CHAPTER 8
SOFTWARE TO IMPLEMENT PROPOSED ALGORITHMS

8.1 Introduction

Three algorithms, namely sum of ratios linear fractional programming algorithm, linear multi objective optimization algorithm, genetic algorithm for linear multi objective optimization are presented and numerical examples illustrated in earlier chapters. For solving large size linear fractional programming problems and to test the computational efficiency of the developed algorithms over the existing algorithms, computer codes have been developed and implemented using Visual C++ programs.

8.2 Structures and Functions of Sum of Ratios Fractional Programming Algorithm

The programs of the sum of ratios fractional programming algorithm contains the following functions:

PROBLEM_INPUT() function enables the user to input the following:

i) Number of objective fractions (t)
ii) Number of variables (n)
iii) Number of constraints(m)
iv) coefficients of decision variables in objective fractions and constant values.
v) coefficients of decision variables in constraints, right hand side value and type of each constraint.
Matrix A is created with coefficients of decision variables in objective fractions and matrix P is created with coefficients of decision variables in constraints. \( p_0 \) vector is created with right hand side values of constraints.

**FORMULATE_PROBLEM()** function adds slack or surplus variables based on constraint types and creates \( C_B \ D_B \) matrix of size \( 2t \times m \) and unit matrix \( U \) of size \( (2t \times 2t) \) matrix and \( B \) matrix of size \( (m \times m) \) are created.

**INITIAL_SOLUTION()** function computes the initial solution \((Z)\) with \( c_0 \) and \( d_0 \) constant values in objective fractions as follows:

\[
Z = \frac{c_{01}}{d_{01}} + \frac{c_{02}}{d_{02}} + \ldots + \frac{c_{0t}}{d_{0t}}
\]

**MAKE_THETA_MATRXO** function creates theta matrix with intercepts of the decision variables \( a_{ij}/b_i \) due to \( i^{th} \) resource \( b_i \) where \( a_{ij}/b_i > 0 \). Minimum of the intercepts for equality constraints, maximum of the intercepts for lower bound constraints and minimum of the intercepts for upper bound constraints are found. Overall minimum value of these minimums are found. Contribution by each variable \( x_j \) to sum of ratios of objective fractions is computed with this minimum value.

**ORDER_PROMIS_VARO** function arranges decision variables in descending order of their contribution to sum of ratios of objective fractions and their subscripts are added to matrix \( J \). Number of such variables is counted and the value is stored in a variable \( l \).

**MAKE_CJNJDJPJ_MATEX()** function creates \( C_{jn}D_{jdj}P_{j} \) vector by transferring coefficients \((C_{j}D_{j})\) of the \( j^{th} \) decision variable in matrix \( A \) followed by coefficients of \( j^{th} \) decision variable in constraint matrix \( P \).
MAKE_ZNJMCJ_MATRIX( ) function creates a vector containing $Z_{nj} - C_j$, $Z_{dj} - D_j$ values by multiplying $1 C_B B^{-1} D_B B^{-1}$ matrix with $C_{nj}D_{dj}P_j$ matrix. It is used to find the promising variables at each pass. For the first pass all the variables are assumed to be promising.

MAKE_C0D0P0_MATRIX() function creates C0D0P0 vector by transferring constant values in each objective fractions ($c_0$ and $d_0$) from matrix A followed by values from $P_0$ vector.

MAKE_ZMATRIX( ) function creates Z matrix by multiplying with by multiplying $[1 C_B B^{-1} D_B B^{-1}]$ matrix with C0D0P0 vector. The resulting vector contains $Z_{ni}$ and $Z_{di}$ values for all the objective fractions.

For subsequent passes, the ratios $R_{ji} = Z_{nj} - C_j / Z_{dj} - D_j$ values are computed using the Znjmcj matrix and the ratios $Z_i = z_{ni} / z_{di}$ are computed using Z matrix. If all the $R_{ji}$ ratios are greater than $Z_i$ ratios, the variables are still promising. If one or more ratios are not greater, then the following improvement formula is computed:

$$
\sum_{i=1}^{n} \sum_{j=1}^{m} \frac{c_{0i} + c_j x_j}{d_{0i} + d_j x_j}
$$

Objective function value $Z$ is computed using the formula:

$$
\sum_{i=1}^{n} \frac{Z_{ni}}{Z_{di}}
$$

If the improvement formula value is greater than or equal to $Z$ value, the variable $x_j$ is still promising and allowed to enter into basis else the variable is skipped.
**COMPUTE_BINVP0( )** function creates a vector binvp0 by multiplying $B^{-1}$ matrix with $P_0$ vector.

**COMPUTE_BINVPj( )** function creates a vector binvpj by multiplying $B^{-1}$ matrix with $P_j$ vector.

**FIND_LEAVE_VAR( )** function finds the leaving variable using binvp0 and binvpj vectors. It computes the following:

i) $r_{ij} = \min \left\{ \frac{B^{-1}P_0}{B^{-1}P_j} ; B^{-1}P_j > 0 \right\}$ is computed when the $i^{th}$ variable in the basis corresponds to an equality constraint.

ii) $r_{2j} = \min \left\{ \frac{B^{-1}P_0}{B^{-1}P_j} ; B^{-1}P_j > 0 \right\}$ is computed when the $i^{th}$ variable in the basis corresponds to lower bound constraint.

iii) $r_{3j} = \min \left\{ \frac{B^{-1}P_0}{B^{-1}P_j} ; B^{-1}P_j > 0 \right\}$ and $\left\{ \frac{B^{-1}P_0}{B^{-1}P_j} ; B^{-1}P_j < 0 \right\}$ is computed when the $i^{th}$ variable in the basis corresponds to an upper bound constraint or the decision variable corresponds to a lower bound constraint and has infeasible value.

If $r_{ij}$ exists then the variable corresponding to $r_{ij}$ is the leaving variable.

If $r_{2j}$ exists then the variable corresponding to $r_{2j}$ is the leaving variable.

If both $r_{ij}$ and $r_{2j}$ do not exist and if $r_{3j}$ exists then the variable corresponding to $r_{3j}$ is the leaving variable.

**MAKE_EMATRIX( )** function creates a transformation matrix $E$. A new unit matrix $E$ of size $(m+2t) \times (m+2t)$ is created. 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, \( \eta \) vector is computed. In the second step, \( \eta_{\text{new}} \) is computed using the pivotal \((r+2t)\) element, where \( r \) is the column vector in which \( j \) variable enters. Matrix obtained by replacing the \( r \) column of \((m+2t) \times (m+2t)\) unit matrix by the \( \eta_{\text{new}} \) vector is the transformation matrix \( E \).

**MAKE_MINV_MATRIX( )** function creates current \( M^{-1} \) matrix of size \((m+2t) \times (m+2t)\) comprising \( C_B B^{-1} \), \( D_B B^{-1}(2t \times m) \), \( U(2t \times 2t) \) matrix, and \( B^{-1} \) \((m \times m)\) are created.

**MAKE_MINV_NEXT_MATRIX( )** function updates current \( M^{-1} \) matrix for next iteration by multiplying transformation matrix \( E \) with current \( M^{-1} \) matrix as \( M^{-1}_{\text{next}} = E \times M^{-1}_{\text{current}} \).

The process is repeated until all the \( l \) number of promising variables\((l)\) in set \( J \) are checked for promisibility and entered into basis or skipped as the case may be.

**SOLUTION( )** computes the optimal solution for the problem after all iterations using the formula:

\[
Z = \sum_{i=1}^{l} \frac{C_{B_i}^\top X_B + c_{0_i}}{D_{B_i}^\top X_B + d_{0_i}}
\]

**MATRIX_DISPLAY( )** function is the general function that displays the content of any matrix in matrix form. It is passed with the matrix along with number of rows and columns in it. It is used by other functions to show the resultant matrices.
8.3 Structures and Functions of Linear Multi Objective Optimization Algorithm

**PROBLEM_INPUT()** function enables the user to input the following:

i) Number of objective functions (t)
ii) Number of variables (n)
iii) Number of constraints (m)
iv) coefficients of decision variables in objective functions and constant values.
v) coefficients of decision variables in constraints, right hand side value and type of each constraint.

Matrix A is created with coefficients of decision variables in objective fractions. P matrix is created with coefficients of decision variables in constraints. P0 vector is created with right hand side values of constraints. The original objective function rows in A matrix is saved in another array A_save.

**FORMULATE_PROBLEM( )** function adds slack or surplus variables based on constraint types and creates C_D matrix of size 2t x m and unit matrix U of size (2t x 2t) matrix and B matrix of size (m x m).

**COMBINO** function generates possible combinations with different orderings of objective functions that can be formed as sum of ratios with objective functions playing the roles of numerator and denominator(Zn and Zd). For a problem with 4 objective functions the combinations are formed are as follows

1. Z1, Z2, Z3, Z4
2. Z1, Z2, Z4, Z3
3. Z1, Z3, Z2, Z4
4. Z1, Z3, Z4, Z2
These orderings are stored in an array CN_SET[tc][t], where tc represents total number of combinations and t represents number of objective functions.

The first combination of objective functions is taken from CN_set matrix. The TRANSFER_OBJECTIVES( ) function transfers the objective function rows from A_save matrix to A matrix as per the combination taken from CN_set.

INITIAL_SOLUTION( ) function computes the initial solution (Z) with $c_0$ and $d_0$ constant values in objective functions as follows:

$$Z = \frac{c_{01}}{d_{01}} + \frac{c_{02}}{d_{02}} + ... + \frac{c_{0t}}{d_{0t}}$$

MAKE_THETA_MATRIX() function creates theta matrix with intercepts of the decision variables $a_{ij}/b_i$ due to $i^{th}$ resource $b_i$ where $a_{ij}/b_i > 0$. Minimum of the intercepts for equality constraints, maximum of the intercepts for lower bound constraints and minimum of the intercepts for upper bound constraints are found. Lowest value among the minimum value is found. Contribution by each variable $x_j$ to sum of ratios of objective fractions is computed with this minimum value.

203
CHECK_IMPROVE_CONDNS() function checks the 16 conditions for improvement given in Table I and determines whether the variables are still promising. If one more conditions are satisfied, then the following improvement formula is computed:

\[
\sum_{i=1}^{l} \sum_{j=1}^{n} \frac{c_{0i} + c_{i}x_{j}}{d_{0i} + d_{i}x_{j}}
\]

Objective function value \( Z \) is computed using the formula:

\[
\sum_{i=1}^{l} \frac{z_{ui}}{z_{di}}
\]

If the improvement formula value is greater than or equal to \( Z \) value, the variable \( x_{j} \) is still promising and allowed to enter into basis else the variable is skipped.

The following functions perform the same operations as in sum of ratios fractional programming algorithm (section 8.2):

ORDER_PROMIS_VAR(),
MAKE_CJNJDJPJ_MATRIX(),
MAKE_ZNJMCJ_MATRIX(),
MAKE_C0D0P0_MATRIX(),
MAKE_ZMATRIX(),
COMPUTE_BINVP0(),
COMPUTE_BINVPj(),
FIND_LEAVE_VAR(),
MAKE_EMATRIX(),
MAKE_MINV_MATRIX(),
MAKE_MINV_NEXT_MATRIX()
SOLUTION() computes the optimal solution for the current combination after all iterations using the formula:

\[ Z = \sum_{i=1}^{l} \frac{C_{B_i}^T X_B + c_{0i}}{D_{B_i}^T X_B + d_{0i}} \]

The solution obtained for the current combination is saved in COMBINE_SOLN array. The objective rows in A_save matrix are transferred to A matrix as per the next combination and the process is repeated until all the combinations are used.

FINAL_SOLUTION() function outputs the optimal solution stored in COMBINE_SOLN array and displays the best optimal solution and its objective function ordering.

MATRIX_DISPLAY() function is the general function that displays the content of any matrix in matrix form. It is passed with the matrix along with number of rows and columns in it. It is used by other functions to show the resultant matrices.

8.4 Structures and Functions of Genetic Algorithm for Multiojective Optimization Algorithm

The programs of multi objective optimization genetic algorithm contains the following functions:

PROBLEM_INPUT() function enables the user to input the following:

i) Number of objective fractions (t)
ii) Number of variables (n)
iii) Number of constraints(m)
iv) Number of generations (ng)
v) Number of parents (np)
vi) Coefficients of decision variables in objective fractions and constant values.
vii) Coefficients of decision variables in constraints, right hand side value and type of each constraint.

Matrix A is created with coefficients of decision variables in objective functions and matrix P is created with coefficients of decision variables and right hand side values in constraints. The type of constraints are also stored.

**FORMULATE_PROBLEM()** function adds slack or surplus variables based on constraint types.

**COMBIN()** function generates possible combinations with different orderings of objective functions that can be formed as sum of ratios with objective functions playing the roles of numerator and denominator ($Z_n$ and $Z_d$). For a problem with 4 objective functions the combinations are formed are as follows

1:: $z_1,z_2,z_3,z_4$
2:: $z_1,z_2,z_4,z_3$
3:: $z_1,z_3,z_2,z_4$
4:: $z_1,z_3,z_4,z_2$
5:: $z_1,z_4,z_2,z_3$
6:: $z_1,z_4,z_3,z_2$
7:: $z_2,z_1,z_4,z_3$
8:: $z_2,z_4,z_3,z_1$
9:: $z_3,z_1,z_4,z_2$
10:: $z_3,z_4,z_2,z_1$
11:: $z_4,z_1,z_3,z_2$
12:: $z_2,z_3,z_4,z_1$
These orderings are stored in an array CN_SET[tc][t], where tc represents total number of combinations and t represents number of objective functions.

The first combination of objective functions is taken from CN_set matrix. The `TRANSFER_OBJECTIVES( )` function transfers the objective function rows from A_save matrix to A matrix as per the combination taken from CN_set.

`MAKE_THETA_MATRIXO` function creates theta matrix with intercepts of the decision variables $a_{ij}/b_i$ due to $i^{th}$ resource $b_i$ where $a_{ij}/b_i > 0$. Minimum of the intercepts for equality constraints, maximum of the intercepts for lower bound constraints and minimum of the intercepts for upper bound constraints are found. Overall minimum value of these minimums are found. Contribution by each variable $x_j$ to sum of ratios of objective fractions is computed with this minimum value. Redundant minimums in each column of the theta matrix is resolved in favour of decision variable with higher contribution to objective function ratio and those variables are selected. The remaining decision variables are left out. Lower bounds of decision variables are set to zero. Upper bounds of selected decision variables are set to the minimum value found in each row of the theta matrix.

The variable $g$ is initialized to 1 and the first generation starts. For the first generation alone, new parents are created using `CREATE_PARENTS()` function. It creates np number of parents. Each parent contains random numbers (chromosomes) for m number of variables selected from theta matrix.

`CONS_CHECKQ` function checks all the constraints by substituting the random numbers in each parent. If any of the constraints are violated, a new set of
random numbers are chosen for the parent. This is repeated until the random numbers chosen satisfy all the constraints.

**FIND_FITNESS_VALUE()** function substitutes values of decision variables in each parent into objective functions and computes their value. The sum of ratios of objective functions in current combination is computed and taken as fitness value. It is repeated for all the parents and the fitness value of each parent is stored in a column next to the last variable in parents[] array.

**FIND_MAX_FITNESS()** finds the parent that has maximum fitness value. It stores generation number, chromosomes in the parent and maximum fitness value into gen_max_fitness array.

**SELECT_PARENTS()** function selects pairs of parents at random using random numbers between 0 to np-1.

**CROSSOVER()** function performs linear cross over by exchanging every odd numbered chromosomes in between a pair of parents. Constraint violation, if any is checked using cons_check() function.

**MUTATION()** function performs random mutation in case of any constraint violation after cross over operation. It slightly changes the random number obtained after crossover and checks all the constraints. Mutation operation is repeated until all the constraints are satisfied.

**FIND_MAX_FITNESS()** function to store the maximum fitness in the current generation into gen_max_fitness, as before.

The maximum among the maximum fitness obtained in each generation is 208.
found. Combination number, chromosomes in parents and fitness value are stored into COMBIN_SOLN array. The process is repeated for each combination of objectives.

**FINAL_SOLUTION()** function outputs the optimal solution stored in COMBIN_SOLN array and displays chromosomes (values of decision variables) and fitness value.

**MATRIX_DISPLAY()** function is the general function that displays the content of any matrix in matrix form. It is passed with the matrix along with number of rows and columns in it. It is used by other functions to show the resultant matrices.
8.5 Flow Chart For Implementing Sum Of Ratios Fractional Programming Algorithm

START

Input \( n_v, n_c \)

Input coefficients of objective functions and constraints. Construct \( C_B, D_B, X_B \) and \( B^{-1} \) matrices

Promising variables are arranged based on constraint type and contribution to sum of objective fractions. Add the subscripts of promising variable to set \( J \).

Let \( I = \) total number of promising variables

Construct Initial \( M^{-1} \) matrix using \( C_B, D_B \) and \( B^{-1} \) matrices.

Is solution optimal?

No

B

Yes

Compute Optimal Solution using the formula

\[
Z = \sum_{i \in J} \frac{C_i \cdot X_B + c_{0i}}{D_{Bi}} \quad \text{or} \quad M^{-1} \times P_0
\]

Output Optimal Solution

STOP
B

p = 1

Select the $p^{th}$ element from set $J$ and let it be subscript $j$ of the promising variable $x_j$

Compute the ratios $\frac{z_{nj} - c_j}{z_{dj} - d_j}$ and $\frac{Z_{ni}}{Z_{di}}$ for each objective fraction

Check if one or more $\frac{z_{nj} - c_j}{z_{dj} - d_j}$ ratios are $\geq$ $\frac{Z_{ni}}{Z_{di}}$ ratios

Yes

Variable is still promising compute $B^1P_0$ and $B^1P_j$

Find Leaving Variable

Update $M^{-1}$ using the relation $M_{next}^{-1} = E^{-1} \times M_{current}^{-1}$

Yes

p = p+1

Is $p \leq l$

No

C
8.6 Flow Chart For Implementing Multi Objective Linear Programming Algorithm

START

Input $n_v$, $n_c$, $t$

Input coefficients of objective functions and constraints. Construct $C_B$, $D_B$, $X_B$ and $B^{-1}$ matrices

Generate the permutations with given number ($t$) of objective functions as $np = t^2$ and store all the orderings of objectives in COMBIN SET array.

Let $c = 1$

Exchange Object function rows as per the permutation number $c$.

Promising variables are arranged based on constraint type and contribution to sum of ratios of objective functions. Add the subscripts of promising variable to set $J$.

Let $I =$ total number of promising variables

Construct Initial $M^{-1}$ matrix using $C_B$, $D_B$ and $B^{-1}$ matrices.

Is solution Optimal?

A Yes B No
Compute Optimal Solution using the formula
\[ Z = \sum_{i=1}^{k} C_{bi} x_{bi} + C_{di} \] or \[ M^{-1} \times P_{c} \]
Store the result in COMBIN_SOLN array in position c.

Output Optimal Solution for current ordering of objective functions.

Is \( c \leq np \) \( \text{Yes} \rightarrow c = c + 1 \) \( \text{No} \) Output the best Optimal Solution among the solutions stored in COMBIN_SOLN array.

STOP
B

p = 1

Select the p\textsuperscript{th} element from set J and let it be subscript j of the promising variable x\textsubscript{j}

Compute the ratios \( \frac{z_{nj} - c_j}{z_{dj} - d_j} \) and \( \frac{z_{ni}}{z_{di}} \) for each objective fraction

Check if one or more ratios are \( \frac{z_{nj} - c_j}{z_{dj} - d_j} \) ratios > \( \frac{z_{ni}}{z_{di}} \) ratios

Variable is still promising compute B\textsuperscript{T}P_{0} and B\textsuperscript{T}P_{j}

Find Leaving Variable

Update M\textsuperscript{T} using the relation
\[ M_{\text{next}} = E^{\text{T}} \times M_{\text{current}} \]

p = p+1

Is p <= l

Yes

C
8.7 Flow Chart For Implementing Multi Objective Linear Programming Genetic Algorithm

START

Input \( n_v, n_c, t, ng, np \) values

Input coefficients of objective functions and constraints.

Compute \( \theta \) matrix and find minimum value for each decision variable based on the type of constraints. Compute \( Z = \sum_{i=1}^{t} Z_{a_i}^{m} \) with minimum value of each decision variable. Resolve redundant minimums in favour of higher \( Z \) value and select \( n_c \) number of decision variables only.

Fix the lower and upper bound values for each decision variable selected in \( \theta \) matrix.

Generate the permutations with given number (\( t \)) of objective functions as \( pn = ^tP_1 \) and store all the orderings of objectives in COMBIN SET array.

Let \( c = 1 \)

If \( c > pn \)

Yes

D

No

Exchange Objective function rows as per the permutation number \( c \).

Output permutation number and chromosome values with maximum fitness from \( \text{gen}_\text{max}_\text{fitness} \) array

STOP
Using random numbers create nc number of chromosomes for parent number parn.

Substitute chromosomes in all the Constraints.

Does all the Constraints Satisfy?

For each parent, compute fitness value as

$$Z = \sum_{i=1}^{n} \frac{Z_{ci} + c_{di}}{Z_{ci} + d_{ci}}$$

Store the chromosome values and maximum fitness value in combination c and generation g into gen_max_fitness array.
8.8 Conclusion

Flowcharts depicting the control flow in proposed algorithms are drawn. The three algorithms given in chapters 3, 4 and 5 are implemented using C and Visual C++ coding. Various functions and structures are designed to find solution for large scale problems.