each i in (2.4) by

$$s_i(\mathbf{c}^i\mathbf{x} + \gamma_i) \leqslant \mathbf{d}^i\mathbf{x} + \delta_i \leqslant t_i(\mathbf{c}^i\mathbf{x} + \gamma_i). \tag{2.9}$$

Then we have

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{p} y_i \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geqslant \mathbf{0} \\
& \left. \begin{array}{l} (t_i + 1)(s_i\mathbf{c}^i - \mathbf{d}^i)\mathbf{x} + u_i y_i \leqslant \alpha_i \\ (s_i + 1)(t_i\mathbf{c}^i - \mathbf{d}^i)\mathbf{x} + v_i y_i \leqslant \beta_i \\ (s_i\mathbf{c}^i - \mathbf{d}^i)\mathbf{x} \leqslant \delta_i - s_i\gamma_i \\ (\mathbf{d}^i - t_i\mathbf{c}^i)\mathbf{x} \leqslant t_i\gamma_i - \delta_i \end{array} \right\} \quad i = 1, \ldots, p.
\end{aligned} \tag{2.10}$$

The equivalence relation between (2.10) and $(\overline{\mathrm{P}})$ follows immediately from the observation that the latter can be rewritten as

$$\begin{aligned} &\text{maximize} \quad z = \sum_{i=1}^{p} y_i \\ &\text{subject to} \quad (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega \cap \Gamma \cap \Delta. \\ &\qquad\qquad y_i \leqslant f_i(\xi_i, \eta_i), \quad y_i \leqslant g_i(\xi_i, \eta_i), \quad i = 1, \ldots, p. \end{aligned}$$

PROPOSITION 2.2. *Problem* (2.10) *is equivalent to* $(\overline{P})$ *in the sense that if* (2.10) *is infeasible, then* $\bar{z} = -\infty$*; otherwise, for any optimal solution* $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ *to* (2.10) *we have*

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

2.3. RECONSIDERATION OF THE TRAPEZOIDAL ALGORITHM

Problem (2.4) has turned out not to be equivalent to $(\overline{\mathrm{P}})$. Then, does the algorithm in [17] which solves (2.4) repeatedly, instead of (2.10), yield incorrect solutions or fail to converge ? – The answer is NO. We should notice that (2.4) is a relaxation of $(\overline{\mathrm{P}})$, and hence provides an upper bound for (P).

PROPOSITION 2.3. *If* (2.4) *is infeasible, then* $= -\infty$*; otherwise, for any optimal solution* $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *to* (2.4) *we have*

$$\overline{z} \leqslant \sum_{i=1}^{p} \tilde{y}_i.$$

Proof. Since any optimal solution $(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\eta}})$ to $(\overline{\mathrm{P}})$ is a point of Ω, there is some $\overline{\mathbf{x}} \in X$ satisfying $\overline{\boldsymbol{\xi}} = \mathbf{C}\overline{\mathbf{x}} + \boldsymbol{\gamma}$ and $\overline{\boldsymbol{\eta}} = \mathbf{D}\overline{\mathbf{x}} + \boldsymbol{\delta}$. Also, for each i, we have either of the following:

$$\phi_i(\overline{\xi}_i, \overline{\eta}_i) = f_i(\overline{\xi}_i, \overline{\eta}_i) \leqslant g_i(\overline{\xi}_i, \overline{\eta}_i), \quad \phi_i(\overline{\xi}_i, \overline{\eta}_i) = g_i(\overline{\xi}_i, \overline{\eta}_i) < f_i(\overline{\xi}_i, \overline{\eta}_i).$$

However, (2.3) implies

$$g_i(\overline{\xi}_i, \overline{\eta}_i) = (s_i + 1)(\overline{\eta}_i - t_i\overline{\xi}_i)/v_i + t_i \leqslant t_i,$$

because $(\overline{\xi}_i, \overline{\eta}_i) \in \Delta_i$ and hence $\overline{\eta}_i \leqslant t_i\overline{\xi}_i$. In both cases, $\phi_i(\overline{\xi}_i, \overline{\eta}_i) \leqslant t_i$ holds. Moreover, Proposition 2.1 implies that $\phi_i(\overline{\xi}_i, \overline{\eta}_i) \geqslant \overline{\eta}_i/\overline{\xi}_i \geqslant s_i$. Therefore, letting $\overline{y}_i = \phi_i(\overline{\xi}_i, \overline{\eta}_i)$, then we have a feasible solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ to (2.4) of value $\overline{z} = \sum_{i=1}^{p} \overline{y}_i$. □

Let us denote by $\tilde{z}$ the optimal value of (2.4) and regard it as $-\infty$ when (2.4) is infeasible. As shown by Proposition 2.3, we have $\tilde{z} \geqslant \overline{z}$; and besides we know that $\overline{z}$ is an upper bound on the optimal value of (P). Therefore,

if $\widetilde{z}$ is less than or equal to the incumbent value of (2.1), we can discard Δ from further consideration. The incumbent can be updated to $\widetilde{\mathbf{x}}$ if necessary, because $\widetilde{\mathbf{x}}$ is a feasible solution to the target problem (2.1). The algorithm works without any trouble as long as we use an exhaustive subdivision rule (see e.g. [10]) such as bisection to divide $\Delta = \Delta_1 \times \cdots \times \Delta_p$. This rule selects $\Delta_k = \{(\xi_k, \eta_k) \in \mathbb{R}^2_+ \mid s_k\xi_k \leqslant \eta_k \leqslant t_k\xi_k\}$ with the longest $[s_k, t_k]$ and sets $w_k = (1-\lambda)s_k + \lambda t_k$ for any fixed ratio $\lambda \in (0,1)$.

As for the ω-division rule, the algorithm seems to fail because an optimal optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ to (2.4) might not satisfy (2.9) for some i. The ω-division rule selects Δ_k such that

$$k \in \arg\max\{\phi_i(\widetilde{\xi}_i, \widetilde{\eta}_i) - \widetilde{\eta}_i/\widetilde{\xi}_i \mid i = 1, \ldots, p\},$$

and sets $w_k = \widetilde{\eta}_k/\widetilde{\xi}_k$, where $\widetilde{\xi}_i = \mathbf{c}^i\widetilde{\mathbf{x}} + \gamma_i$ and $\widetilde{\eta}_i = \mathbf{d}^i\widetilde{\mathbf{x}} + \delta_i$ for each i. Hence, if (2.9) is not satisfied for all i, then w_k deviates from the interval $[s_k, t_k]$ and Δ cannot be divided. In that case, however, Δ contains no feasible solution better than $(\widetilde{\boldsymbol{\xi}}, \widetilde{\boldsymbol{\eta}})$ and can be discarded from further consideration.

PROPOSITION 2.4. *Assume that (2.4)has an optimal solution* $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ *and let* $\widetilde{\boldsymbol{\xi}} = \mathbf{C}\widetilde{\mathbf{x}} + \boldsymbol{\gamma}$ *and* $\widetilde{\boldsymbol{\eta}} = \mathbf{D}\widetilde{\mathbf{x}} + \boldsymbol{\delta}$. *If* $(\widetilde{\boldsymbol{\xi}}, \widetilde{\boldsymbol{\eta}})$ *satisfies*

$$\widetilde{\eta}_i/\widetilde{\xi}_i \notin (s_i, t_i), \quad i = 1, \ldots, p, \tag{2.11}$$

then Δ *contains no feasible solution to* (P) *better than* $(\widetilde{\boldsymbol{\xi}}, \widetilde{\boldsymbol{\eta}})$.

Proof. We first remark that $\widetilde{\eta}_i/\widetilde{\xi}_i \geqslant s_i$ for each i. Otherwise, we have

$$\widetilde{y}_i \leqslant (t_i + 1)(\widetilde{\eta}_i - s_i\widetilde{\xi}_i)/u_i + s_i < s_i,$$

which is inconsistent with the constraint $y_i \geqslant s_i$ in (2.4). Therefore, (2.11) implies

$$\widetilde{\eta}_i/\widetilde{\xi}_i > t_i, \quad i = 1, \ldots, p.$$

We then have $f_i(\widetilde{\xi}_i, \widetilde{\eta}_i) > t_i$ immediately, and

$$\begin{aligned} g_i(\widetilde{\xi}_i, \widetilde{\eta}_i) &= (t_i\widetilde{\eta}_i + \widetilde{\eta}_i - s_it_i\widetilde{\xi}_i - s_i\widetilde{\xi}_i)/u_i + s_i \\ &> (t_i - s_i)(\widetilde{\xi}_i + \widetilde{\eta}_i)/u_i + t_i \geqslant t_i \end{aligned}$$

by noting $\widetilde{\xi}_i + \widetilde{\eta}_i \geqslant u_i$. We see from these that $\widetilde{y}_i = t_i$ holds for each i. On the other hand, from Proposition 2.1, we have $\phi_i(\widetilde{\xi}_i, \widetilde{\eta}_i) < \widetilde{\eta}_i/\widetilde{\xi}_i$ because $(\widetilde{\xi}_i, \widetilde{\eta}_i) \notin \Delta_i$. Consequently, we have

$$\sum_{i=1}^{p} \eta_i/\xi_i \leqslant \widetilde{z} = \sum_{i=1}^{p} t_i < \sum_{i=1}^{p} \widetilde{\eta}_i/\widetilde{\xi}_i$$

for any feasible solution $(\boldsymbol{\xi}, \boldsymbol{\eta})$ to (P). □

As observed above, the trapezoidal algorithm works correctly on (2.1) in spite of the theoretical flaw in [17]. Furthermore, solving (2.4) has the advantage over (2.10) in some respect. Each successor of (P), say (P′) obtained by dividing Δ_k, is different from (P) only in either s_k or t_k. To solve the relaxed problem of (P′), starting from an optimal solution to the relaxed problem of (P), we usually restore its feasibility and then reestablish the optimality, using sensitivity analysis of the simplex algorithm (see e.g. [4]). This process can be done more efficiently on (2.4) because two of the four constraints involving s_k or t_k are just bounding constraints on y_k in (2.4). However, it is also a fact that the upper bound $\tilde{z}$ given by (2.4) is inferior to $\bar{z}$. It will cause relatively rapid growth of branching trees. In the next section, we will discuss a procedure for tightening the upper bound $\tilde{z}$.

3. Tightening of the Upper Bound and its Applications

Let us suppose that the relaxed problem (2.4) has an optimal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, and let $\tilde{\boldsymbol{\xi}} = \mathbf{C}\tilde{\mathbf{x}} + \boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\eta}} = \mathbf{D}\tilde{\mathbf{x}} + \boldsymbol{\delta}$. If $\tilde{\eta}_i/\tilde{\xi}_i \notin (s_i, t_i)$ for each i, we need not tighten the upper bound $\tilde{z} = \sum_{i=1}^{p} \tilde{y}_i$ any more, as seen in Proposition 2.4. Therefore, we assume for some $j \in \{1, \ldots, p\}$ that

$$s_j < \tilde{\eta}_j/\tilde{\xi}_j < t_j.$$

Then we can easily check

$$s_j < f_j(\tilde{\xi}_j, \tilde{\eta}_j), \quad s_j < g_j(\tilde{\xi}_j, \tilde{\eta}_j) < t_j,$$

and see that the value of $\tilde{y}_j$ depends on neither s_j nor t_j directly, but on $\phi_j(\tilde{\xi}_j, \tilde{\eta}_j)$. Let us try improving this upper bound $\phi_j(\tilde{\xi}_j, \tilde{\eta}_j)$ on $\tilde{\eta}_j/\tilde{\xi}_j$ by noting that the value of ξ_j/η_j is constant along the half line defined by $\eta_j = (\tilde{\eta}_j/\tilde{\xi}_j)\xi_j$ (see Figure 2).

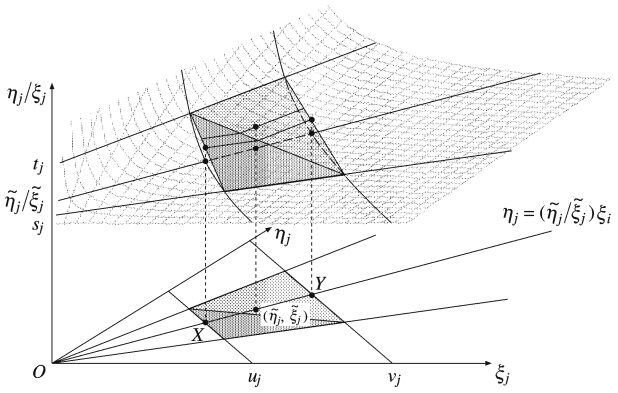

Figure 2. Tightening the upper bound.

Let us denote the intersection points of $\eta_j = (\tilde{\eta}_j/\tilde{\xi}_j)\xi_j$ with $\xi_j + \eta_j = u_j$ and $\xi_j + \eta_j = v_j$ respectively by

$$X = (u_j\tilde{\xi}_j, u_j\tilde{\eta}_j)/(\tilde{\xi}_j + \tilde{\eta}_j), \quad Y = (v_j\tilde{\xi}_j, v_j\tilde{\eta}_j)/(\tilde{\xi}_j + \tilde{\eta}_j).$$

Both the values of η_j/ξ_j at X and Y are $\tilde{\eta}_j/\tilde{\xi}_j$. Also, the values of ϕ_j at X and Y, given by f_j and g_j respectively, coincide as follows:

$$\phi_j(X) = f_j(X) = [(s_j + t_j + 1)\tilde{\eta}_j - s_j t_j \tilde{\xi}_j]/(\tilde{\xi}_j + \tilde{\eta}_j) = g_j(Y) = \phi_j(Y).$$

Since ϕ_j is concave and overestimates η_j/ξ_j on $\Gamma_j \cap \Delta_j$ (Proposition 2.1), we have

$$\phi_j(\tilde{\xi}_j, \tilde{\eta}_j) \geqslant \phi_j(X) = \phi_j(Y) \geqslant \tilde{\eta}_j/\tilde{\xi}_j.$$

If we replace $\tilde{y}_j = \phi_j(\tilde{\xi}_j, \tilde{\eta}_j)$ by $\tilde{y}_j = \phi_j(X)$, the upper bound $\tilde{z} = \sum_{j=1}^{p} \tilde{y}_j$ improves and will suppress the rapid growth of branching trees.

Let

$$\psi_i(\xi_i, \eta_i) = [(s_i + t_i + 1)\eta_i - s_i t_i \xi_i]/(\xi_i + \eta_i), \quad i = 1, \ldots, p.$$

In general, the following relationship holds among ϕ_i, ψ_i and η_i/ξ_i:

PROPOSITION 3.1. *For any* $(\xi_i, \eta_i) \in \Gamma_i$ *we have*

$$\begin{aligned} &\phi_i(\xi_i, \eta_i) \geqslant \psi_i(\xi_i, \eta_i) \geqslant \eta_i/\xi_i \quad \text{if } (\xi_i, \eta_i) \in \Delta_i \\ &\phi_i(\xi_i, \eta_i) < \psi_i(\xi_i, \eta_i) < \eta_i/\xi_i \quad \text{otherwise.} \end{aligned} \tag{3.1}$$

Especially when (ξ_i, η_i) *is an interior point of* $\Gamma_i \cap \Delta_i$, *the first two inequalities hold strictly*.

Proof. Let (ξ_i', η_i') be a point in Γ_i. We have already shown that the first two inequalities hold when (ξ_i', η_i') lies on the interior of Δ_i. If it is a boundary point of Δ_i, both the values of ϕ_i and ψ_i are s_i or t_i. Now, suppose (ξ_i', η_i') is an interior point of $\Gamma_i \cap \Delta_i$ and show that the inequalities hold strictly. There are two cases to consider:

$$f_i(\xi_i', \eta_i') \leqslant g_i(\xi_i', \eta_i') \tag{3.2}$$

$$f_i(\xi_i', \eta_i') > g_i(\xi_i', \eta_i'). \tag{3.3}$$

In case (3.2), we have $\phi_i(\xi_i', \eta_i') = f_i(\xi_i', \eta_i')$ and

$$f_i(\xi_i', \eta_i') - \psi_i(\xi_i', \eta_i') = (t_i + 1)(\eta_i' - s_i\xi_i')(\xi_i' + \eta_i' - u)/[u_i(\xi_i' + \eta_i')] > 0$$

because $s_i\xi_i' < \eta_i'$ and $u_i < \xi_i' + \eta_i'$. Also we have

$$\psi_i(\xi_i', \eta_i') = (\eta_i' - s_i\xi_i')(t_i\xi_i' - \eta_i')/[\xi_i'(\xi_i' + \eta_i')] > 0$$

because $s_i\xi_i' < \eta_i' < t_i\xi_i'$. In case (3.3), we may replace $\phi_i(\xi_i', \eta_i') = f_i(\xi_i', \eta_i')$ by $g_i(\xi_i', \eta_i')$. The last two inequalities in (3.1) can be proved in the same way. □

3.1. REVISED TRAPEZOIDAL ALGORITHM

Let us revise the trapezoidal branch-and-bound algorithm using the procedure ψ_i for tightening the upper bound $\tilde{z}$. We denote by $\epsilon \geqslant 0$ a given tolerance for the optimal value of the target problem (2.1).

```
algorithm TRAPEZOID
begin
  for i = 1, ..., p do begin
      compute s_i, t_i, u_i and v_i;
      Γ_i := {(ξ_i, η_i) ∈ R²_+ | u_i ⩽ ξ_i + η_i ⩽ v_i};
      Δ_i := {(ξ_i, η_i) ∈ R²_+ | s_i ξ_i ⩽ η_i ⩽ t_i ξ_i};
  end;
  Γ := Γ_1 × ··· × Γ_p; Δ := Δ_1 × ··· × Δ_p; D := {Δ}; z^ε := 0;
  while D ≠ do begin
     select Δ ∈ D and set D := D \ {Δ}; define a subproblem (P) with Δ;
     for i = 1, ..., p do
      determine the overestimator φ_i of η_i/ξ_i on Γ_i ∩ Δ_i;
      construct the linear programming problem (2.4) using φ_i's;
      solve (2.4) to obtain an upper bound z̃ on the value of (P);
      if z̃ − z^ε > ε then begin
      set ξ̃ = Cx̃ + γ and η̃ = Dx̃ + δ for an optimal solution (x̃, ỹ) to
     (2.4);
     if Σ_{i=1}^p η̃_i/ξ̃_i > z^ε then
         update z^ε := Σ_{i=1}^p η̃_i/ξ̃_i and x^ε := x̃;
     for i = 1, ..., p do
         if s_i < η̃_i/ξ̃_i < t_i then ỹ_i := ψ_i(ξ̃_i, η̃_i);
     tighten the upper bound to z̃ := Σ_{i=1}^p ỹ_i;
     if z̃ − z^ε > ε then
         select k ∈ {1, ..., p} and w_k ∈ [s_k, t_k];
         Δ'_k := {(ξ_k, η_k) ∈ R²_+ | s_k ξ_k ⩽ η_k ⩽ w_k ξ_k};
         Δ''_k := {(ξ_k, η_k) ∈ R²_+ | w_k ξ_k ⩽ η_k ⩽ t_k ξ_k};
         Δ' := Δ_1 × ··· × Δ'_k × ··· × Δ_p;
         Δ'' := Δ_1 × ··· × Δ''_k × ··· × Δ_p; D := D ∪ {Δ', Δ''}
      end
    end
  end
end;
```

Although the rule for selecting $\Delta \in \mathcal{D}$ is not specified in this description, we can use any one of the usual selection rules in branch-and-bound

algorithms (see e.g. [10]). Typical ones are the depth-first rule, where Δ is taken from the head of the list $\mathcal{D}$ and $\{\Delta', \Delta''\}$ is put back there, and the best-bound rule, where Δ with largest $\tilde{z}$ is taken out of $\mathcal{D}$. In general, the depth-first rule requires less memory than the best-bound rule, but does not guarantee convergence of branch-and-bound algorithms unless the tolerance $\epsilon > 0$. To select $k \in \{1, \ldots, p\}$ and $w_k \in [s_k, t_k]$, we can adopt either bisection or ω-division, as in the original algorithm shown in the previous section. Essentially, algorithm TRAPEZOID has the same structure as the original, except for the tightened upper bound, and hence behaves similarly as follows (see Theorem 5.1 and Corollary 5.2 in [17] for the proofs):

THEOREM 3.2. *When $\epsilon > 0$, algorithm TRAPEZOID terminates after a finite number of iterations and yields a globally ϵ-optimal solution $\mathbf{x}^\epsilon$ to problem (2.1).*

COROLLARY 3.3. *Suppose $\epsilon = 0$. If the best-bound rule is adopted in selecting $\Delta \in \mathcal{D}$, the sequence of $\mathbf{x}^\epsilon$'s generated by algorithm TRAPEZOID has limit points, each of which is a globally optimal solution to problem* (2.1).

3.2. SIMPLIFICATION OF THE ALGORITHM

The procedure ψ_i is based on a simple observation but highly effective in suppressing the growth of branching trees, as will be indicated by numerical results in Section 4. Those suggest an expectation that the algorithm might still work well by means of ψ_i even if we further relax the problem (2.4). Here, instead of (2.4), we propose to solve the following problem in the bounding operation:

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{p} y_i \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0} \\ &\qquad \left.\begin{aligned} &(s_i + 1)(t_i \mathbf{c}^i - \mathbf{d}^i)\mathbf{x} + v_i y_i \leqslant \beta_i \\ &s_i \leqslant y_i \leqslant t_i \end{aligned}\right\} \quad i = 1, \ldots, p. \end{aligned} \tag{3.4}$$

This is associated with a simplification of $(\overline{\mathrm{P}})$:

$$\begin{aligned} (\overline{\mathrm{P}}') \quad &\text{maximize} \ \ z = \sum_{i=1}^{p} g_i(\xi_i, \eta_i) \\ &\text{subject to} \ (\boldsymbol{\xi}, \boldsymbol{\eta}) \in \Omega \cap \Gamma' \cap \Delta, \end{aligned}$$

where

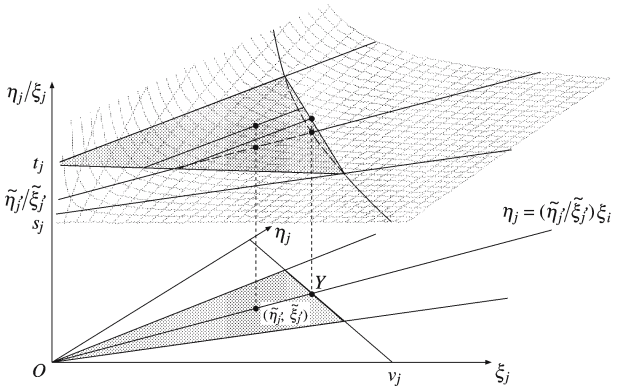

Figure 3. Simplified overestimator.

$$\Gamma_i' = \{(\xi_i, \eta_i) \in \mathbb{R}_+^2 | \xi_i + \eta_i \leqslant v_i\}, \quad \Gamma' = \Gamma_1' \times \cdots \times \Gamma_p'.$$

The reason why we drop f_i from each overestimator $\phi_i = \min\{f_i, g_i\}$ is that f_i is supposed to be less active than g_i in the maximization problem. Since p constraints are removed, (3.4) is easier to solve than (2.4), though the upper bound loosens. The following is an immediate result from the inclusion relation between the feasible sets of (2.4) and (3.4).

PROPOSITION 3.4. *If* (3.4) *is infeasible, then* $\bar{z} = -\infty$*; otherwise, for any optimal solution* $(\tilde{\mathbf{x}}', \tilde{\mathbf{y}}')$ *to* (3.4) *we have*

$$\bar{z} \leqslant \tilde{z} \leqslant \sum_{i=1}^{p} \tilde{y}_i'.$$

Suppose that (3.4) is feasible. Let $\tilde{z}' = \sum_{i=1}^{p} \tilde{y}_i'$, $\tilde{\boldsymbol{\xi}}' = \mathbf{C}\tilde{\mathbf{x}}' + \boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\eta}}' = \mathbf{D}\tilde{\mathbf{y}}' + \boldsymbol{\delta}$ for an optimal solution $(\tilde{\mathbf{x}}', \tilde{\mathbf{y}}')$ to (3.4). Even this loose upper bound $\tilde{z}'$ can be tightened if we replace $\tilde{y}_j'$ by $\psi_j(\tilde{\xi}_j', \tilde{\eta}_j')$ for each j such that $s_j < \tilde{\eta}_j'/\tilde{\xi}_j' < t_j$ (see Figure 3). Since $(\tilde{\xi}_j', \tilde{\eta}_j') \in \Delta_j$, by Proposition 3.1 we have

$$\tilde{y}_j' = g_j(\tilde{\xi}_j', \tilde{\eta}_j') \geqslant \phi_j(\tilde{\xi}_j', \tilde{\eta}_j') \geqslant \psi_j(\tilde{\xi}_j', \tilde{\eta}_j') \geqslant \tilde{\eta}_j'/\tilde{\xi}_j',$$

where the last two inequalities hold strictly if $(\tilde{\xi}_j', \tilde{\eta}_j')$ is an interior point of $\Gamma_j' \cap \Delta_j$. If we use (3.4) in place of (2.4), we can omit computing u_i's in algorithm TRAPEZOID. Since they usually need solving p linear programming problems, preprocessing time will be reduced considerably by this simplification. Some other effectiveness will be shown in the next section.

4. Numerical Experiment

Let us report computational results of having compared algorithm TRAPEZOID and its simplification with the original algorithm in [17].

Those algorithms were tested on problems of varied sizes, each of which was of the following form and generated randomly in the same way as [17]:

$$\begin{aligned} \text{maximize} \quad & z = \sum_{i=1}^{p} \frac{\sum_{j=1}^{n'} d_{ij}x_j + c}{\sum_{j=1}^{n'} c_{ij}x_j + c} \\ \text{subject to} \quad & \sum_{j=1}^{n'} a_{kj}x_j \leqslant 1.0, \quad k = 1, \ldots, m \\ & x_j \geqslant 0.0, \quad j = 1, \ldots, n', \end{aligned} \tag{4.1}$$

where $c_{ij}, d_{ij} \in [0.0, 0.5]$ and $a_{kj} \in [0.0, 1.0]$ are uniformly random numbers. The constant terms of denominators and numerators were all set to the same number c, which ranged between 2.0 and 80.0.

The algorithms were coded using GNU Octave (version 2.0.17) [20], a Matlab-like computational tool, according to the descriptions in the previous section and [17]. The tolerance ϵ needed in the backtracking criterion of each algorithm was fixed at 10^{-5}. As to the initial values of s_i and t_i, we exploited the structure of (4.1) and determined them, together with u_i and v_i, by solving a single linear programming problem for each i. First, u_i was set to $2c$ because both $\sum_{j=1}^{n'} c_{ij}x_j + c$ and $\sum_{j=1}^{n'} d_{ij} + c$ have the same minimum value c in (4.1). Then, a linear programming problem was solved to determine the maximum value v_i of $\sum_{j=1}^{n'}(c_{ij} + d_{ij})x_j + 2c$. Finely, s_i and t_i were set to $c/(v_i - c)$ and $(v_i - c)/c$, respectively. This method is easy to implement and makes no difference in the preprocessing time of three algorithms, though the resulting Δ is somewhat loose to wrap up Ω. As the rule for selecting $\Delta \in \mathcal{D}$, depth first was adopted, and both bisection and ω-devision rules were tried to divide Δ into Δ' and Δ'', in each algorithm. Therefore, a total of six codes were written and run on a computer (Pentium M, 900 MHz) with Linux 2.4.20.

4.1. EFFECT OF CHANGES IN p AND c

It has been reported in [17] that the performance of the trapezoidal branch-and-bound algorithm is strongly affected by changes in p and c. In order to check how much it improves with the procedure ψ_i, we solved (4.16) of size $(m, n') = (60, 40)$ as changing $p \in \{2, 4, \ldots, 10, 11, \ldots, 15\}$ and $c \in \{2.0, 4.0, \ldots, 10.0, 20.0, 40.0, 60.0, 80.0\}$.

Figure 4 shows variation in the average CPU seconds taken to solve ten instances with c fixed at 10.0 for each p using bisection. The results of algorithm TRAPEZOID (*trap*) and its simplified version (*simp*) are represented by the solid and broken lines, respectively; and the dotted line indicates the result of the original algorithm (*orig*). Figure 5 shows the average CPU seconds taken to solve the same instances using ω-division.

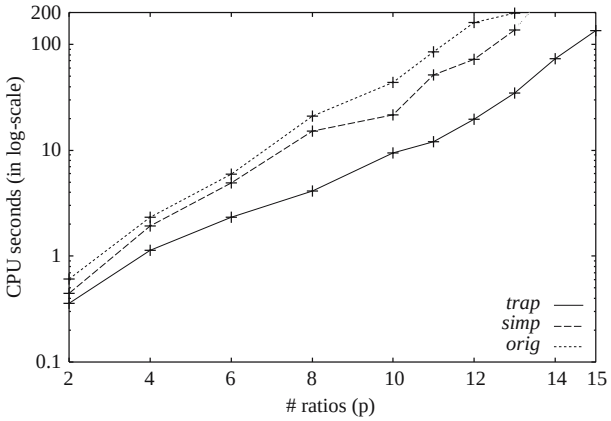

Figure 4. Behavior of algorithms using bisection when $(m, n') = (60, 40)$ and $c = 10.0$.

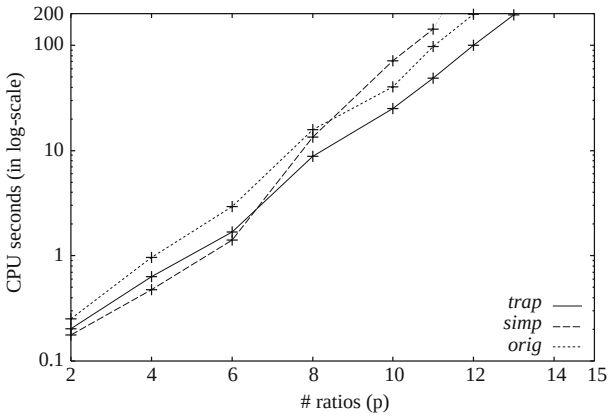

Figure 5. Behavior of algorithms using ω-division when $(m, n') = (60, 40)$ and $c = 10.0$.

Whichever division rule it uses, the computational time of each algorithm increases as an exponential function in p. However, we should notice that algorithm TRAPEZOID requires far less CPU time than the original for every p. Even the simplified version takes less CPU time than the original for each p when using bisection, and for $p \leqslant 6$ when using ω-division.

Figures 6 and 7, respectively, show variation in the average CPU seconds taken to solve ten instances with p fixed at 6 for each value of c, using bisection and ω-division. Since the overestimator ϕ_i is rather poor in estimating η_i/ξ_i near the origin of the ξ_i-η_i space [17], the computational time of the original algorithm increases explosively if the value of c decreases below around ten. We can see from these line plots that this weakness is overcome considerably with the procedure ψ_i. Especially when using ω-division, even the simplified algorithm requires fairly less CPU time than the original for $c \leqslant 20.0$, despite the removal of f_i from the overestimator ϕ_i.

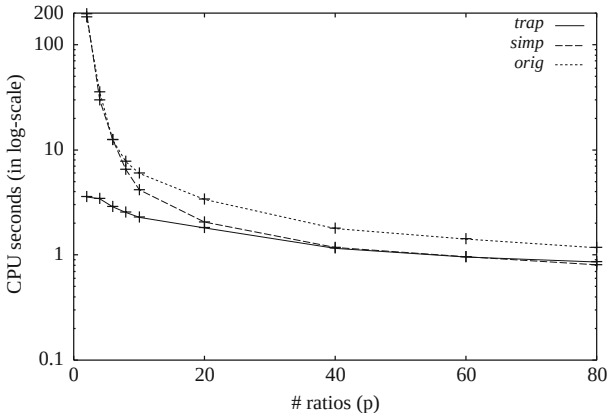

Figure 6. Behavior of algorithms using bisection when $(m, n') = (60, 40)$ and $p = 6$.

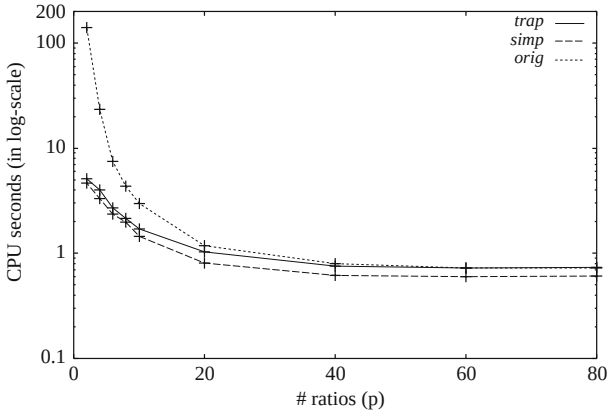

Figure 7. Behavior of algorithms using ω-division when $(m, n') = (60, 40)$ and $p = 6$.

4.2. EFFECT OF CHANGES IN (m, n')

We next compare three algorithms on (4.1) of size (m, n') larger than $(60, 40)$. Except for the above observation, we can study from Figures 4 and 5 that the ω-division rule is more efficient than bisection when p is a small number, say less than eight. Also, we have seen that it is demanding for the original algorithm to solve instances with $c < 10.0$. Therefore, we employed the ω-division rule in each algorithm and solved ten instances with $c = 10.0$ for each $(m, n') \in \{(40, 60), (80, 60), (60, 80), (100, 80), (80, 100), (120, 100)\}$, as changing $p \in \{4, 5, 6, 7\}$.

Table 1 shows the computational result, which contains the average CPU seconds (*sec*) and the average number of branching operations (#) needed by algorithm TRAPEZOID (*trap*), the simplified version (*simp*) and the original algorithm (*orig*) for each (m, n') and p. Both figures of each algorithm increase mildly with increase in the size of (m, n'), in contrast to their rapid change depending on p. As expected, the performance of algorithm

Table 1. Computational results of three algorithms when $c = 10.0$

$m \times n'$		$p = 4$			$p = 5$			$p = 6$			$p = 7$		
		trap	*simp*	*orig*	*trap*	*simp*	*orig*	*trap*	*simp*	*orig*	*trap*	*simp*	*orig*
40×60	*sec*	0.741	0.609	1.180	1.517	1.366	3.006	3.162	3.524	9.654	7.949	9.770	19.13
	#	70.4	78.4	139.8	154.4	199.8	358.8	354.6	570.2	1,115	906.6	1,763	2,327
80×60	*sec*	1.256	0.944	1.907	2.357	1.887	4.805	3.619	3.767	7.462	7.338	9.989	16.03
	#	81.4	77.6	163.2	196.0	221.0	454.4	355.2	511.4	797.0	739.0	1,604	1,657
60×80	*sec*	1.485	1.203	2.697	2.732	2.036	5.764	5.412	5.173	15.34	11.14	24.07	37.40
	#	85.6	98.0	194.4	198.0	193.0	474.0	463.2	680.6	1,353	925.0	2,993	3,112
100×80	*sec*	2.172	1.969	3.681	3.983	3.899	13.86	6.861	9.376	15.57	13.46	20.85	46.07
	#	72.2	83.8	200.0	179.2	229.8	894.0	357.2	705.0	1,055	864.6	1,996	2,897
80×100	*sec*	2.338	2.068	4.094	4.049	3.926	8.077	8.468	10.54	25.78	15.02	30.17	45.88
	#	86.0	89.0	216.4	194.4	257.2	494.0	441.2	807.2	1,512	799.0	2,397	2,536
120×100	*sec*	3.221	2.866	4.722	4.918	4.689	8.671	10.30	14.10	30.85	22.51	35.16	78.17
	#	86.6	81.4	188.8	162.4	230.6	414.6	433.4	891.8	1,383	976.8	2,376	3,731

TRAPEZOID is superior to that of the original for every (m, n') and p. It should be noted that the simplified version takes less CPU time than algorithm TRAPEZOID for each (m, n') when p is less than six. Since the former almost always requires more branching operations than the latter, this is due to the ease of solving the relaxed problem (3.4) compared with (2.4). Unfortunately, the performance of the original algorithm impeded further comparisons on instances of larger (m, n', p). Algorithm TRAPEZOID, however, could solve them rather efficiently unless p exceeds ten.

5. Concluding Remarks

In this paper, we pointed out a theoretical flaw in our previous paper [17]. The linear program (2.4) has been asserted in [17] to be equivalent to the relaxed problem $(\overline{\mathrm{P}})$ of the linear sum-of-ratios problem (P). Actually, it is not equivalent to but is a relaxed problem of $(\overline{\mathrm{P}})$. The trapezoidal branch-and-bound algorithm proposed in [17] solves this incorrect (2.4) repeatedly. Nevertheless, it converges to a correct globally optimal solution to (P), as we proved in Section 2. To tighten the upper bound yielded as the optimal value of (2.4), we proposed the procedure ψ_i, which exploits a relationship between η_i/ξ_i and its overestimator ϕ_i. This is a simple procedure but significantly effective in suppressing the growth of branching trees, as shown in the previous session. We also showed that ψ_i enables us to further relax (2.4) without damaging the performance of the algorithm not so much.

The procedure ψ_i is not only applicable to (2.4), but can also be used for the original relaxed problem $(\overline{\mathrm{P}})$. In that case, since we can expect a still tighter upper bound, the number of branching operations would be further decreased. Also, we could design a procedure similar to ψ_i for improving the overestimator of η_i/ξ_i proposed by Benson [3]. His overestimator is defined on rectangles but using two affine functions like ours. Details of these matters will be reported elsewhere.

Acknowledgment

The author is grateful to Professors L.D. Muu, J. Shi and two anonymous reviewers for their valuable comments, which have greatly improved the earlier version of this paper.

References

1. Almogy, Y. and Levin, O. (1970), Parametric analysis of a multi-stage stochastic shipping problem. In: Lawrence, J. (ed.), *Operational Research '69*, Tavistock Publications, London, 359–370.

2. Avriel, M., Diewert, W.E., Schaible, S. and Zang, I. (1988), *Generalized Convexity*, Plenum Press, New York.
3. Benson, H.P. (2002), Using concave envelopes to globally solve the nonlinear sum of ratios problem, *Journal of Global Optimization* 22, 343–364.
4. Chvátal, V. (1983), *Linear Programming*, Freeman, New York.
5. Crouzeix, J.P., Ferland, J.A. and Schaible, S. (1985), An algorithm for generalized fractional programs, *Journal of Optimization Theory and Applications* 47, 35–49.
6. Dür, R. Horst and Thoai, N.V. Solving sum-of-ratios fractional programs using efficient points, *Optimization* 49, 447–466.
7. Falk, J.E. and Palocsay, S.W. (1994), Image space analysis of generalized fractional programs, *Journal of Global Optimization* 4, 63–88.
8. Freund, R.W. and Jarre, F. (2001), Solving the sum-of-ratios problem by an interior-point method, *Journal of Global Optimization* 19, 83–102.
9. HoaiPhuong, N.T. and Tuy, H. (2003), A unified monotonic approach to generalized linear fractional programming, *Journal of Global Optimization* 26, 229–259.
10. Horst, R. and Tuy, H. (1993), *Global Optimization: Deterministic Approaches*, 2nd ed., Springer-Verlag Berlin.
11. Konno, H. and Abe, N. (1999), Minimization of the sum of three linear fractional functions, *Journal of Global Optimization* 15, 419–432.
12. Konno, H. and Fukaishi, K. A branch-and-bound algorithm for solving low rank linear multiplicative and fractional programming problems.
13. Konno, H., Thach, P.T. and Tuy, H. (1997), *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers Dordrecht.
14. Konno, H. and Watanabe, H. (1996), Bond portfolio optimization problems and their applications to index tracking, *Journal of the Operations Research Society of Japan* 39, 295–306.
15. Konno, H., Yajima, Y. and Matsui, T. (1991), Parametric simplex algorithms for solving a special class of nonconvex minimization problems, *Journal of Global Optimization* 1, 65–81.
16. Konno, H. and Yamashita, H. (1999), Minimization of the sum and the product of several linear fractional functions, *Naval Research Logistics* 46, 583–596.
17. Kuno, T. (2002), A branch-and-bound algorithms for maximizing the sum of several linear fractional functions, *Journal of Global Optimization* 22, 155–174.
18. Majihi, J., Janardan, R., Smid, M. and Gupta, P. (1999), On some geometric optimization problems in layered manufacturing, *Computational Geometry* 12, 219–239.
19. Muu, L.D., Tam, B.T. and Schaible, S. (1995), Efficient algorithms for solving certain nonconvex programs dealing with the product of two affine fractional functions, *Journal of Global Optimization* 6, 179–191.
20. Octave Home Page, http://www.octave.org/.
21. Schaible, S. (1995), Fractional programming. In: Horst, R. and Pardalos, P.M. (eds.), *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht pp. 495–608.
22. Schaible, S. and Shi, J. (2003), Fractional programming: the sum-of-ratios case, *Optimization Methods and Software* 18, 219–229.
23. Schwerdt, J., Smid, M., Janardan, R., Johnson, E. and Majihi, J. Protecting critical facets in layered manufacturing, *Computational Geometry* 16, 187–210.
24. Tuy, H. (2000), Monotonic optimization: problems and solution approaches, *SIAM Journal of Optimization* 11, 464–494.