Journal of Global Optimization (2005) 31:253–270

Conical Partition Algorithm for Maximizing the Sum of dc Ratios

YANG DAI[1],*, JIANMING SHI[2],* and SHOUYANG WANG[3],*
[1]*Department of Bioengineering (MC063), The University of Illinois at Chicago, 851 S. Morgan Street, Chicago, IL 60607-7052, USA (e-mail: yangdai@uic.edu).*[2]*Department of Computer Science and Systems Engineering, Muroran Institute of Technology, 27-1 Mizumoto-cho, Muroran 050-8585, Japan (e-mail: shi@csse.muroran-it.ac.jp).*[3]*Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Zhongguancun, Beijing 100080, China (e-mail: swang@iss04.iss.ac.cn).*

(Received 30 March 2004; accepted in revised form 1 April 2004)

Abstract. The following problem is considered in this paper: $\max_{x\in D}\{\sum_{j=1}^{m} g_j(x)|h_j(x)\}$, where $g_j(x)\geqslant 0$ and $h_j(x)>0,\ j=1,\ldots,m$, are d.c. (difference of convex) functions over a convex compact set D in R^n. Specifically, it is reformulated into the problem of maximizing a linear objective function over a feasible region defined by multiple reverse convex functions. Several favorable properties are developed and a branch-and-bound algorithm based on the conical partition and the outer approximation scheme is presented. Preliminary results of numerical experiments are reported on the efficiency of the proposed algorithm.

AMS Subject Classifications: 90C32, 90C30, 65K05.

Key words: cutting plane, fractional programming, global optimization, outer approximation, sum of ratios.

1. Introduction

Many problems arising in engineering, economics, management science, and other disciplines can be stated as the problem of optimizing a sum of ratios of functions (the sum of ratios problem) [29]. Due to the general importance of this form of optimization, both in theory and in applications, fractional programming has received growing emphasis during the last three decades. One can find the details of this development in references [6, 24, 29] and the corresponding bibliographies appearing therein.

In this paper, we consider the following specific problem:

$$(\mathrm{P})\quad \left|\begin{array}{l} \max\ f(x) := \displaystyle\sum_{j=1}^{m}\frac{g_j(x)}{h_j(x)} \\ \text{s.t. } x\in D, \end{array}\right. \tag{1}$$

*The authors were partially supported by a Grant-in-Aid (Yang Dai: C-13650444; Jianming Shi and Shouyang Wang: C-14550405) of the Ministry of Education, Science, Sports and Culture of Japan.

where $0 \leqslant g_j : R^n \mapsto R$ and $0 < h_j : R^n \mapsto R$ are d.c. (difference of convex) functions for $j = 1, \ldots, m$, and D is a convex, compact, nonempty subset of R^n. It is well known that every function in $\mathcal{C}^2$ is d.c. [13], and several classes of problems in fractional programming can be reduced to (P).

By d.c. function, we mean that $g_j := g_{j_1} - g_{j_2}$ and $h_j := h_{j_1} - h_{j_2}$ for some convex functions $g_{j_k} : R^n \mapsto R$ and $h_{j_k} : R^n \mapsto R$, $j = 1, \ldots, m$, $k = 1, 2$. If $g_{j_2} \equiv 0$ and $h_{j_2} \equiv 0$ for all j, then both g_j and h_j become convex. Moreover, the problem (P) with $m \equiv 1$ and $g_{1_1} \equiv 0$ and $h_{1_2} \equiv 0$ is called the concave single-ratio fractional (CSF) programming problem. Although there exists an abundance of publications on the study of fractional programming, most of them are concentrated on the CSF problem, especially on the linear case. The problem (P) considered here is much more difficult to treat than the single-ratio problem.

A canonical strategy for solving a single-ratio problem $\max\{g(x)/h(x) | x \in D\}$ is the so-called parametric method, a procedure which attempts to find the root λ^* of the following equation:

$$\pi(\lambda) := \max_{x \in D}(g(x) - \lambda h(x)) = 0.$$

The validity of this method is based on the fact that λ^* is the optimal value of (P) if and only if $\pi(\lambda^*) = 0$. It is easy to see that the function $\pi(\lambda)$ has the following properties: $\pi(\lambda)$ is continuous; $\pi(\lambda)$ is nonincreasing and convex, and there exist two positive points λ_- and λ_+ such that $\pi(\lambda_-) \leqslant 0$ and $\pi(\lambda_+) \geqslant 0$. With these properties and under certain conditions, algorithms for the determination of the root of $\pi(\lambda) = 0$ using, for example, the bisection and the Newton-like methods can be designed. Many results on this scheme can be found in [8, 14, 19, 28], and details dealing with the quadratic case are given in [15, 26]. Notwithstanding these results, the method of solution cited above can not be extended to the sum of ratios problem [9].

The sum of ratios problem has been studied by many researchers; this group includes Cambini et al. [2], Chen et al. [4], Falk and Palocsay [9], Konno and Kuno [16], Konno and Yamashita [17], Kuno [18], Ritter [20], and others. However, most of the corresponding studies are confined to the linear case, namely, the sum of linear ratios with linear constraints. Recently, Freund and Jarre [10] proposed an interior-point approach for the convex–concave ratios with convex constraints. However, since their algorithm is based on the structure of the convex–concave ratios, it is difficult to apply that method directly to our problem.

In their paper [3], Charnes and Cooper introduced an epi-multiple transformation for a linear fractional programming problem. Schaible adapted it to solve the concave single-ratio case [22, 23]. After transformation, the objective function becomes concave, while the feasible region remains convex. Therefore, the concave fractional programming is transformed into a corresponding concave programming, thereby rendering it solvable by concave programming methods. This approach could be further exploited to solve our problem, however, it requires the introduction of an excessively large number of new variables in order to give an equivalent representation of the problem.

The goal of this research is two-fold. First, we present a transformation of the problem based on an equivalent d.c. representation of the ratios. The properties of the d.c. constraints are further studied, and they enable us to recast the original problem in terms of maximizing a linear objective function over a convex set with several reverse convex constraints in an $(n + 6m)$-dimensional space. Second, we propose a branch-and-bound algorithm based on the conical partition and the outer approximation. On the basis of an appropriate decomposition of the structure of the problem, we are able to perform the conical partition in the $(n + 6)$-dimensional space, a process that requires much less computational effort comparing to the conical partition in the $(n + 6m)$-dimensional space.

The remainder of the paper is organized as follows. Section 2 introduces the method for the transformation mentioned above. In Section 3, we review the algorithm for the linear programming with several reverse convex constraints and present the modified algorithm to the problem (P) with a polytope D. We develop a solution method in Section 4 and based on the method, Section 5 describes our new algorithm based on the conical partition and the outer approximation method; it also discusses its validity and convergence. Section 6 gives an illustrative example for the proposed algorithm and indicates the implementation information in detail. A brief conclusion is provided in Section 7 which outlines some potential extensions.

2. The Equivalent Transformation

Throughout this paper, for the sake of convenience in our discussion, we assume that the feasible region D of the problem (P) is defined by a convex function d, i.e., $D := \{x \in R^n | d(x) \leqslant 0\}$. Without loss of generality, we assume further $g_j(x) \geqslant 0$ for $j = 1, \ldots, m$. In fact, since $h_j(x) > 0$ for all $x \in D$, $j = 1, \ldots, m$, and

$$\max \sum_{j=1}^{m} \frac{g_j(x)}{h_j(x)} \Longleftrightarrow \max \sum_{j=1}^{m} \frac{g_j(x)}{h_j(x)} + M_j \Longleftrightarrow \max \sum_{j=1}^{m} \frac{g_j(x) + M_j h_j(x)}{h_j(x)},$$

by choosing a sufficiently large value $M_j > 0$, the numerators $g_j(x) + M_j h_j(x)$ are always nonnegative. By introducing three extra variables y_j, z_j and s_j, $j = 1, \ldots, m$, we rewrite (P) as follows:

$$(\mathrm{P})\quad \left|\begin{array}{lll} \max & \sum_{j=1}^{m} s_j & \\ \text{s.t.} & g_j(x) \geqslant y_j, & \text{for } j = 1, \ldots, m, \\ & h_j(x) \leqslant z_j, & \text{for } j = 1, \ldots, m, \\ & \frac{y_j}{z_j} \geqslant s_j, & \text{for } j = 1, \ldots, m, \\ & d(x) \leqslant 0. & \end{array}\right. \tag{2}$$

Note that for $z_j > 0$,

$$\frac{y_j}{z_j} \geqslant s_j \Longleftrightarrow 2y_j \geqslant (z_j + s_j)^2 - (z_j^2 + s_j^2)$$

for $j = 1, \ldots, m$.

Assume that $g_j(x) \equiv g_j'(x) - g_j''(x)$ and $h_j(x) \equiv h_j'(x) - h_j''(x)$ for some convex functions $g_j'(x), g_j''(x), h_j'(x)$ and $h_j''(x)$. Then (2) can be written in the following form.

$$(\mathrm{P})\quad \left|\begin{array}{lll} \max & \sum_{j=1}^{m} s_j & \\ \text{s.t.} & g_j'(x) - g_j'' \geqslant y_j, & \text{for } j = 1, \ldots, m, \\ & h_j'(x) - h_j''(x) \leqslant z_j, & \text{for } j = 1, \ldots, m, \\ & 2y_j - (z_j + s_j)^2 + z_j^2 + s_j^2 \geqslant 0, & \text{for } j = 1, \ldots, m, \\ & d(x) \leqslant 0. & \end{array}\right. \tag{3}$$

This program has a linear objective function with $(3m + 1)$ constraints and $(n + 3m)$ variables. In his paper [27], Shi considered the following epi-multiple functions:

$$G(x, \lambda) := \begin{cases} \lambda g(\lambda^{-1} x) & \text{if } \lambda > 0, \\ 0 & \text{if } \lambda = 0,\ x = 0, \\ -\infty & \text{otherwise} \end{cases} \tag{4}$$

in order to rewrite the objective function $\frac{g(x)}{h(x)}$ as a convex function for single-ratio fractional programming. The transformation (2) is considerably less complex than the epi-multiple-based method proposed in [27], a procedure which needs $(n + 3)m$ variables for a similar d.c. representation. In (3), the objective function is linear and the constraints are of d.c. type. Therefore, it is a d.c. optimization problem, i.e., the maximization of a linear function over a compact d.c. set. In fact, the introduction of the new variables u_j, v_j and w_j allows the restatement of (3) as follows:

$$
(\mathrm{P})\quad\left|\begin{array}{lll}
\max & \sum_{j=1}^{m} s_j & \\
\text{s.t.} & g_j'(x) - u_j - y_j \geqslant 0, & \text{for } j = 1, \ldots, m, \\
& g_j''(x) - u_j \leqslant 0, & \text{for } j = 1, \ldots, m, \\
& h_j'(x) - v_j - z_j \leqslant 0, & \text{for } j = 1, \ldots, m, \\
& h_j''(x) - v_j \geqslant 0, & \text{for } j = 1, \ldots, m, \\
& z_j^2 + s_j^2 + 2y_j - w_j \geqslant 0, & \text{for } j = 1, \ldots, m, \\
& (z_j + s_j)^2 - w_j \leqslant 0, & \text{for } j = 1, \ldots, m, \\
& d(x) \leqslant 0. &
\end{array}\right. \tag{5}
$$

All functions on the left-hand side of the constraints of (5) are convex. Especially, the statements $g_j'(x) - u_j - y_j \geqslant 0, h_j''(x) - v_j \geqslant 0$ and $z_j^2 + s_j^2 + 2y_j - w_j \geqslant 0$ for $j = 1, \ldots, m$ are called *reverse convex constraints*;. The feasible region of (5) consists of $(3m + 1)$ convex constraints and $3m$ reverse convex constraints. Accordingly, we have a total of $(n + 6m)$ variables. Define that

$$
\begin{aligned}
G_j^- &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | g_j'(x) - u_j - y_j < 0\}, \\
G_j &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | g_j''(x) - u_j \leqslant 0\}, \\
H_j^- &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | h_j''(x) - v_j < 0\}, \\
H_j &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | h_j'(x) - v_j - z_j \leqslant 0\}, \\
\Gamma_j^- &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | z_j^2 + s_j^2 + 2y_j - w_j < 0\}, \\
\Gamma_j &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | (z_j + s_j)^2 - w_j \leqslant 0\}, \\
D &:= \{(x, y_j, z_j, s_j, u_j, v_j, w_j) \in R^{n+6} | d(x) \leqslant 0\}.
\end{aligned} \tag{6}
$$

By using (6), we can rewrite (5) as

$$
(\mathrm{P})\quad\left|\begin{array}{ll}
\max & \sum_{j=1}^{m} s_j \\
\text{s.t.} & (x, y_j, z_j, s_j, u_j, v_j, w_j) \in G_j \cap H_j \cap \Gamma_j \cap D, \\
& (x, y_j, z_j, s_j, u_j, v_j, w_j) \notin G_j^- \cup H_j^- \cup \Gamma_j^-, \\
& \text{for } j = 1, \ldots, m.
\end{array}\right. \tag{7}
$$

For the simplicity, we denote $\varpi_j := (y_j, z_j, s_j, u_j, v_j, w_j)$ and $\varpi_j^* := (y_j^*, z_j^*, s_j^*, u_j^*, v_j^*, w_j^*)$ for all j. For a set S, denote the boundary of S by ∂S. Under removal of the reverse constraints, problem (7) becomes a convex program, which is polynomial solvable. Therefore, without loss of generality, we assume that

(A1) there exist an index $j_0 \in \{1, \ldots, m\}$ and a point $(x^\star, \varpi_1^\star, \ldots, \varpi_m^\star)$ with $(x^\star, y_j^\star, z_j^\star, s_j^\star, u_j^\star, v_j^\star, w_j^\star) \in (G_j \cap H_j \cap \Gamma_j \cap D) \cup (G_{j_0}^- \cup H_{j_0}^- \cup \Gamma_{j_0}^-)$ for all j

satisfying $\sum_j^m s_j < \sum_j^m s_j^*$ for any $(x, \varpi_1, \ldots, \varpi_m)$ in the feasible region of (7).

The next lemma plays hereafter a crucial role in our algorithm.

LEMMA 1. *If problem (7) has an optimal solution* $(x^*, \varpi_1^*, \ldots, \varpi_m^*)$ *then* $(x^*, \varpi_j^*) \in \partial(G_j^- \cup H_j^- \cup \Gamma_j^-)$ *for* $j = 1, \ldots, m$.

Proof. Suppose that $(x^*, \varpi_1^*, \ldots, \varpi_m^*)$ is an optimal solution of problem (7) and that $(x^*, \varpi_{j_0}^*) \notin \partial(G_{j_0}^- \cup H_{j_0}^- \cup \Gamma_{j_0}^-)$ for some j_0.
Consider a point

$$p(t) := (x^*, \varpi_1^*, \ldots, \varpi_m^*) + t[(x^*, \varpi_1^*, \ldots, \varpi_m^*) - (x^*, \varpi_1^*, \ldots, \varpi_m^*)]$$

for $0 \leqslant t \leqslant 1$. From the assumption that $p(1) \in (G_{j_0}^- \cup H_{j_0}^- \cup \Gamma_{j_0}^-)$ and $p(0) \notin (G_{j_0}^- \cup H_{j_0}^- \cup \Gamma_{j_0}^-)$, there must exist some $t_0 \in (0, 1)$ such that $p(t_0) \in \partial(G_{j_0}^- \cup H_{j_0}^- \cup \Gamma_{j_0}^-)$, and $p(t_0) \neq p(0)$. Denote the objective function value of problem (7) at $p(t)$ by $F(p(t))$. Since the objective function is linear, we have

$$F(p(0)) < F(p(t_0)).$$

This contradicts the fact that $F(p(0))$ is the optimal value of problem (7). □

3. Linear Programming with Several Reverse Convex Constraints

In this section, we review a method of global optimization proposed by Dai et al. [7] for the solution of a linear programming problem with several reverse convex constraints. Consider the problem $\max\{bx | x \in D, f_j(x) \geqslant 0, \ j = 1, \ldots, m\}$, where D is a polytope in R^n and $f_j(x)$, $j = 1, \ldots, m$, are convex. Define $C_j = \{x | f_j(x) < 0, \ j = 1, \ldots, m\}$. Then C_j, $j = 1, \ldots, m$, are open and convex. This problem can be represented as follows.

$$(\mathrm{P}) \quad \left| \begin{array}{ll} \max & bx \\ \text{s.t.} & x \in D, \\ & x \notin \cup_{j=1}^m C_j. \end{array} \right. \tag{8}$$

If there exists an interior point x_0 of the set $\cap_{j=1}^m C_j$, then the approach proposed in [7] can be used to approximate the feasible region by a concavity cut reduction. The algorithm is a combination of the conical branch-and-bound scheme and the concavity cut reduction. In order to use the polyhedral inner approximation and the conical partition, we enlist the following two assumptions:

(**A2**) int $(D \setminus \cup_{j=1}^{m} C_j) \neq \emptyset$.

(**A3**) $\cap_{j=1}^{m} C_j \neq \emptyset$ and a point $x^0 \in \cap_{j=1}^{m} C_j$ is available.

Let x^0 and $\text{Ray}^i = \{r_v^i | v = 1, \dots, n\}$ be $n+1$ affinely independent points, we define

$$\text{cone}(x^0; \text{Ray}^i) := \left\{ x \middle| x = \sum_{v=1}^{n} \alpha_v^i (r_v^i - x^0) + x^0, \alpha_v^i \geqslant 0 \right\}.$$

or cone(Ray^i) or c_i simply.

Before describing the algorithm, we establish certain notations. Let $\mathcal{C}$ be a collection of cones $\{c_1, \dots, c_p\}$ as defined above. Such a collection $\mathcal{C}$ is called a *conical partition* of a given set D if $\cup_{i=1}^{p} (c_i \cap D) = D$ and $\text{int}(c_i) \cap \text{int}(c_j) = \emptyset$ for all $i \neq j$. We say a partition $\mathcal{C}'$is a refinement of $\mathcal{C}$ if for any cone $c' \in \mathcal{C}'$ there exists a cone $c \in \mathcal{C}$ such that $c' \subset c$. A refinement process is called *exhaustive* if for every strictly nested sequence $\{c_k\}_{k=1,2,\dots,}$ satisfying $c_k \in \mathcal{C}_k$ and $c_{k+1} \subset c_k$ for every k, there exists a half-line $\mathcal{R}$ emanating from x^0 such that

$$\{x \in R^n | \liminf_{k\to\infty} \delta(x, c_k) = 0\} = \{x \in R^n | \lim_{k\to\infty} \delta(x, c_k) = 0\} = \mathcal{R}, \tag{9}$$

where $\delta(x, c_k) := \inf_{y \in c_k} \delta(x, y)$, and $\delta(x, y)$ denote the Euclidean distance between points x and y.

Fix a convex set C_j and a point $x_0 \in C_j$. We denote the intersection point of the vth ray $\{x | x = \alpha(r_v^i - x^0) + x^0, \alpha \geqslant 0\}$ of c_i and the boundary $\partial(C_j)$ of C_j by $t_v(C_j, c_i)$ or t_v for $v = 1, \dots, n$. Moreover, define the hyperplane H and the half-plane H_+ respectively as follows:

$$H(C_j, c_i) := \left\{ x \middle| x = \sum_{v=1}^{n} \alpha_v^i t_v, \sum_{v=1} \alpha_v^i = 1, \alpha_v^i \geqslant 0 \right\}$$

and

$$H_+(C_j, c_i) := \left\{ x \middle| x = \sum_{v=1}^{n} \alpha_v^i t_v, \sum_{v} \alpha_v^i \geqslant 1, \alpha_v^i \geqslant 0 \right\}.$$

Given a cone c_i, a convex compact set D, and an open convex set C_j, we have

$$c_i \cap (D \setminus C_j) \subseteq (D \cap c_i) \cap H_+(C_j, c_i). \tag{10}$$

The following lemma is of basic importance for the calculation of an upper bound of the objective function over a cone.

LEMMA 2. *For the convex sets $C_1, \dots, C_m$ and a fixed cone c_i,*

$$c_i \cap (D \setminus \cup_{j=1}^{m} C_j) \subseteq (D \cap (\cap_{j=1}^{m} H_+(C_j, c_i))) \cap c_i,$$

under Assumption(A3).

Proof. Directly from (9). □

We denote the relaxed convex feasible region over c_i by

$$F(c_i) := (D \cap (\cap_{j=1}^{m} H_+(C_j, c_i))) \cap c_i. \tag{11}$$

It is straightforward that

$$\max\{bx | x \in D, x \notin \cup_{j=1}^{m} C_j, x \in c_i\} \leqslant \max\{bx | x \in F(c_i)\}.$$

Now we are all ready to present the algorithm.

ALGORITHM MRC (For Optimization with Multiple Reverse Convex Constraints)

Step 0. Compute the point x^0 and construct a conical partition $\mathcal{C} = \{c_1, \ldots, c_p\}$ of D. Set tolerance ε. Set $\mathcal{M} := \{(1, \ldots, p)\}, k := 0, L := -\infty, U := \infty$.

Step 1. Select a μ from $\mathcal{M}$. Calculate $U_\mu := \max\{bx | x \in F(c_\mu)\}$. and $x^k := \arg\max\{bx | x \in F(c_\mu)\}$.

Step 2. If $U_\mu \leqslant L$ then delete μ from $\mathcal{M}$ and goto Step 1. Otherwise calculate a lower bound L_μ of bx over $F(c_\mu)$.

Step 3. If $L_\mu > L$ then $L := L_\mu$; Delete all j from $\mathcal{M}$ with $U_j - L < \varepsilon$; If $\mathcal{M} = \emptyset$ then stop; $x^* := x^k$ is an optimal solution.

Step 4. Divide c_μ into c_{p+2k} and c_{p+2k+1}. Let $\mathcal{M} := (\{\mathcal{M}\} \backslash \{\mu\}) \cup \{p + 2k, p + 2k + 1\}$. $k := k + 1$. go to Step 1.

Unlike the general rectangle-based branch-and-bound approaches, one of the favorable properties of ALGORITHM MRC is its ability to cut off an infeasible region deeply. The computational cost of this advantage is the calculation of the points $t_v(C_j, c_i)$. Since $C_i (i = 1, \ldots, p)$ are convex sets, we can employ a convex optimization to obtain the points $t_v(C_j, c_i)$ appearing in the definitions of $F(c_i)$ and $H_+(C_j, c_i)$.

In Step 2, we calculate a lower bound L_μ of bx over $F(c_\mu)$ for a fixed cone c_μ. A heuristic way to achieve this goal is to calculate the value

$$L_\mu := \max\{bx | x = t_v(C_j, c_\mu),\ x \in D,\ v = 1, \ldots, n,\ j = 1, \ldots, m\}. \tag{12}$$

It is easy to see that if all n points $t_v (v = 1, \ldots, n)$ for every C_j are infeasible, then there is no feasible point in c_μ; accordingly, we set $\mathrm{L}_\mu := -\infty$. When $L \leqslant bt_v(C_i, c_\mu)$ for all v and i, we can remove cone c_μ from further consideration.

In Step 4, we divide c_μ into c_{p+2k} and c_{p+2k+1}. Many exhaustive dividing processes, for instance, bisection and w-dividing methods [12], can be used here.

THEOREM 3. (Theorem 3.7 of [7]) *Assume that the conical partitions in* ALGORITHM MRC *are exhaustive. Then every cluster point of the sequence* $\{x^k\}$ *is an optimal solution of the problem (8).*

4. Solution Method

Now we return to problem (6) or the corresponding compact form (7). The difference between (7) and (8) is that (7) contains a general convex set. This makes it impossible to use the algorithm of the previous section. In order to circumvent this limitation, a set of linear inequalities can be used for the approximation. Moreover, an appropriate decomposition of the convex set in (7) can be employed to enable efficient design of the algorithm. We will address these issues in this section.

As stated above, Assumption (A3) is fundamentally crucial for ALGORITHM MRC. Without this assumption, difficulty in generating the conical partition for the feasible region may be encountered.

LEMMA 4. *Assumption (A3) is satisfied for the problem (8).*

Proof. We use the representation of (6) for obtaining such a point in (A3). Take an arbitrary point $\overline{x}$ from D. We can select $\overline{u}_j$ and $\overline{v}_j$ such that $(\overline{x}, \cdot, \cdot, \cdot, \overline{u}_j, \overline{v}_j, \cdot)$ is in G_j and H_j^- for all j. With $(\overline{x}, \cdot, \cdot, \cdot, \overline{u}_j, \overline{v}_j, \cdot)$, we take a suitable positive $\overline{z}_j$ such that$(\overline{x}, \cdot, \overline{z}_j, \cdot, \overline{u}_j, \overline{v}_j, \cdot) \in H_j$. Notice that the variable z_j is free in the previous two sets G_j and H_j^-. Therefore, the selected point is contained in the two sets. We are then ready to select positive $\overline{s}_j$ and $\overline{w}_j$ such that

$$(\overline{z}_j + \overline{s}_j)^2 - \overline{w}_j \leqslant 0, \tag{13}$$

a statement that defines Γ_j . Take a large $\overline{y}_j$, such that $(\overline{x}, \overline{y}_j, \overline{z}_j, \overline{s}_j, \overline{u}_j, \overline{v}_j, \overline{w}_j) \in H_j^-$. Note that a large $\overline{y}_j$ may violate the inequality

$$\overline{z}_j^{\,2} + \overline{s}_j^{\,2} + 2\overline{y}_j - \overline{w}_j < 0 \tag{14}$$

which defines Γ_j^-. To prevent such a potential violation, we replace $\overline{w}_j$ by $\overline{w}_j + 2\overline{y}_j$. Consequently, we have the points $(\overline{x}, \overline{y}_j, \overline{z}_j, \overline{s}_j, \overline{u}_j, \overline{v}_j, \overline{w}_j + \overline{y}_j) \in G_j^- \cap H_j^- \cap \Gamma_j^-$ for all j that satisfy Assumption (A3). □

Since $G_j \cap H_j \cap \Gamma_j \cap D$ is not a polytope, ALGORITHM MRC can not be directly applied to the problem (7). Hence, we use a sequence of polytopes to approximate the convex set $G_j \cap H_j \cap \Gamma_j \cap D$. Let $\mathcal{P}_j^0$ be a polytope in R^{n+6} that contains $G_j \cap H_j \cap \Gamma_j \cap D$, $j = 1, \ldots, m$. The polytope $\mathcal{P}_j^0$ is customarily assumed in past studies to be defined by a system of linear inequalities. The suitability of this assumption rests on the approximation of $G_j \cap H_j \cap \Gamma_j \cap D$ and the easy calculation of the vertices of $\mathcal{P}_j^0$. For the sake of efficiency, we assume that $\mathcal{P}_j^0$ is a simplex in R^{n+6} and that $G_j \cap H_j \cap \Gamma_j \cap D$ is defined by a convex function $S_j(x, \varpi_j) \leqslant 0$. Let $V(\mathcal{P}_j^0)$ be the set of the vertices of $\mathcal{P}_j^0$. At the kth iteration, we select $v_j^k \in \arg\max\{S_j(v_j) | v_j \in V(\mathcal{P}_j^k)\}$, where $v_j^k := (x^k, \varpi_j^k)$. Similarly, we denote $v_j := (x, \varpi_j)$. If $v_j^k \notin G_j \cap H_j \cap \Gamma_j \cap D$, we compute a subgradient $B_j(v_j^k)$ of S_j at v_j^k and let

$$\ell_j^k(v_j) = (v_j - v_j^k)B_j(v_j^k) + S(v_j^k).$$

The inequality

$$\ell_j^k(v_j) \leqslant 0$$

contains every feasible point of (7) but v_j^k. By setting

$$\mathcal{P}_j^{k+1} := \mathcal{P}_j^k \cap \{v_j | \ell^k(v_j) \leqslant 0\},$$

we can generate a nested sequence of polytopes $\{\mathcal{P}^k\}_{k=1,2,\dots}$ such that $\dots \subset \mathcal{P}_j^{k+1} \subset \mathcal{P}_j^k \subset \dots$ and

$$\lim_{k\to\infty} \mathcal{P}_j^k = G_j \cap H_j \cap \Gamma_j \cap D$$

for all j. Suppose $V(\mathcal{P}_j^k)$ is available. Then $V(\mathcal{P}_j^{k+1})$ can be calculated by the algorithms such as in [5].

LEMMA 5. *Suppose* $v_j^k \in \mathcal{P}_j^k$ *and* $\lim_{k\to\infty} v_j^k = v_j^*$, *then* $v_j^* \in G_j \cap H_j \cap \Gamma_j \cap D$.

Proof. From Lemmas 3.2 and 3.5 of [7], we know that $\ell_j^k(v_j)$ are uniformly equicontinuous and that there exists a continuous function ℓ_j such that

$$\lim_{k\to\infty} \ell_j^k(v_j^k) = \lim_{k\to\infty} \ell_j^k(v_j^{k+1}) = \ell_j(v_j^*).$$

We observe that $\ell_j^k(v_j^k) \leqslant 0$ and $\ell_j^k(v_j^{k+1}) \geqslant 0$. Therefore, $\ell_j(v_j^*) = 0$. Hence, by Lemma 3.4 of [7], we have $v_j^* \in G_j \cap H_j \cap \Gamma_j \cap D$. □

5. The Algorithm and its Convergence

Based on the discussion above, we are prepared to describe our algorithm.

ALGORITHM SUMRATIOS (For Optimization of Sum of Ratios)

Step 0. Compute the point $(x_0, \varpi_1^0, \dots, \varpi_m^0)$. Construct a polytope $\mathcal{P}_j^0$ for each j. Construct a conical partition $\mathcal{C}_j = \{c_1^j, \dots, c_p^j\}$ of $\mathcal{P}_j^0$ for each $j (j = 1, \dots, m)$. Set tolerance ε. Set $\mathcal{M}_j := \{1 + jp, \dots, (j+1)p\}$ and $\mathcal{M} := \{(\mu_1, \dots, \mu_m) | \mu_j \in \mathcal{M}_j\}$. Set $k := 0, L := -\infty$.

Step 1. Select a $(\mu_1, \dots, \mu_m)$ from $\mathcal{M}$ and calculate, $U_{(\mu_1,\dots,\mu_m)} = \max\Big\{\sum_{j=1}^m s_j | \ (x, y_j, z_j, s_j, u_j, v_j, w_j) \in F(c_{v_j}),\ j = 1, \dots, m\}$. Let $(x^k, (y_j, z_j, s_j, u_j, v_j, w_j)^k)$, $j = 1, \dots, m$, be the optimal solution. Set $U_k = U_{(\mu_1,\dots,\mu_m)}$.

Step 2. If $U_k \leqslant L$, then delete $(\mu_1, \dots, \mu_m)$ from $\mathcal{M}$ and go to Step 1. Otherwise calculate a lower bound L_k of $\sum_j s_j$ by (12).

Step 3. If $L_k > L$, then $L := L_k$; Delete all $(\mu_1, \ldots, \mu_m)$ from $\mathcal{M}$ such that $U_{(\mu_1,\ldots,\mu_m)} - L < \epsilon$; If $\mathcal{M} = \emptyset$, then stop. $x^* := x^k$ is an optimal solution.

Step 4. Select a μ_j from $\{\mu_1, \ldots, \mu_m\}$. Divide c_μ into c_{p+2k} and c_{p+2k+1}. Set $\mathcal{M} := \mathcal{M} \backslash \{(\mu_1, \ldots, \mu_m)\}$, $\mathcal{M} := \mathcal{M} \cup \{(\mu_1, \ldots, \mu_{j-1}, p+2k, \mu_{j+1}, \ldots, \mu_m)\} \cup \{(\mu_1, \ldots, \mu_{j-1}, p+2k+1, \mu_{j+1}, \ldots, \mu_m)\}$. Set $k := k+1$ and go to Step 1.

The main computational tasks of the algorithm are the conical partition and the calculation of $U_{(\mu_1,\ldots,\mu_m)}$ in Step 1. Since the problem possesses a nice decomposition structure in the feasible region, the refinement of the conical partition can be carried out for c_{μ_j} individually (see Step 4). This operation only involves the cone in an $(n+6)$-dimensional space, which has much lower dimensionality compared with the original $(n+6m)$ space.

The upper bound U_k is obtained by solving the following linear programming in a space with $(n+6m)$ dimensions in Step 1 :

$$\max\left\{\sum_j^m s_j \middle| (x, y_j, z_j, s_j, u_j, v_j, w_j) \in F(c_{\mu_j}), \quad j = 1, \ldots, m\right\}.$$

Note that every $F(c_{\mu_j})$ is written as a system of linear inequalities [13], hence, the above implementation is not computationally costly.

LEMMA 6. *Assume that the conical partitions in* ALGORITHM SUM-RATIOS *are exhaustive and* $\{(x^*, \varpi_1^*, \ldots, \varpi_m^*)^k\}$ *is an accumulation point of the sequence* $\{(x^\star, \varpi_1^\star, \ldots, \varpi_m^\star)^k\}$, *then* $v_j^* \notin G_j^- \cap H_j^- \cap \Gamma_j^-$.

Proof. See the first part of Theorem 3.7 in [7]. □

THEOREM 7. *Assume that the conical partitions in* ALGORITHM SUM-RATIOS *are exhaustive, then every accumulation point of the sequence* $\{(x, \varpi_1, \ldots, \varpi_m)^k\}$ is an optimal solution.

Proof. Suppose that $\lim(x, \varpi_1, \ldots, \varpi_m)^k = (x^*, \varpi_1^*, \ldots, \varpi_m^*)$. From Lemma 5 we see that $v_j^* \in G_j \cap H_j \cap \Gamma_j \cup D$ for all j. In association with Lemma 6, we see that $(x^*, \varpi_1^*, \ldots, \varpi_m^*)$ is a feasible solution of problem (7). Note that maximum value over c_i is maintained from the definition of v_j^k in the algorithm. Therefore, the $(x^*, \varpi_1^*, \ldots, \varpi_m^*)$ is an optimal solution. □

6. An Illustrative Example

The following example [1] is used to illustrate the algorithm.

EXAMPLE.

$$f(x_1, x_2) = \frac{-x_1^2 + 3x_1 - x_2^2 + 3x_2 + 3.5}{x_1 + 1.0} + \frac{x_2}{x_1^2 - 2x_1 + x_2^2 - 8x_2 + 20.0}$$

That is, $m = 2, n = 2, g_1(x_1, x_2) = -x_1^2 + 3x_1 - x_2^2 + 3x_2 + 3.5, g_2(x_1, x_2) = x_2, h_1(x_1, x_2) = x_1 + 1.0, h_2(x_1, x_2) = x_1^2 - 2x_1 + x_2^2 - 8x_2 + 20.0$. The feasible region D is defined by the following linear inequalities:

$$D := \{(x_1, x_2) | 2x_1 + x_2 \leqslant 6, 3x_1 + x_2 \leqslant 8, x_1 - x_2 \leqslant 1, x_1 \geqslant 1, x_2 \geqslant 2\}.$$

$(x_1^*, x_2^*) = (1.00, 2.00), f(x_1^*, x_2^*) = 4.0357$. Let

$$\begin{aligned}
g_1'(x_1, x_2) &= 3x_1 + 3x_2 + 3.5,\\
g_1''(x_1, x_2) &= x_1^2 + x_2^2,\\
h_1'(x_1, x_2) &= x_1 + 1,\\
h_1''(x_1, x_2) &= 0,\\
g_2'(x_1, x_2) &= x_2,\\
g_2''(x_1, x_2) &= 0,\\
h_2'(x_1, x_2) &= x_1^2 + x_2^2,\\
h_2''(x_1, x_2) &= 2x_1 + 8x_2 - 20.
\end{aligned}$$

The equivalent problem (2.4) is as follows.:

$$\begin{array}{ll}
\max & s_1 + s_2\\
\text{s.t.} & x_1^2 + x_2^2 - u_1 \leqslant 0\\
& x_1^2 + x_2^2 - v_2 - z_2 \leqslant 0\\
& -u_2 \leqslant 0\\
& x_1 + 1 - v_1 - z_1 \leqslant 0\\
& (s_1 + z_1)^2 - w_1 \leqslant 0\\
& (s_2 + z_2)^2 - w_2 \leqslant 0\\
& 2x_1 + x_2 \leqslant 6\\
& 3x_1 + x_2 \leqslant 8\\
& x_1 - x_2 \leqslant 1\\
& x_1 \geqslant 1, x_2 \geqslant 2\\
& \\
& 3x_1 + 3x_2 + 3.5 - u_1 - y_1 \geqslant 0\\
& 2x_1 + 8x_2 - 20 - v_2 \geqslant 0\\
& x_2 - u_2 - y_2 \geqslant 0\\
& -v_1 \geqslant 0\\
& s_1^2 + z_1^2 + 2y_1 - w_1 \geqslant 0\\
& s_2^2 + z_2^2 + 2y_2 - w_2 \geqslant 0\\
& \text{all variables are nonnegative but } u_j \text{ and } v_j,\ i = 1, 2.
\end{array}$$

The second group of the constraints corresponds to the reverse constraints. However, since the first four are linear, the real reverse constraints are only the last two. Let

$$v_{jv} = 47e_v, \quad v = 1, \ldots, 8, \; j = 1, 2,$$

where e_v is the vth unit vector in R^8. Then $v_{jv}, v = 1, \ldots, 8, v_{j0} = [0, 0, 0, 0, 0, 0, 0, 0] \in R^8$, for $j = 1, 2$, will be the vertices of two initial polytopes in R^8 in Step 1 of the algorithm, respectively. Let

$$(x_1^0, x_2^0, y_j^0, z_j^0, s_j^0, u_j^0, v_j^0, w_j^0)^T = (-1, -1, 1, 1, -1, 2, 2, 5)^T, \quad j = 1, 2.$$

$$s_1^2 + z_1^2 + 2y_1 - w_1 = -1 < 0,$$
$$s_2^2 + z_2^2 + 2y_2 - w_2 = -1 < 0.$$

Therefore, they can serve as the vertices of the initial cones. For each $j = 1, 2$, let the extreme rays from $(x_1^0, x_2^0, y_j^0, z_j^0, s_j^0, u_j^0, v_j^0, w_j^0)^T$ be

$$v_{jv} - (x_1^0, x_2^0, y_j^0, z_j^0, s_j^0, u_j^0, v_j^0, w_j^0)^T, \quad v = 1, \ldots, 8.$$

The intersecting points of the rays with the constraint $s_j^2 + z_j^2 + 2y_j - w_j = 0, \quad j = 1, 2$, are respectively as follows:

$$\begin{aligned}
t_{j1} &= [47, 0, 0, 0, 0, 0, 0, 0]^T,\\
t_{j2} &= [0, 47, 0, 0, 0, 0, 0, 0]^T,\\
t_{j3} &= [-0.968442, -0.968442, 1.48323,\\
&\qquad 0.968442, 0.968442, 1.93688, 1.93688, 4.84221]^T,\\
t_{j4} &= [-0.978613, -0.978613, 0, 1.98378, 0.978613,\\
&\qquad 1.95723, 1.95723, 4.89307]^T,\\
t_{j5} &= [-0.978613, -0.978613, 0, 0.978613, 1.98378, 1.95723,\\
&\qquad 1.95723, 4.89307]^T,\\
t_{j6} &= [0, 0, 0, 0, 0, 47, 0, 0]^T,\\
t_{j7} &= [0, 0, 0, 0, 0, 0, 47, 0]^T,\\
t_{j8} &= [22.065, 22.065, 0, -22.065, -22.065,\\
&\quad -44.13, -44.13, 973.73]^T, \quad j = 1, 2.
\end{aligned}$$

Let

$$T_j = [t_{j1}, t_{j2}, t_{j3}, t_{j4}, t_{j5}, t_{j6}, t_{j7}, t_{j8}], \quad j = 1, 2$$

and

$$\alpha_j = [\alpha_{j1}, \alpha_{j2}, \alpha_{j3}, \alpha_{j4}, \alpha_{j5}, \alpha_{j6}, \alpha_{j7}, \alpha_{j8}]^T, \quad j = 1, 2.$$

The variables in the linear program can be represented as

$$(x_1, x_2, y_j, z_j, s_j, u_j, v_j, w_j)^T = T_j\alpha_j, \sum_{v=1}^{8} \alpha_{jv} \geqslant 1, \alpha_{jv} \geqslant 0, \quad j = 1, 2. \tag{15}$$

Replacing all variables with (15) yields the following LP, whose optimal value is an upper bound of $s_1 + s_2$.

$$
\begin{aligned}
\max \quad & 0.968442\alpha_{13} + 0.978613\alpha_{14} + 1.98378\alpha_{15} - 22.065\alpha_{18} \\
& \quad + 0.968442\alpha_{23} + 0.978613\alpha_{24} \\
& \quad + 1.98378\alpha_{25} - 22.065\alpha_{28} \\
\text{s.t.} \quad & 94\alpha_{11} + 47\alpha_{12} - 2.90533\alpha_{13} - 2.93584\alpha_{14} \\
& \quad - 2.93584\alpha_{15} + 66.195\alpha_{18} \leqslant 6 \\
& 141\alpha_{11} + 47\alpha_{12} - 3.87377\alpha_{13} - 3.91445\alpha_{14} \\
& \quad - 3.91445\alpha_{15} + 88.26\alpha_{18} \leqslant 8 \\
& 47\alpha_{11} - 47\alpha_{12} \leqslant 1 \\
& -47\alpha_{11} + 0.968442\alpha_{13} + 0.978613\alpha_{14} \\
& \quad + 0.978613\alpha_{15} - 22.065\alpha_{18} \leqslant -1 \\
& -47\alpha_{12} + 0.968442\alpha_{13} + 0.978613\alpha_{14} + 0.978613\alpha_{15} \\
& \quad - 22.065\alpha_{18} \leqslant -2 \\
& -141\alpha_{11} - 141\alpha_{12} + 9.23076\alpha_{13} + 7.82891\alpha_{14} \\
& \quad + 7.82891\alpha_{15} + 47\alpha_{16} - 176.52\alpha_{18} \leqslant 3.5 \\
& 47\alpha_{11} - 3.87377\alpha_{13} - 4.91962\alpha_{14} \\
& \quad - 3.91445\alpha_{15} - 47\alpha_{17} + 88.26\alpha_{18} \leqslant -1 \\
& 0.968442\alpha_{13} + 1.98378\alpha_{14} + 0.978613\alpha_{15} - 22.065\alpha_{18} \leqslant 0 \\
& 47\alpha_{11} + 47\alpha_{12} + 9.23076\alpha_{13} + 8.83407\alpha_{14} \\
& \quad + 8.83407\alpha_{15} + 47\alpha_{16} + 47\alpha_{17} + 907.535\alpha_{18} \leqslant 47 \\
& -94\alpha_{21} - 376\alpha_{22} + 11.6213\alpha_{23} + 11.7434\alpha_{24} \\
& \quad + 11.7434\alpha_{25} + 47\alpha_{27} - 264.78\alpha_{28} \leqslant -20 \\
& -47\alpha_{22} + 4.38855\alpha_{23} + 2.93584\alpha_{24} + 2.93584\alpha_{25} \\
& \quad + 47\alpha_{26} - 66.195\alpha_{28} \leqslant 0 \\
& 47\alpha_{21} + 47\alpha_{22} + 10.1992\alpha_{23} + 9.81269\alpha_{24} \\
& \quad + 9.81269\alpha_{25} + 47\alpha_{26} + 47\alpha_{27} + 885.47\alpha_{28} \leqslant 47 \\
& 47\alpha_{11} - 0.968442\alpha_{13} - 0.978613\alpha_{14} \\
& \quad - 0.978613\alpha_{15} + 22.065\alpha_{18} - 47\alpha_{21} + 0.968442\alpha_{23} \\
& \quad + 0.978613\alpha_{24} + 0.978613\alpha_{25} - 22.065\alpha_{28} = 0 \\
& 47\alpha_{12} - 0.968442\alpha_{13} - 0.978613\alpha_{14} - 0.978613\alpha_{15} \\
& \quad + 22.065\alpha_{18} - 47\alpha_{22} + 0.968442\alpha_{23} + 0.978613\alpha_{24} \\
& \quad + 0.978613\alpha_{25} - 22.065\alpha_{28} = 0 \\
& \alpha_{11} + \alpha_{12} + \alpha_{13} + \alpha_{14} + \alpha_{15} + \alpha_{16} + \alpha_{17} + \alpha_{18} \geqslant 1 \\
& \alpha_{21} + \alpha_{22} + \alpha_{23} + \alpha_{24} + \alpha_{25} + \alpha_{26} + \alpha_{27} + \alpha_{28} \geqslant 1 \\
& \quad \text{all variables are nonnegative.}
\end{aligned}
$$

The optimal value of this problem is 5.24, and the original variables are

$$
\begin{aligned}
&(x_1, x_2, y_1, z_1, s_1, u_1, v_1, w_1, y_2, z_2, s_2, u_2, v_2, w_2) \\
&\quad = (1.0, 4.0, 0, 0, 0.8191, 0, 2, 39.1809, \\
&\qquad 0, 0.2, 4.42177, 4.0, 4.0, 27.578),
\end{aligned}
$$

an outcome that does not satisfy the convex constraints. Since $(x_1, x_2) = (1, 4)$ satisfies the original linear constraints, the objective value at this point is a lower bound of the optimal objective function, which has the magnitude 2.0833.

Next we select the first cone to divide and obtain two cones. Since $x_1^2 + x_2^2 - u_1 = 25 > 0$ at the current LP solution, the quadratic constrains $x_1^2 + x_2^2 - u_1 \leqslant 0$ is violated. As the subgradient at the optimal solution is $(2, 8, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0)$, the linear cut

$$2(x_1 - 1) + 8(x_2 - 4) - u_1 + 25 \leqslant 0$$

was added to the constraint set of the first LP after replacing the original variables by (15).

The approximate solution can be found by repeating this procedure until the difference between the lower and upper bounds is below some small prescribed value ε. In above example, we set $\varepsilon = 0.01$, an solution (1.00, 2.00, 7.50, 2.00, 3.75, 5.00, 0.00, 33.06, 2.00, 7.00, 0.29, 0.00, −2.00, 53.08), that is, $(x_1, x_2) = (1.00, 2.00), (y_1, z_1, s_1, u_1, v_1, w_1) = (7.50, 2.00, 3.75, 5.00, 0.00, 33.06), (y_2, z_2, s_2, u_2, v_2, w_2) = (2.00, 7.00, 0.29, 0.00, -2.00, 53.08)$ was found after 1016 iterations and 478 seconds.

In order to investigate the behavior of the algorithm SUMRATIOS, another set of problems with different number of ratios were considered. The variable x in problem (P) for all test problems was 2-dimensional. All numerators and denominators of the ratios of the test problems were quadratic polynomial or linear functions. More precisely, either numerator or denominator is quadratic for each ratio. All coefficients of the quadratic polynomial and linear functions were random integers between -10 and 10. Figure 1 shows the results of the algorithm for this data set. Each point is the average from three problems. The curve in Figure 1 depicts an approximation function $26.199e^{1.4004x}$ of CPU time. The algorithm was coded in MATLAB 6.5 and tested on a Win XP PC (Xeon(TM) CPU 2.80Hz, 1.00 GB RAM). Initial points in Step 0 were calculated by the constraints of (5). The relaxed LP subproblems in Step 1 were solved by the procedure of `linprog` in Optimization ToolBox.

Remark. Preliminary numerical reports on other types of sum-of-ratios problems can be found in the literature [18, 10]. But the algorithms in [18,

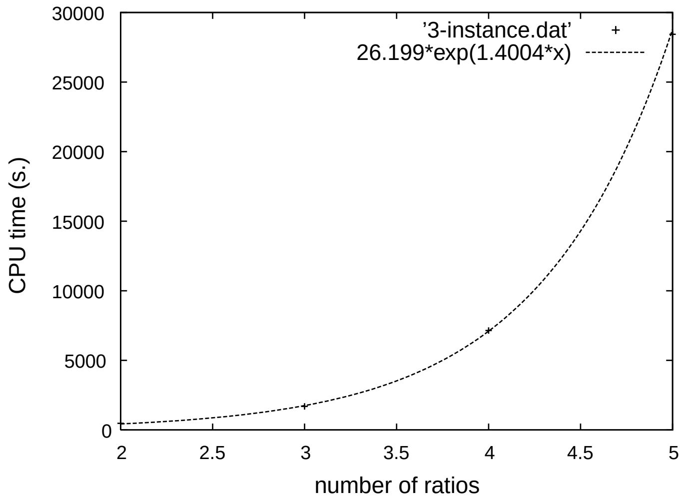

Figure 1. CPU time vs. number of ratios

10] are designed for maximization of sum of linear ratios and convex/concave ratios problems, respectively. It is reported recently that Kuno's algorithm works well for sum of convex/concave ratios problems [25], however, its extension to the d.c. case is still under developing.

7. Concluding Remarks

A general form for the optimization of the sum of ratios, in which both the denominator and the numerator are d.c. functions, has been discussed. In particular, the properties of the transformed problem are examined. An algorithm designed for the solution of this problem by the branch-and-bound algorithm that is based on the combination of the conical partition and the outer approximation is proposed. To investigate the efficiency of the proposed algorithm, a preliminary numerical experiment was carried out. The results indicate that the algorithm proposed in this paper solves problems with small-sizes in reasonable amount of time.

The basic question of dealing with the ratios is the key to the transformation. Benson [1] proposed to use a concave envelope to approximate the ratio t/s, a procedure which is equivalent to $g(x)/h(x)$ with the constraints $t = g(x)$ and $s = h(x)$. His original ratio has the structure of concave/convex which can be "converted" readily to convex form by the method used to treat the single-ratio problem. These results suggest the possible extension of the concave envelope method to the sum of ratios problem with the structure of d.c./d.c.

Acknowledgments

The author wish to thank Professor S. Schaible for his valuable discussions. They are also grateful to the referees for their detailed comments and valuable suggestions.

References

1. Benson, H.P. (2001), Using concave envelopes to globally solve the nonlinear sum of ratios problem, Manuscript, Warrington College of Business Administration, University of Florida.
2. Cambini, A., Martein, L. and Schaible, S. (1989), On maximizing a sum of ratios, *Journal of Information and Optimization Sciences*, 10, 65–79.
3. Charnes, A. and Cooper, W.W. (1962), Programming with linear fractional functionals, *Naval Research Logistics Quarterly* 9, 181–186.
4. Chen, D.Z., Daescu, O., Dai, Y., Katoh, N., Wu, X. and Xu, J. (2000), Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications, In: *Proceedings of the 11th Annual SIAM-ACM Symposium on Discrete Algorithms (SODA)*, San Francisco, pp. 707–716.
5. Chen, P.C., Hansen, P. and Jaumard, B. (1991), On-line and off-line vertex enumeration by adjacency lists, *Operations Research Letters* 10, 403–409.
6. Craven, B.D. (1988), *Fractional Programming*, Sigma Series in Applied Mathematics 4, Heldermann Verlag.
7. Dai,Y., Shi, J. and Yamamoto, Y. (1996), Global optimization problem with several reverse convex constraints and its application to out-of-roundness problem, *Journal of the Operations Research Society of Japan* 39, 356–371.
8. Dinkelbach, W. (1967), On nonlinear fractional programming, *Management Science* 13, 492–498.
9. Falk, J.E. and Palocsay, S.W. (1992), Optimizing the sum of linear fractional functions, In: *Recent Advances in Global Optimization*, Kluwer Academic Publishers, pp. 221–258.
10. 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.
11. Hartman, P. (1959), On functions representable as a difference of convex functions, *Pacific Journal of Mathematics* 9, 707–713.
12. Horst, R. and Pardalos, P.M. (1995), *Handbook of Global Optimization*, Kluwer Academic Publishers.
13. Horst, R. and Tuy, H. (1993), *Global Optimization: Deterministic Approaches*, Springer-Verlag.
14 Ibaraki, T. (1983), Parametric approaches to fractional programs, *Mathematical Programming* 26, 345–362.
15. Ibaraki, T., Ishii, H., Iwase, J., Hasegawa, T. and Mine, H. (1976), Algorithm for quadratic fractional programming problems, *Journal of the Operations Research Society of Japan* 19, 174–191.
16. Konno, H. and Kuno, T. (1990), Generalized linear multiplicative and fractional programming, *Annals of Operations Research* 25, 147–161.
17. Konno, H. and Yamashita, H. (1999), Minimizing sums and products of linear fractional functions over a polytope, *Naval Research Logistics* 46, 583–596.
18. Kuno, T. (2002), A branch-and-bound algorithm for maximizing the sum of several linear ratios, *Journal of Global Optimization* 22, 155–174.

19. Megiddo, N. (1979), Combinatorial optimization with rational objective functions, *Mathematics of Operations Research* 4, 414–424.
20. Ritter, K. (1967), A parametric method for solving certain nonconcave maximization problems, *Journal of Computer and System Sciences*, 1, 44–54.
21. Rockafellar, R.T. and Wets, R.J-R. (1998), *Variational Analysis*, Springer.
22. Schaible, S. (1973), Fractional programming: transformations, duality and algorithmic aspects, Technical Report 73–79, Department of Operations Research, Stanford University.
23. Schaible, S. (1976), Fractional programming I: duality, *Management Science* 22, 858–867.
24. Schaible, S. (1995), Fractional programming, In: *Handbook of Global Optimization*, Kluwer Academic Publishers, pp. 495–608.
25. Schaible, S. and Shi, J. (2003), Fractional programming: the sum-of-ratios case, *Optimization Methods and Software* 18, 219–229.
26. Sekitani, K., Shi, J. and Yamamoto, Y. (1995), General fractional programming: min–max convex–convex quadratic case, In: *APORS-Development in Diversity and Harmony*, World Scientific, pp. 505–514.
27. Shi, J. (2001), A combined algorithm for fractional programming, *Annals of Operations Research* 103, 135–147.
28. Sniedovich, M. (1987), A new look at fractional programming, *Journal of Optimization Theory and Applications* 54, 113–120.
29. Stancu-Minasian, I.M. (1997), *Fractional Programming*, Kluwer Academic Publishers.