International Journal of Operations Research

International Journal of Operations Research Vol. 11, No. 1, 028–039 (2014)

A Revised Algorithm for Solving the Sum of Linear Ratios Problem with Lower Dimension using Linear Relaxation*

Yongwen Hu[1], Jianming Shi[2]†, and Shinya Watanabe[1]

[1]School of Engineering, Muroran Institute of Technology, Muroran, Japan

[2]School of Management, Tokyo University of Science, Tokyo, Japan

Received February 2014; Revised April 2014; Accepted April 2014

Abstract — In this paper, we propose a revision of the linear relaxation algorithm [Carlsson and Shi (2013): A linear relaxation algorithm for solving the sum-of-linear-ratios problem with lower dimension. OR Letters, 41(4): 381-389] for solving the Sum-of-Linear-Ratios (SOLR) problem. Carlsson and Shi casted the SOLR problem into an equivalent problem with linear objective and a set of linear and nonconvex quadratic constraints. By dropping out the nonconvex quadratic constraints, they proposed a linear relaxation for the SOLR problem and designed a branch-and-bound algorithm to solve the SOLR problem with lower dimension.

To circumvent the nonconvex quadratic constraints, we do not drop them out but make a linear relaxation for the nonconvex constraints with some extra variables. Therefore, this linear relaxation is generally tighter than the previous one. With the help of the new relaxation, we propose an algorithm for solving the SOLR problem and prove the convergence of the algorithm. The numerical experiments are conducted and the results indicate that our method is more efficient than the previous.

Keywords — sum-of-linear-ratios, linear relaxation, nonconvex quadratic, branch and bound

1. INTRODUCTION

Over the past decades, the Sum-of-Linear-Ratio (SOLR) problem has attracted theoretical and practical interest of many researchers and practitioners. The SOLR problem poses some theoretical and computational challenges in nonconvex optimization because of its NP-hardness (Matsui, (1996)). The 3-partition problem (Michael, (1979)), Traveling Salesman Problem (TSP) can be written in the form of the SOLR problem. It is well known that the SOLR problem generally has multiple local optimizers which are not globally optimal even for low dimension (Schaible, (1977)). From a practical point view, the SOLR problem has a wide variety of applications such as multistage stochastic shipping problem (Almogy *et al.* (1970)), multi-objective bond portfolio optimization problem (Konno and Inori, (1989), Konno and Watanabe, (1996)), minimum ratio spanning tree problem (Skiscim and Palocsay, (2001)), finance and investment (Choi and Bricker, (1996)), and a number of geometric problems and others problems (Chen *et al.* (2000)).

In this paper, we focus on a revision of the linear relaxation algorithm (Carlsson and Shi, (2013)) with lower dimension. A variety of problems in application domains including *layered manufacturing* (Majhi *et al.*, (1999)), *camera resectioning* (Kim and Hong, (2007)), *homograph estimation*, *star cover problem*, *triangulation problem* and others can be appropriately formulated as the SOLR problem with lower dimension. The problems of this class are characterized by a small number of variables and a large number of ratios. The following are some specific examples with detailed description.

Camera Resectioning A recursive *Camera Resectioning* algorithm is proposed for solving this problem (Kim and Hong, (2007)). In the algorithm, the estimation of camera motion can be simply computed by the following expectation

$$E[X_t \mid Y_t] \approx \sum_{i=1}^{M} \omega_t^{(i)} x_t^{(i)}.$$

Here M is the number of samples (usually M is large), $\omega_t^{(i)}$ is the weight of the *i-th* sample, and $x_t^{(i)}$ is the model of the probability distribution of camera system state at time t for sample i. Clearly, this problem can be formulated into the SOLR problem in R^3 if the probability distribution is linear ratio for each sample.

* This research was partially supported by JSPS KAKENHI Grant Number (C) 23510152.

† Corresponding author's email: shi@rs.tus.ac.jp.

1813-713X Copyright © 2014 ORSTW

The star cover problem This problem was introduced by Karen Daniels at the 5th MSI workshop on Computational Geometry in 1995 (Daniels, (1995)). Later, Chen *et al.* (2005) converted this problem into $O(n^2)$ number of the SOLR subproblems, where each the SOLR problem has an objective function with the number $O(n)$ of ratios in R^2.

Triangulation Kuno and Masaki (2013) built a pinhole camera model for this problem and formulated it into minimizing the following form

$$\sum_{i=1}^{p}\left|\frac{n_i^{\mathrm{T}}x+a_i}{d_i^{\mathrm{T}}x+b_i}\right|^q,$$

where x is subject to a compact convex region in R^3 and q=1 or 2. Obviously, this problem is equivalent to the SOLR problem when q=1 and the symbol of each ratio does not change in its feasible region. Actually, the Charnes-Cooper transformation (Charnes and Cooper, (1962)) can be applied to solve this problem even the symbol of each value can be changed on its feasible region. Several other problems such as optimal penetration problem (Chen et al., (2001)) and others also share the same characters mentioned above. Therefore, the SOLR problem with lower dimension is an important class in the application domain.

Over the past decades, many global algorithms have been proposed to solve the SOLR problem. Most of these algorithms were designed to solve problems derived from economic field where the number of ratios is small (Konno and Abe, (1999), Benson, (2007)). It is well known that the SOLR problem with a single ratio is equivalent to a linear programming problem (Charnes and Cooper, (1962)). Some algorithms have been developed by using simplex-like method pivoting or the parametric simplex method for two ratios (see, for instance, (Cambini *et al.*, (1989), Konno *et al.*, (1991)). Various algorithms are developed for globally solving the problem when the number of ratios is less than 4. Some of them transform the problem into a relaxation linear programming problem by introducing new extra variables, then methods such as branch-and-bound can be exploited to obtain a global optimal solution within a given tolerance. More precisely, the branch-and-bound method is performed on a p-dimensional region rather than the native n-dimension with introducing p variables (e.g., Falk (1994), Wu *et al.*, (2008)). It was reported (Kuno, (2002)) that the SOLR problem cannot be solved in a reasonable time with a number of ratios greater than 16 (e.g. p>16) by using such approach. In addition, Depetrini and Locatelli (2011) proposed an algorithm like sampling method to obtain an approximation optimal solution, but their algorithm takes much CPU time for solving the SOLR problem even the number of ratios is 3. For an excellent review of the applications, theoretical results, algorithms for the SOLR problem, the reader is referred to Schaible and Shi (2003).

Recently, Carlsson and Shi (2013) developed an algorithm to solve the SOLR problem with lower dimension based on a linear relaxation of the objective function. The SOLR problem can be reformulated as a problem with a linear objective and a set of linear and bilinear constraints by introducing auxiliary variables, then the SOLR problem can be globally solved by branch-and-bound approach in a space of native variables with a lower dimension. It is reported that their approach can solve the problem efficiently even the number of ratios goes up to 75 (Carlsson and Shi, (2013)). However, a set of quadratic (bilinear) constraints is discarded to make a linear relaxation problem in their research. In this paper, we make a linear relaxation of the quadratic constraints to create a tighter lower bound of the minimum, which will improve the efficiency of the previous algorithm. In addition, Kuno and Masaki (2013) also focused on the problem with lower dimension and proposed an algorithm for solving a kind of the SOLR problem with different branch rules.

This paper is organized as follows. In Section 2, we give a standard form of the SOLR problem and show how to make a relaxation of its equivalent problem. In Section 3, we give a branch and bound algorithm for solving the problem and prove the convergence of the proposed algorithm. Section 4 gives a numerical example to illustrate the efficiency of the new algorithm and reports the results of numerical experiments with randomly generated datasets. We conclude this study and outline the further work in Section 5.

2. EQUIVALENT TRANSFORMATION AND LINEAR RELAXATION

In this paper, we consider the SOLR problem defined as follows

$$(P_0)\left|\begin{array}{ll}\text{minimize} & f(x)=\displaystyle\sum_{i=1}^{p}\frac{n_i^{\mathrm{T}}x+a_i}{d_i^{\mathrm{T}}x+b_i}\\ \textit{subject to} & x\in X=\left\{x\in R^n \mid Ax\le c, x\ge 0\right\}\end{array}\right.$$

where $p\ge 2$, n_i, d_i are vectors in R^n for $i=1,\cdots,p$, a_i, b_i are real numbers for $i=1,\cdots,p$, A is an $m\times n$ matrix, c is a vector in R^m. We suppose that $d_i^T x+b_i>0$ on X for all $i=1,\cdots,p$.

Because the set X is nonempty and bounded, we can construct a rectangle $B=[l,u]$ which contains the feasible region X. We denote that $l=[l_1,l_2,\cdots,l_n]^{\mathrm{T}}, u=[u_1,u_2,\cdots,u_n]^{\mathrm{T}}$, where l_j, u_j are the optimal values of the linear programming problem (1) and (2), respectively.

$$l_j := \text{minimize } x_j \quad \text{subject to } x \in \mathrm{X} \tag{1}$$

$$u_j := \text{maximize } x_j \quad \text{subject to } x \in \mathrm{X} \tag{2}$$

Considering problem (P_0) with the box $B = [l, u]$ as follows.

$$(P_1)\left|\begin{array}{ll} \text{minimize} & \sum_{i=1}^{p} \dfrac{n_i^{\mathrm{T}} x + a_i}{d_i^{\mathrm{T}} x + b_j} \\ \text{subject to} & \quad Ax \le c \\ & \quad l \le x \le u \end{array}\right. \tag{3}$$

where $0 \le l \le u$. Because $X \subseteq B$ from (1) and (2), problem (P_0) is equivalent to problem (P_1). Let us apply the Charnes-Cooper transformation (Charnes and Cooper, (1962)) to (3), by introducing $2p$ variables as follows:

$$y^i := z_i x,\ z_i := \frac{1}{d_i^{\mathrm{T}} x + b_i}, i = 1, 2, \cdots, p.$$

It is easy to see that $Ax \le c$ if and only if $Ay - cz \le 0$ and that $x \in [l, u]$ if and only if $-y + lz \le 0$ and in the sense that $y = x / (d_i^{\mathrm{T}} x + b_i),\ z = 1 / (d_i^{\mathrm{T}} x + b_i)$.

Denote that $\alpha_i := \min\left\{ d_i^{\mathrm{T}} x + b_i \mid x \in X \right\}$ and $\beta_i := \max\left\{ d_i^{\mathrm{T}} x + b_i \mid x \in X \right\}$. Then we have

$$(P_2)\left|\begin{array}{l} \text{minimize } \sum_{i=1}^{p} \left(n_i^{\mathrm{T}} y^i + a_i z_i \right) \\ \left.\begin{array}{l} \text{subject to } d_i^{\mathrm{T}} y^i + b_i z_i = 1, \\ \qquad Ay^i - cz_i \le 0, \\ \qquad \dfrac{1}{\beta_i} \le z_i \le \dfrac{1}{\alpha_i}, \\ \qquad y^i z_j = y^j z_i, \\ \qquad z_i l \le y^i \le z_i u, \end{array}\right\} i, j = 1, 2, \ldots, p. \end{array}\right. \tag{4}$$

We will see that problem (P_2) is equivalent to problem (P_1) from the following theorem.

Theorem 1. *Problem* (P_2) *is equivalent to problem* (P_1).

Proof: The following proof is similar to Theorem 1 in (Carlsson and Shi, (2013)). For self-contained we give a proof below.

Let x^* be an optimal solution to problem (P_1). Define that $(y^i)^* := x^* / (d_i^{\mathrm{T}} x^* + b_i),\ z_i^* := 1 / (d_i^{\mathrm{T}} x^* + b_i)$ then we see that $((y^i)^*, z_i^*)$ are feasible for problem (P_2). Suppose there is a feasible solution $((y^i)', z_i')$ of (P_2) such that

$$\sum_{i=1}^{p} \left(n_i^{\mathrm{T}} (y^i)' + a_i z_i' \right) < \sum_{i=1}^{p} \left(n_i^{\mathrm{T}} (y^i)^* + a_i z_i^* \right). \tag{5}$$

Thus, we see that with $x' = (y^i)' / z_i'$

$$\sum_{i=1}^{p} \left(n_i^{\mathrm{T}} (y^i)' + a_i z_i' \right) = \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x' + a_i}{d_i^{\mathrm{T}} x' + b_i},\quad \sum_{i=1}^{p} \left(n_i^{\mathrm{T}} (y^i)^* + a_i z_i^* \right) = \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^* + a_i}{d_i^{\mathrm{T}} x^* + b_i}$$

and x' is also a feasible solution to (P_1). Therefore, the inequality (5) contradicts the optimality of x^* for (P_1). And similar to vice versa.

From Theorem 1, in order to globally solve (P_0), we may solve (P_2) instead because all (P_0), (P_1) and (P_2) are equivalent. However, the constrains $y^i z_j = y^j z_i$ for $i, j = 1, \cdots, p$ in (4) are quadratic and nonconvex. Therefore, this problem can be solved in the category of nonconvex quadratic programming, which is general, of course, NP -hard.

It is trivial that we can make a relaxation for (P_2) by discarding $y^i z_j = y^j z_i$ for all $i, j = 1, \cdots, p$ as follows:

$$Q_0(l,u)\quad \left|\begin{array}{l} \text{minimize } \sum_{i=1}^{p} \left(n_i^{\mathrm{T}} y^i + a_i z_i\right) \\ \textit{subject to } \left.\begin{array}{l} d_i^{\mathrm{T}} y^i + b_i z_i = 1, \\ A y^i - c z_i \le 0, \\ \dfrac{1}{\beta_i} \le z_i \le \dfrac{1}{\alpha_i}, \\ z_i l \le y^i \le z_i u, \end{array}\right\} i = 1,2,\dots,p \end{array}\right. \tag{6}$$

The problem (6) is a linear programming problem. In their paper (Carlsson and Shi, (2013)), Carlsson and Shi fundamentally exploited this linear form (6) to obtain the lower bounds of (4) and design a branch and bound algorithm for solving the SOLR problem.

In this study, we devise a linear relaxation of $y^i z_j = y^j z_i$ with $-y^i + l z_i \le 0$ and $y^i - u z_i \le 0$. Note that $1/\beta_i \le z_i \le 1/\alpha_i$ and $z_i > 0$ for all *i*. Thus we see that for each *i*

$$1/\beta_i \le y_i \text{ and } 1/\alpha_i \le y_i \text{ for all } i, j = 1, \cdots, p.$$

We consider two sets B_{curv} and B_{tria} that are defined below.

$$B_{\mathrm{curv}} := \left\{(t_{ij}, y^i, z_j) \mid t_{ij} = y^i z_j, 1/\beta_j \le z_j \le 1/\alpha_i, l/\beta_i \le y_i \le u/\alpha_i\right\}$$

and

$$B_{tria} := \left\{\left(t_{ij}, y^i, z_j\right) \left| \begin{array}{l} t_{ij} \le \dfrac{1}{\alpha_j} y^i + \dfrac{l}{\beta_i} z_j - \dfrac{l}{\alpha_j \beta_i}, \\ t_{ij} \le \dfrac{1}{\beta_j} y^i + \dfrac{u}{\alpha_i} z_j - \dfrac{u}{\alpha_i \beta_j}, \\ t_{ij} \ge \dfrac{1}{\alpha_j} y^i + \dfrac{u}{\alpha_i} z_j - \dfrac{u}{\alpha_i \alpha_j}, \\ t_{ij} \ge \dfrac{1}{\beta_j} y^i + \dfrac{l}{\beta_i} z_j - \dfrac{l}{\beta_i \beta_j}, \end{array} \right. 1/\beta_j \le z_j \le 1/\alpha_j, l/\beta_i \le y^i \le u/\alpha_i \right\}.$$

It is easy to see that

Lemma 1. *The relation* $B_{\mathrm{curv}} \subseteq B_{\mathrm{tria}}$ *holds.*

Proof: We first prove the case that $t_{ij} \le \dfrac{1}{\alpha_j} y^i + \dfrac{l}{\beta_i} z_j - \dfrac{l}{\alpha_j \beta_i}$. For $\forall (t_{ij}, y^i, z_j) \in B_{\mathrm{curv}}$, $i, j = 1, \cdots, p$.

$$\begin{aligned} &\frac{1}{\alpha_j} y^i + \frac{l}{\beta_i} z_j - \frac{l}{\alpha_j \beta_i} - t_{ij} \\ &= \frac{1}{\alpha_j} y^i + \frac{l}{\beta_i} z_j - \frac{l}{\alpha_j \beta_i} - y^i z_j \\ &= (\frac{1}{\alpha_j} - z_j)(y^i - \frac{l}{\beta_i}) \end{aligned}$$

Since $z_j \le 1/\alpha_j$ and $y_i \ge l/\beta_i$, we have $\dfrac{1}{\alpha_j} y^i + \dfrac{l}{\beta_i} z_j - \dfrac{l}{\alpha_j \beta_i} - t_{ij} \ge 0$. Using the similar way, we can easily prove that the other cases in the lemma hold. □

Lemma 1 indicate that the following problem $Q_1(l,u)$ can be exploited to find an lower bound for problem $Q_0(l,u)$ with a box $[l,u]$:

$$Q_1(l,u)\quad\left|\begin{array}{l} \min imize\ \sum_{i=1}^{p}\left(n_i^{\mathrm{T}}y^i + a_i z_i\right) \\ \\ \left.\begin{array}{l} subject\ to\ \ d_i^{\mathrm{T}}y^i + b_i z_i = 1, \\ \quad Ay^i - cz_i \le 0, \\ \quad \dfrac{1}{\beta_i} \le z_i \le \dfrac{1}{\alpha_i}, \\ \quad -y^i + lz_i \le 0,\ y^i - uz_i \le 0, \\ \quad t_{ij} \le \dfrac{1}{\alpha_j}y^i + \dfrac{l}{\beta_i}z_j - \dfrac{l}{\alpha_j\beta_i},\ t_{ij} \le \dfrac{1}{\beta_j}y^i + \dfrac{u}{\alpha_i}z_j - \dfrac{u}{\alpha_i\beta_j}, \\ \quad t_{ij} \ge \dfrac{1}{\alpha_j}y^i + \dfrac{u}{\alpha_i}z_j - \dfrac{u}{\alpha_i\alpha_j},\ t_{ij} \ge \dfrac{1}{\beta_j}y^i + \dfrac{l}{\beta_i}z_j - \dfrac{l}{\beta_i\beta_j}, \\ \quad t_{ij} = t_{ji}, \end{array}\right\}\ i,\ j = 1,2,\ldots,p. \end{array}\right. \tag{7}$$

Next we will develop a branch and bound algorithm to find the optimal solution to (3) by solving a series of the linear programming problems (7).

3. ALGORITHM AND ITS CONVERGENCE

In our algorithm, the branching process is executed in the space of *R*ⁿ. Suppose $B^k = \left\{x \in R\ |: \underline{x_i} \le x_i \le \overline{x_i}, i = 1,2,\cdots,n\right\}$ is a rectangle that contains an optimal solution in the process. Then B^k is divided into two subrectangles B^{2k}, B^{2k+1} according to the following bisection branch rules *R*1-*R*3:

*R*1. Let $i_0 \in \arg\max\left\{\overline{x_i} - \underline{x_i}\ |\ i = 1,2,\cdots,n\right\}$.

*R*2. Let $\gamma_{i_0} = \frac{1}{2}\left(\overline{x_{i_0}} + \underline{x_{i_0}}\right)$.

*R*3. Let $B^{2k} = \left\{x \in R^n\ |\ \underline{x_i} \le x_i \le \overline{x_i}, i \ne i0, \underline{x_{i_0}} \le x_{i_0} \le \gamma_{i_0}\right\}$,

$B^{2k+1} = \left\{x \in R^n\ |\ \underline{x_i} \le x_i \le \overline{x_i}, i \ne i_0, \gamma_{i_0} \le x_{i_0} \le \overline{x_{i_0}}\right\}$.

Starting from $B^1 = \{[l^1,u^1] = [l,u]\}$ that is generated by solving problem (1) and (2), we solve $Q_1(l^k,u^k)$ for $k = 1,2,\cdots$. The rectangle B^k will be discarded at the *k*-th iteration if the optimal value of $Q_1(l^k,u^k)$ is greater than the current best value at the solution $x^{i_0} = y^{i_0}/z_{i_0}$, $i_0 \in \arg\max\left\{f\left(x^{i_k}\right)|\ i_k = 1,2,\cdots,p\right\}$. Now the algorithm is ready to be described in detail as follows:

Let $\delta\left(\left[l^k,u^k\right]\right) := \max\left\{u_i^k - l_i^k\ |\ i = 1,2,\cdots,n\right\}$, which denote the size (diameter) of box $\left[l^k,u^k\right]$. Hereafter, we also use δ to denote the size of box. The convergence of **Algorithm 1** can be shown by the following Lemma 2 and Theorem 2.

Algorithm 1 Branch and bound algorithm for SOLR

```
1: Initial settings: Set k=1. Let $\mathcal{B}^k = \{[l^k,u^k]\} = \{[l,u]\}$ be the initial rectangle. $L = -\infty$, $U = +\infty$.
2: Solve $Q_1(l^j,u^j)$ with $B^j \in \mathcal{B}^k$. If $Q_1(l^j,u^j)$ is feasible then obtain the optimal value $L^k$ and
   $x^{i_k} = y^{i_k}/z_{i_k}$, and $U^k = \min\left\{f\left(x^{i_k}\right)|\ i_k = 1,2,\cdots,p\right\}$. $\mathcal{L}^k = L^k, U = U^k$. If $Q_1(l^k,u^k)$ is infeasible then
   terminate.
3: Set a tolerance $\varepsilon \ge 0$.
4: while $U - \mathcal{L}^k > \varepsilon$ do
5:     Discard rectangles $B^j \in \mathcal{B}^k$ such that the value of $Q_1(l^j,u^j) > U^k$.
```

```
6:  Select $B^{j_0} \in \mathcal{F}^k$ with $L^{j_0} = \mathcal{L}^k$.
7:  Divide $B^{j_0}$ into two subrectangles $B^{2k}$, $B^{2k+1}$ according branching rules R1-R3, and update
    $\mathcal{F}^k = (\mathcal{F}^k \setminus \{B^{j_0}\}) \cup \{B^{2k}\} \cup \{B^{2k+1}\}$.
8:  Solve $Q_1(l^j, u^j)$ for $j = k, 2k+1$. If $Q_1(l^j, u^j)$ is not feasible, then discard $B^j$. Otherwise, obtain
    $L^k, U^k, L^{2k+1}, U^{2k+1}$ and keep the corresponding incumbent best solution $x^{j_k} = y^{j_k} / z_{j_k}$.
9:  Update $\mathcal{L}^k = \min\{L^k, L^{2k+1}\}$ if $\mathcal{L}^k \le \min\{L^k, L^{2k+1}\}$, $U = \min\{U^k, U^{2k+1}\}$ if
    $U > \min\{U^k, U^{2k+1}\}$ and update the incumbent best solution.
10: Set $k=k+1$.
11: end while
12: Return $U$ as an $\varepsilon-\text{minimize}$ of $(P_0)$ with a minimizer $x^{j_k} = y^{j_k} / z_{j_k} \in [l, u]$.
```

Lemma 2. Suppose that (y^i, z_i^i) for $i = 1, \cdots, p$ are obtained from solving (7) and $x = y^{i_0} / z_{i_0} \in [l, u]$. Then

$$\left| \sum_{i=1}^{p} (n_i^{\mathrm{T}} y^i + a_i z_i) - \sum_{i=1}^{p} \frac{n_{i_0}^{\mathrm{T}} x + a_{i_0}}{d_{i_0}^{\mathrm{T}} x + b_{i_0}} \right| \to 0 \quad \text{as} \quad \delta\left(\left[l^k, u^k\right]\right) \to 0. \tag{8}$$

Proof: This proof is almost as the same as Theorem 1 in (Carlsson and Shi, (2013)). For self-contained we give the proof below.

Without loss of generality, we suppose that $i_0 = 1$. Let $x^i = y^i / z_i$. Since y^i, z_i are feasible solutions of (7), it follows that x^i for $i = 1, \cdots, p$ are feasible for (3). We see that for each *i*,

$$(n_i^{\mathrm{T}} y^i + a_i z_i) = \frac{n_{i_0}^{\mathrm{T}} x^i + a_i}{d_{i_0}^{\mathrm{T}} x^i + b_i}.$$

Let $x_\delta = x - x^i$, then $\|x_\delta\| \le \delta([l^k, u^k])$. Denote that $\tau_i := \text{maximize} \left\{ \left| n_i^T x^i + a_i \right| \mid x^i \in X \right\}$ and $\zeta := \text{maximize} \left\{ \|d_i\| \mid i = 1, \cdots, p \right\}$, $\mu := \text{maximize} \left\{ \|d_i\| \mid i = 1, \cdots, p \right\}$. Suppose $x^i = y^i / z_i$ for all $i = 1, \cdots, p$ and x are points in $[l^k, u^k]$. Then we can confirm the following inequalities.

$$\begin{aligned}
&\left| \sum_{i=1}^{p} (n_i^{\mathrm{T}} y^i + a_i z_i) - \sum_{i=1}^{p} \frac{n_{i_0}^{\mathrm{T}} x + a_{i_0}}{d_{i_0}^{\mathrm{T}} x + b_{i_0}} \right| \\
&= \left| \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^i + a_i}{d_i^{\mathrm{T}} x^i + b_i} - \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x + a_i}{d_i^{\mathrm{T}} x + b_i} \right| \\
&= \left| \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^i + a_i}{d_i^{\mathrm{T}} x^i + b_i} - \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^i + a_i + n_i^{\mathrm{T}} x_\delta}{d_i^{\mathrm{T}} x^i + b_i + d_i^{\mathrm{T}} x_\delta} \right| \\
&\le \left| \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^i + a_i}{d_i^{\mathrm{T}} x^i + b_i} - \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x^i + a_i}{d_i^{\mathrm{T}} x^i + b_i + d_i^{\mathrm{T}} x_\delta} \right| + \left| \sum_{i=1}^{p} \frac{n_i^{\mathrm{T}} x_\delta}{d_i^{\mathrm{T}} x^i + b_i + d_i^{\mathrm{T}} x_\delta} \right| \\
&\le \sum_{i=1}^{p} \left| \frac{n_i^{\mathrm{T}} x^i + a_i}{d_i^{\mathrm{T}} x^i + b_i} \cdot \frac{d_i^{\mathrm{T}} x_\delta}{d_i^{\mathrm{T}} x^i + b_i + d_i^{\mathrm{T}} x_\delta} \right| + \sum_{i=1}^{p} \left| \frac{n_i^{\mathrm{T}} x_\delta}{d_i^{\mathrm{T}} x^i + b_i + d_i^{\mathrm{T}} x_\delta} \right| \\
&\le \sum_{i=1}^{p} \left| \frac{n_i^{\mathrm{T}} x^i + a_i}{\alpha_i^2} \right| \left| d_i^{\mathrm{T}} x_\delta \right| + \sum_{i=1}^{p} \left| \frac{n_i^{\mathrm{T}} x_\delta}{\alpha_i} \right| \\
&\le \delta \sum_{i=1}^{p} \left(\frac{\tau_i \zeta}{\alpha_i^2} + \frac{\mu}{\alpha_i} \right)
\end{aligned}$$

Therefore, the assertion holds as $\delta \to 0$. □

Theorem 2. For a given tolerance ε , a global ε -minimizer of problem (P_0) can be found within finitely many iterations.

Proof: A sufficient condition for a global optimization to be convergent to the global minimum, for instance, stated in Hosrt and Tuy (1996), is that the bounding operation must be consistent and the selection operation must be bound improving. A bounding operation is called consistent if at every step any unfathomed partition can be further refined, and if at any infinitely decreasing sequence of successively refined partition elements satisfies:

$$\lim_{k \to +\infty} (U - \mathcal{L}^k) = 0, \tag{9}$$

where U and $\mathcal{L}^k$ are the computed upper bound and the current best bound at iteration k , respectively.

Since the subdivision process is bisection, the process is exhaustive. Clearly, Lemma 2 keeps (9) holding, which implies that the employed bounding operation in our algorithm is consistent.

A selection operation is called bound improving if at least one partition element where the actual lower bound is attained will be selected for further partition after a finite number of refinements. Clearly, the partition element at step 6 in Algorithm 1 where the current lower bound is attained will be selected for further partition at the next iteration in our algorithm. Therefore, the employed selection operation is bound improving. We complete the proof of convergence. □

4. NUMERICAL EXPERIMENTS

In this section, we will give a numerical example to compare the efficiency between the revision and its previous algorithm. We also report the results of the numerical experiments which were conducted using randomly generated data sets to verify the performance of the revised algorithm. The algorithm is coded in MATLAB® and implemented on Panasonic CF-SX3 with a quad-core Core(i7)-4600U in Muroran Institute of Technology.

We use the following Example 1 to see the empirical evidence that the new algorithm works efficiently well.

Example 1. (Chen *et al.*, (2005). *p*.78)

$$\begin{aligned}
\text{minimize} \quad & \frac{-x_1 + 2x_2 + 2}{3x_1 - 4x_2 + 5} + \frac{4x_1 - 3x_2 + 4}{-2x_1 + x_2 + 3} \\
\textit{subject to} \quad & x_1 + x_2 \le 1.5 \\
& x_1 \le x_2 \\
& 0 \le x_1 \le 1, 0 \le x_2 \le 1.
\end{aligned}$$

Example 1 was solved by Algorithm SOLiRat in (Carlsson and Shi, (2013)) and its revision Algorithm 1 that is proposed in this study, respectively. That is, Example 1 was solved based on two different linear relaxations Q_0 in (6) and Q_1 in (7) with $\varepsilon = 0.05$. The minimum of Example 1 is 1.62318. In Figure 1, the horizontal and vertical axises are the number of iterations in the process and the lower & upper bounds the algorithms obtained at the iterations, respectively. The blue line depicts the lower bounds and the red is for upper bounds which were calculated by the revised algorithm. We see that the gaps between the red and blue lines become very tiny after about only 5 iterations in this example. The gaps, in contrast, between the sky-blue and black lines keep a relative wider range even after 60 iterations. Figure 1 indicates that

- The new linear relaxation Q_1 is likely to be more efficient than Q_0.

To investigate the difference in the ability to produce a tighter lower bound between Q_0 and Q_1, we solve problem (P_0) with a variety of size (diameter) δ of the box constraints $l \le x \le u$. The problem used in this numerical experiment is with $p=5, n = 3$. Figure 2 indicates that

- The new linear relaxation Q_1 makes a better lower bound for any size δ of the box constraints.
- For both Q_0 and Q_1, the larger the size of the box constraints is, the worse the lower bounds become.
- The larger the size of box constraints is, the larger the difference in the lower bounds obtained from Q_0 and Q_1 is.

We also conducted the numerical experiments to evaluate the general behavior of the revised algorithm with lower dimension problems.

The datasets of the test problems used in this study are set as follows: The dimension of variables is 3 and the tolerance $\varepsilon = 0.05$. The coefficients n_i, d_i are *i.i.d.* generated in the ranges of $-5 \le n_{ij}, d_{ij} \le 5$, for $i = 1, \cdots, p$ and $j = 1,2,3$. The constants a_i $(i = 1, \cdots, p)$ are randomly chosen from $[0,50]$ and b_i $(i = 1, \cdots, p)$ are fixed to 50.

Problem (P_0) with a variety of $p = 2,5,10,15$ was solved. For a fixed p, a set of 10 instances of the problem was solved by model Q_0 and model Q_1, respectively. The recorded CPU times in second, the number of iterations and the number of branches in the execution are reported in Table 1. The columns titled 'AVERG.' provide the information about the average value of CPU time, the number of iterations and the number of branches out of the 10 runs in the execution. As their names indicate, the rows Q_1 delivers the results that obtained from model Q_1 while rows Q_0 for the results obtained from model Q_0. Table 1 indicates the following observations.

- The number of branches is heavily reduced at least about to 5% of the previous algorithm. A smaller number of branches in a branch-and-bound algorithm usually results in less time-consuming in implementation. In this study, the proposed algorithm consists of two parts: bounding based on a LP relaxation and branching. The LP can be solved in polynomial time, so the reduction of the number of branches is fundamentally important to design an algorithm.
- The proposed LP relaxation is very efficient for solving problem (P_0) when p is not large ($p \le 15$). The proposed algorithm achieves superiority over the previous algorithm in CPU time, number of iterations

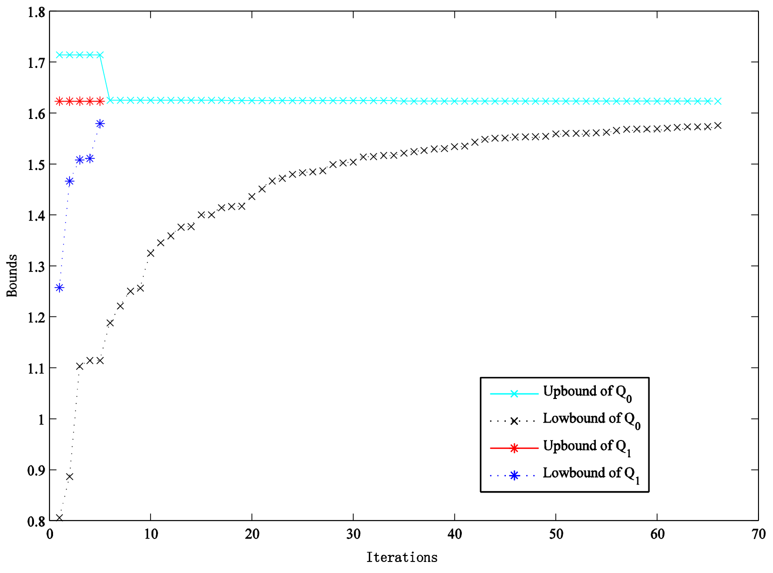

Figure 1: Bounds and iterations with $\varepsilon = 0.05$ for solving Example 1

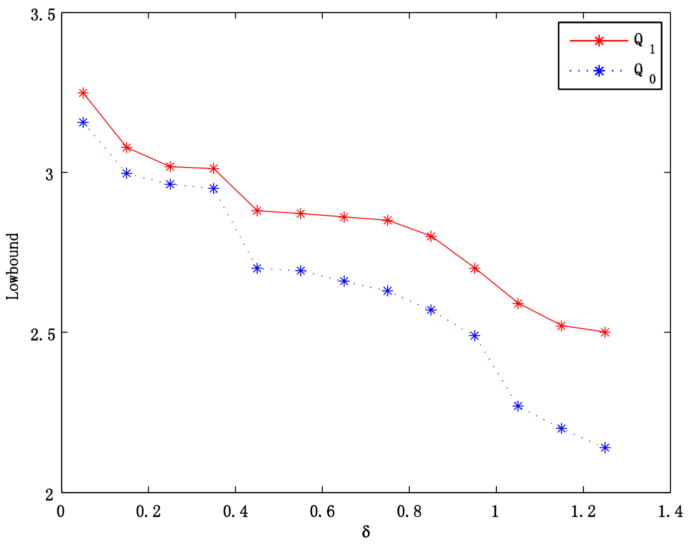

Figure 2: Average lowbound of two models with different size of box p=5, n=3

and number of branches on average, as well as max and min values.

Because that Q_1 is a linear programming with a number of $p^2n + pn + p$ variables, Q_1 has a large number of variables when p is large. We see that such a large number has two sides:

1) Theoretically, the size of $p^2n + pn + p$ is a polynomial of the size of input data, and LP can be solved by a polynomial time of the input data size, therefore we strongly expect that the proposed algorithm keeps its good efficiency even for a large-scale problem if we use a sophisticated software to solve the involved LP problems.

2) In reality, in our numerical experiments solving the problems with $p \geq 15$ the implementation reached the maximum number of iterations in MATLAB®. To take full advantage of this LP relaxation developed in this study the software products that can solve large scale linear programming efficiently are highly recommended to use.

Table 1: Numerical results for solving problem (P_0) with Q_1 and Q_0 with various p and $n=3$

Model	#of p	CPU time(s)			# of Iterations			# of Branches		
		MIN.	AVERG.	MAX.	MIN.	AVERG.	MAX.	MIN.	AVERG.	MAX.
Q_1	2	0.13	0.25	0.54	1	1.71	2	1	1.67	2
Q_0	2	0.45	2.30	6.18	3	17.00	48	3	27.33	89
Q_1	5	0.97	6.12	11.85	1	2.80	5	1	2.16	5
Q_0	5	3.55	20.32	36.10	12	54.58	117	14	88.33	200
Q_1	10	16.90	52.78	203.20	5	9.00	24	7	18.00	54
Q_0	10	27.14	125.46	243.32	46	192.42	362	71	317.67	622
Q_1	15	33.38	84.63	173.05	5	12.80	28	7	21.00	47
Q_0	15	549.43	3684.40	12537.86	538	3610.80	12434	893	6882.40	23972

Table 2: Results of the 4 instance of the numerical experiments with p=60 and n=3

Instance	Model	CPU time(s)	# of Iterations	# of Branches
I_1	Q_1	729.45	7	12
	Q_0	9162.30	4383	7424
I_2	Q_1	1278.31	12	18
	Q_0	27843.25	11166	18992
I_3	Q_1	436.94	5	9
	Q_0	1675.83	792	1142
I_4	Q_1	882.48	9	17
	Q_0	9066.27	3055	4957

To evaluate the performance of the proposed algorithm with a larger p, we set $p = 60$ and conducted the numerical experiments by using GUROBI to solve all LP problems for both model Q_1 and Q_0. Table 2 shows the results of 4 random instances with the fixed data $a_i = b_i = 48$ for all i, and the other datasets that were *i.i.d.* generated as same as previous. Table 2 indicates the following observations.

- The number of branches on average in Q_1 is only 0.34% of Q_0. Such a great reduction of the number of branches results in a sharp decrease in the CPU time for Q_1. Actually, the CPU time for Q_1 is about 12% of Q_0 on average.
- The proposed relaxation is much more efficient for solving problem (P_0) in the four instances with $p = 60$. Though we have only conducted the experiments with a fixed p, we firmly believe that, if we use an efficient LP solver for the large-scale problem, the proposed algorithm performs well because we have only a limited number of LP problems to solve in the branches of Q_1.

5. CONCLUSION AND FURTHER WORK

In this paper, we have made a new linear relaxation for the SOLR problem and designed a branch and bound algorithm for solving the SOLR problem with lower dimension. The proposed algorithm shares the similarities to the previous algorithm (Carlsson and Shi, (2013)) that its branching process works on a space with dimensions n, the dimensions of native variables while its bounding process works on a space with dimensions of $p^2n + pn + p$, p is the number of terms of ratios. Theoretically, the proposed algorithm finds an ε -minimizer for any pre-given $\varepsilon > 0$ within finitely many iterations.

We conducted the numerical experiments to investigate the behavior of the proposed algorithm. The results obtained from the small-scale experiments indicate that the proposed algorithm is superior to the previous algorithm in CPU time, number of iterations and numbers of branches. The proposed algorithm solving the large-scale problems with $p = 60$ is also very efficient. The numerical experiments show that the CPU time is only about 12% of the previous on average.

In this study the experiments are not very large-scale. We plan to conduct large-scale experiments to look into the detailed behavior of the new algorithm as further work. It is fundamentally important to make a theoretical analysis on the number of iterations in which the new algorithm finds a minimizer. An error bound between a minimizer and an ε -minimizer is an important feature. We leave them as our further work.

REFERENCES

1. Almogy, Y. and Levin, O. (1970). Parametric analysis of a multi-stage stochastic shipping problem. *Operational Research*, 69: 359–370.
2. Benson, H. P. (2007). Solving sum of ratios fractional programs via concave minimization. *Jouranl Optimization Theory Application*, 135(1): 1–17.
3. Cambini, A., Martein, L., and Schaible, S. (1989). On maximizing a sum of ratios. *Journal of Information and Optimization Sciences*, 10(1): 65–79.
4. Carlsson, J. G. and Shi, J. (2013). A linear relaxation algorithm for solving the sum-of-linear-ratios problem with lower dimension. *Operations Research Letters*, 41(4): 381–389.
5. Charnes, A. and Cooper, W. W. (1962). Programming with linear fractional functionals. *Naval Research logistics quarterly*, 9(3-4): 181–186.
6. Chen, D. Z., Daescu, O., Dai, Y., Katoh, N., Wu, X., and Xu, J. (2000). Optimizing the sum of linear fractional functions and applications. In Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, pages: 707–716. Society for Industrial and Applied Mathematics.
7. Chen, D. Z., Daescu, O., Dai, Y., Katoh, N., Wu, X., and Xu, J. (2005). Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications. *Journal of Combinatorial Optimization*, 9(1): 69–90.
8. Chen, D. Z., Daescu, O., Hu, X. S., Wu, X., and Xu, J. (2001). Determining an optimal penetration among weighted regions in two and three dimensions. *Journal of Combinatorial Optimization*, 5(1): 59–79.
9. Choi, J. C. and Bricker, D. L. (1996). Effectiveness of a geometric programming algorithm for optimization of machining economics models. *Computers & operations research*, 23(10): 957–961.
10. Daniels, K. (1995). The restrict evaluate subdivide paradigm for translational containment. In *Fifth MSI Stony Brook Workshop on Computational Geometry*.
11. Depetrini, D. and Locatelli, M. (2011). Approximation of linear fractional-multiplicative problems. *Mathmatical Programming*, 128(1-2): 437–443.
12. Falk, J. E. and Palocsay, S. W. (1994). Image space analysis of generalized fractional programs. *Journal of Global Optimization*, 4(1): 63–88.
13. Horst, R. and Tuy, H. (1996). *Global optimization: Deterministic approaches*. Springer.
14. Kim, J.-S. and Hong, K.-S. (2007). A recursive camera resectioning technique for on-line video-based augmented reality. *Pattern recognition letters*, 28(7): 842–853.
15. Konno, H. and Abe, N. (1999). Minimization of the sum of three linear fractional functions. *Journal of Global Optimization*, 15(4): 419–432.
16. Konno, H. and Inori, M. (1989). Bond portfolio optimization by bilinear fractional programming. *Journal of the Operations Research Society of Japan*, 32(2): 143–158.
17. Konno, H. and Watanabe, H. (1996). Bond portfolio optimization problems and their applications to index tracking: a partial optimization approach. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 39(3): 295–306.
18. 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(1): 65–81.
19. Kuno, T. (2002). A branch-and-bound algorithm for maximizing the sum of several linear ratios. *Journal of Global Optimization*, 22(1-4): 155–174.
20. Kuno, T. and Masaki, T. (2013). A practical but rigorous approach to sum-of-ratios optimization in geometric applications. *Computational Optimization and Applications*, 54(1): 93–109.
21. Majhi, J., Janardan, R., Schwerdt, J., Smid, M., and Gupta, P. (1999). Minimizing support structures and trapped area in two-dimensional layered manufacturing. *Computational Geometry*, 12(3): 241–267.
22. Matsui, T. (1996). Np-hardness of linear multiplicative programming and related problems. *Journal of Global Optimization*, 9(2): 113–119.
23. Michael, R. G. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness. *WH Freeman & Co., San Francisco.*
24. Schaible, S. (1977). A note on the sum of a linear and linear-fractional function. *Nava Research Logistics Quarterly*, 24(4):

691–693.
25. Schaible, S. and Shi, J. (2003). Fractional programming: the sum-of-ratios case. *Optimization Methods and Software*, 2(18): 219–229.
26. Skiscim, C. C. and Palocsay, S. W. (2001). Minimum spanning trees with sums of ratios. *Journal of Global Optimization*, 19: 103–120.
27. Wu, W.-Y., Sheu, R.-L., and Birbil, Ş. İ. (2008). Solving the sum-of-ratios problem by a stochastic search algorithm. *Journal of Global Optimization*, 42(1): 91–109.