ECONSTOR
Make Your Publications Visible.

A Service of
ZBW
Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Lachhwani, Kailash; Nehra, Suresh

Article

Modified FGP approach and MATLAB program for solving multi-level linear fractional programming problems

Journal of Industrial Engineering International

Provided in Cooperation with:
Islamic Azad University (IAU), Tehran

Suggested Citation: Lachhwani, Kailash; Nehra, Suresh (2015) : Modified FGP approach and MATLAB program for solving multi-level linear fractional programming problems, Journal of Industrial Engineering International, ISSN 2251-712X, Springer, Heidelberg, Vol. 11, pp. 15-36, https://doi.org/10.1007/s40092-014-0084-4

This Version is available at:
https://hdl.handle.net/10419/157427

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

http://creativecommons.org/licenses/by/2.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

J Ind Eng Int (2015) 11:15–36
DOI 10.1007/s40092-014-0084-4

TECHNICAL ARTICLE

Modified FGP approach and MATLAB program for solving multi-level linear fractional programming problems

Kailash Lachhwani · Suresh Nehra

Received: 6 November 2013 / Accepted: 23 July 2014 / Published online: 11 September 2014
© The Author(s) 2014. This article is published with open access at Springerlink.com

Abstract In this paper, we present modified fuzzy goal programming (FGP) approach and generalized MATLAB program for solving multi-level linear fractional programming problems (ML-LFPPs) based on with some major modifications in earlier FGP algorithms. In proposed modified FGP approach, solution preferences by the decision makers at each level are not considered and fuzzy goal for the decision vectors is defined using individual best solutions. The proposed modified algorithm as well as MATLAB program simplifies the earlier algorithm on ML-LFPP by eliminating solution preferences by the decision makers at each level, thereby avoiding difficulties associate with multi-level programming problems and decision deadlock situation. The proposed modified technique is simple, efficient and requires less computational efforts in comparison of earlier FGP techniques. Also, the proposed coding of generalized MATLAB program based on this modified approach for solving ML-LFPPs is the unique programming tool toward dealing with such complex mathematical problems with MATLAB. This software based program is useful and user can directly obtain compromise optimal solution of ML-LFPPs with it. The aim of this paper is to present modified FGP technique and generalized MATLAB program to obtain compromise optimal solution of ML-LFP problems in simple and efficient manner. A comparative analysis is also carried out with numerical example in order to show efficiency of proposed modified approach and to demonstrate functionality of MATLAB program.

Keywords Multi-level linear fractional programming problem · Fuzzy goal programming · Compromise optimal solution · MATLAB program

K. Lachhwani (✉)
Department of Mathematics, Government Engineering College Bikaner, Bikaner 334 004, Rajasthan, India
e-mail: kailashclachhwani@yahoo.com

S. Nehra
Research Scholar, Department of Electronics and Communication Engineering, Government Engineering College Bikaner, Bikaner 334 004, Rajasthan, India
e-mail: sknehra75@gmail.com

Introduction

Multi-level programming problem (MLPP) concerns with decentralized programming problems with multiple decision makers (DMs) in multi-level or hierarchical organizations, where decisions have interacted with each other. Multi-level organization or hierarchical organization has the following common characteristics: Interactive decision-making units exist within a predominantly hierarchical structure; the execution of decisions is sequential from higher level to lower level; each decision-making unit independently controls a set of decision variables and is interested in maximizing its own objective but is affected by the reaction of lower level DMs. So the decision deadlock arises frequently in the decision-making situations of multi-level organizations.

Numerous methods were suggested by researchers in literature (Anandilingam 1988, 1991; Lai 1996; Pramanik and Roy 2007; Shih et al. 1983; Shih and Lee 2000; Sinha 2003a, b) on MLPPs and also on multi-criteria decision-making problems (MCDM) and multi-objective programming problems with their applications like Zoraghi et al. (2013) presented a fuzzy multi-criteria decision making (MCDM) model by integrating both objective and subjective weights for evaluating service quality in hotel industries. Sadjadi et al. (2005) proposed a multi-objective

linear fractional inventory model using fuzzy programming. Fattahi et al. (2006) proposed a Pareto approach to solve multi-objective job shop scheduling. Aryanezhad et al. (2011) considered the portfolio selection where fuzziness and randomness appear simultaneously in optimization process. Tohidi and Razavyan (2013) presented necessary and sufficient conditions to have unbounded feasible region and infinite optimal values for objective functions of multi-objective integer linear programming problems. Khalili-Damghani and Taghavifard (2011) proposed a multi-dimensional knapsack model for project capital budgeting problem in uncertain situation through fuzzy sets. Makul et al. (2008) presented the use of multiple objective linear programming approach for generating the common set of weights under the DEA framework. Each method appears to have some advantages as well as disadvantages. So, the issue of choosing a proper method in a given context is still a subject of active research. In context of such hierarchical problems, Fuzzy goal programming (FGP) approach seems to be more appropriate than other methodologies. The FGP introduced by Mohamed (1997) was extended to solve multi-objective linear fractional programming problems in Pal et al. (2003), bi-level programming problems in Moitra and Pal (2002), bi-level quadratic programming problems in Pal and Moitra (2003), and also extended to solve multi-level programming problems (MLPPs) with single objective function in each level in Pramanik and Roy (2007). In recent years, Aghdaghi and Jolai (2008) presented a goal programming approach and heuristic algorithm to solve vehicle routing problem with backhauls. Babeai et al. (2009) investigated the optimum portfolio for an investor using lexicographic goal programming approach. Ghosh and Roy (2013) formulated weighted goal programming as goal programming with logarithmic deviational variables. Lachhwani and Poonia (2012) proposed FGP approach for multi-level linear fractional programming problem. Lachhwani (2013) presented an alternate algorithm to solve multi-level multi-objective linear programming problems (ML-MOLPPs) which is simpler and requires less computational efforts than that of suggested by Baky (2010). Baky (2010) suggested two new techniques with FGP approach based on solution preferences by the decision maker at each level to solve new type of multi-level multi-objective linear programming (ML-MOLP) problems through the fuzzy goal programming (FGP) approach. Abo-Sinha and Baky (2007) presented interactive balance space approach for solving multi-level multi-objective programming problems. Baky (2009) proposed FGP algorithm for solving decentralized bi-level multi-objective programming (DBL-MOP) problems with a single decision maker at the upper level and multiple decision makers at the lower level. The main disadvantage of the FGP algorithms is that the possibility of rejecting the solution again and again by the upper level DMs and re-evaluation of the problem repeatedly, by redefining the tolerance values on decision variables, needed to reach the satisfactory decision frequently arises. To overcome such computational difficulties, we modified FGP approach for ML-LFPP in which solution preferences by decision maker at each level and sequential order of decision-making process in finding satisfactory solutions are not taken into account of proposed technique and we straightforwardly obtain compromise optimal solution of the problem with higher degree of membership function values. In this paper, we proposed modified FGP approach for multi-level linear fractional programming problem (ML-LFPP) in which solution preferences by decision maker at each level and sequential order of decision-making process in finding satisfactory solutions are not taken into account of proposed technique. Using modified technique, we straightforwardly obtain compromise optimal solution of the problem with higher degree of membership function values. This modified approach simplifies the solution procedure and reduces the computational efforts with it. Here, we also present coding of generalized MATLAB program based on proposed modified approach for solving ML-LFPPs which is the unique toward dealing with such complex mathematical problems with MATLAB. This software based program is useful and user can directly obtain compromise optimal solution of ML-LFPPs. The aim of this paper is to present modified FGP algorithm and generalized MATLAB program which is simple, efficient and requires less computational efforts for solving multi-level linear fractional programming problems (ML-LFPPs).

The paper is organized in following sections: MLPPs and related literature reviews are presented in introduction section. Formulation of ML-LFPP and related notations are discussed in next Sect. 2. Characterization of membership functions, solution approach based on FGP and formulation of FGP models are discussed in next section. Proposed MATLAB program and its functionality are discussed in Sect. 4. Numerical example on modified FGP technique and its comparison with solution technique suggested by Lachhwani and Poonia (2012) are discussed in numerical example Sect. 5. Concluding remarks are given in the last section. Coding of main function and recurresive simplex function are presented in appendices.

Formulation of ML-LFPPs

We consider a T-level maximization type multi-level linear fractional programming problem (ML-LFPP). Mathematically it can be defined as:

$$\underset{\overline{X_1}}{Max}\, Z_1(\overline{X}) = \frac{\overline{C_{11}}\,\overline{X_1} + \overline{C_{12}}\,\overline{X_2} + \cdots + \overline{C_{1T}}\,\overline{X_T} + \alpha_1}{\overline{D_{11}}\,\overline{X_1} + \overline{D_{12}}\,\overline{X_2} + \cdots + \overline{D_{1T}}\,\overline{X_T} + \beta_1} = \frac{N_1(\overline{X})}{D_1(\overline{X})}$$

$$\underset{\overline{X_2}}{Max}\, Z_2(\overline{X}) = \frac{\overline{C_{21}}\,\overline{X_1} + \overline{C_{22}}\,\overline{X_2} + \cdots + \overline{C_{2T}}\,\overline{X_T} + \alpha_2}{\overline{D_{21}}\,\overline{X_1} + \overline{D_{22}}\,\overline{X_2} + \cdots + \overline{D_{2T}}\,\overline{X_T} + \beta_2} = \frac{N_2(\overline{X})}{D_2(\overline{X})}$$

.

.

$$\underset{\overline{X_T}}{Max}\, Z_T(\overline{X}) = \frac{\overline{C_{T1}}\,\overline{X_1} + \overline{C_{T2}}\,\overline{X_2} + \cdots + \overline{C_{TT}}\,\overline{X_T} + \alpha_T}{\overline{D_{T1}}\,\overline{X_1} + \overline{D_{T2}}\,\overline{X_2} + \cdots + \overline{D_{TT}}\,\overline{X_T} + \beta_T} = \frac{N_T(\overline{X})}{D_T(\overline{X})}$$

Subject to, $\overline{A_{i1}}\,\overline{X_1} + \overline{A_{i2}}\,\overline{X_2} + \cdots + \overline{A_{iT}}\,\overline{X_T}(\leq, =, \geq)\, b_i$

$\forall i = 1, 2, \ldots, m$

and $\overline{X_1} \geq 0, \overline{X_2} \geq 0, \ldots, \overline{X_T} \geq 0.$ (1)

$\overline{X_1} = \{X_1^1, X_1^2, \ldots, X_1^{N_1}\}'$

decision variables under the control of first level DM.

.

.

$\overline{X_T} = \{X_T^1, X_T^2, \ldots, X_T^{N_T}\}'$

decision variable under the control of t - level DM.

where $'$denotes transposition, $\overline{A_{ij}}$ $i = 1, 2, \ldots, m, j = 1, 2, \ldots, T$ are m row vectors, each of dimension$(1 \times N_j)$. $\overline{A_{it}}\,\overline{X_t}$, $t = 1, 2, \ldots, T$ is a column vector of dimension $(M \times 1)$.$\overline{C_{11}}, \overline{C_{21}}, \ldots, \overline{C_{T1}}$ all are row vectors of dimension of $(1 \times N_1)$. Similarly $\overline{C_{1T}}, \overline{C_{2T}}, \ldots, \overline{C_{TT}}$ and $\overline{D_{1T}}, \overline{D_{2T}}, \ldots, \overline{D_{TT}}$ are row vectors of dimension of $(1 \times N_T)$. We take $\overline{X} = \overline{X_1} \cup \overline{X_2} \cup \cdots . \cup \overline{X_T}$ and $N = N_1 + N_2 + \cdots + N_T$. Here one DM is located on each level. Decision vector $\overline{X_t}$, $t = 1, 2, \ldots, T$ is control of tth level DM having N_t number of decision variables. Here, it is assumed that the denominator of objective functions is positive at each level for all the values of decision variables in the constraint region.

Modified FGP methodology for ML-LFPP

The proposed modified FGP procedure is based on finding the compromise optimal solution as described by Lachhwani (2013) for multi-level multi-objective linear programming (ML-MOLP) problems. Here, we need to express the definitions related to efficient solution and compromise optimal solution in context of MLPP as:

Definition 1 $X^* \in S$ is an efficient solution to MLPP if and only if there exists no other $X \in S$ such that $Z_t \geq Z_t^*$ $\forall t = 1, 2, \ldots, T$.

Definition 2 For a problem (1), a compromise optimal solution is an efficient solution selected by the decision maker (DM) as being the best solution where the selection is based on the DM's explicit or implicit criteria.

Zeleny (1982) as well as most authors describes the act of finding a compromise optimal solution to problem as "……an effort or emulate the ideal solution as closely as possible".

Our FGP model for determining compromise optimal (efficient) solution is based on the finding of the totality or subset of efficient solutions with the DM, then choosing one best solution on some explicit or implicit algorithm.

FGP formulation for ML-LFPP

To formulate the modified FGP models of ML-LFPP, the objective numerator $f_{tN}(\overline{X}) + \alpha_t$, $\forall t = 1, 2, \ldots, T$, objective denominator $f_{tD}(\overline{X}) + \alpha_t$, $\forall t = 1, 2, \ldots, T$ at each level and the decision vector $\overline{X_t}$, $(t = 1, 2, .., T-1)$ would be transformed into fuzzy goals by assigning an aspiration level to each of them. Then, they are to be characterized by the associated membership functions by defining tolerance limits for the achievement of the aspired levels of the corresponding fuzzy goals. Here, decision vector $\overline{X_t}$ of up to (*T–1*) levels is transformed into fuzzy goals in order to avoid decision deadlock situations.

Characterization of membership functions

To build membership functions, fuzzy goals and their aspiration levels should be determined first. Using the individual best solution without considering inference of decision variables on lower levels, we find the maximum and minimum values of all the numerator and denominator objective functions at each level solution and construct payoff matrices as:

$$\begin{bmatrix} \overline{X_t} & \overline{N_t} \\ \overline{X_1} & \overline{N_1}(\overline{X_1}) \\ \overline{X_2} & \overline{N_2}(\overline{X_2}) \\ . & . \\ . & . \\ \overline{X_T} & \overline{N_T}(\overline{X_T}) \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \underline{X_t} & \underline{N_t} \\ \underline{X_1} & \underline{N_1}(\underline{X_1}) \\ \underline{X_2} & \underline{N_2}(\underline{X_2}) \\ . & . \\ . & . \\ \underline{X_T} & \underline{N_T}(\underline{X_T}) \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \overline{X_t} & \overline{D_t} \\ \overline{X_1} & \overline{D_1}(\overline{X_1}) \\ \overline{X_2} & \overline{D_2}(\overline{X_2}) \\ . & . \\ . & . \\ \overline{X_T} & \overline{D_T}(\overline{X_T}) \end{bmatrix} \tag{4}$$

and

$$\begin{bmatrix} \underline{X_t} & \underline{D_t} \\ \overline{X_1} & \underline{D_1}(\overline{X_1}) \\ \underline{X_2} & \underline{D_2}(\overline{X_2}) \\ \cdot & \cdot \\ \cdot & \cdot \\ \underline{X_T} & \underline{D_T}(\underline{X_T}) \end{bmatrix} \tag{5}$$

The maximum values of each row $N_t(\overline{X_t})$ and $D_t(\overline{X_t})$ $\forall t = 1, 2, \ldots, T$ give upper and lower tolerance limit or aspired level of achievement for the membership function of tth level numerator and denominator objective function respectively. Similarly, the minimum values of each row $N_t(\underline{X_t})$ and $D_t(\underline{X_t})$ $\forall t = 1, 2, \ldots, T$ give lower and upper tolerance limit or lowest acceptable level of achievement for the membership function of tth level numerator and denominator objective function respectively. Hence, linear membership functions for the defined fuzzy goals (as shown in Fig. 1a, b respectively) are:

$$\mu_{Z_t}(N_t(\overline{X})) = \begin{cases} 1 & \text{if } N_t(\overline{X}) \geq \overline{N_t} \\ \dfrac{N_t(\overline{X}) - \underline{N_t}}{\overline{N_t} - \underline{N_t}} & \text{if } \underline{N_t} \leq N_t(\overline{X}) \leq \overline{N_t} \\ 0 & \text{if } N_t(\overline{X}) \leq \underline{N_t} \end{cases} \quad \forall t = 1, 2, \ldots, T \tag{6}$$

$$\mu_{Z_t}(D_t(\overline{X})) = \begin{cases} 0 & \text{if } D_t(\overline{X}) \geq \overline{D_t} \\ \dfrac{\overline{D_t} - D_t(\overline{X})}{\overline{D_t} - \underline{D_t}} & \text{if } \underline{D_t} \leq D_t(\overline{X}) \leq \overline{D_t} \\ 1 & \text{if } D_t(\overline{X}) \leq \underline{D_t} \end{cases} \quad \forall t = 1, 2, \ldots, T \tag{7}$$

Now, the linear membership functions for the decision vector $X_T (t = 1, 2, \ldots, T-1)$ (as shown in Fig. 2) are formulated in modified form as:

$$\mu_{X_t}(X_t) = \begin{cases} 1 & \text{for } X_t \geq \overline{X_t} \\ \dfrac{X_t - \underline{X_t}}{\overline{X_t} - \underline{X_t}} & \text{for } \underline{X_t} \leq X_t \leq \overline{X_t} \\ 0 & \text{for } X_t \leq \underline{X_t} \end{cases} \tag{8}$$

where $\overline{X_t}$ and $\underline{X_t}$ are taken as the values of the corresponding decision vectors at each level which yield the highest and lowest values of the numerator part of objective functions ($(\overline{N_t}(\overline{X})$ and $\underline{N_t}(\underline{X})$ $\forall t = 1, 2, \ldots, T-1$) at each level respectively defined as:

$$\overline{N_t} = \underset{\overline{X_t} \in X}{\text{Max}} \{\overline{N_t}(\overline{X_t}),\ \forall t = 1, 2, \ldots, T\} \tag{9}$$

$$\underline{N_t} = \underset{\underline{X_t} \in X}{\text{Min}} \{\underline{N_t}(\underline{X_t}),\ \forall t = 1, 2, \ldots, T\} \tag{10}$$

Here, it is important to note that for simplicity of proposed technique and in order to avoid decision deadlock situation in the whole solution methodology, the solution preferences by the decision maker in terms of values of decision vector at each level with respect to the values of decision vector at lower levels are not considered. This results that large amount of computational tasks is reduced into limited simple calculation in modified FGP model formulation. Also, linear membership functions are

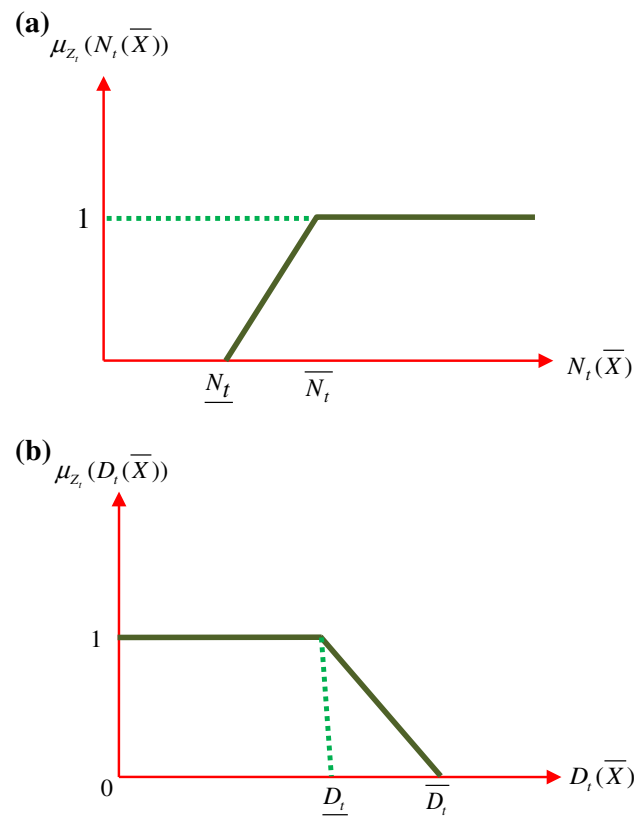

Fig. 1 **a** Membership functions of maximization type numerator objective functions. **b** Membership functions of maximization type denominator objective functions

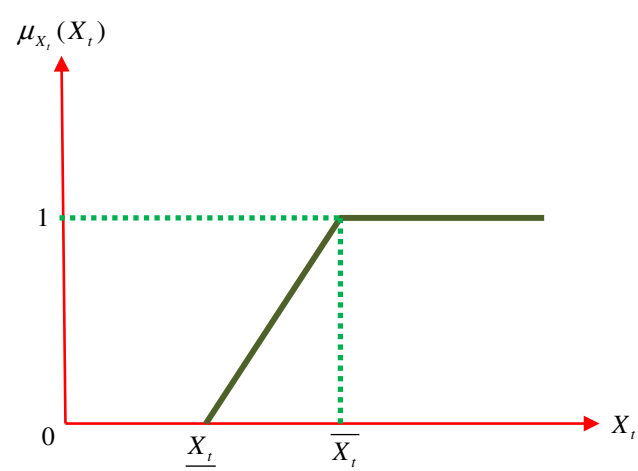

Fig. 2 Membership function for $\mu_{X_t}(X_t)$ $\forall t = 1, 2, \ldots, T-1$

considered because these are more suitable than nonlinear ones in context of complex ML-LFPPs and it further reduces computational difficulties in modified method.

FGP solution approach

In GP approach, decision policy for minimizing the regrets of the DMs for all the levels is taken into consideration. Then each DM should try to maximize his or her membership function by making them as close as possible to unity by minimizing its negative deviational variables. Therefore, in effect, we are simultaneously optimizing all the objective functions. So, for the defined membership functions in (6), (7) and (8), the flexible membership goals having the aspired level unity can be represented as:

$$\mu_t(N_t) + d_t^{N-} - d_t^{N+} = 1 \quad \forall t = 1, 2, \ldots, T \tag{11}$$

$$\mu_t(D_t) + d_t^{D-} - d_t^{D+} = 1 \quad \forall t = 1, 2, \ldots, T \tag{12}$$

$$\mu_{X_t}(X_t) + d_t^{-} - d_t^{+} = \bar{I} \quad \forall t = 1, 2, \ldots, T-1 \tag{13}$$

where $d_t^{N-}, d_t^{D-}, d_t^{N+} d_t^{D+} (\geq 0)$ $(\forall t = 1, 2, \ldots, T)$ and $d_t^{-}, d_t^{+} (\geq 0)$ $\forall t = 1, 2, \ldots, T-1$ represent the under and over deviational variables respectively from the aspired levels. $\bar{I}$ is the column vector having all components equal to 1 and its dimension depends on X_t. Thus ML-LFP problem (1) changes into:

$$\min \lambda = \sum_{t=1}^{T} (d_t^{N-} + d_t^{D-}) + \sum_{t=1}^{T-1} d_t^{-} \tag{14}$$

$$\begin{aligned}
\text{Subject to,} \quad & \mu_t(N_t) + d_t^{N-} - d_t^{N+} = 1 \quad \forall t = 1, 2, \ldots, T \\
& \mu_t(D_t) + d_t^{D-} - d_t^{D+} = 1 \quad \forall t = 1, 2, \ldots, T \\
& \mu_{X_t}(X_t) + d_t^{-} - d_t^{+} = \bar{I} \quad \forall t = 1, 2, \ldots, T-1 \\
& \overline{A_{i1}}\,\overline{X_1} + \overline{A_{i2}}\,\overline{X_2} + \cdots + \overline{A_{iT}}\,\overline{X_T} (\leq, =, \geq)\, b_i \\
& \forall i = 1, 2, \ldots, m \\
\text{and} \quad & \overline{X_1} \geq 0, \overline{X_2} \geq 0, \ldots, \overline{X_T} \geq 0.
\end{aligned}$$

In this FGP approach, only the sum of under deviational variables is required to be minimized to achieve the aspired level. It may be noted that when a membership goal is fully achieved, negative deviational variable becomes zero and when its achievement is zero, negative deviational variable becomes unity in the solution. Now if the most widely used and simplest version of GP (i.e. minsum GP) is introduced to formulate the model of the problem under consideration, the FGP model formulation becomes:

FGP model I

$$\min \lambda = \sum_{t=1}^{T} (d_t^{N-} + d_t^{D-}) + \sum_{t=1}^{T-1} d_t^{-}$$

$$\begin{aligned}
\text{Subject to,} \quad & -\overline{N_t} + N_t + d_t^{N-}(\overline{N_t} - \underline{N_t}) \geq 0 \quad \forall t = 1, 2, \ldots, T \\
& \underline{D_t} - D_t + d_t^{D-}(\overline{D_t} - \underline{D_t}) \geq 0 \quad \forall t = 1, 2, \ldots, T \\
& -\overline{X_t} + X_t + d_t^{-}(\overline{X_t} - \underline{X_t}) \geq 0 \quad \forall T = 1, 2, \ldots, T-1 \\
& \overline{A_{i1}}\,\overline{X_1} + \overline{A_{i2}}\,\overline{X_2} + \cdots + \overline{A_{iT}}\,\overline{X_T} (\leq, =, \geq)\, b_i \\
& \forall i = 1, 2, \ldots, m \\
\text{and} \quad & \overline{X_1} \geq 0, \overline{X_2} \geq 0, \ldots, \overline{X_T} \geq 0.
\end{aligned}$$

FGP model II

$$\min \lambda = \sum_{t=1}^{T} (\dot{w}_t d_t^{N-} + \ddot{w}_t d_t^{D-}) + \sum_{t=1}^{T-1} d_t^{-}$$

$$\begin{aligned}
\text{Subject to,} \quad & -\overline{N_t} + N_t + d_t^{N-}(\overline{N_t} - \underline{N_t}) \geq 0 \quad \forall t = 1, 2, \ldots, T \\
& \underline{D_t} - D_t + d_t^{D-}(\overline{D_t} - \underline{D_t}) \geq 0 \quad \forall t = 1, 2, \ldots, T \\
& -\overline{X_t} + X_t + d_t^{-}(\overline{X_t} - \underline{X_t}) \geq 0 \quad \forall T = 1, 2, \ldots, T-1 \\
& \overline{A_{i1}}\,\overline{X_1} + \overline{A_{i2}}\,\overline{X_2} + \cdots + \overline{A_{iT}}\,\overline{X_T} (\leq, =, \geq)\, b_i \\
& \forall i = 1, 2, \ldots, m \\
\text{and} \quad & \overline{X_1} \geq 0, \overline{X_2} \geq 0, \ldots, \overline{X_T} \geq 0.
\end{aligned}$$

where λ (in FGP model II) represents the fuzzy achievement function consisting of the weighted under deviational variables and the numerical weights $\dot{w}_t, \ddot{w}_t > 0$, $(\forall t = 1, \ldots, T)$ represent the relative importance of achieving the aspired level of the respective fuzzy goals subject to the constraints in the decision-making situation. To assess the relative importance of the fuzzy goals properly, the weighted scheme suggested by Mohamed (1997) can be used to assign the values to $\dot{w}_t, \ddot{w}_t > 0$, $(\forall t = 1, \ldots, T)$. In the present formulation $\dot{w}_t, \ddot{w}_t > 0$ can be determined as:

$$\dot{w}_t = \frac{1}{\overline{N_t} - \underline{N_t}} \quad \forall t = 1, \ldots, T \tag{15}$$

$$\ddot{w}_t = \frac{1}{\overline{D_t} - \underline{D_t}} \quad \forall t = 1, \ldots, T \tag{16}$$

MATLAB Program for ML-LFPPs based on modified FGP approach

Here, we discuss the coding and functionality of generalized MATLAB program for finding the compromise optimal solution of any ML-LFPPs based on proposed modified FGP approach. Using this program, the user needs to input data related to the problem and then user can directly obtain compromise optimal solution of ML-LFPP in single iteration with this program. To run this program, the two files are imported (used for main function and simplex function respectively and as shown in appendices) in the current MATLAB folder as:

1 opt_pro.m
2 simplex_function1.m

Then go to the MATLAB command prompt and type opt_pro to execute the program. The main coding of this program is partitioned into following two parts as:

(a) Main function (MATLAB coding of main function is shown in appendices)
(b) Simplex function (MATLAB coding of simplex function is shown in appendices)

The functionality of MATLAB program to obtain optimized values of decision variables and corresponding objective functions can be described in following stepwise algorithm as:

Step 1: In first step, main function takes input values from the user and converts them into suitable matrices. Then these matrices are passed to the simplex function as its input arguments in single matrix containing constraints as well as objective functions.
Step 2: For each level, the simplex function is called two times to compute minimized and maximized values of numerator and denominator objective functions.
Step 3: In this step, firstly the simplex function separates the constraints matrix and objective function matrix and then it computes the optimized solution based on the simplex method after some iteration. Then it provides these optimized solutions to the main function as its output argument in a single matrix containing values of decision variables and values of corresponding objective functions.
Step 4: So, in this way, repeating the step 2 and step 3, we get the maximum and minimum values for each objective function.
Step 5: Using these optimized values, the main function takes the decision variables up to *(T*-1*)* levels.
Step 6: Using these values of decision variables and values of numerator and denominator objectives, it constructs the type I, type II and type III constraints as defined in (6), (7) and (8).
Step 7: It recognizes all these constraints along with the initial constraints to construct a single constraint matrix.
Step 8: Then it constructs an objective function to minimize the sum of all the *D* variables (negative deviational variables) which are formulated during type I, II and III constraints.
Step 9: Now it again passes a matrix containing both the constraints as well as objective functions to the simplex function as its input argument.
Step 10: Now the simplex function decodes the input matrix to find the constraint matrix and objective function.
Step 11: Using the above constraints matrix and objective function, it computes the optimal solution using the usual simplex method. This optimized solution is then passed to the main function as its output argument.
Step 12: Now the main function, using these optimized values of main decision variables, computes the corresponding values of objective functions and displays them as output values.

Numerical Example

In this section, we illustrate the same numerical example considered in Lachhwani and Poonia (2012) in order to show efficiency of modified method over earlier technique as well as to demonstrate proposed MATLAB program on numerical example.

Illustration 1

$$\underset{x_1,x_2}{\text{Max}}\, Z_1 = \frac{7x_1 + 3x_2 - 4x_3 + 2x_4}{x_1 + x_2 + x_3 + 1}$$
$$\underset{x_3}{\text{Max}}\, Z_2 = \frac{x_2 + 3x_3 + 4x_4}{x_1 + x_2 + x_3 + 2}$$
$$\underset{x_4}{\text{Max}}\, Z_3 = \frac{2x_1 + x_2 + x_3 + x_4}{x_1 + x_2 + x_3 + 3}$$
$$\text{Subject to,} \quad x_1 + x_2 + x_3 + x_4 \le 5$$
$$x_1 + x_2 - x_3 - x_4 \le 2$$
$$x_1 + x_2 + x_3 \ge 1$$
$$x_1 - x_2 + x_3 + 2x_4 \le 4$$
$$x_1 + 2x_3 + 2x_4 \le 3$$
$$x_4 \le 2$$
$$x_1,\ x_2,\ x_3,\ x_4 \ge 0$$

Following the procedure, FGP model I and II can be described as:

FGP model I

$$\text{Minimize} \quad \lambda = (d_1^{N-} + d_2^{N-} + d_3^{N-} + d_1^{D-} + d_2^{D-} + d_3^{D-}) + d_1^{-}$$

$$\begin{aligned}\text{Subject to,} \quad & 7x_1 + 3x_2 - 4x_3 + 2x_4 + 23d_1^{N-} \ge 17\\ & x_2 + 3x_3 + 4x_4 + 9.5d_2^{N-} \ge 9.5\\ & 2x_1 + x_2 + x_3 + x_4 + 4d_3^{N-} \ge 5\\ & x_1 + x_2 + x_3 - 4d_1^{D-} \ge 1\\ & x_1 + x_2 + x_3 - 4d_2^{D-} \ge 1\\ & x_1 + x_2 + x_3 - 4d_3^{D-} \ge 1\\ & x_1 + 2.3333d_1^{-} \ge 2.3333\\ & x_1,\ x_2,\ x_3,\ x_4,\ d_1^{N-},\ d_2^{N-},\ d_3^{N-},\ d_1^{D-},\ d_2^{D-},\\ & d_3^{D-},\ d_1^{-} \ge 0\end{aligned}$$

Solving this programming problem, the compromise optimal solution obtained is: $\lambda = 1.8596$, $x_1 = 2.3333$,$x_2 = 0$,$x_3 = 0$, $x_4 = 2.3750$ with the values of objective functions as: $Z_1 = 5.0999$, $Z_2 = 0.3076$, $Z_3 = 0.9374$. Also the achieved values of membership functions are $\mu_1(N_1) = 0.9999$, $\mu_2(N_2) = 0.1403$,

Table 1 Comparison of values between modified FGP approach and FGP technique by Lachhwani and Poonia (2012) for numerical example 1

Parameters	Modified FGP technique (FGP model I, II)	FGP technique suggested by Lachhwani and Poonia (2012)		Better values	Better values
		Method II b	Method I	(comparison with method IIb)	(comparison with method I)
Z_1	5.0999	4.5	3.42738	$Z_1^* = 5.0999$	$Z_1^* = 5.0999$
Z_2	0.3076	1.3333	1.642437	$Z_2 = 1.3333$	$Z_2 = 1.642437$
Z_3	0.9374	0.75	0.751564	$Z_3^* = 0.9374$	$Z_3^* = 0.9374$
$\mu_{z_1}(N_1)$	0.9999	0.6521	0.7285	$\mu_{z_1}^*(N_1) = 0.9999$	$\mu_{z_1}^*(N_1) = 0.9999$
$\mu_{z_2}(N_2)$	0.1403	1	0.7155	$\mu_{z_2}(N_1) = 1$	$\mu_{z_2}(N_1) = 0.7155$
$\mu_{z_3}(N_3)$	0.9999	0.42105	0.7154	$\mu_{z_3}^*(N_3) = 0.9999$	$\mu_{z_3}^*(N_3) = 0.9999$
$\mu_{z_1}(D_1)$	0.9999	1	0.7155	$\mu_{z_1}(D_1) = 1$	$\mu_{z_1}^*(D_1) = 0.9999$
$\mu_{z_2}(D_2)$	0.9999	0.5	0.7154	$\mu_{z_2}^*(D_2) = 0.9999$	$\mu_{z_2}^*(D_2) = 0.9999$
$\mu_{z_3}(D_3)$	0.9999	1	0.7154	$\mu_{z_3}(D_3) = 1$	$\mu_{z_3}^*(D_3) = 0.9999$
at $(x_1, x_2, x_3, x_4) = (2.3333, 0, 0, 2.3750)$		at $(x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$	at $(x_1, x_2, x_3, x_4) = (0.4471, 1.1691, 0, 1.2764)$		

* indicates values from modified FGP approach

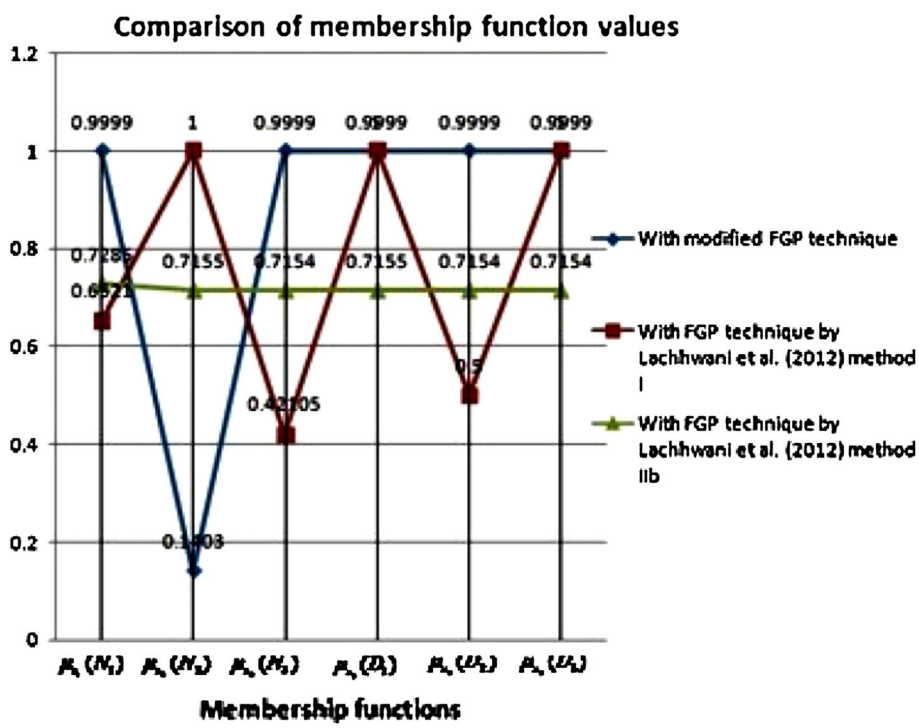

Fig. 3 Comparison of membership function values

$\mu_3(N_3) = 0.9999$, $\mu_1(D_1) = 0.9999$, $\mu_2(D_2) = 0.9999$ and $\mu_3(D_3) = 0.9999$. The same compromise optimal solution of this problem is obtained using FGP model II with the corresponding weights as defined in (15) and (16).

Note that the satisfactory solutions of the same problem using FGP technique proposed by Lachhwani and Poonia (2012) are: $(x_1, x_2, x_3, x_4) = (0.4471, 1.169105, 0, 1.2764)$ with $(Z_1, Z_2, Z_3) = (3.42738, 1.642437, 0.7515643)$ (For proposed method I) and $(x_1, x_2, x_3, x_4) =$

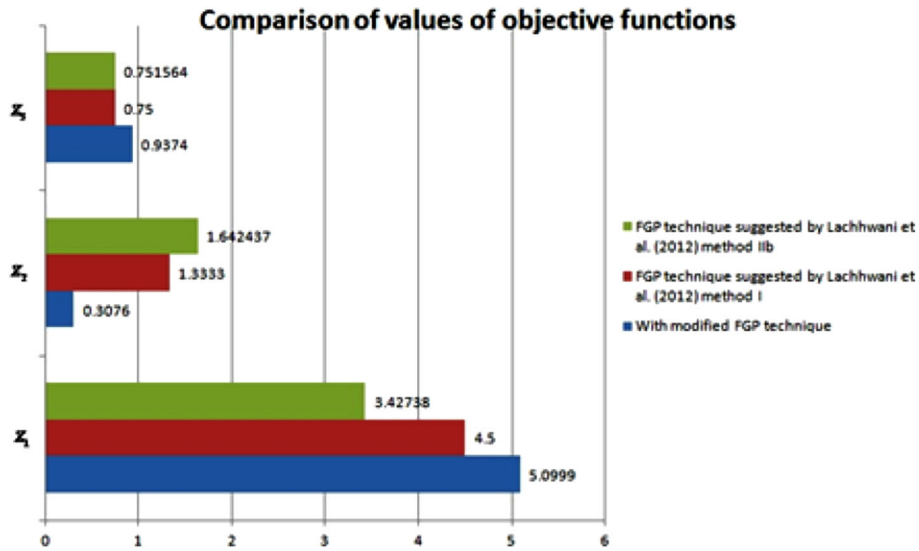

Fig. 4 Comparison of objective function values

$(1, 0, 0, 1)$ with $(Z_1, Z_2, Z_3) = (4.5, 1.3333, 0.75)$ (For proposed method II b). These satisfactory solutions of ML-LFPP are dependent on the tolerance values (p_1^-, p_1^+),(p_1^-, p_1^+) on the decision variables and type of FGP model used.

Table (Table 1) and graphs (Figs. 3, 4) show that the modified FGP technique yields better values of most of the membership functions and individual objective functions in comparison of FGP technique (method I and IIb) suggested by Lachhwani and Poonia (2012). It is clear that both the approaches are close to one another but the modified methodology is efficient and requires less computations than earlier technique in terms of considering the solution preferences by the decision maker at each level.

Again, If we compare main advantages of proposed modified FGP methodology on different parameters as shown in table (Table 2) considering theoretical aspects of techniques and numerical example, it shows that the proposed modified technique has advantages of simplicity, efficiency, construction of MATLAB program, without decision deadlock situations, less computational efforts etc. than the technique suggested by Lachhwani and Poonia (2012) on each of these parameters.

Now, if we use the proposed MATLAB program on this numerical example and input the total no. of variables, total no. of constraints, numerator/denominator objective matrices, decision variables in matrix format for each stage etc. (as shown in Fig. 5). Then we get the compromise optimal solution $(\lambda, x_1, x_2, x_3, x_4) = (1.8596, 2.3333, 0, 0, 0.3333)$ which is the same as illustrated above with our proposed methodology. This validates our proposed MATLAB program.

Conclusions

This paper presents an improved FGP technique (in terms of achieving higher values of membership functions, simplicity, computational efforts etc.) as well as generalized MATLAB program to obtain compromise optimal solution of ML-LFPPs. The proposed technique is simple, efficient and requires less computational works than that of earlier techniques. Also the proposed MATLAB program is unique and latest for solving these complex mathematical problems. This software based program is useful and user can directly obtain compromise optimal solution of ML-LFPPs with it. However, the main demerit of this MATLAB program is that construction of its coding is difficult and complex which also depends on the complexity of the problem.

Certainly there are many points for future research in the areas of MLPPs, based on modified FGP approach and should be studied. Some of these areas are:

(1) The proposed technique can be extended to more complex hierarchical programming problems like multi-level quadratic fractional programming problems (ML-QFPPs), multi-level multi-objective programming problems (ML-MOPPs) etc. and related computer programs can also be constructed in MATLAB or other programming platforms.

(2) Further modifications can be carried out in recent techniques for ML-LFPPs in order to improve efficiency of solution algorithm.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

Table 2 Comparison between modified FGP approach and FGP technique by Lachhwani and Poonia (2012) on general parameters

S. no.	Parameters	Proposed modified FGP technique (degree of parameter)	FGP technique suggested by Lachhwani and Poonia (2012) (degree of parameter)
1	Simplicity (in terms of linear/non linear structure of FGP models, repetition of tolerance values, structure of membership function etc.)	Simple (all FGP models are linear, No repetition of values & only linear membership functions are used)	Complex (repetition of tolerance values again and again, triangular membership functions used etc.)
2	Efficiency (in terms of yielding values of membership functions and objective functions)	More efficient (as shown in comparative table 1, graph 1 and graph 2)	Less efficient
3	Decision deadlock situation	Decision deadlock situations do not occur at any stage of algorithm	Frequently arise decision deadlock situations
4	Possibility of construction of MATLAB program of technique	Easily possible and coding of program is given in appendices.	Difficult to construct MATLAB program of the technique
5	Computational efforts	Less (simple FGP models, No repetition of values, solution preferences of the decision vectors are not considered)	Much (repetition of values, comparative complex FGP models, solution preferences of the decision vectors are considered)

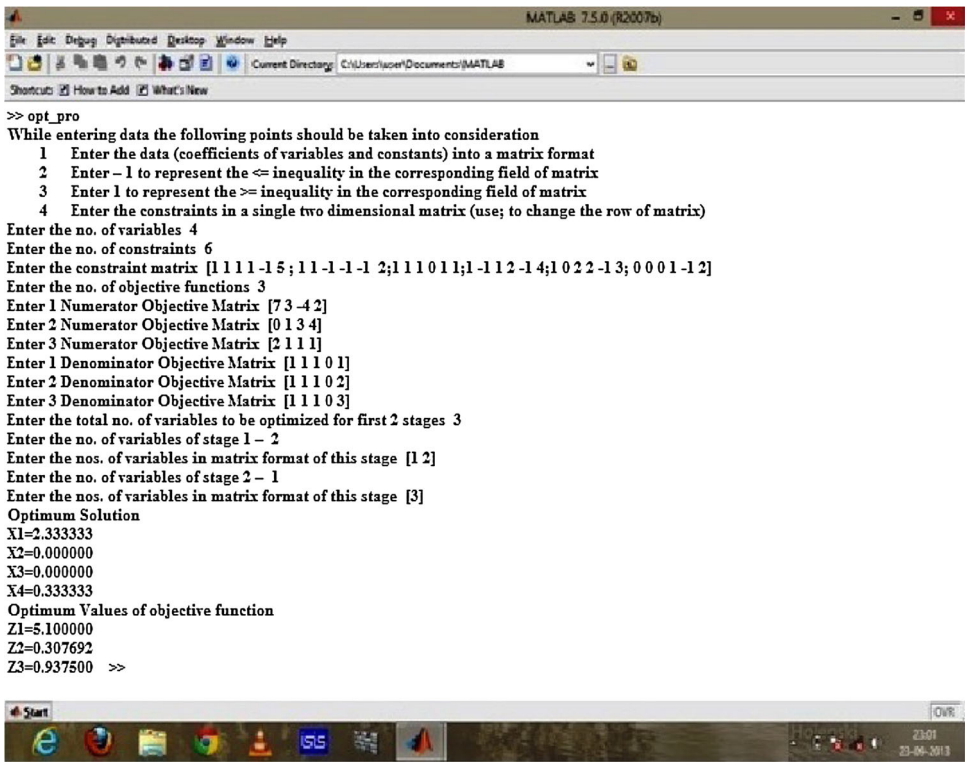

Fig. 5 Compromise optimal solution of ML-LFPP using proposed MATLAB program (trial version)

Appendices

(a) Main function(opt_pro)

```
fprintf('While entering data the following points should be taken into
consideration');
fprintf('\n1- Enter the data (coefficients of variables and constants) into a
matrix format');
fprintf('\n2- Enter -1 to represent the <= inequality in the corresponding
field of matrix ');
fprintf('\n3- Enter  1 to represent the >= inequality in the corresponding
field of matrix ');
fprintf('\n4- Enter the constraints in a single two dimensional matrix (use ;
to change the row  of matrix)');
nvar = input('\nEnter the no of variables  ');
% No. of variables
ncon = input('Enter the no of constraints ');
% No. of constraint
c=zeros(ncon,nvar+2);
 % Constraint Matrix
c =input('Enter the constraint matrix ');
nobj = input('Enter the no of objective functions  ');
% No. of objective function
onum = zeros(0,0);
% For storing the numerator objective functions
oden = zeros(0,0);
% For storing the denominator objective functions
Xnb = zeros(0,0);
% For storing the numerator objective functions
Xnl = zeros(0,0);
% For storing the denominator objective functions
Xdb = zeros(0,0);
% For storing the numerator objective functions
Xdl = zeros(0,0);
% For storing the denominator objective functions
out1=zeros(0); % For storing all optimum values and cost
k=1;
x=0;
%%
%Now we take the objective function from the user and for each objective
%function we calculate the maximum & minimum values and corresponding
%values of variables by calling the simplex function through looping
%for each time output of simplex function is stored in matrix 'out1'
constnum=zeros(nobj,1);
%for storing constant term in corresponding objective function of numerator
to be used later
constden=zeros(nobj,1);
%for storing constant term in corresponding objective function of denominator
to be used later
for i=1:2*nobj
if i <= nobj
fprintf('Enter %d Numerator Objective Matrix',i);
else
fprintf('Enter %d Denominator Objective Matrix',k);
k=k+1;
```

```
end
max1=zeros(0,0);
max=zeros(1,nvar);
max1 = input('  ');
% Objective Function
%For checking whether the objective function contains the constant term
%If the constant term in objective function is present then we separate it
from objective function and after optimization we add it to the cost(maximum
or minimum value)
%To store this constant we use a variable named 'x'
if length(max1)== nvar
max=max1;
if i <= nobj
onum=[onum; max1 0];
else
oden=[oden; max1 0];
end
end
if length(max1)> nvar
x = max1(:,nvar+1);
if i <= nobj
onum=[onum; max1];
constnum(i,1)= x;
else
oden=[oden; max1];
constden(i-nobj,1)=x;
end

max1(:,nvar+1)=[];
max = max1;
end
%Now we alternatively call the simplex function for finding maximum and
minimum
%value by passing a 'c1'matrix in which the constraints matrix and objective
matrix are merged
%output matrix of simplex function is stored in 'out' matrix and then
%merge with the 'out1' matrix
for j=1:2
if j == 1
c1=[c;max j 0];
% For Passing maximize objective function
else
max=-1*max;
% For Passing minimize objective function
c1=[c;max j 0];
end
out= simplexfun1(c1);
out(nvar+1)=out(nvar+1)+x;
% Adding the constant of objective function to the cost(Maximum or Minimum
value)
out1=[out1; out];
% Adding each optimum matrix to out1
end
end
%%
%Now using matrix algebra we separate the cost and variable values from
```

```
%matrix 'out1'
%cost-- matrix for maximum & minimum values
%Xnl-- for storing values of variables for minimum of numerator objective
functions
%Xnb-- for storing values of variables for maximum of numerator objective
functions
%Xdl-- for storing values of variables for minimum of denominator objective
functions
%Xdb-- for storing values of variables for maximum of denominator objective
functions
cost=out1(:,nvar+1);
out1(:,nvar+1)=[];
for i=1:4*nobj
if i<=2*nobj
if  mod(i,2)==0
Xnl= [Xnl; out1(i,:)];
else
Xnb= [Xnb; out1(i,:)];
end
else
if mod(i,2)==0
Xdl= [Xdl; out1(i,:)];
else
Xdb= [Xdb; out1(i,:)];
end
end
end
%similarly as above we separate maximized and minimized values for numerator
and denominator separately from 'cost' matrix
%using matrix algebra in following matrices
%Xnl-- for storing minimized values of numerator objective functions
%Xnb-- for storing maximized values of numerator objective functions
%Xdl-- for storing minimized values of denominator objective functions
%Xdb-- for storing maximized values of denominator objective functions
j=1;
k=1;
l=1;
m=1;
Nl =zeros(0);
Nb =zeros(0);
Dl =zeros(0);
Db =zeros(0);
for i=1:4*nobj
if i<= 2*nobj
if mod(i,2) == 0
Nl(j)=cost(i);
j=j+1;
else
Nb(k) = cost(i);
k=k+1;
end
end
if i>2*nobj
if mod(i,2) == 0
Dl(l)=cost(i);
l=l+1;
else
```

```
Db(m) = cost(i);
m=m+1;
end
end
end
%%
%Now we formulate the type I constraints
d1=zeros(nobj,2+nobj);
%for storing Dt1, inequality, constant columns
j=0;
%No. of Dt1 variables
for i=1:nobj
%For numerator part of each objective function
d1(i,1+j)=Nb(i)-Nl(i);
%Calculating Dt1*(Ntb-Ntl)
d1(i,1+nobj)=1;
%for greater than equality
d1(i,2+nobj)=Nb(i)+onum(i,5);
%calculating constant column i.e.=[Ntb-(constant term of objective function)]
j=j+1;
end
%Now resizing d1 matrix to add the variable matrix of objective function
onum(:,5)=[];
d11=[onum d1];
%formulated type II constraints
%%
%Now we formulate the type II constraints
d2=zeros(nobj,2+nobj);
%for storing Dt2, inequality, constant columns
j=0;
%No. of Dt2 variables
for i=1:nobj
%For denominator part of each objective function
d2(i,1+j)= -(Db(i)-Dl(i));
%Calculating -Dt2*(Dtb-Dtl)
d2(i,1+nobj)=-1;
%Inequality sign
d2(i,2+nobj)=(Dl(i)-oden(i,5));
%Calculating constant column
j=j+1;
end
%Now resizing d1 matrix to add the variable matrix of objective function
oden(:,5)=[];
d22=[oden d2];
%formulated type II constraints

%%
%Now we will formulate the type III constraints
fprintf('Enter the total no. of variables to be optimized for first %d stages
',(nobj-1) );
tvar = input('' );
d3=zeros(tvar,tvar+6); %Matrices for storing type III constraints
%Here we are using looping to compute the type III constraints corresponding
to the given stage
j=1;
%Storing no. of valid constraints
```

```
sno=1;
for st=1:(nobj-1)
fprintf('Enter the no. of variables of stage %d- ',sno );
%To identify the stage
stage = input('');
stagevn = input('Enter the nos. of variable in matrix format of this stage  '
);
%To identify the variable of above variables
sno=sno+1;
for i=1:stage
if  Xnb(st,stagevn(i))-Xnl(st,stagevn(i))~=0
%Checking validity of the constraint
d3(j,0+stagevn(i))=1/(Xnb(st,stagevn(i))-Xnl(st,stagevn(i)));
%calculating Xt/(Xtb-Xtl)
d3(j,4+j)=1;
%representing Dt31 variable
d3(j,tvar+5)=1;
%+sign for greater than equality
d3(j,tvar+6)=1+ (Xnl(st,stagevn(i))/(Xnb(st,stagevn(i))-Xnl(st,stagevn(i))));
%calculating [1+(Xtl/(Xtb-Xtl))]
j=j+1;
end
end
end
%Rearranging and resizing d3 matrix to get standard d3 matrix
j=j-1;
valid =j;
invalid=tvar-valid;
for i=1:invalid
%removing invalid columns
d3(:,5+j)=[];
end
for i=1:invalid
%removing invalid rows
d3(j+1,:)=[];
end
d33 =d3; %formulated type III constraints
%%
%resizing d11 matrix to get complete type I constraints matrix including %
%all variables %
d0=zeros(nobj,(nobj)+valid);
%zero matrices for remaining variables
d13=zeros(0,0);
%temporary matrices for storage
for i=1:2
%extracting last two columns and resizing matrix d11
dx=d11(:,nvar+nobj+1);
d13=[d13 dx];
d11(:,nvar+nobj+1)=[];
end
d11=[d11 d0 d13];
%combining three matrices to get complete type I constraints matrix
%%
%resizing d22 matrix to get complete type II constraints matrix
d01=zeros(nobj,(nobj));
%zero matrices for Dt1 variables
d02=zeros((nobj),valid);
```

```
%zero matrices for Dt31 variables
d21=zeros(0,0);
%for temporary storage
for i=1:nvar
%extracting first x-variable columns of d22 and resizing d22 matrix
dx=d22(:,1);
d21=[d21 dx];
d22(:,1)=[];
end
d23=zeros(0,0);
%for storing Dt2 variable columns
dx=zeros(0,0);
%for temporary storage
for i=1:nobj
%loop for extracting Dt2 variable columns from d22 and resizing d22
dx=d22(:,1);
d23=[d23 dx];
d22(:,1)=[];
end
d22=[d21 d01 d23 d02 d22];
%adding these matrices to get complete type II constraints matrix
%%
%resizing d11 matrix to get complete type III constraints matrix including %
%all variables %
d30=zeros(valid,nvar);
%for extracting first x-variable columns of d33 matrix
d32=zeros(valid,valid);
%for adding diagonal matrix of Dt31 variables
d34=zeros(valid,0);
%for extracting last two columns of d33 matrix
d31=zeros(valid,2*nobj);
%for adding zero matrix for Dt1 and Dt2 variables
dy=zeros(valid,1);
for i=1:valid
%extracting d30
for j=1:nvar
d30(i,j)=d3(i,j) ;
end
end
for i=1:valid
%forming d32(diagonal matrix of order (valid))
for j=1:valid
if i==j
d32(i,j) =1;
end
end
end

j=1;
for i=1:2
%extracting d34 matrix from d33
dy=d3(:,nvar+valid+j);
d34=[d34 dy];
end
d33=[d30 d31 d32 d34];
%adding these matrices to get complete type III constraints matrix
```

```
dfinal=[d11;d22;d33];
%%
%Now we rearrange and combine the initial constraints matrix c with the new
(d) constraints to form new
%constraints matrix cfinal using c1,c2,dz as temporary matrices
c2=zeros(ncon,0);
c1=zeros(ncon,2*nobj+valid);
dz=zeros(ncon,1);
for i=1:2
dz=c(:,nvar+1);
c2=[c2 dz];
c(:,nvar+1)=[];
end
c=[c c1 c2];
cfinal=[c;dfinal];
%Now we merge the objective function matrix with constraint matrix to cc
matrix to pass
%the data into the simplex function to provide optimum solution into the
%final out put matrix, using of1,of2,ofinal are temporary matrices
of1=zeros(1,nvar);
of2=ones(1,2*nobj+valid);
of2=(-1).*of2;              % Multiplying by -1 to convert maximized
objective into minimized objective function
ofinal=[of1 of2];
cc=[cfinal;ofinal 1 1];
finalout= simplexfun1(cc);
opvar=zeros(1,nvar);
for i=1:nvar
opvar(i)=finalout(i);
end
%%
%Now we display the values of optimum solution from final output matrices
fprintf('\nOptimun Solution\n');
for i=1:nvar
fprintf('x%d=%f',i,opvar(i));
fprintf('\n');
end
%finally we calculate the value of objective functions using above
%optimum solution, objective function and constant values in objective
%function
opnumvalue=zeros(nobj,1);
opdenvalue=zeros(nobj,1);
for i=1:nobj
for j=1:nvar
opnumvalue(i,1)=opnumvalue(i,1) + onum(i,j).*opvar(j);
opdenvalue(i,1)=opdenvalue(i,1) + oden(i,j).*opvar(j);
end
end
for i=1:nobj
opnumvalue(i,1)=opnumvalue(i,1)+ constnum(i,1);
opdenvalue(i,1)=opdenvalue(i,1)+ constden(i,1);
end
fprintf('\nOptimun Values of objecive function\n');
zfinal=zeros(nobj,1);
for i=1:nobj
fprintf('z%d=%f',i,(opnumvalue(i,1)./opdenvalue(i,1)));
fprintf('\n');
end
```

(b) Recursive simplex function

```
function out= simplexfun1(c1)
%This function calculates the optimized values by a matrix containing both
%the objective function and constraint matrix as an input argument 'c1'
%Here we decode the c1 matrix to get the values of following variables
% ncon= no. of constraints
% var= no. of variables
% c= original constraint matrix
% max= original objective function matrix
q=zeros(1,2);
q=size(c1);
ncon=q(1,1)-1;
var=q(1,2)-2;
c=zeros(var+2,ncon);
max=c1((ncon+1), :);
% Extracting objective matrix
c1((ncon+1), :)=[];
c=c1;
mvariable=max(var+1);
max(:,var+1)=[];
max(:,var+1)=[];
sl=0;
% Initialize No. of slack variables
sp=0;
% Initialize No. of surplus variables
spm=zeros(ncon,1);
% Used for storing coefficient of surplus variable
artloc = zeros(0,0);
% To specify location of artificial variable
% Calculating No. slack and surplus variables
j=1;
for i=1:ncon
if c(i,(var+1)) == -1
sl=sl +1;
end
if c(i,(var+1)) == 1
sp=sp +1;
spm(i)=-1;
artloc(j) = i;
j=j+1;
end
nartvar=j-1;
% No. of artificial variables
end
col1=var+1;
nbfs= ncon;
% No. of basic feasible solution
z_row=nbfs+2;
c1=zeros(ncon,col1);
for i=1:ncon
```

```
% Rearranging constraint matrix to convert into simplex table
for j=1:(var)
c1(i,j+1)=c(i,j);
end
end
for i= 1:ncon
c1(i,1)=c(i,var+2);
end
if sp ~= 0
svpm=zeros(ncon,sp);
j=1;
for i=1:ncon
if spm(i)==-1
svpm(i,j)=-1;
% 2-d matrix representing weight of artificial variables
j=j+1;
end
end
end
c1=[c1 svpm];
%resizing the c1 matrix
z1=zeros(1,sp);
z2 =zeros(1,nbfs);
%Now we provide a large -ve cost to the artificial variables and to store
%the cost corresponding to the artificial variable, we use a z2 matrix
if sp ~=0
j=1;
for i= 1:nbfs
if (j<=nartvar) && (i==artloc(j))
z2(1,artloc(j))= -500;
j=j+1;
end
end
end
z1=zeros(1,sp);
j=1;
for i=1:nbfs
%change
if (j<=nartvar) && (i==artloc(j))
cs(artloc(j),1)= -500;
j=j+1;
end
end
bfs =zeros(nbfs,1);
for i=1:nbfs
% Basic feasible variables
bfs(i,1)=var+sp+i;
end
id = eye(nbfs);
%ID Matrix of order basic feasible variable
c1=[c1 id];
tcol =var + nbfs +sp+ 3;
% Total columns
col=tcol;
d=zeros(1,tcol);
% zj -cj Row
a=[0 0 0 max z; cs bfs c1 ; d];  % First simplex table
```

```
%Calculating Zj-Cj
for i=4:col
for j=2:(nbfs+1)
a(z_row,i)=a(z_row,i)+a(j,1)*a(j,i);
end
a(z_row,i)=a(z_row,i)-a(1,i);
end
t1=-5;
%Finding next simplex tables
for i=1:35
j=1;
%Store Zj-Cj
for i=4:col
f(j)=a(z_row,i);
j=j+1;
end
t=1000;
%Find least negative number
for i=1:(col-3)
if f(i)< t
t=f(i);
end
end
%Find entering vector
ev=0;
for i=1:(col-3)
if f(i)== t
ev=i;
break;
end
end
ev=ev+3;
x=1;
%Find Bij
for i=2:(nbfs+1)
if a(i,ev) >0
c(x)=(a(i,3))/(a(i,ev));
x=x+1;
else
c(x)=10000;
x=x+1;
end
end
t1=1000;
%Find min Bij
for i=1:nbfs
if c(i)< t1
t1=c(i);
end
end
%Find key element row
ke=0;
for i=1:nbfs
if c(i)== t1
ke=i;
break;
end
```

```
end
ke=ke+1;
%Find key element
kele= a(ke,ev);
%For next step let b Matrix
%
%Changing col 1 col2
a(ke,2)=ev-3;
a(ke,1)=a(1,ev);
%Calculating number of rows to made zero
nzrow=(nbfs-1);
j=1;
%Finding which rows are to be zero
for i=1:nbfs
if (i+1) ~= (ke)
zrow(j) = i+1;
j=j+1;
end
end
%Dividing key element row by key element
for i=3:col
a(ke,i)=(a(ke,i))/kele;
end
%Rest two rows are to be made zero
for i=1:nzrow
h=a(zrow(i),ev);
for j= 3:col
a(zrow(i),j)=a(zrow(i),j)- (h*(a(ke,j)));
end
end
for i = 4:col
a(z_row,i)=0;
end
%Calculating new Zj-Cj
v= nbfs+1;
for i=4:col
for j=2:v
a(z_row,i)=a(z_row,i)+a(j,1)*a(j,i);
end
a(z_row,i)=a(z_row,i)-a(1,i);
end
%Store Zj-Cj
for i=4:col
f(j)=a(z_row,i);
j=j+1;
end
t1=1000;
%Find least negative number
u=col-3;
for i=1:u
if f(i)< t1
t1=f(i);
end
end
%Condition for termination of tables
if t >=0
break;
```

```
end


end %end for simplex tables
j=1;
%For storing optimum variables
for i=2:v
ov(j)=a(i,2);
j=j+1;
end
%Displaying optimum variables and their values
for i=1:var
optvar(i)=i;
end
optvarvalue = zeros(1,var);
for i=1:var
for j=1:nbfs
s=ov(j);
t=a((j+1),3);
if optvar(i) == s
optvarvalue(1,i)=t;
end
end
end
%Calculating optimum cost
cost=0;
for i=2:v
cost=cost+(a(i,3)*a(i,1));
end
if mvariable ==1
cost=cost;
else
cost=-cost;
end
optvarvalue(var+1)=cost;
for i=1:var+1
out(i)= optvarvalue(i);
end
end
```

References

Abo-Sinha MA, Baky IA (2007) Interactive balance space approach for solving multi-level multi-objective programming problems. Inf Sci 177:3397–3410

Aghdaghi M, Jolai F (2008) A goal programming model for vehicle routing problem with backhauls and soft time windows. J Ind Eng Int 4(6):7–18

Anandilingam G (1988) A mathematical programming model of decentralized multi-level system. J Oper Res Soc 39(11):1021–1033

Anandilingam G, Apprey V (1991) Multi-level programming and conflicting resolution. Eur J Oper Res 51:233–247

Aryanezhad MB, Malekly H, Karimi-Nasab M (2011) A fuzzy random multi-objective approach for portfolio selection. J Ind Eng Int 7(13):12–21

Babeai H, Tootooni M, Shahanaghi K, Bakhsha A (2009) Lexicograhic goal programming approach for portfolio optimization. J Ind Eng Int 5(9):63–75

Baky IA (2009) Fuzzy goal programming algorithm for solving decentralized bi-level multiobjective programming problems. Fuzzy Sets Syst 160:2701–2710

Baky IA (2010) Solving multi-level multi-objective linear programming problems through fuzzy goal programming approach. Appl Math Model 34:2377–2387

Fattahi P, Mehrabad MS, Aryanezhad MB (2006) An algorithm for multi-objective job shop scheduling problem. J Ind Eng Int 2(3):43–53

Gosh P, Roy TK (2013) A goal geometric programming problem (G^2P^2) with logarithmic deviational variables and its applications on two industrial problems. J Ind Eng Int 9:5. doi:10.1186/2251-712X-9-5

Khalili-Damghani K, Taghavifard M (2011) Solving a bi-objective project capital budgeting problem using a fuzzy multi-dimensional knapsack. J Ind Eng Int 7(13):67–73

Lachhwani K (2013) On solving multi-level multi objective linear programming problems through fuzzy goal programming approach. OPSEARCH Oper Res Soc India. doi:10.1007/s12597-013-0157-y (In press)

Lachhwani K, Poonia MP (2012) Mathematical solution of multi-level fractional programming problem with fuzzy goal programming approach. J Ind Eng Int 8:12. doi:10.1186/2251-712x-8-16

Lai YJ (1996) Hierarchical optimization: A satisfactory solution. Fuzzy Sets Syst 77:321–335

Makul A, Alinezhad A, Zohrehbandian M (2008) Practical common weights MOLP approach for efficiency analysis. J Ind Eng Int 4(6):57–63

Mohamed RH (1997) The relationship between goal programming and fuzzy programming. Fuzzy Sets Syst 89:215–222

Moitra BN, Pal BB (2002) A fuzzy goal programming approach for solving bilevel programming problems. In: Pal NR, Sugeno M (eds) AFSS, vol 2275., Lecture notes in artificial intelligenceSpringer, Berlin, pp 91–98

Pal BB, Moitra BN (2003) A fuzzy goal programming procedure for solving quadratic bilevel programming problems. Int J Intell Syst 18(5):529–540

Pal BB, Moitra BN, Malik U (2003) A goal programming procedure for fuzzy multiobjective linear fractional programming problems. Fuzzy Sets Syst 139(2):395–405

Pramanik S, Roy TK (2007) Fuzzy goal programming approach to multi-level programming problems. Eur J Oper Res 176:1151–1166

Sadjadi SJ, Aryanezhad MB, Sarfaraz A (2005) A fuzzy approach to solve a multi-objective linear fractional inventory model. J Ind Eng Int 1(1):43–47

Shih HS, Lee ES (2000) Compensatory fuzzy multiple decision making. Fuzzy Sets Syst 14:71–87

Shih HS, Lai YJ, Lee ES (1983) Fuzzy approach for multi-level programming problems. Comput Oper Res 23(1):773–791

Sinha S (2003a) Fuzzy mathematical approach to multi-level programming problems. Comput Oper Res 30:1259–1268

Sinha S (2003b) Fuzzy programming approach to multi-level programming problems. Fuzzy Sets Syst 136:189–202

Tohidi G, Razavyan S (2013) An L1-norm method for generating all the efficient solutions of multi-objective integer linear programming problem. J Ind Eng Int 8:17. doi:10.1186/2251-712X-8-17

Zeleny M (1982) Multiple criteria decision making. McGraw-Hill book company, New York

Zoraghi N, Amiri M, Talebi G, Zowghi M (2013) A fuzzy MCDM model with objective and subjective weights for evaluating service quality in hotel industries. J Ind Eng Int 9:38. doi:10.1186/2251-712X-9-38