Asian Journal of Information Technology 7 (2): 83-86, 2008
ISSN: 1682-3915

Genetic Algorithm Based Method to Solve Linear Fractional Programming Problem

A.M. Sameeullah, S. Divyaa Devi and B. Palaniappan
Department of Computer Science and Engineering, University of Annamalai,
Annamalainagar Chidambaram 608002, TamilNadu, India

Abstract: Linear fractional programming problem finds application in all fields such as animal husbandry problem, power aware embedded system, etc. Finding solution to this problem using conventional methods requires more computational effort. Therefore, a heuristic approach based on Genetic Algorithm is developed to solve the linear fractional programming problem. A set of solution points are generated using random numbers and feasibility of each solution point is verified and the fitness values for all the feasible solution points are obtained. Among the feasible solution points, the best solution point is found out and it replaces the worst solution point. Then pair wise solution points are used for crossover and a new set of solution points is obtained. These steps are repeated for a certain number of generations and the best solution for the given problem is obtained.

Key words: Linear fractional programming, genetic algorithm, crossover, mutation, θ matrix

INTRODUCTION

An objective function of problems concerning optimization which is a ratio of 2 functions subject to a set of constraints and non-negative variables are called Fractional Programming Problems. If the objective function of numerator and denominator and the constraints are linear, the problem is called Linear Fractional Programming Problem.

Mathematically it can be represented as:

$$\text{Extremize} \quad Z = \frac{C^T X + C_0}{D^T X + D_0}$$

$$\text{Subject to} \quad AX \leq P_0$$

Where: $X \geq 0$

- X is the decision vector of order n×1.
- A is the constraint matrix of order m×n.
- C and B are contribution coefficient vector of order n×1.
- P_0 is the resource vector of order m×1.
- C_0 and $D_0$are scalars.
- n and m are the number of variables and constraints, respectively.

Decision making situations occur in all fields like science, technology and management, etc. The main objective is to maximize or minimize a task like maximizing the profit or minimizing the waste. Over the past 6 decades, many optimization procedures have been developed to solve the problems like linear programming, nonlinear programming, dynamic programming, transportation problems, assignment problems, replacement problems, inventory problems, queuing problems and scheduling problems.

Most of the traditional optimization procedures end its search in the local optima rather than finding the global optima. To overcome this situation in the past 4 decades many number of non-traditional search and optimization algorithms were developed.
They are,

- Genetic Algorithm (GA).
- Simulated Annealing (SA).
- Tabu Search (TS).
- Ant Colony Optimization (ACO).
- Particle Swarm Optimization (PSO).
- Scatter Search (SS), etc.

Genetic Algorithms (GA) are computerized search and optimization algorithms based on the mechanics of natural genetics and natural selection. It was inspired by Darwin's theory about evolution. Prof. Holland of University of Michigan envisaged the concept of Genetic Algorithm. Many students and researchers have contributed to develop this field.

Corresponding Author: Professor A.M. Sameeullah, Department of Computer Science and Engineering, University of Annamalai, Annamalainagar Chidambaram 608002, TamilNadu, India

The classical methods of Operation Research (OR) often fail due to the exponentially growing computational effort. Therefore, in practice heuristics are commonly used. Heuristic techniques that mimic natural processes developed over the last 30 years have produced good results in reasonable short runs for this class of optimization problems. In this study, it is proposed to use Genetic Algorithm approach to solve the Linear Fractional Programming problem.

MATERIALS AND METHODS

Various methods available to solve linear fractional programming problems are briefly presented in this study.

Charnes-Cooper method (Charnes and Cooper, 1962) converts a linear fractional programming problem into two equivalent linear programming problems assuming that the denominator

$$\sum_{j=1}^{n} d_j x_j + d_o$$

of the objective function is never zero. The modified linear programming problem with (m + 1) constraints can be solved using simplex method.

Isbell and Marlow, Dinkelbach and Martos determined the optimal solution to the linear fractional programming problem using a sequence of linear programs. The common characteristic of all these algorithms is that they concentrate solely on determining the optimal solution, without providing the decision maker with additional information, which is valuable for decision making.

The parametric method (Harmut, 1986) differs in this respect from the usual algorithms. It involves a two-step process. In the first step, the linear parametric problem is solved, which contains the parameters that define non-empty set of solutions. The second step determines the optimal solution to the problem starting from the function obtained in the first step.

The KantiSwarup (1965) method assumes that the denominator is positive for all feasible solutions. A new objective function is formed based on the numerator and denominator of the original objective function and that is used to find the entering variable at each iteration.

In Ratio method (Sankaraiyer, 1994), the ratio of the contribution coefficients of the decision variables in the numerator and denominator of the objective function is used to select the entering variable based on the assumption that the denominator value of the objective function is always positive and the numerator can assume either positive or negative value. Eight different optimality conditions are derived. This algorithm preserves feasibility until it reaches the optimality.

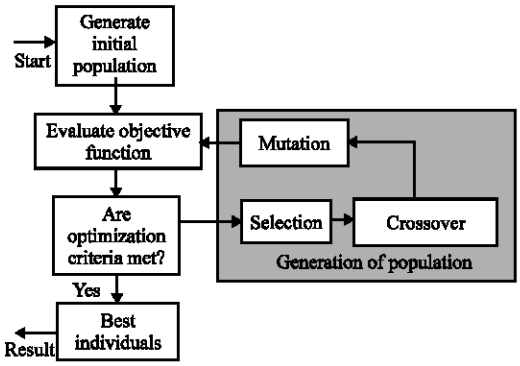

Fig. 1: Structure of genetic algorithm

Ratio multiplex algorithm (Sakthivel and Naganathan, 1998) employs multiple column selection procedure to select a set of entering and leaving variables. Ratio multiplex method selects a set of entering variables based on the ratio of the numerator and denominator coefficients of the decision variables of the objective function. In this method, the basic solution obtained in each pass is tested for optimality and feasibility at the end of each pass. If it fails to satisfy the optimality conditions, the next set of promising variables is brought in and is continued until optimality is reached. If it does not satisfy the feasibility then the dual multiplex procedure is invoked to make the solution to move to the feasible region.

Unlike the methods explained so far, Modified ratio algorithm (Jeyaraman, 2001) provides a technique to solve linear fractional programming problems even when the denominator of the objective function changes from positive to negative or vice versa.

Genetic algorithm: To solve the optimization problems many traditional optimization methods are available like bracketing methods, region elimination methods, point estimation methods, gradient based methods, direct search methods and linearized search techniques. It is found that the genetic algorithm outperforms the non-traditional optimization techniques.

Simple genetic algorithm-overview: Figure 1 shows the structure of Genetic Algorithm.

A simple genetic algorithm starts with a set of randomly generated initial population.The basic steps involved in the genetic algorithm is shown below.

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem].
2. [Evaluate] Evaluate the fitness F(X) of each chromosome X in the population.
3. [New Population] Create a new population by repeating following steps until the new population is complete.

i. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
ii. [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed offspring is the exact copy of parents.
iii. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

Accepting: Place new offspring in the new population.

4. [Replace]Use new generate population for a further run of the algorithm.
5. [Test] If the end condition is satisfied, stop and return the best solution in current population.
6. [Loop] Go to step 2.

Crossover or recombination: Crossover operator produces new offspring in combining the information contained in 2 parents. The crossover occurs only if the random number generated is less than the crossover probability P_c (like flipping of a coin with a probability) otherwise the 2 strings repeated without any change. Depending on the representation of the variables, the offspring will be subjected to crossover.

After crossover operation is performed, the string is subjected to mutation operation. This is to prevent falling all solutions of the population into a local optimum of solved problem. Mutation operator alters a chromosome locally to hopefully create a better string. The bit wise mutation is performed with probability of mutation P_m. Mutation occurs only if the random number generated is less than the mutation probability P_m (like flipping of a coin with probability) otherwise the bit is kept unchanged. A simple genetic algorithm treats the mutation only as a secondary operator with role of restoring lost genetic materials. The mutation is also used to maintain diversity in the population.

Method to find the upper and lower limit for variables

Step 1: The matrix of intercepts of the decision variables along the respective axes called θ matrix with respect to the chosen basis is to be constructed. A typical intercept for the j^{th} variable, x_j due to the i^{th} resource, b_i is:

$$\frac{b_i}{a_{ij}}, a_{ij} > 0$$

The expanded form of θ matrix is:

$$\begin{array}{c} \\ x_1 \\ x_2 \\ \cdots\cdots \\ x_j \\ \cdots\cdots \\ x_n \end{array} \begin{array}{c} \begin{array}{ccccccc} S_1 & S_2 & \cdots\cdots & S_i & \cdots\cdots & S_m \end{array} \\ \begin{bmatrix} b_1/a_{11} & b_2/a_{21} & \cdots\cdots & b_i/a_{i1} & \cdots\cdots & b_m/a_{m1} \\ b_1/a_{12} & b_2/a_{22} & \cdots\cdots & b_i/a_{i2} & \cdots\cdots & b_m/a_{m2} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ b_1/a_{1j} & b_2/a_{2j} & \cdots\cdots & b_i/a_{ij} & \cdots\cdots & b_m/a_{mj} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ b_1/a_{1n} & b_2/a_{2n} & \cdots\cdots & b_i/a_{in} & \cdots\cdots & b_m/a_{mn} \end{bmatrix} \end{array}$$

Each row of the θ matrix represents m number of intercepts of the decision variables along their respective axes and each column represents the intercepts formed by the decision variables of each of the m constraints.

Step 2: Each row of the θ matrix is scanned and the minimum intercept due to upper bound constraints (called r1min), maximum intercept due to lower bound constraints (called r2max) and minimum intercept due to equality constraints (called r3min) are identified.

Step 3(i): If r2max doesn't exists for a row, then the lower limit for the variable corresponding to a row is zero and the upper limit of that variable is equal to r1min.

Step 3(ii): If r2max exists then the lower bound for that variable is r2max and the upper limit of that variable is r1min.

Step 4: Repeat step 3 till the upper and lower limit for all the variables are fixed.

Step-by-step procedure: The various steps involved in the application of Genetic Algorithm are:

1. Find the upper limit and lower limit for all the variables (as discussed earlier).
2. Decide the number of bits for representing each variable as well as base to be used.
3. Move zero to number of solution points, number of generations.
4. Generation of population: Using random numbers generate that many numbers of bits that are decided in step 2 for each variable. This combination of bits will become a value for a variable.
5. Repeat step 4 as many as the number of variables so that the values for a set of variables are obtained.
6. For the solution obtained using steps 4 and 5 check the feasibility.

Table 1: The results obtained for problems of different sizes (variables and constraints)

Number of variables	Number of constraints	Objective	Maximum number of generations	Simple genetic algorithm (sec)	Number of generations at which the optimum obtained
31	17	Max	100	52	08
96	22	Max	100	133	45
209	30	Max	100	389	42
372	43	Max	200	530	60
675	60	Max	200	718	84
928	74	Max	300	1057	98

   a. If feasible, the corresponding objective value is found out. Update the number of solution points by 1.
   b. If infeasible, drop that solution.
7. If number of solution points is less than the number of populations (say n) go to step 4. Else step 8.
8. Find the solution whose objective function value is the maximum and also the corresponding solution point.
9. Select the solution points pair-wise using random numbers and do the crossover.
10. For the new set of solution points check the feasibility.
    a. If feasible, find the fitness function value.
    b. If infeasible, leave that solution and using steps 4 and 5 find a feasible solution and then find the fitness function value.
11. Update the number of generation by 1.
12. If the number of generation is less than or equal to the maximum number of generations (say m) go to step 8. Else step 13.
13. Find the best among m different solution in step 8.

RESULTS

The Software has been developed using c Language and it is tested with different types of problems. The results obtained for problems of different sizes (variables and constraints) are given in the Table 1.

CONCLUSION

The conventional methods usually take more iterations and time. This method finds the best solution in 98 generations for a maximization problem with 928 variables and 74 constraints. Fixing the upper bound and lower bound for each of the variable using θ matrix is the highlight of this method. Further research is going on in comparing the efficiency of this method with other methods.

REFERENCES

Charnes, A. and W.W. Cooper, 1962. Programming with Linear Fractional functionals. Naval, Res. Logist Quart., 9: 181-186.

Harmut Wolf, 1986. Parametric Analysis in Linear Fractional programming, Operat. Res., 34 (6): 930-937.

Jeyaraman, K., 2001. A Modified Ratio Algorithm to Solve Linear Fractional Programming Problems, A Ph.D. thesis, Alagappa University, Karaikudi.

KantiSwarup, 1965. Linear Fractional Programming. Operat. Res., 13: 1029-1036.

Sakthivel, S. and E.R. Naganathan, 1998. A Ratio Multiplex Algorithm to Solve Linear Fractional Programming Problem. Opsearch J., 35 (3): 215-225.

Sakthivel, S. and T.R. Natesan, 1983. A multiplex algorithm to solve large scale linear a programming problems. Third Symposium-IFAC/IFORS Large Scale Systems: Theory, Applications and Impact, Warsaw, Poland.

Sankaraiyer, P., 1994. A computer Oriented Ratio algorithm to solve Linear Fractional Programming Problems, A Ph.D. thesis, Alagappa University, Karaikudi.