Egyptian Informatics Journal (2017) **18**, 75–86

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

FULL-LENGTH ARTICLE

Solving stochastic programming problems using new approach to Differential Evolution algorithm

Ali Wagdy Mohamed

Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza 12613, Egypt

Received 2 October 2015; revised 24 August 2016; accepted 27 September 2016
Available online 27 October 2016

KEYWORDS

Differential evolution;
Stochastic programming;
Fractional programming;
Multi-objective programming

Abstract This paper presents a new approach to Differential Evolution algorithm for solving stochastic programming problems, named DESP. The proposed algorithm introduces a new triangular mutation rule based on the convex combination vector of the triangle and the difference vector between the best and the worst individuals among the three randomly selected vectors. The proposed novel approach to mutation operator is shown to enhance the global and local search capabilities and to increase the convergence speed of the new algorithm compared with conventional DE. DESP uses Deb's constraint handling technique based on feasibility and the sum of constraint violations without any additional parameters. Besides, a new dynamic tolerance technique to handle equality constraints is also adopted. Two models of stochastic programming (SP) problems are considered: Linear Stochastic Fractional Programming Problems and Multi-objective Stochastic Linear Programming Problems. The comparison results between the DESP and basic DE, basic particle swarm optimization (PSO), Genetic Algorithm (GA) and the available results from where it is indicated that the proposed DESP algorithm is competitive with, and in some cases superior to, other algorithms in terms of final solution quality, efficiency and robustness of the considered problems in comparison with the quoted results in the literature.

1. Introduction

Stochastic or probabilistic programming (SP) deals with situations where some or all of the parameters of the optimization problem are described by random or probabilistic variables rather than by deterministic quantities [1]. The mathematical models of these problems may follow any particular probability distribution for model coefficients [2]. The main objective is to find the optimal value for model parameters influenced by random event. The basic idea used in stochastic programming is to convert the stochastic problem into an equivalent deterministic problem which can be solved by using an appropriate classical and/or modern numerical technique. SP is widely used in many real-world decision making problems of management science, engineering, and technology. Also, it has been applied to various areas such as finance [3], manufacturing product

E-mail address: aliwagdy@gmail.com

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2016.09.002

and capacity planning [4], electrical generation capacity planning [5], supply chain management [6], water resource allocation modeling [7], portfolio selection [8], project selection [9], transportation [10], Telecommunications [11], energy planning [12], healthcare management [13] and marketing [14]. A constrained linear stochastic fractional programming (LSFP) problem involves optimizing the ratio of two linear functions subject to some constraints in which at least one of the problem data is random in nature with nonnegative constraints on the variables. In addition, some of the constraints may be deterministic. Thus, the LSFP framework attempts to model uncertainty in the data by assuming that the input or a part thereof is specified by a probability distribution, rather than being deterministic [15]. There are many algorithms to solve LSFPP such as [16–19]. Moreover, SP has been applied to the problems having multiple, conflicting and noncommensurable objectives where generally there does not exist a single solution which can optimize all the objectives [20]. Various algorithms to solve Multiobjective Stochastic Linear Programming (MOSLP) problems have been discussed in the literature [20–23]. On the other hand, over the last three decades, Evolutionary Algorithms (EAs) have been developed to solve nonlinear, high-dimensional and complex computational optimization problems. Virtually, EAs have been originally proposed to overcome the challenges of global optimization problems such as nonlinearity, non-convexity, non-continuity, non-differentiability, and/or multimodality, while traditional numerical optimization techniques had difficulties with these problems. Differential Evolution (DE) is a stochastic population-based search method, proposed by Storn and Price [24]. DE is considered the most recent EAs for solving real-parameter optimization problems [25]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [26]. In this research, two models of stochastic programming (SP) problems are considered: Linear Stochastic Fractional Programming Problems and Multi-Objective Stochastic Linear Programming Problems. For LSFPP and MOSLP, the models proposed by Charles and Dutta [19] and by Charles et al. [20], respectively, are followed. The deterministic equivalent models of these two classes of stochastic programming models are solved using a new approach to Differential Evolution algorithm, named DESP. The proposed algorithm introduces a new triangular mutation rule based on the convex combination vector of the triangle and the difference vector between the best and the worst individuals among the three randomly selected vectors. The proposed novel approach to mutation operator is shown to enhance the global and local search capabilities and to increase the convergence speed of the new algorithm compared with conventional DE. DESP uses Deb's constraint handling technique based on feasibility and the sum of constraint violations without any additional parameters. Besides, a new dynamic tolerance technique to handle equality constraints is also adopted. The results obtained by DESP algorithms are compared with basic versions of DE and PSO and also with the results quoted in the literature [19,20]. It is worth noting that although this work is an extension and modification of our previous work in [27], there are significant differences as follows: (1) previous work in [27] is proposed for unconstrained problems, whereas this work is proposed for constrained problems. Hence, (2) a new dynamic tolerance technique to handle equality constraints is also adopted, (3) the crossover rate in [27] is given by a dynamic nonlinear increased probability scheme, but in this work, the crossover rate is fixed 0.95. (4) The triangular mutation rule is only used in this work, but in the previous work [27], the triangular mutation strategy is embedded into the DE algorithm and combined with the basic mutation strategy DE/rand/1/bin through a nonlinear decreasing probability rule. (5) In previous work [27] a restart mechanism based on Random Mutation and modified BGA mutation is used to avoid stagnation or premature convergence, whereas this work does not. The remainder of this paper is organized as follows: Section 2 presents the problem of our interest and the Deb's procedure for handling constraints. In Section 3, the standard DE algorithm is introduced with a review of its operators and parameters. Next, in Section 4, the proposed DESP algorithm is described. The problem definitions are given in Section 5. In Section 6, the effectiveness of the proposed method, the experimental settings and numerical results are discussed. Finally, the conclusions and future works are drawn in Section 7.

2. Problem statement

In general, constrained optimization problem can be expressed as follows (without loss of generality minimization is considered here):

$$\text{Minimize} \quad f(\vec{x}), \quad \vec{x} = (x_1, x_2, \ldots, x_n) \in \Re^n \tag{1}$$

Subject to:

$$g_j(\vec{x}) \leqslant 0, \quad j = 1, \ldots, q \tag{2}$$

$$h_j(\vec{x}) = 0, \quad j = q+1, \ldots, m \tag{3}$$

where $\vec{x} \in \Omega \subseteq S$, Ω is the feasible region, and S is an n-dimensional rectangular space in $\Re^n$ defined by the parametric constraints $l_i \leqslant x_i \leqslant u_i, 1 \leqslant i \leqslant n$ where l_i and u_i are lower and upper bounds for a decision variable x_i, respectively. For an inequality constraint that satisfies $g_j(\vec{x}) \leqslant 0$ $(j \in 1, \ldots, q)$ at any point $\vec{x} \in \Omega$, we say it is active at $\vec{x}$. All equality constraints $h_j(\vec{x}) = 0, (j = q+1, \ldots, m)$ are considered active at all points of Ω. Most constraint-handling approaches used with EAs tend to deal only with inequality constraints. Therefore equality constraints are transformed into inequality constraints of the form $\left|h_j(\vec{x})\right| - \varepsilon \leqslant 0$ where ε is the tolerance allowed (a very small value) [28]. In order to handle constraints, we use Deb's constraint handling procedure. Deb [29] proposed a new efficient feasibility-based rule to handle constraint for genetic algorithm where pair-wise solutions are compared using the following criteria:

- Between two feasible solutions, the one with the highest fitness values wins.
- If one solution is feasible and the other one is infeasible, the feasible solution wins.
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

As a result, Deb [29] has introduced the superiority of feasible solutions selection procedure based on the idea that any individual in a constrained search space must first comply with

the constraints and then with the objective function. Practically, Deb's selection criterion does not need to fine-tune any parameter. Typically, from the problem formulation above, there are m constraints and hence the amount of constraint violation for an individual is represented by a vector of m dimensions. Using a tolerance (ε) allowed for equality constraints, the constraint violation of a decision vector or an individual $\vec{x}$ on the jth constraint is calculated by

$$cv_j(\vec{x}) = \begin{cases} \max(0, g_j(\vec{x})), & j = 1, \ldots, q \\ \max(0, |h_j(\vec{x})| - \varepsilon), & j = q+1, \ldots, m \end{cases} \tag{4}$$

If a decision vector or an individual $\vec{x}$ satisfies the jth constraint, $c_j(\vec{x})$ is set to zero. As discussed [30], in order to consider all the constraints at the same time or to treat each constraint equally, each constraint violation is then normalized by dividing it by the largest violation of that constraint in the population. Thus, the maximum violation of each constraint in the population is given by

$$c^j_{\max} = \max_{x^{\rightarrow} \in s} \quad c_j(\vec{x}) \tag{5}$$

These maximum constraint violation values are used to normalize each constraint violation. The normalized constraint violations are added together to produce a scalar constraint violation $c(\vec{x})$ for that individual which takes a value between 0 and 1

$$c(\vec{x}) = \frac{1}{m} \sum_{j=1}^{m} \frac{c_j(\vec{x})}{c^j_{\max}} \tag{6}$$

3. Methodology

3.1. Differential Evolution (DE)

This section provides a brief summary of the basic Differential Evolution (DE) algorithm. In simple DE, generally known as DE/rand/1/bin [31,32], an initial random population consists of NP vectors $\vec{X}_i$, $\forall i = 1, 2, \ldots, NP$, and is randomly generated according to a uniform distribution within the lower and upper boundaries $\left(x^L_j, x^U_j\right)$. After initialization, these individuals are evolved by DE operators (mutation and crossover) to generate a trial vector. A comparison between the parent and its trial vector is then done to select the vector which should survive to the next generation [33]. DE steps are discussed below:

3.2. Initialization

In order to establish a starting point for the optimization process, an initial population must be created. Typically, each decision variable in every vector of the initial population is assigned a randomly chosen value from the boundary constraints:

$$x^0_{ij} = x^L_j + rand_j \cdot \left(x^U_j - x^L_j\right) \tag{7}$$

where $rand_j$ denotes a uniformly distributed number between $[0, 1]$, generating a new value for each decision variable.

3.3. Mutation

At generation G, for each target vector x^G_i, a mutant vector v^{G+1}_i is generated according to the following:

$$v^{G+1}_i = x^G_{r_1} + F * \left(x^G_{r_2} - x^G_{r_3}\right), \quad r_1 \neq r_2 \neq r_3 \neq i \tag{8}$$

with randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$. F is a real number to control the amplification of the difference vector $\left(x^G_{r_2} - x^G_{r_3}\right)$. According to Storn and Price [34], the range of F is in $[0, 2]$. If a component of a mutant vector violates search space, then the value of this component is generated a new using (7) or new other repair method.

3.4. Crossover

The binomial crossover, the target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u^{G+1}_i.

$$u^{G+1}_{ij} = \begin{cases} v^{G+1}_{ij}, & rand(j) \leqslant CR \text{ or } j = randn(i), \\ x^G_{ij}, & rand(j) > CR \text{ and } j \neq randn(i), \end{cases} \tag{9}$$

where $j = 1, 2, \ldots, D$, $rand(j) \in [0, 1]$ is the jth evaluation of a uniform random generator number. $CR \in [0, 1]$ is the crossover rate, $randn(i) \in \{1, 2, \ldots, D\}$ is a randomly chosen index which ensures that u^{G+1}_i gets at least one element from v^{G+1}_i; otherwise, no new parent vector would be produced and the population would not alter.

3.5. Selection

DE adapts a greedy selection strategy. If and only if the trial vector u^{G+1}_i yields as good as or a better fitness function value than x^G_i, then u^{G+1}_i is set to x^{G+1}_i. Otherwise, the old vector x^G_i is retained. The selection scheme is as follows (for a minimization problem):

$$x^{G+1}_i = \begin{cases} u^{G+1}_i, & f\left(u^{G+1}_i\right) \leqslant f\left(x^G_i\right) \\ x^G_i, & \text{otherwise} \end{cases} \tag{10}$$

A detailed description of standard DE algorithm is given in Fig. 1.

4. DESP algorithm

All evolutionary algorithms, including DE, are stochastic population-based search methods. Accordingly, there is no guarantee that the global optimal solution will be reached consistently. Furthermore, they are not originally designed to solve constrained optimization problems. Nonetheless, adjusting control parameters such as the scaling factor, the crossover rate and the population size, alongside with developing an appropriate mutation scheme and coupling with suitable and effective constraint-handling techniques can considerably improve the search capability of DE algorithms. Moreover, the possibility of achieving promising and successful results in complex numerical and engineering constrained optimization problems increases. Therefore, in this paper, three modifications are introduced in order to significantly enhance the overall performance of the standard DE algorithm.

01. Begin
02. G=0
03. Create a random initial population $\vec{x}_i^{\,G}\ \forall i, i=1,...,NP$
04. Evaluate $f(\vec{x}_i^{\,G})\ \forall i, i=1,...,NP$
05. **For** G=1 to GEN **Do**
06. **For** i=1 to NP **Do**
07. Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
08. j_{rand}= randint(1,D)
09. **For** j=1 to D **Do**
10. **If** ($rand_j[0,1] <$ CR **or** j= j_{rand}) **Then**
11. $u_{i,j}^{G+1} = x_{r1,j}^{G} + F \cdot (x_{r2,j}^{G} - x_{r3,j}^{G})$
12. **Else**
13. $u_{i,j}^{G+1} = x_{i,j}^{G}$
14. **End If**
15. **End For**
16. **If**($f(\vec{u}_i^{G+1}) \leq f(\vec{x}_i^{G})$)**Then**
17. $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$
18. **Else**
19. $\vec{x}_i^{G+1} = \vec{x}_i^{G}$
20. **End If**
21. **End For**
22. G=G+1
23. **End For**
24. **End**

Figure 1 Description of standard **DE** algorithm. rand [0, 1) is a function that returns a real number between 0 and 1. randint (min, max) is a function that returns an integer number between min and max. *NP*, *GEN*, *CR* and *F* are user-defined parameters. *D* is the dimensionality of the problem.

4.1. Triangular mutation

Practical experience through many developed evolutionary algorithms and experimental investigation prove that the success of population-based search algorithms is based on maintaining a balance between two contradictory aspects: exploration and exploitation [35]. Moreover, the mutation scheme plays a vital role in the DE search capability and the convergence rate. However, even though the DE algorithm has good global exploration ability, it suffers from weak local exploitation ability and its convergence velocity is still too low as the region of the optimal solution is reached [36]. Clearly, from the mutation Eq. (8), it can be observed that three vectors are randomly chosen for mutation and the base vector is then randomly selected among the three. Consequently, the basic mutation strategy DE/rand/1/bin is able to maintain population diversity and global search capability, but it slows down the convergence of DE algorithms. Hence, in order to enhance the local search tendency, to improve the global exploration ability and to accelerate the convergence of standard DE technique, a new triangular mutation rule is proposed based on the convex combination vector of the triangle and the difference vector between the best and the worst individuals among the three randomly selected vectors. The modified mutation scheme is as follows:

$$v_{i,j}^{G+1} = \bar{x}_{c,j}^{G} + 2F \cdot \left(x_{best,j}^{G} - x_{worst,j}^{G}\right) \tag{11}$$

where $\bar{x}_c^G$ is a convex combination vector of the triangle and x_{best}^G and x_{worst}^G are the best and the worst individuals among the three randomly selected vectors, respectively. F is a uniform random variables, $u(0,1)$ returns a real number between 0 and 1 with uniform random probability distribution and G is the current generation number. This modification is intended to replace the random base vector $x_{r_1}^G$ in Eq. (8) by the convex combination vector $\bar{x}_c^G$ and the remaining two vectors are

replaced by the best and worst vectors of the three randomly selected vectors to yield the difference vector. The mutation operation in Eq. (11) is originally performed according to the following mutation equation:

$$v_{i,j}^{G+1} = \bar{x}_{c,j}^{G} + F \cdot \left(x_{best,j}^{G} - x_{better,j}^{G}\right) + F \cdot \left(x_{better,j}^{G} - x_{worst,j}^{G}\right) + F \cdot \left(x_{best,j}^{G} - x_{worst,j}^{G}\right) \quad (12)$$

where x_{best}^{G}, x_{better}^{G} and x_{worst}^{G} are the tournament best, better and worst three randomly selected vectors, respectively. The convex combination vector $\bar{x}_{c}^{G}$ of the triangle is a linear combination of the three randomly selected vectors and is defined as follows:

$$\bar{x}_{c}^{G} = w_1 \cdot x_{best}^{G} + w_2 \cdot x_{better}^{G} + w_3 \cdot x_{worst}^{G} \quad (13)$$

where the real weights w_i satisfy $w_i \geqslant 0$ and $\sum_{i=1}^{3} w_i = 1$. Where the real weights w_i are given by $w_i = p_i / \sum_{i=1}^{3} p_i$, $i = 1, 2, 3$. Where $p_1 = 1$, $p_2 = rand(0.5, 1)$ and $p_3 = rand(0, 0.5)$, $rand(a, b)$ are functions that return a real number between a and b, where a and b are not included. For constrained optimization problems at any generation $g > 1$, the tournament selection of the three randomly selected vectors and assigning weights follow one of the following three scenarios that may exist through generations, Without loss of generality, we only consider minimization problems:

Scenario 1: If the three randomly selected vectors are feasible, then sort them in ascending according to their objective function values and assign w_1, w_2, w_3 to x_{best}^{G}, x_{better}^{G} and x_{worst}^{G}, respectively.

Scenario 2: If the three randomly selected vectors are infeasible, then sort them in ascending order according to their constraint violation (CV) values and assign w_1, w_2, w_3 to x_{best}^{G}, x_{better}^{G} and x_{worst}^{G}, respectively.

Scenario 3: If the three randomly selected vectors are mixed (feasible and infeasible), then the vectors are sorted by using the three criteria: (a) Sort feasible vectors in front of infeasible solutions, (b) sort feasible solutions according to their objective function values, and (c) sort infeasible solutions according to their constraint violations. Accordingly, assign w_1, w_2, w_3 to x_{best}^{G}, x_{better}^{G} and x_{worst}^{G}, respectively. Obviously, from mutation Eqs. (11) and (13), it can be observed that the incorporation of the objective function value and constraints violation in the mutation scheme has two benefits. Firstly, the perturbation part of the mutation is formed by the three sides of the triangle in the direction of the best vector among the three randomly selected vectors. Therefore, the directed perturbation in the proposed mutation resembles the concept of gradient as the difference vector is oriented from the worst to the best vectors [36]. Thus, it is considerably used to explore the landscape of the objective function being optimized in different sub-region around the best vectors within constrained search-space through optimization process. Secondly, the convex combination vector $\bar{x}_{c}^{G}$ is a weighted sum of the three randomly selected vectors where the best vector has the highest contribution. Therefore, $\bar{x}_{c}^{G}$ is extremely affected and biased by the best vector more than the remaining two vectors. Consequently, the global solution can be easily reached if all vectors follow the direction of the best vectors besides they also follow the opposite direction of the worst vectors among the randomly selected vectors. Indeed, the new mutation process exploits the nearby region of each $\bar{x}_{c}^{G}$ in the direction of $\left(x_{best}^{G} - x_{worst}^{G}\right)$ for each mutated vector. In a nutshell, it concentrates the exploitation of some sub-regions of the search space. Thus, it has better local search tendency so it accelerates the convergence speed of the proposed algorithm. Besides, the global exploration ability of the algorithm is significantly enhanced by forming many different sizes and shapes of triangles in the feasible region through the optimization process. Thus, the proposed directed mutation balances both global exploration capability and local exploitation tendency.

4.2. Modification of scaling factor

In the mutation Eq. (8), the constant of differentiation F is a scaling factor of the difference vector. It is an important parameter that controls the evolving rate of the population. Regarding global unconstrained optimization, in the original DE algorithm in [30], the constant of differentiation F was chosen to be a value in $[0, 2]$. The value of F has a considerable influence on exploration: small values of F lead to premature convergence. Besides, it can lead to speed up the convergence, and high values slow down the search [36]. In this paper, for the difference vector in the mutation Eq. (11), it can be seen that it is a directed difference vector from the worst to the best vectors in the entire population. Hence, F must be a positive value in order to bias the search direction for all trial vectors in the same direction. Therefore, F is introduced as a uniform random variable within $(0, 1]$. Instead of keeping F constant during the search process, F is set as a random variable for each trial vector so as to perturb the random base vector by different directed weights. Therefore, the new directed mutation resembles the concept of gradient as the difference vector is oriented from the worst to the best vectors [37,38].

4.3. Equality constraint approach

For any feasible region limited by a set of constraints, if one of these constraints is represented by equality, the feasible region becomes a line segment. Furthermore, if all these constraints are equality constraints, then the feasible region is reduced to a point i.e. the intersection point of all constraints. Therefore, although the equality constraints are transformed into inequality constraints of the form $\left|h_j(\vec{x})\right| - \varepsilon \leqslant 0$ where ε is the tolerance allowed (a very small value), it still makes the feasible region very small compared with the search space, which exacerbates the optimization process using any evolutionary algorithm to find the feasible solutions. As a result, the larger the tolerance value, the more reliable feasible solutions found [28,39]. In other words, the temporary increase in the feasible space during the initial generations of the search process there will be some feasible individuals from the beginning of the optimization process. Then, by reducing the search space after some generations by decreasing the tolerance value (ε), the algorithm will improve the previous feasible solutions to fit into the new feasible region and satisfy the new equality constraints. Hence, in order to deal with equality constraints, the researchers introduce the new approach that dynamically changes the value of (ε) through generations. The tolerance value (ε) is adapted and decreased through generations by using the following mathematical expression:

$$Factor = \begin{cases} -1 * (\log_{10}(4)/\log_{10}(10)), & 0.00 \leqslant G/GEN < 0.05 \\ -1 * (\log_{10}(3)/\log_{10}(10)), & 0.05 \leqslant G/GEN < 0.10 \\ -1 * (\log_{10}(2)/\log_{10}(10)), & 0.10 \leqslant G/GEN < 0.15 \\ -1 * (\log_{10}(1)/\log_{10}(10)), & 0.15 \leqslant G/GEN < 0.20 \\ 1, & 0.20 \leqslant G/GEN < 0.40 \\ 2, & 0.40 \leqslant G/GEN < 0.60 \\ 3, & 0.60 \leqslant G/GEN < 0.80 \\ 4, & 0.80 \leqslant G/GEN \leqslant 1.00 \end{cases} \quad (14)$$

where $\varepsilon(t) = 10^{-Factor}$, G is the current generation number, GEN is the maximum number of generations. As can be easily derived from the above expression, as G/GEN increases toward 1, the tolerance value $\varepsilon(t)$ decreases from 4 to reach $10^{-(4)}$ and stays fixed in the later generations. In other words, as G/GEN increases, the widening feasible region considerably shrinks to the true feasible region. Indeed, as number of equality constraints increases, the initial tolerance has also a significant impact in creating the suitable initial feasible region that can attract too many infeasible points which can be improved through generations to be feasible. Moreover, as previously discussed, it can be observed that, the temporary increase in the feasible space is rapidly decreased during the initial 20% of total generations of the search process in constant rate of 5% of total generations. However, to remove the improper solutions and keep the actual optimal value, the feasible region is reduced gradually in constant rate of 20% of the total generations during the remaining generations. The description of DESP is presented in Fig. 2.

5. Problem definition

This section is divided into two subsections; in Section 1, the problem formulation of LSFPPs is given and the general model of the multi-objective SP problems with two test examples is given in Section 2.

5.1. Linear Stochastic Fractional Programming model

A linear stochastic fractional programming (LSFP) problem involves optimizing the ratio of two linear functions subject to some constraints in which at least one of the problem data is random in nature with non-negative constraints on the variables. Additionally, some of the constraints may be deterministic [40]. The LSFP framework attempts to model uncertainty

01. Begin
02. G=0 ,
03. Create a random initial population $\vec{x}_i^G \ \forall i, i = 1,...,NP$, CR= 0.95 .
04. Evaluate $f(\vec{x}_i^G), cv(\vec{x}_i^G), \forall i, i = 1,...,NP$
05. **For** G=1 to GEN **Do**
06. Compute the Factor of the tolerance value (ε) using equation (14).
07. **For** i=1 to NP **Do**
08. F =rand[0,1]
09. j_{rand}= randint(1,D)
10. Select randomly $r1 \neq r2 \neq r3 \neq i \in [1, NP]$
11. **For** j=1 to D **Do**
12. **If** (rand$_j$[0,1]< CR **or** j= j_{rand}) **Then**
13. Determine the tournament x_{best}^G ,x_{better}^G and x_{worst}^G based on their $f(\vec{x}^G), cv(\vec{x}^G)$ values , using one of three scenarios and compute $\overline{x}_c^G$ according to eq.(13).
14. $u_{i,j}^{G+1} = \overline{x}_{c,j}^G + 2F \cdot (x_{best,j}^G - x_{worst,j}^G)$
15. **Else**
16. $u_{ij}^{G+1} = x_{ij}^G$
17. **End If**
18. **End For**
19. **If** ($\vec{u}_i^{G+1}$ is better than $\vec{x}_i^G$ (based on Deb's selection criteria)) **Then**
20. $\vec{x}_i^{G+1} = \vec{u}_i^{G+1}$
21. **Else**
22. $\vec{x}_i^{G+1} = \vec{x}_i^G$
23. **End If**
24. Evaluate $f(\vec{x}_i^{G+1})$ and $cv(\vec{x}_i^{G+1}), \forall i, i = 1,...,NP$.
25. **End For**
26. G=G+1
27. **End For**
28. **End**

Figure 2 Description of **DESP** algorithm.

in the data by assuming that the input or a part there of is specified by a probability distribution, rather than being deterministic. The problem of optimizing sum of more than one ratios of function is called stochastic sum-of-probabilistic fractional programming (SSFP) problem when the data under study are random in nature. The following section gives the general model of the SSFP problems.

The mathematical model of a stochastic SSFP problem can be expressed as follows [19]:

$$\max_{x\in S} R(X) = \sum_{y=1}^{k} R_y(X),$$

where $R_y(X) = \frac{N_y(X)+\alpha_y}{D_y(X)+\beta_y}$, $y = 1,2,\ldots,k$.

Subject to:

$$p\left(\sum_{j=1}^{n} t_{ij}x_j \leqslant b_i^{(1)}\right) \geqslant 1 - p_i^{(1)}, \quad i = 12,\ldots,m; \tag{15}$$

$$\sum_{j=1}^{n} t_{ij}x_j \leqslant b_i^{(2)}, \quad i = m+1,\ldots,h \tag{16}$$

where $0 \leqslant X_{n\times 1} = \|x_j\| \subset R^n$ is a feasible set and $R: R^n \to R^n$,

$$T_{m\times n} = \left\|t_{ij}^{(1)}\right\|,$$

$$b_{m\times 1}^{(1)} = \left\|b_i^{(1)}\right\|, \quad i = 1,2,\ldots,m, \quad j = 1,2,\ldots,n;$$

$$b_{h-(m+1)\times 1}^{(2)} = \left\|b_i^{(2)}\right\|, \quad i = m+1,\ldots,h; \alpha_y, \beta_y \text{ are scalars.}$$

$$N_y(X) = \sum_{j=1}^{n} c_{yj}x_j \text{ and } D_y(X) = \sum_{j=1}^{n} d_{yj}x_j$$

In this model, out of $N_y(X)$, $D_y(X)$, T and $b^{(1)}$ at least one may be a random variable.

$S = \{X \mid$ Eqs. (15) and (16), $X \geqslant 0$, $X \subset R^n\}$ is non-empty, convex and compact set in R^n.

Test Example 1 (SSFP1)

$$\text{Max } R(X) = \sum_{y=1}^{2} \frac{c_{y1}x_1 + c_{y2}x_2 + \alpha_y}{d_{y1}x_1 + d_{y2}x_2 + \beta_y}$$

Subject to: $a_{11}x_1 + a_{12}x_2 \leqslant 1$, $a_{21}x_1 + a_{22}x_2 \leqslant b_2$, $16x_1 + x_2 \leqslant 4$, $x_1, x_2 \geqslant 0$.

The deterministic model of the above problem may be given as follows:

$$\text{Max } F(x) = \lambda_1 + \lambda_2$$

Subject to:

$$(\lambda_1 + 2\lambda_2 - 5)x_1 + (\lambda_1 + 3\lambda_2 - 4)x_2 + 2\lambda_1 + 4\lambda_2 + 1.28\sqrt{x_1^2 + x_2^2} \leqslant 3,$$

$$(2x_1 + x_2) + 1.645\sqrt{x_1^2 + x_2^2} \leqslant 1, (3x_1 + 4x_2) + 0.84\sqrt{2x_1^2 + 3x_2^2 + 2} \leqslant 3, 16x_1 + x_2 \leqslant 4, x_1, x_2, \lambda_1, \lambda_2 \geqslant 0.$$

Test example 2 (SSFP2)

$$\text{Max } R(X) = \sum_{y=1}^{3} \frac{c_{y1}x_1 + c_{y2}x_2 + \alpha_y}{d_{y1}x_1 + d_{y2}x_2 + \beta_y}$$

Subject to:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \leqslant b_1, \; a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \leqslant 20,$$
$$x_1 + x_2 + x_3 \leqslant b_3,$$
$$5x_1 + 3x_2 + 4x_3 \leqslant 15, \; x_1, x_2, x_3 \geqslant 0$$

The deterministic model of the above problem may be given as follows:

$$\text{Max } F(x) = \lambda_1 + \lambda_2 + \lambda_3$$

Subject to:

$$(\lambda_1 + 2\lambda_2 + 4\lambda_3 - 17)x_1 + (\lambda_1 + \lambda_2 + \lambda_3 - 19)x_2 + (\lambda_1 + 4\lambda_2 + 7\lambda_3 - 23)x_3 + 2\lambda_1 + 10\lambda_2 + 5\lambda_3 + 1.645\sqrt{(\lambda_2^2 + 0.5\lambda_3^2)x_1^2 + (0.5\lambda_2^2 + 2\lambda_3^2)x_2^2 + (2\lambda_2^2 + 3\lambda_3^2)x_3^2} \leqslant 12,$$

$$(4x_1 + 2x_2 + 4x_3) + 1.645\sqrt{0.5x_1^2 + 0.25x_2^2 + 0.5x_3^2 + 0.25} \leqslant 12,$$

$$(6x_1 + 4x_2 + 6x_3) + 1.28\sqrt{x_1^2 + 0.5x_2^2 + 0.75x_3^2} \leqslant 20,$$

$$x_1 + x_2 + x_3 \leqslant 3.16, \; 5x_1 + 3x_2 + 4x_3 \leqslant 15, \; x_1, x_2, x_3, \lambda_1, \lambda_2, \lambda_3 \geqslant 0.$$

Test example 3 (SSFP3)

$$\text{Max } R(X) = \sum_{y=1}^{2} \frac{c_{y1}x_1 + c_{y2}x_2 + \alpha_y}{d_{y1}x_1 + d_{y2}x_2 + \beta_y}$$

Subject to: $a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \leqslant 27$, $5x_1 + 3x_2 + x_3 \leqslant 12$, $x_1, x_2, x_3 \geqslant 0$.

The deterministic model of the above problem may be given as follows:

$$\text{Max } F(x) = \lambda_1 + \lambda_2$$

Subject to:

$$(20 - 2\lambda_1 + 4\lambda_2)x_1 + (16 - 3\lambda_1 - 2\lambda_2)x_2 + (12 - 5\lambda_1 - 2\lambda_2) - 10\lambda_1 - 12\lambda_2 - 1.28\sqrt{(\lambda_1^2 + \lambda_2^2 + 10)x_1^2 + (2\lambda_1^2 + \lambda_2^2 + 4)x_2^2 + (3\lambda_1^2 + 2\lambda_2^2 + 5)x_3^2} \geqslant 3,$$

$$(3x_1 + 4x_2 + 8x_3) + 1.645\sqrt{2x_1^2 + x_2^2 + x_3^2} \leqslant 27, \; 5x_1 + 3x_2 + x_3 \leqslant 12,$$

$$x_1, x_2, x_3, \lambda_1, \lambda_2 \geqslant 0.$$

For more details on the above examples, please refer [19].

5.2. Multi-Objective Stochastic Linear Programming model

The mathematical model of the MOSLP problem used in this study is given in the following subsection.

The general mathematical model of a constrained MOSLP may be given as [20]:

$$\text{Maximize } z_k = \sum_{j=1}^{n} c_j^k x_j, \quad k = 1,2,\ldots,k$$

Subject to

$$P\left(\sum_{j=1}^{n} a_{1j}x_j \leqslant b_1, \quad \sum_{j=1}^{n} a_{2j}x_j \leqslant b_2, \ldots, \sum_{j=1}^{n} a_{mj}x_j \leqslant b_m,\right) \geqslant P, \quad x_j \geqslant 0, \quad j = 1,2,\ldots,n.$$

where $0 < p < 1$ is usually close to 1. It has been assumed that the parameters a_{ij} and c_j are deterministic constants and b_j are random variables. For more details, the interested reader may refer to [20].

Test example 1(MOSLP1)

$$\text{Maximize } z_1 = 5x_1 + 6x_2 + 3x_3,$$
$$\text{Maximize } z_2 = 6x_1 + 3x_2 + 5x_3,$$
$$\text{Maximize } z_3 = 2x_1 + 5x_2 + 8x_3$$

Subject to

$$P(3x_1 + 2x_2 + 2x_3 \leqslant b_1) \geqslant 0.90,\ P(2x_1 + 8x_2 + 5x_3 \leqslant b_2) \geqslant 0.98,$$
$$P(5x_1 + 3x_2 + 2x_3 \leqslant b_3) \geqslant 0.95,\ P(0.5x_1 + 0.5x_2 + 0.25x_3 \leqslant b_4) \geqslant 0.90,$$
$$P(8x_1 + 3x_2 + 4x_3 \leqslant b_5) \geqslant 0.99,\ x_1, x_2, x_3 \geqslant 0.$$

Here, b_1 follow power Function distribution, b_2 follow Pareto distribution, b_3 follow Beta distribution, b_4 follow Weibull distribution, and b_5 follow Bure type XII distribution. The problem is converted to deterministic model as follows:

$$\text{Maximize } z_1 = \lambda_1(5x_1 + 6x_2 + 3x_3) + \lambda_2(6x_1 + 3x_2 + 5x_3) + \lambda_3(2x_1 + 5x_2 + 8x_3)$$

Subject to

Test example 2(MOSLP2)

$$\text{Maximize } z_1 = 3x_1 + 8x_2 + 5x_3,$$
$$\text{Maximize } z_2 = 7x_1 + 4x_2 + 3x_3,$$
$$\text{Maximize } z_3 = 6x_1 + 7x_2 + 10.5x_3$$

Subject to

$$P(5x_1 + 4x_2 + 2x_3 \leqslant b_1) \geqslant 0.95,\ P(7x_1 + 3x_2 + x_3 \leqslant b_2) \geqslant 0.95,$$
$$P(2x_1 + 7x_2 + 3x_3 \leqslant b_3) \geqslant 0.95,\ P(2x_1 + 3x_2 + 2.5x_3 \leqslant b_4) \geqslant 0.95,$$
$$P(5x_1 + 2x_2 + 1.5x_3 \leqslant b_5) \geqslant 0.95,\ x_1, x_2, x_3 \geqslant 0.$$

Here, b_1 follow power Function distribution, b_2 follow Pareto distribution, b_3 follow Beta distribution of the first kind, b_4 follow Weibull distribution, and b_5 follow Bure type XII distribution. The problem is converted to deterministic model as follows:

$$\text{Maximize } z_1 = \lambda_1(3x_1 + 8x_2 + 5x_3) + \lambda_2(7x_1 + 4x_2 + 3x_3) + \lambda_3(6x_1 + 7x_2 + 10.5x_3)$$

Subject to

$$\left[\frac{y_1^2}{9}\right]\left[\frac{y_2^2 - 100}{y_2^2}\right]\left[\frac{y_3 - 5}{10}\right]\left[\frac{e^{2y_4} - 1}{e^{2y_4}}\right]\left[\frac{3y_5^2}{1 + 3y_5^2}\right] \geqslant 0.95$$
$$5x_1 + 4x_2 + 2x_3 = y_1,\ 7x_1 + 3x_2 + x_3 = y_2,$$
$$2x_1 + 7x_2 + 3x_3 = y_3,$$
$$2x_1 + 3x_2 + 2.5x_3 = y_4,\ 5x_1 + 2x_2 + 1.5x_3 = y_5,$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1,$$
$$x_1, x_2, x_3, y_1, y_2, y_3, y_4, y_5, \lambda_1, \lambda_2, \lambda_3 \geqslant 0.$$

6. Experimental settings and numerical results

In this section the computational results of DESP are discussed as well in comparison with other algorithms.

6.1. Parameter settings

The three main control parameters of DESP algorithm, population size NP is 50 with exception to MOSLP2 with 75 as the number of decision variables increases (11 decision variables), crossover rate Cr is fixed 0.95 and F is a random variable following uniform distribution as aforementioned. For each algorithm, the stopping criterion is to terminate the search process when one of the following conditions is satisfied:

(i) the maximum number of generations is reached (assumed 1000 generations),
(ii) $|f_{\max} - f_{\min}| < 10^{-4}$ where f is the value of objective function.

A total of 50 runs for each experimental setting were conducted and the best solution throughout the run was recorded as global optimum. Results obtained by the DESP versions are compared with LDE [2], basic DE, basic PSO and also previously quoted results [19,20]. The results provided by these approaches were directly taken from [2]. Refer to [2] for more details about LDE algorithms.

6.2. Numerical results

DESP algorithm is compared with basic DE, basic PSO and LDE algorithms through various performance metrics like average fitness function value and standard deviation (SD). To compare the convergence speed of algorithms, we considered the average number of function evaluations (NFEs).

6.3. Result analysis of SSFP

In order to study the performance of the proposed algorithm DESP on stochastic sum-of-fractional programming (SSFP) problems, performance comparison of DESP algorithm with basic DE, basic PSO, LDE algorithms and results in [19] is given in Tables 1–3 in terms of best objective function value of deterministic model ($F(X)$), optimal decision variable values, standard deviation and average NFEs.

For the best results obtained by DESP, the constraints are [0, 0, −0.495616788561048, −0.597382431501665] to ensure that it is feasible solution, and the objective function value is 1.83246. It can be seen from Table 1 that DESP, LDE1,

Table 1 Results of stochastic sum-of-fractional programming problem 1 (SSFP1).

	DE	LDE1	LDE2	PSO	Results in [19]	DESP
$F(X)$	1.83245	1.83246	1.83246	1.83218	1.7533	1.83246
x_1	0.202128	0.20199	0.202329	0.20254	0.0602	0.202324
x_2	0.165738	0.165968	0.165426	0.164923	0.3292	0.165433
λ_1	1.83245	1.883246	1.83246	1.83218	1.7533	1.83246
λ_2	0	0	0	0	0	0
SD	1.2639E−05	1.2143E−05	1.2147E−05	0.120788	NA	4.23E−08
Average NFE	15,925	11,790	5455	37,269	NA	4630

Table 2 Results of stochastic sum-of-fractional programming problem 2 (SSFP2).

	DE	LDE1	LDE2	PSO	Results in [19]	DESP
$F(X)$	15.2256	15.0329	15.2256	15.2231	15.1931	15.2256
x_1	0	0.0101852	0	0	0	0
x_2	1.42539	1.76232	1.43882	1.46877	1.4284	1.424836
x_3	1.68131	1.3875	1.67478	1.65862	1.6818	1.681578
λ_1	15.2256	15.0329	15.2256	15.2231	15.1931	15.2256
λ_2	0	7.278E−08	7.268E−08	0	0	0
λ_3	6.039E−07	5.641E−12	0	0	0	0
SD	1.4346E−05	1.2029E−05	1.1060E−05	1.06941	NA	1.03E−05
Average NFE	38,855	4540	10,460	42,352	NA	10,170

Table 3 Results of stochastic sum-of-fractional programming problem 3 (SSFP3).

	DE	LDE1	LDE2	PSO	Results in [19]	DESP
$F(X)$	2.32854	2.23663	2.33083	2.23657	3.6584	2.330846
x_1	1.88002	2.39981	1.781	2.4	2.4	1.783185
x_2	0.866625	0.00031	1.03165	0	0	1.028024
x_3	0	0	0	0	0	0
λ_1	2.32854	2.23663	2.33083	2.23657	3.6584	2.330846
λ_2	2.095E−07	1.923E−06	0	0	0	0
SD	0.05265	0.03543	0.03305	0.335365	NA	0.018375
Average NFE	5725	5350	4850	44,656	NA	5903

LDE2 and DE are able to find the optimal solution consistently in all runs but DESP has the lower standard deviation. Moreover, DESP has the better "best" objective function value than this obtained in [19] which is slightly infeasible solution. It is to be noted that the improvement percentage of DESP in terms of NFEs in comparison with DE, LDE1, LDE2 and PSO is 70.9%, 60.7%, 15.12% and 87.5%, respectively. Therefore, DESP is considered the most efficient with the smallest (NFE).

For the best results obtained by DESP, the constraints are [0, 0, −1.944514465098703, −0.053585156014804, −3.999177179444022] to ensure that it is feasible solution, and the objective function value is 15.2256. It can be seen from Table 2 that DESP, LDE2 and DE are able to find the optimal solution consistently in all runs but DESP has the lower standard deviation. Besides, although the improvement percentage of LDE1 in terms of NFEs in comparison with DESP is 55.35%, the best solution obtained by LDE1 is inferior to the best solution obtained by DESP. Moreover, DESP has the better "best" objective function value than those obtained by PSO and [19]. It is to be noted that the improvement percentage of DESP in terms of NFEs in comparison with DE, LDE2 and PSO is 73.8%, 0.027% and 75.98%, respectively. Therefore, DESP is considered the most efficient with the smallest (NFE).

For the best results obtained by DESP, the constraints are [0, −13.058526571600890, 0] to ensure that it is feasible solution, and the objective function value is 2.330846. It can be seen from Table 3 that DESP and LDE2 are able to find the optimal solution consistently in all runs but DESP has the lower standard deviation. However, LDE2 is considered the most efficient with the smallest (NFEs). Besides, although the improvement percentage of DE and LDE1 in terms of NFEs in comparison with DESP is 0.03% and 0.09%, respectively, which is not a significant difference for the comparison, the best solutions obtained by DE and LDE1 are inferior to the best solution obtained by DESP. Moreover, DESP has the better "best" objective function value than that obtained by PSO. The improvement percentage of DESP in terms of NFEs in comparison with PSO is 86.78%. Moreover, it must be noted that the best solution obtained by [19] is infeasible solution as the constraints are [23.999557387890611, −14.216684855751021, 0], and the objective function value is 3.6584.

6.4. Results analysis of MOSLP

In order to study the performance of the proposed algorithm DESP on Multi-objective Stochastic Linear Programming Problems (MOSLP) problems, performance comparison of DESP algorithm with basic DE, basic PSO, LDE algorithms and results in [20] is given in Tables 4 and 5 in terms of best objective function value of deterministic model ($F(X)$), optimal decision variable values, standard deviation and average NFEs.

For the best results obtained by DESP, the constraints are [−3.077116474765064, −0.000000085756271, −1.4790101 31249243, −0.533838884207691, −3.567129578475143, −0.000000137249561] to ensure that it is feasible solution, and the objective function value is 12.9312. It can be seen from Table 4 that DESP and PSO are able to find the optimal solution consistently in all runs but DESP has the lower standard deviation. Thus, DESP is more robust than PSO. Besides, the improvement percentage of DESP in terms of NFEs in comparison with PSO is 43.66%. Additionally, the best solutions obtained by DE and LDE1, LDE2 and the quoted result

Table 4 Results of Multi-Objective Stochastic Linear Programming Problem 1 (MOSLP 1).

	DE	LDE1	LDE2	PSO	Results in [20]	DESP
z	9.48978	10.1553	12.0647	12.9299	8.5089	12.9312
z_1	6.18688	6.22744	6.12031	4.84872	6.4834	4.84872
z_2	9.48978	9.34841	9.39769	8.0812	8.3125	8.36666
z_3	12.5073	12.2764	12.4212	12.9299	10.5140	12.9299
x_1	0.352147	0.35354	0.344071	0	0.3727	2.43E−06
x_2	2.124E−07	0.0293907	0	0	0.2319	1.48E−06
x_3	1.47538	1.4278	1.46665	1.61624	1.0761	1.616237
SD	2.06789	1.62183	1.95275	3.91715	NA	1.74E−04
Average NFE	14,691	6932	7250	20,573	NA	11,590

Table 5 Results of Multi-Objective Stochastic Linear Programming Problem 2 (MOSLP 2).

	DE	LDE1	LDE2	PSO	Results in [20]	DESP	DESP1
z	6.87235	7.13425	7.73912	7.13425	3.2081	16.689347	15.299629
z_1	3.32379	4.18588	4.46621	4.18588	3.8139	7.946516	7.3175884
z_2	1.99418	2.46762	2.57927	2.46762	3.0717	4.767910	5.5454252
z_3	6.9787	8.19198	8.12872	8.19198	5.1968	16.687682	15.300104
x_1	2.65E−05	9.44E−04	3.08E−04	9.44E−04	0.1939	9.06E−09	0.222094
x_2	1.27E−04	0.061029	0.127573	0.061029	0.2810	1.23E−07	2.08E−05
x_3	0.664552	0.738963	0.688939	0.738963	0.1968	1.589303	1.330228
y_1	1.32963	1.72678	1.88791	1.72678	2.4872	3.178698	3.771069
y_2	0.664947	0.928675	1.07383	0.928675	2.3971	1.589204	2.884872
y_3	1.99454	2.64598	2.96046	2.64598	2.9454	4.767809	4.435026
y_4	1.66177	2.03239	2.10569	2.03239	1.7229	3.9733268	3.769867
y_5	0.9971	1.0	1.0	1.0	1.8267	2.3839530	3.105837
SD	2.20171	1.77604	1.41855	0.490581	NA	4.954E−02	3.250082
Average NFE	30,794	50,050	50,050	50,050	NA	64,929	61,200

[20], where the problem is solved by Genetic Algorithm (GA), are inferior to the best solution obtained by DESP regardless of their average NFEs. The improvement percentage of DESP in terms of objective function value in comparison with DE, LDE 1, LDE 2 and quoted result [20] is 136.2%, 127%, 1.07% and 152%, respectively.

Actually, this problems is very difficult to solve as it has the following common features: moderate dimensionality (eleven decision variables) which is almost twice the number of decision variables in MOSLP 1 problem (six decision variables), nonlinear objective functions, more than one equality constraint (six constraints) and one nonlinear inequality constraint, and feasibility metric of zero as the feasible region is reduced to a point i.e. the intersection point of all constraints. Thus, it is very difficult to generate feasible solutions during the initial search process with these types of problem. However, by using large tolerance value i.e. by widening the true feasible area, more feasible solutions can be generated to satisfy the equality constraints in the initial generation. Then, through generation, while the initial feasible region is contracted, the algorithm improves the previous feasible solutions to satisfy the re-defined feasible space. As a result, it can be deduced that the proposed equality constrained approach is mainly responsible for the poor or good performance of DESP with equality constrained problems with initial tolerance value of $\varepsilon(t) = 4$. From Table 5, it can be obviously seen that the best results obtained by DESP are the optimal solution for this challenge problem as the constraints are −0.000000065533720, −0.000008358091382, −0.000001611376663, −0.000000290843902, −0.000030808461554, −0.000098702360594, −0.000000037835539] to ensure that it is the feasible and optimal solution (the intersection point of all equality constraints), and the objective function value is 16.689347. Thus, regardless of the average (NFEs) of all compared algorithms, DESP algorithm is superior to others in terms of objective function value and it is the only algorithm that is able to find new and the optimal solution to MOSLP 2 problem while the results obtained by others are infeasible solutions. In order to verify the efficiency of the proposed dynamic tolerance rule to handle equality constraints, another version of DESP algorithm, named DESP1, solves the MOSLP 2 problem without using

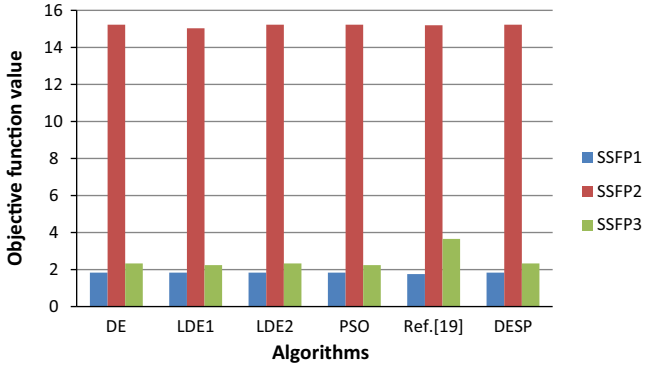

Figure 3 Performance of DE, LDE1, LDE2, PSO, Ref. [19] and DESP algorithms in terms of objective function value for SSFPs.

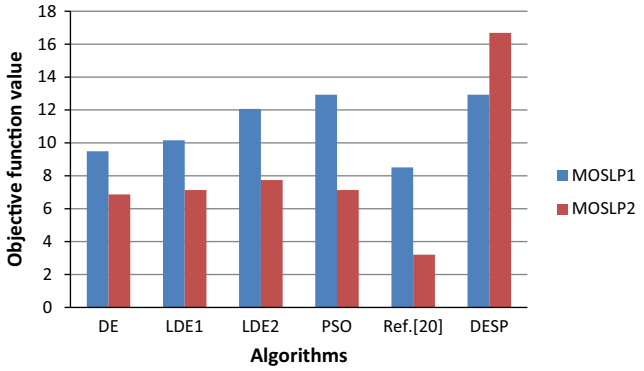

Figure 4 Performance of DE, LDE1, LDE2, PSO, Ref. [19] and DESP algorithms in terms of objective function value for MOSLPs.

the proposed rule, applying a constant tolerance value of 0.0001 along the whole search process. Table 5 presents the results of DESP without new dynamic tolerance rule using the same parameter settings. The presented results in Table 5 show that DESP1 was only able to find the best solution which is very close to the optimal solution but it was not able to reach it consistently in all runs as the standard deviation is 3.250082. Consequently, it can be deduced that the proposed dynamic tolerance rule has the main role in solving constrained problems with equality constraints. However, the improvement percentage of DESP1 in terms of objective function value in comparison with DE, LDE 1, LDE 2, PSO and quoted result [20] is 222.62%, 214.45%, 197.69%, 214.45% and 476.9%, respectively. Thus, it is clearly that DESP1 is superior with others in solving this challenge problem. Based on the above analysis, results and comparisons, it can be concluded that DESP is able to consistently find the global optimal in all stochastic programming problems with a very small standard deviation and with a relatively small average (NFEs) which indicates that the proposed DESP has a remarkable ability to solve considered stochastic programming problems with a perfect performance in terms of high quality solution, rapid convergence speed, efficiency and robustness. Besides, its performance is superior and competitive with all compared algorithms. Finally, it is noteworthy that the optimal solution of MOSLP 2 which is the most difficulty constrained problem is found. Figs. 3 and 4 show the performance of DESP, DE, LDE, PSO algorithms and the quoted results in [19,20] in terms of objective function value for SSFPs and MOSLPs, respectively.

7. Conclusions and future work

Stochastic programming (SP) is a framework for modeling optimization problems that involve uncertainty. In this research, two models of SP problems were considered: (i) SSFPs and (ii) MOSLPs problems. The deterministic equivalent models of these two classes of stochastic programming models are solved using a new approach to Differential Evolution algorithm, called DESP. The proposed DESP algorithm has been compared with basic DE, basic PSO, LDE1, LDE2 and the quoted algorithm from the literature. The experimental results and comparisons have shown that the DESP algorithm performs better in constrained SP problems with different types, complexity and dimensionality; it performs better with regard to the search process efficiency, the final solution quality, the convergence rate, and robustness, when compared with other algorithms. Finally, the performance of the DESP algorithm is superior to and competitive with other compared algorithm. Besides, it is noteworthy to mentioning that DESP algorithm finds the optimal solution for the most difficult MOSLP problem due to the proposed procedure for handling equality constraints. This proves that The DESP algorithm is considered as perfect alternative for solving SP problems. Several current and future works can be developed from this study. Firstly, current research effort focuses on how to control the crossover rate by self-adaptive mechanism. Besides, another benchmark constrained and mixed integer programming problems will be solved using DESP. Additionally, future research will investigate the performance of the DESP algorithm in solving unconstrained and constrained multi-objective optimization problems as well as real world applications. Additionally, the promising research direction is joining the proposed triangular mutation with evolutionary algorithms, such as genetic algorithms, harmony search and particle swarm optimization, as well as foraging algorithms such as artificial bee colony, bees algorithm and ant colony optimization.

References

[1] Rao SS. Engineering optimization: theory and practice. 4th ed. Hoboken (NJ): John Wiley & Sons Inc.; 2009.
[2] Thangaraj R, Pant M, Bouvry P, Abraham A. Solving stochastic programming problems using modified differential evolution algorithms. Log J IGPL 2012;20(4):732–46.
[3] Carino DR, Kent T, Meyers DH, Stacy C, Sylvanus M, Turner AL, et al. The Russell-Yasuda Kasai model: an asset liability model for a Japanese insurance company using multistage stochastic programming. Interfaces 1994;24(1):29–49.
[4] Eppen GD, Martin RK, Schrage L. A scenario approach to capacity planning. Oper Res 1989;37(4):517–27.
[5] Murphy FH, Sen S, Soyster AL. Electric utility capacity expansion planning with uncertain load forecasts. IIE Trans 1982;14(1):52–9.
[6] Fisher M, Hammond J, Obermeyer W, Raman A. Configuring a supply chain to reduce the cost of demand uncertainty. Prod Oper Manage 1997;6(3):211–25.
[7] Abdelaziz FB, Mejri S. Application of goal programming in a multi-objective reservoir operation model in Tunisia. Eur J Oper Res 2001;133(2):352–61.
[8] Abdelaziz FB, Aouni B, Fayedh RE. Multi-objective stochastic programming for portfolio selection. Eur J Oper Res 2007;177(3):1811–23.
[9] Mukherjee K, Bera A. Application of goal programming in project selection decision - a case study from the Indian coal mining industry. Eur J Oper Res 1995;82(1):18–25.
[10] Yang L, Feng Y. A bicriteria solid transportation problem with fixed charge under stochastic environment. Appl Math Mod 2007;31(12):2668–83.
[11] Sen S, Doverspike RD, Cosares S. Network planning with random demand. Telecommun Syst 1994;3(1):11–30.
[12] Birge JR, Rosa CH. Modeling investment uncertainty in the costs of global CO_2 Emission policy. Eur J of Oper Res 1995;83(3):466–88.
[13] Abdelaziz FB, Masmoudi M. A multiobjective stochastic program for hospital bed planning. J Oper Res Soc 2012;63(4):530–8.

[14] Bhattacharya UK. A chance constraints goal programming model for the advertising planning problem. Eur J Oper Res 2009;192(2):382–95.
[15] Charles V, Yadavalli VSS, Rao MCL, Reddy PRS. Stochastic fractional programming approach to a mean and variance model of a transportation problem. Math Prob Eng 2011, 12pages 657608.
[16] Charles V, Dutta D. A method for solving linear stochastic fractional programming problem with mixed constraints. Acta Cienci Indica Math 2004;30(3):497–506.
[17] Charles V, Dutta D. Linear stochastic fractional programming problem with branch-and-bound technique. In: Proceedings of national conference on mathematical and computational models. Coimbatore (India): PSG College of Technology; 2001. p. 31–139.
[18] Charles V, Dutta D, Appal RK. Linear stochastic fractional programming problem. In: Proceedings of international conference on mathematical modeling. India: University of Roorkee; 2001. p. 211–7.
[19] Charles V, Dutta D. Linear stochastic fractional programming with sum-of-probabilistic-fractional objective. Optim Online; 2005. http://www.optimizationonline.org.
[20] Charles V, Ansari SI, Khalid MM. Multi-objective stochastic linear programming with general form of distributions. Int J Oper Res Optim 2011;2(2):261–78.
[21] Suwarna H, Biswal MP, Sinha SB. Fuzzy programming approach to multiobjective stochastic linear programming problems. Fuzzy Set Syst 1997;88(1):173–81.
[22] Baba N, Morimoto A. Stochastic approximations methods for solving the stochastic multiobjective programming problem. Int J Syst Sci 1993;24(4):789–96.
[23] Caballero R, Cerdá E, Munoz MM, Rey L, Stancu MIM. Efficient solution concepts and their relations in stochastic multiobjective programming. J Optim Theory Appl 2001;110(1):53–74.
[24] Storn R, Price K. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report TR-95-012. ICSI; 1995.
[25] Storn R, Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11(4):341–59.
[26] Engelbrecht AP. Fundamentals of computational swarm intelligence. John Wiley & Sons Ltd; 2005.
[27] Mohamed AW. An improved differential evolution algorithm with triangular mutation for global numerical optimization. Comput Ind Eng 2015;85:359–75.
[28] Mohamed AW, Sabry HZ. Constrained optimization based on modified differential evolution algorithm. Inf Sci 2012;194:171–208.
[29] Deb K. An efficient constraint handling method for genetic algorithms. Comput Method Appl Mech 2000;186(2–4):311–38.
[30] Venkatraman S, Yen GG. A generic framework for constrained optimization using genetic algorithms. IEEE Trans Evol Comput 2005;9(4):424–35.
[31] Storn R, Price K. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11(4):341–59.
[32] Mohamed AW, Sabry HZ, Abd-Elaziz T. Real parameter optimization by an effective differential evolution algorithm. Egypt Inf J (Elsevier) 2013;14:37–53.
[33] Das S, Suganthan PN. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 2011;15(1):4–31.
[34] Price KV, Storn RM, Lampinen JA. Differential evolution - a practical approach to global optimization. Berlin: Springer; 2005.
[35] Mohamed AW. An efficient modified differential evolution algorithm for solving constrained non-linear integer and mixed-integer global optimization problems. Int J Mach Learn Cyber 2015:1–19. http://dx.doi.org/10.1007/s13042-015-0479-6.
[36] Feoktistov V. Differential evolution. In: Search of solutions. Springer; 2006.
[37] Mohamed AW, Sabry HZ, Khorshid M. An alternative differential evolution algorithm for global optimization. J Adv Res 2012;3(2):149–65.
[38] Mohamed AW. RDEL: restart differential evolution algorithm with local search for global numerical optimization. Egypt Infom J 2014;15:175–88.
[39] Brest J. Constrained real-parameter optimization with ε-self-adaptive differential evolution. In: Mezura-Montes Efrén, editor. Constraint-handling in evolutionary computation. Studies in computational intelligence, vol. 198. Berlin: Springer; 2009. p. 73–93, chapter 4, ISBN 978-3-642-00618-0.
[40] Charles V. E-model for transportation problem of linear stochastic fractional programming. Optim Online; 2007. http://www.optimizationonline.org.