International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

A New Computer Oriented Technique to Solve Sum of Ratios Non-Linear Fractional Programming Problems

R.Ramesh Babu[1], S.Divyaa Devi [2], V.Palanisamy[3]
[1]Department of Computer science and Engineering, Alagappa University, Karaikudi, India
[2] Department of Computer science and Engineering, Anna University of Technology Madurai, Dindigul, India
[3] Department of Computer science and Engineering, Alagappa University, Karaikudi, India
E-mail:svmr_babu@yahoo.com

Abstract - Normally the sum of ratios problems are reduced into a sequence of single-ratio problems and then solved by existing methods. Because of their combinatorial nature, the computational complexity grows exponentially. A sum of non-linear fractional function optimization problem with several fractions is proved to be a NP-complete problem indicating that an efficient algorithm may not exist. The non-linear sum of fractional functions are linearized by piecewise linearization technique and converted into a linear sum of fractional programming problem. Optimal solution is found at for the new problem. The number of iterations depends on the contribution of decision variables to objective function value.

I. INTRODUCTION

Sum-of-ratios fractional programs have attracted the interest of both practitioners and researchers, especially during the past 20 years. From a practical point of view, these problems have a variety of important applications. For instance, sum-of-ratios fractional programs arise whenever several rates are to be optimized simultaneously and a compromise solution is sought which optimizes a weighted sum of these rates [1]. Other applications of sum-of-ratios fractional programs arise in optimal clustering problems [2], in shipping problems [3], in government contracting problems [4], and in bond portfolio optimization [5]. From a research point of view, interest in sum of-ratios problems has been strong, especially in recent years. For detailed reviews of sum-of-ratios fractional programs, see [6, 7]. A branch and bound-outer approximation algorithm for globally solving problem (1) To solve problem (1), the algorithm instead globally solves an equivalent problem that seeks to minimize an indefinite quadratic function over a nonempty, compact convex set. To solve this problem, the algorithm combines a branch and bound search with an outer approximation process. From a computational point of view, the main work of the algorithm involves solving a sequence of lower bounding convex programming problems. Since the feasible regions of these convex programs are identical to one another except for certain linear constraints, to solve them, an optimal solution to one problem can potentially be used as an effective starting solution for the next problem.

II. NON-LINEAR PROGRAMMING PROBLEM WITH SUM OF RATIOS

A sum of non-linear fractional optimization problem has a number of fractional functions which are to be extremized.
Let
$Zn_1(x)\ Zn_2(x).....Zn_t(x),\ Zd_1(x),\ Zd_2(x).....Zd_t(x)$ *and* $g_1(x),\ g_2(x)...$ *and* $g_m(x)$
be real valued functions of n variables.

A sum of non-linear fractional objective functions with t number of fractions in its general form: all are non-linear, or some are non-linear then the problem of determining $(x_1, x_2, \ldots, x_n)$ which extremizes Z and satisfies 1 and .2 is called a General form of Non-Linear Fractional Programming Problem.

$x_j \geq 0,\ j = 1, 2, \ldots, n$

$If\ Z_{n1}(x_1, x_2, \ldots, x_n),\ Z_{n2}(x_1, x_2, \ldots, x_n)\ \ldots.\ Z_{nt}(x_1, x_2, \ldots, x_n)$

$Z_{d1}(x_1, x_2, \ldots, x_n),\ Z_{d2}(x_1, x_2, \ldots, x_n)\ \ldots.\ Z_{dt}(x_1, x_2, \ldots, x_n)$

$g_i(x_1, x_2, \ldots, x_n),\ i = 1, 2, \ldots, m$

$$\text{ie} \qquad g_{ij}(x_j) = \sum_{k=1}^{1} w_{i,k} x_j^{\,k}$$

III. SOLUTION FOR SEPARABLE SUM OF NON-LINEAR FRACTIONAL PROGRAMMING PROBLEM

IV.

An algorithm that has been suggested to solve separable sum of non-linear fractional programming problem consists of three phases.

1) In the first phase the sum of non-linear fractional programming problem is converted into a sum of linear fractional programming problem.

International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

2) In phase two, the promising variables are arranged according to the total contribution of the variables to the objective function.
3) In phase three, the ordered variables are entered into the basis one by one if it is still promising.

V. CONVERSION OF SEPARABLE SUM OF NON-LINEAR FRACTIONAL PROGRAMMING PROBLEM INTO A LINEAR SUM OF FRACTIONAL PROGRAMMING PROBLEM

This consists of two steps. In step one, the maximum value t_j of the variable x_j which satisfies all the constraints are computed. In step two the interval $[0,t_j]$ is divided into number of mesh points. For each mesh point compute piecewise linear approximation for each single variable objective function and the constraints. Corresponding to each linear approximation a variable is introduced. Also the addition constraints for the mesh point variables for the j^{th} variable should be equal to one. Further in this solution one or at the maximum two mesh points variables can be in the solution. If two mesh point variables are in the solution they should correspond to two successive mesh points and not random mesh points. The determination of t_j for the variable x_j depends not only on the polynomial function of the variable x_j in the i^{th} constraint and the power of that variable but also the type of the constraint.

VI. STEP BY STEP PROCEDURE FOR SEPARABLE NON-LINEAR FRACTIONAL PROGRAMMING PROBLEM

Step1: For all the promising variables determine t_j, the maximum value of the variable x_j, satisfying the given constraints.

Let $g_{ij}(x_j)$ be the polynomial function of the variable x_j in the i^{th} constraint and the degree be *l*.

where $w_{i,k}$ is the coefficient of the polynomial and l is a nonnegative integer. Let b_i be the resource available for the i^{th} constraint, if the i^{th} constraint is of equality type or of upper bound type if the i^{th} constraint is of equality type or of lower bound type

$$Extremize\ Z = \frac{Z_{a1}(x_1,x_2,\ldots,x_n)+Z_{a10}}{Z_{d1}(x_1,x_2,\ldots,x_n)+Z_{d10}} + \frac{Z_{a2}(x_1,x_2,\ldots,x_n)+Z_{a20}}{Z_{d2}(x_1,x_2,\ldots,x_n)+Z_{d20}} + \ldots\ldots \frac{Z_{a1}(x_1,x_2,\ldots,x_n)+Z_{a10}}{Z_{d1}(x_1,x_2,\ldots,x_n)+Z_{d10}}$$

subject to

$$g_1(x_1,x_2,\ldots,x_n) \le b_1$$
$$g_2(x_1,x_2,\ldots,x_n) \le b_2$$
$$\vdots$$
$$g_m(x_1,x_2,\ldots,x_n) \le b_m$$

$$(x_j)_k = \max_{k=1,2,\ldots,l} \left\{ \left(\frac{b_i}{w_{i,k}} \right)^{1/k} ;\ w_{i,k} > 0 \right\}$$

$$\sum_{j=1}^{n} m_j + n$$

$$t_j = \min_{i=1,2,\ldots,m} \left((x_j)_i \right)$$

$$(x_j)_i = \min_{k=1,2,\ldots,l} \left\{ \left(\frac{b_i}{w_{i,k}} \right)^{1/k} ;\ w_{i,k} > 0 \right\}$$

Step 2: Divide each $x_j\ (0 \le x_j \le t_j)$ values into m_j number of mesh points $(x_j)_k$ using the formula 5.

Step 3: For each point $(x_j)_k$ compute piecewise linear approximation for each $Zni(x_{j,k})$, $Zdi(x_{j,k})$ and the constraints $g_{ij}(x_{j,k})$.

The contribution coefficient of the new variable $x_{j,k}$ in the numerator function can be calculated by using the formula, Each variable x_j is replaced by $m_j + 1$ variables, the subscripts of the variables are computed using the relation The total number of variables in the problem will become

$$\begin{array}{c} x_1 \\ x_2 \\ \cdots \\ x_j \\ \cdots \\ x_n \end{array} \begin{bmatrix} b_1/a_{11} & b_2/a_{21} & \ldots & b_i/a_{i1} & \ldots & b_m/a_{m1} \\ b_1/a_{12} & b_2/a_{22} & \ldots & b_i/a_{i2} & \ldots & b_m/a_{m2} \\ \cdots & \cdots & & \cdots & & \cdots \\ b_1/a_{1j} & b_2/a_{2j} & \ldots & b_i/a_{ij} & \ldots & b_m/a_{mj} \\ \cdots & \cdots & & \cdots & & \cdots \\ b_1/a_{1n} & b_2/a_{2n} & \ldots & b_i/a_{in} & \ldots & b_m/a_{mn} \end{bmatrix}$$

Step 4: The contribution coefficient of the variable $x_{j,k}$ in thedenominator function is calculated by using the formula, The coefficient of the variable $x_{j,k}$ in the i^{th} constraint is calculated by using the formula

$$d\left(\sum_{p=0}^{j} m_p + k\right) = f_2((x_j)_k)$$

$$a_i\left(\sum_{p=0}^{j-1} m_p + k + j\right) = g_{ij}((x_j)_k)$$

The additional constraint for the j^{th} variable is

Step 5: Repeat Step 1 to 4 till all the variables are linearized.

Step 6: The linearized fractional programming problem will be as given below.

P_0 = *m x 1, matrices.*

Here n = number of variables in the non-linear programming problem and

m = number of constraints

Y = number of variables in this linearized fractional programming problem

C^T and D^T are the contribution coefficients of the variables in the linearized problem

International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

VII. IDENTIFICATION OF PROMISING VARIABLES

A matrix (known as θ matrix) [8,9,10] of intercepts of the decision variables along the respective axes with respect to the chosen basis is constructed. A typical intercept at start for the j^{th} variable x_j due i^{th} resource b_i *is* b_i / a_{ij}, $a_{ij} > 0$.

At start, the θ matrix is $S_1 \quad S_2 \quad S_3 \quad \ldots \quad S_m$

In this θ matrix, the row elements represent the k(≤ m) intercepts of the variable along the respective axes and the column represents the l(≤ n) intercept formed by each of the constraints.

In the subsequent passes the θ matrix is constructed using the following rules:

a) If the promising variable is a decision variable then:

$$\theta_{ki} = \min \left\{ \frac{(B^{-1}P_0)_i}{(B^{-1}P_j)_i}; \quad (B^{-1}P_j)_i > 0 \right\}$$

where i = 1 to m, k = 1 to *l* = 1.. m j = 1 to n.

b) If the promising variable is a decision variable then

$$\theta_{ki} = \min \left\{ \frac{(B^{-1}P_0)_i}{b_{ij}}; \quad b_{ij} > 0 \right\}$$

where b_{ij} is the i^{th} row and j^{th} column element of B^{-1}The promising variables are identified by following the steps given below:

Computation of 'θ matrix'

$$Maximize\ Z = \frac{C^T Y + c_0}{D^T Y + d_0}$$

$$subject\ to\ AY \le P_0$$

$$Y = 0 \quad or \quad 1$$

$$where \quad C^T = 1\ x \left(\sum_{j-1}^{n} m_j + n \right)$$

$$Y = \left(\sum_{j-1}^{n} m_j + n \right)\ x\ 1$$

$$D^T = 1\ x \left(\sum_{j-1}^{n} m_j + n \right)$$

$$A = m\ x \left(\sum_{j-1}^{n} m_j + n \right)$$

$$\sum y \left(\sum_{p=0}^{j-1} m_p + k + j \right) = 1$$

Step 1: The θ matrix is scanned row by row and the position of the minimum intercept in each row is identified.

Step 2: The first entering vector will be the most promising vector in the set, say j^{th} vector and the corresponding row in the matrix be k. This is an entering vector in the place of a vector corresponding to the column in which the minimum intercept lies in the θ matrix, say column i.

Step 3: If there are any other rows with minimum value in the i^{th} column, then all such rows as well as the k^{th} row and the i^{th} column are deleted.

Step 4: Steps 1 to 3 are repeated until all the rows are covered. The collection of vectors corresponding to the k^{th} row will be entering vectors and the vectors corresponding to i^{th} column are the leaving vectors.

$$Extremize\ Z = \frac{Z_{n1}(x_1, x_2, \ldots, x_n) + Z_{n10}}{Z_{d1}(x_1, x_2, \ldots, x_n) + z_{d10}} + \frac{Z_{n2}(x_1, x_2, \ldots, x_n) + Z_{n20}}{Z_{d2}(x_1, x_2, \ldots, x_n) + Z_{d20}} + \ldots\ldots\ldots \frac{Z_{ni}(x_1, x_2, \ldots, x_n) + Z_{ni0}}{Z_{di}(x_1, x_2, \ldots, x_n) + Z_{di0}}$$

$$subject\ to$$

$$g_1(x_1, x_2, \ldots, x_n) \le b_1$$
$$g_2(x_1, x_2, \ldots, x_n) \le b_2$$
$$\vdots$$
$$g_m(x_1, x_2, \ldots, x_n) \le b_m$$

Ordering of promising variables

Promising variables are ordered by following the steps given below:

Step 1: For each of the decision variables x_j, j = 1..n the values of the net evaluations of the numerator and denominator are computed using the formula:

Step 2: Compute the ratios $\frac{z_{nj} - c_j}{z_{dj} - d_j}$ for all the non-basic variables. Collect all the promising variables which do not satisfy the optimality condition. Let '*l*' denote the total number of such promising variables.

$$\begin{bmatrix} z_{n1} - c_1 \\ z_{d1} - d_1 \\ . \\ . \\ . \\ z_{nt} - c_t \\ z_{dt} - d_t \end{bmatrix} = \begin{bmatrix} 1\ 0\ 0\ \ldots\ldots\ 0\ \ 0\ C_{B1}B^{-1} \\ 0\ 1\ 0\ \ldots\ldots\ 0\ \ 0\ D_{B1}B^{-1} \\ . \\ . \\ . \\ 0\ 0\ 0\ \ldots\ldots\ 1\ \ 0\ C_{Bt}B^{-1} \\ 0\ 0\ 0\ \ldots\ldots\ 0\ \ 1\ D_{Bt}B^{-1} \end{bmatrix} \begin{bmatrix} -c_1 \\ -d_1 \\ . \\ . \\ . \\ -c_t \\ -d_t \\ P_j \end{bmatrix}$$

Step 3a: For the j^{th} promising variable when the i^{th} constraint is an equality constraint and the i^{th} basis variable is not a decision variable, compute

International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

$$r_{\min} = \min_{i=1..m} \left\{ \frac{(B^{-1}P_0)_i}{(B^{-1}P_j)_i}; (B^{-1}P_j)_i > 0 \right\}$$

If r_{min} exists then compute

$$u_j = \sum_{k=1}^{t} \frac{-(z_{nk} - c_k) * r_{\min} + z_{nk}}{-(z_{dk} - d_k) * r_{\min} + z_{dk}}$$

Else $u_j = -1$

If $u_j < Z$ then let $u_j = -1$.

3b: Compute u_j for all promising variables by repeating step 3a and then go to step 4.

Step 4: Arrange promising variables in the descending order of u_j values ($u_j > 0$) and place the corresponding subscripts in set J.

J = {Subscripts of the promising variables arranged in descending order of $u_j s$ }

Let l_1 represents the number of elements in set J and. $0 \le l_1 \le p,\ 1 \le p \le l.$ Terminate the procedure if $l_1 = l$ else go to step 5a.

Step 5a: For the j^{th} promising, if u_j is negative and the i^{th} constraint is lower bound constraint and the i^{th} basis variable is not a decision variable then, compute

$$r_{\min} = \max_{i=1..m} \left\{ \frac{(B^{-1}P_0)_i}{(B^{-1}P_j)_i}; (B^{-1}P_j)_i > 0 \right\}$$

If r_{min} exists then compute

$$v_j = \sum_{k=1}^{t} \frac{-(z_{nk} - c_k) * r_{\min} + z_{nk}}{-(z_{dk} - d_k) * r_{\min} + z_{dk}}$$

else $v_j = -1$

5b: Repeat step 5a to compute v_js for all the promising variables and then go to step 6.

Step 6 : Arrange the promising variables in the descending order of their v_j values ($v_j > 0$) and add their subscripts to set J from $(l_1+1)^{th}$ position onwards, next to u_js.

J = {Subscripts of the promising variables arranged in descending order v_js followed by u_js}

all v_js. Let l_2 denote the number of elements in the enlarged set J and $0 \le l_2 \le p,\ 1 \le p \le l$.

Terminate the procedure if l_2 = total number of promising variables l else go to step 7a.

Step 7a: For the j^{th} promising variable, if both u_j and v_j are negative and either i^{th} basic variable is a decision variable or the corresponding constraint is an upper bound constraint then compute:

$$r_1 = \min_{i=1..m} \left\{ \frac{(B^{-1}P_0)_i}{(B^{-1}P_j)_i}; (B^{-1}P_j)_i > 0 \right\}$$

If the i^{th} basis variable is a feasible slack variable then compute

$$r_2 = \min_{i=1..m} \left\{ \frac{(B^{-1}P_0)_i}{(B^{-1}P_j)_i} \right\}$$

$$r_{\min} = \min\{r_1, r_2\}$$

If r_{min} exists compute

$$w_j = \sum_{k=1}^{t} \frac{-(z_{nk} - c_k) * r_{\min} + z_{nk}}{-(z_{dk} - d_k) * r_{\min} + z_{dk}}$$

else $w_j = -1$.

7b: Repeat step 7a until all w_js are computed for all the promising variables and then go to step 8.

Step 8 : Arrange the promising variables in the descending order of their w_j values ($w_j > 0$) and add their subscripts to set J from $(l_2+1)^{th}$ position onwards.

J = {Subscripts of the promising variables arranged in descending order w_js followed by v_js followed by u_js}

Let l_3 denote the number of elements in the enlarged set J

Now $l_3 = l$, the total number of promising variables.

Arrange promising variables in descending order of their w_j values and add it to the set J. Let l_3 denote the number of elements in enlarged set J. Now $l_3 = l$, the total number of promising variables.

8. Algorithm to solve sum of ratios fractional programming problem

Step 1 : Convert the sum of ratios non linear fractional programming problem into sum of linear fractional programming problem using piecewise linearization procedure.

2: Slack variables are added and the initial basic solution X_B is determined. B matrix, C_{Bi} and D_{Bi} matrices are formed.

2a: The optimality conditions of the solution are checked and promising variables are identified and arranged using the procedures 3.5 and 3.6. The set J is constructed. Let l be the number of elements in set J.

Let p=1 and go to step 3a.

2b: Under non optimal conditions increment p, go to step 9 without updating the value of Z.

2c: Under optimal conditions, the feasibility is verified. The process is terminated if the solution is feasible.

Step 3a: Take the variable corresponding to the p^{th} element of the set J. Let the subscript of the selected variable be j.

For each of the decision variables x_j, the values of the net evaluations of the numerator and denominator are computed using the formula:

International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

$$\begin{bmatrix} z_{n1}-c_1 \\ z_{d1}-d_1 \\ . \\ . \\ . \\ z_{nt}-c_t \\ z_{dt}-d_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \ldots\ldots & 0 & 0 & C_{B1}B^{-1} \\ 0 & 1 & 0 & \ldots\ldots & 0 & 0 & \mathbf{D}_{B1}B^{-1} \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ 0 & 0 & 0 & \ldots\ldots & 1 & 0 & C_{Bt}B^{-1} \\ 0 & 0 & 0 & \ldots\ldots & 0 & 1 & \mathbf{D}_{Bt}B^{-1} \end{bmatrix} \begin{bmatrix} -\mathbf{c}_1 \\ -\mathbf{d}_1 \\ . \\ . \\ . \\ -\mathbf{c}_t \\ -\mathbf{d}_t \\ P_j \end{bmatrix}$$

3b: Compute the ratios

$$R_i = \frac{(z_{ni}-c_i)}{(z_{di}-d_i)} \quad \{i=1,2,3,..t\}$$

If all the ratios do not satisfy optimality conditions, then go to step 7.

If at least one ratio satisfies the optimality conditions, then x_j may still be promising.

3c: A new column α is constructed using the formula $\alpha_i = \left(B^{-1}P_j\right)_i$ where i=1,2...m. The solution vector $\left(B^{-1}P_0\right)$ is also obtained.

Step 4a: Procedure to find leaving vector.

If the i^{th} constraint is an equality constraint and the i^{th} basic variable is not a decision variable, find

$$r_{1j} = \min_{i=1...m}\left\{\frac{(B^{-1}P_0)_i}{\alpha_i} \; ; \; \alpha_i > 0\right\}$$

If r_{1j} exists then the i^{th} basic variable corresponding to r_{1j} is the leaving variable, and r_{1j} is the value of the leaving variable. i.e., $x_j = r_{1j}$, Go to step 5, else go to step 4b.

4b: If the i^{th} constraint is a lower bound constraint and the i^{th} variable is not a decision variable, find

$$r_{2j} = \max_{i=1...m}\left\{\frac{(B^{-1}P_0)_i}{\alpha_i}; \alpha_i > 0\right\}$$

If r_{2j} exists then the i^{th} variable corresponding to r_{2j} is the leaving variable and $x_j = r_2$. Go to Step 5 else go to Step 4c.

4c: If the i^{th} basic variable is a decision variable or i^{th} constraint is an upper bound constraint, then find

$$\theta_i = \left\{\frac{(B^{-1}P_0)_i}{\alpha_i}; \alpha_i > 0\right\}$$

If the i^{th} basic variable is the feasible slack variable, then find

$$\theta_i = \left\{\frac{(B^{-1}P_0)_i}{\alpha_i} \; ; \; if \;\; \alpha_i < 0, (B^{-1}P_0)_i < 0\right\}$$

Compute $r_{3j} = \min_{i=1...m}\{\theta_i\}$

If r_{3j} exists, then i^{th} variable corresponding to r3 is the leaving variable and $x_j = r_{3j}$

Step 5: Compute improvement formula

$$u_j = \sum_{i=1}^{t} \frac{-(z_{ni}-c_i)*x_j + z_{ni}}{-(z_{di}-d_i)*x_j + z_{di}}$$

If value of u_j is greater than previous value of Z then go to step 5 else go to step 8.

Step 6: Computation of E matrix

The transformation matrix corresponding to the new entering and leaving variables can be obtained by using the product form of inverse. In the first step η vector is computed. Let j be the subscript of the entering variable and r be the column vector in which j^{th} variable enters. The column vector corresponding to the j^{th} vector is used to calculate η vector by using the relation.

$$\eta_1 = \mathbf{z}_{n1} - \mathbf{c}_1$$
$$\eta_2 = \mathbf{z}_{d1} - \mathbf{d}_1$$
$$\eta_3 = \mathbf{z}_{n2} - \mathbf{c}_2$$
$$\eta_4 = \mathbf{z}_{d2} - \mathbf{d}_2$$
$$\eta_n = \mathbf{z}_{nt} - \mathbf{c}_t$$
$$\eta_d = \mathbf{z}_{dt} - \mathbf{d}_t$$
$$\eta_{t+2} = \left[B^{-1}P_j\right]_i \qquad i = 1 \ldots m$$

Computation of η_{new}

Since the variable corresponding to the r^{th} row is the leaving variable $(r+2t)^{th}$ element in the η vector is the pivotal element. η_{new} can be obtained using the following relation:

$$\eta_{i\,new} = \frac{-\eta_{iold}}{(B^{-1}P_j)_r} \text{ when } i \neq r$$
$$\eta_{r\,new} = \frac{1}{(B^{-1}P_j)_r}$$

Replace the r^{th} column of $(m+2t)\;x\;(m+2t)$ unit matrix by the η_{new} vector. The resulting matrix is the transformation matrix E.

Step 7: Computation of M^{-1}

Compute the inverse matrix for the next iteration as

$$M^{-1}_{next} = E * M^{-1}_{current}$$

Step 8: Let $p = p + 1$ If $p \leq l$ go to step 3a else go to step 2a.

Step 9: Compute the optimal solution using the formula

$$\mathbf{Z} = \sum_{i=1}^{t} \frac{C_{Bi}{}^T X_B + c_{0i}}{D_{Bi}{}^T X_B + d_{0i}}$$

VIII. CONCLUSION

A new algorithm for solving linear fractional optimization problem with sum of ratios in its objective function is presented. Typically execution time grows very rapidly with the number of ratios. Methods in the literature handles problems with fewer number of variables or fewer number of ratios only. The proposed algorithm can handle any number of ratios and any number of variables. The algorithm is scalable to arbitrary number of ratios and variables, provided the necessary computing power is at hand. The proposed

International Journal of Computing Algorithm
Volume: 01 Issue: 01 June 2012 Page No.1-6
ISSN: 2278-2397

algorithm is computationally tested and it is found that the execution time and number of iterations are saved by this algorithm.

References

[1] Frenk, J.B.G., Scheible, S.: Fractional programming. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, vol. II, pp. 162–172. Kluwer, Dordrecht (2001)

[2] Rao, M.R.: Cluster analysis and mathematical programming. J. Am. Stat. Assoc. 66, 622–626 (1971)

[3] Almogy, Y., Levin, O.: Parametric analysis of a multi-stage stochastic shipping problem. In: Proceedings of the 5th IFORS Conference (IFORS), pp. 359–370 (1964)

[4] Colantoni, C.S., Manes, R.P., Whinston, A.: Programming, profit rates, and pricing decisions. Account. Rev. 44, 467–481 (1969)

[5] Konno, H., Inori, M.: Bond portfolio optimization by bilinear fractional programming. J. Oper. Res. Soc. Jpn. 32, 143–158 (1989)

[6] Schaible, S.: Fractional programming. In: Horst, R., Pardalos, P. (eds.) Handbook of Global Optimization, pp. 495–608. Kluwer, Dordrecht (1995)

[7] Schaible, S.: Fractional programming with sums of ratios. In: Castagnoli, E., Giorgi, G. (eds.) Scalar and Vector Optimization in Economic and Financial Problems, Proceedings of theWorkshop in Milan, 1995, pp. 163–175. Elioprint, Modena (1995)

[8] Naganathan E R (1997). A Ratio Multiplex Algorithm to solve large scale linear programming problems, Ph.D. thesis, Department of Computer Science and Engineering, Alagappa University, Karaikudi

[9] Nagarajan S, “A Complex Algorithm to Solve Large Scale Linear Programming Problems”, Ph.D. thesis, Department of Computer Science and Engineering, Alagappa University, Karaikudi

[10] Sakthivel,S., “A Multiplex algorithm to solve large scale linear programming problems”, Ph.D. thesis, Anna University, India, 1984.

[11] Meyyappan. T “A new Computer Oriented Technique to Solve Large Scale Multi Objective Optimization Problems”, Ph.D. thesis, Department of Computer Science and Engineering, Alagappa University, Karaikudi

[12] Palanisamy.V “Computer oriented algorithms to solve special cases of non-linear fractional programming problems” Ph.D. thesis, Department of Computer Science and Engineering, Alagappa University, Karaikudi