Received: January 10, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Monte Carlo Simulation-Based Bat Algorithm for Solving Stochastic Multi-Objective Optimization Problems

Marwa Mostafa Sabry[1]* **Assem Tharwat[2]** **Ihab El-Khodary[1]**

[1]*Faculty of Computers and Information, Cairo University, Egypt*
[2]*College of Business Administration, American University in the Emirates, UAE*
* Corresponding author's Email: m.sabry@fci-cu.edu.eg

Abstract: This paper develops a stochastic Bat algorithm based on Monte Carlo simulation for solving stochastic multi-objective problems, where model parameters are random variables follow any arbitrary continuous distribution. The traditional Bat algorithm requires the deterministic equivalence of the problem which is only possible if the random variables follow specific distributions. However, in the developed algorithm, the stochastic model is solved directly without obtaining the deterministic equivalence. The feasibility of the stochastic constraints is checked using Monte Carlo simulation, and the developed algorithm is used to obtain the optimal solution. The developed algorithm combined the advantageous of the Bat algorithm and Monte Carlo techniques to solve complex stochastic multi-objective problems without the deterministic equivalent, which is difficult in most cases, and is tested on a numerical example that was previously solved by other algorithms and the obtained results are compared and showed that the proposed algorithm is more efficient and robust.

Keywords: Stochastic programming, Chance constraint, Continuous random variable, Monte Carlo simulation, Bat algorithm.

1. Introduction

In a real-world decision situation, the decision maker is often faced with multiple objectives under the problem of uncertain parameter values due to the imprecision in human judgments as well as the uncertainty in nature of the parameters involved in the problem. One approach for solving such problems is stochastic programming that handles the probabilistically uncertain data. Stochastic programming deals with situations where some or all of the parameters of a mathematical programming problem are described by stochastic variables rather than by deterministic values. One of the available techniques is the chance constrained programming (CCP) which can be used to solve problems involving chance constraints, that is, constraints having pre-determined level of probability to be attained. As such, CCP assumes that the stochastic system of constraints is satisfied at a pre-determined confidence level, α. CCP was initially developed by Charnes and Cooper [1], and today is being used intensively to model and solve problems in many applications of engineering, telecommunication, finance, etc.

The general form for a multi-objective stochastic programming problem with chance constraints is as follows:

$$\begin{aligned} &\text{Maximize: } z^k(x) = \sum_{j=1}^{n} c_j^k x_j, && k = 1, 2, \dots, K \\ &\text{Subject to:} \\ &Prob\left[\sum_{j=1}^{n} a_{ij} x_j \le b_i\right] \ge \alpha_i \\ &x_j \ge 0 && j = 1, \dots, n \\ &\alpha_i \in (0, 1) && i = 1, \dots, m \end{aligned} \quad (1)$$

Where, c_j^k is the coefficient of decision variable x_j for the k^{th} objective function, a_{ij} and b_i are continuous random variables with known probability distributions and α_i is a pre-determined probability (confidence) level.

In many situations, it turns out to be very difficult and complex to get the deterministic equivalence of the chance constraints, especially if the random components cannot be separated from the decision variables, as in this case, the multivariate integration would lead to improper calculations.

Metaheuristic algorithms offer successful methodologies to deal with complex optimization problems. They have two important characteristics: intensification and diversification. The selection criterion of the first characteristic, intensification, which is used interchangeably with exploitation, is mainly based on searching around its neighborhood from the current best solutions. While the selection criterion of second characteristic, diversification, also used interchangeably with exploration, is based on randomization and thus assures the effectiveness of the search process in the search space [2, 4]. The efficiency of these algorithms is due to the fact that they successfully simulate the naturally evolved characteristics in nature, especially of the biological systems.

Most widely used Meta-heuristic algorithms are Genetic algorithm (GA), simulated annealing (SA), Tabu search (TS) and Particle Swarm Optimization (PSO). Genetic algorithm (GA) emulates the evolutionary process in nature, whereas Tabu search (TS) exploits the memory structure in living beings, simulated annealing (SA) imitates the annealing process in crystalline solids and particle swarm optimization (PSO) inspires social behavior of bird flocking or fish schooling [5]. A general drawback for these algorithms, with the exception of PSO, is the slow convergence towards the optimal solution and some of them have poor performance with large search spaces problems [6]. A crucial drawback with GA is its unguided mutation operator where there is no directed mechanism for fine tuning its parameters (trial and error only), thus does not guarantee reaching optimal solution. Although TS tries to avoid trapping into local optimum but it is very slow taking much time when performing especially with large space problems. SA, on the other hand, requires high accuracy when choosing its tuning parameters and its great need of computer time for many runs.

With the recent developments in the field of metaheuristic algorithms, the Bat algorithm (BA) which was proposed by Yang [7]. BA is a new metaheuristic algorithm that simulates the echolocation behavior of microbats. The microbat's natural phenomena of echolocation represent the main focus in the search process. In complete darkness, these bats can easily find their prey and even differentiate different types of other insects. The research studies indicated that BA is superior in solving nonlinear constrained optimization problems and showed very promising results that outperform many existing algorithms [8]. It has some advantages over other algorithms since it uses a fewer number of adjustable parameters. A frequency tuning technique is used in the BA which increases the diversity of the solutions in the population. During searching for a solution (prey), the algorithm automatically modifies the pulse emission rates and loudness of bats through an internal zooming technique, and thus it always achieves both the exploration and exploitation of solution through the search process. Consequently, BA has been widely used in many applications like engineering design optimization, fuzzy clustering, predictions and other real problems.

In this paper, a stochastic multi-objective bat algorithm (SMOBA) is developed for solving stochastic constrained optimization problems. The generation of random numbers is done based on the given probability distributions of their continuous random variables. Then the Monte Carlo simulation technique is applied to ensure the feasibility of the chance constraints. Then the multi-objective Bat algorithm is used to obtain the optimal solution. The probability distributions for the random variables need not be the same for different constraints.

The organization of this paper is as follows: Section 2 presents method of generating random numbers for some continuous probability distributions. Section 3 introduces the technique of Monte Carlo stochastic simulation for the chance constraints. Section 4 gives the working technique of the Bat algorithm. Section 5 discusses the developed algorithm (SMOBA) in details. Numerical example and concluding remarks are presented in Sections 6 and 7, respectively.

2. Generating random numbers for certain probability distributions

In this section, we will quickly give some algorithms for generating random numbers for some continuous probability distributions:

a) Uniform Distribution:

A random variable x has a uniform distribution when its probability distribution function (pdf) is given by:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & otherwise. \end{cases} \qquad (2)$$

The pdf is denoted as U*(a, b)*, where *a* and *b* are given real numbers with *a* < *b*. Thus, to generate a uniformly distributed random number on an interval [*a, b*], the algorithm is as follows:

Step 1. *m = rand ()*.
Step 2. Return *a + m (b − a)*.

b) Exponential Distribution:

A random variable *x* has an exponential distribution when its pdf is given by:

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-x/\beta}, & x > 0 \\ 0, & otherwise \end{cases} \quad (3)$$

The pdf is denoted as Exp(β), where $\beta > 0$. β is the mean and β^2 is he variance of the distribution.
To generate an exponentially distributed random number, the algorithm is as follows:
Step 1. *Generate m from U (*0, 1*).*
Step 2. *Return* $-\beta$ ln*(m).*

c) Gamma Distribution:

A random variable *x* has a gamma distribution when its pdf is given by:

$$f(x) = \begin{cases} \frac{x^{\alpha-1} e^{-x/\beta}}{\beta^{\alpha}\,\Gamma(\alpha)}, & x > 0 \\ 0, & otherwise \end{cases} \quad (4)$$

The pdf is denoted as $G(\alpha, \beta)$, where $\alpha, \beta > 0$ and $\alpha\beta$, $\alpha\beta^2$ are the mean and variance of the distribution.
To generate a Gamma distributed random number, the algorithm is as follows:
Step 1. *Set x* = 0.
Step 2. *Generate m from* EXP*(*1*).*
Step 3. *x* ← *x* + *m*.
Step 4. $\alpha \leftarrow \alpha - 1$.
Step 5. *Repeat Steps 2–4 until* α = 1.
Step 6. *Return βx.*

d) Weibull Distribution

A random variable *x* has a Weibull distribution when its pdf is given by:

$$f(x) = \begin{cases} \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{(-x/\beta)^{\alpha}}, & x > 0 \\ 0, & otherwise \end{cases} \quad (5)$$

The pdf is denoted as $W(\alpha, \beta)$, where $\alpha, \beta > 0$. The algorithm for a Weibull distributed random number can be generated as:
Step 1. *Generate m from* EXP*(*1*).*
Step 2. *Return* $\beta m^{1/\alpha}$.

e) Normal Distribution

A random variable *x* has a Normal distribution when its pdf is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)2}{2\sigma^2}}, \quad -\infty < x < \infty \quad (6)$$

The pdf is denoted as $N(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance.
To generate a Normally distributed random number, the algorithm is as follows:
Step 1. *Generate* m_1 *and* m_2 *from U(*0, 1*).*
Step 2. $y = \sqrt{-2\ln(m_1)}\sin(2\pi m_2)$
Step 3. *Return* $(\mu + \sigma y)$.

3. Monte Carlo stochastic simulation for the chance constraints

Stochastic optimization problems are usually solved with the methodology of chance constrained programming. The problem here arises in obtaining the deterministic equivalent of the stochastic constraints so that it can be solved by any nonlinear programming (NLP) technique. The major challenge towards solving chance constrained optimization problems lies in the computation of the probability and its slopes of satisfying inequality constraints.

In this section we will overcome this problem by using the Monte Carlo simulation technique. Monte Carlo methods are a broad class of computational algorithms that rely on repetitive random sampling to obtain numerical results. They are much dominant when the problem has a probabilistic interpretation and basic techniques become difficult to be used for solving [9].

The chance constraints in (1), $Prob\left[\sum_{j=1}^{n} a_{ij}x_j \le b_i\right] \ge \alpha_i$, could be re-written as $Prob\,[g_i(x, r) \le 0] \ge \alpha_i$, $i = 1, \ldots, m$, and $r = (R_1, R_2, \ldots, R_t)$ is a continuous random vector where each R_i^t has a known probability distribution. The Monte Carlo simulation technique is used as follows to check the feasibility of the above chance constraints.

- Generate L independent random vectors from the probability distributions of the random

vector component of each constraint $r^{(z)} = \left(R_1^{(Z)}, R_2^{(Z)}, \dots, R_t^{(Z)}\right), \quad z = 1, \dots, L$ (7)

- Count the number of times that the constraint $g_i\left(x, r^{(z)}\right) \leq 0$ is satisfied and let it L_i'
- Then by definition of probability, the stochastic constraint $Prob\ [g_i(x,r) \leq 0] \geq \alpha_i$, will hold only if $L_i'/\ L > \alpha_i, i = 1, 2, \dots, m$

Therefore, the algorithm for the stochastic Monte Carlo simulation of chance constraints could be presented as:

Step 1. Initialize the counter, $L_i' = 0, \quad i = 1, 2, \dots, m$.

Step 2. Generate random numbers from the probability density function of the random vector component R_t.

Step 3. Estimate the value of constraint, $g_i(x, r)$. If $g_i(x, r)$ is ≤ 0, then increase the counter by one, i.e., $L_i' = L_i' + 1, i = 1, 2, \dots, m$.

Step 4. Repeat steps 2 and 3 L times.

Step 5. Find the value of $L_i'/\ L, i = 1, 2, \dots, m$.

Step 6. If $L_i'/_L > \alpha_i$, then the constraint i is satisfied at the pre-determined level of probability α_i.

4. Review of some swarm algorithms

a) Genetic Algorithms

The Genetic Algorithm (GA) mainly is a search algorithm that depends on the mechanics of the natural selection process. Its basic idea is to mimic the concept of the 'survival of the fittest' where it simulates the natural processes observed in the system in which the strong survives while the weak vanishes. GA is a population-based approach in which members of the population are ranked based on the quality of their solutions which is evaluated in terms of their fitness. In GA, specific genetic operators such as crossover, reproduction, and mutation are used to form a new population.

In each generation, the new chromosome (a member of the population) is formed from the fittest chromosomes of the previous population. GA generates an initial population of feasible solutions and merge them in a way to guide their search toward more promising zones of the search space.

Each of these feasible solutions is encoded as a chromosome, and is evaluated through a fitness function (objective function). The value of fitness function of a chromosome determines its capability to produce offspring. The high fitness value indicates the better solution for maximization and the low fitness value shows the better solution for minimization problems. A basic GA has five main components: a random number generator, a fitness evaluation function, a reproduction operation, a crossover operation, and a mutation operation.

b) Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an optimization technique that uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for global optimal solutions. PSO proved to be an efficient optimization algorithm that searches the entire problem space which is high-dimensional. It is a robust stochastic optimization technique based on the movement and intelligence of swarms. It uses the concept of social interaction to problem solving and does not use the gradient of the problem being optimized, that's why it does not require the optimization problem to be differential, as is required by classic optimization methods. The PSO algorithm starts by initializing the population first. Then the second step is calculating the fitness values of each particle, followed by updating individual and global best solutions. Finally, the velocity and the position of the particles are updated. The second to fourth steps are repeated until the termination condition is satisfied.

c) Differential Evolution

The Differential Evolution (DE) algorithm is a population-based algorithm that is somehow similar to GA as it uses crossover, mutation, and selection operators which are similar operators of GA. The main difference between DE and GA is in producing better solutions, where DE depends on mutation operation while GA depends on crossover operation.

It uses the mutation as its basic search mechanism and benefits from the advantage of the selection operator to direct the search towards promising areas in the search space.

5. Bat algorithm: background

BA is a recent optimization algorithm based on swarm intelligence, basically inspired from the behavior of microbats, in which they use an echolocation which is a kind of sonar. It is a very useful capability that enables them to notice their prey, turn over obstacles, and locate their paths in the dark. The echo plays an important role in the search process, where these bats produce a very high sound pulse and then wait to listen for the echo that returns back from the neighboring objects that they

hit. In a magical way, they have the ability to determine the distance and the location of their prey or target.

There are almost three essential basic rules for the echolocation characteristics of microbats, which are [10]:

a) The echolocation is mainly used by all bats to determine the distances and location of the prey. Not only determining the distances, but also to differentiate between their prey and any other background surroundings.
b) Randomly is the flying of the bats with velocity v_i and with frequency f_{min} at the position x_i, changing wavelength λ and loudness A_0 to search for their prey. The wavelength of their emitted sounds is automatically modified by them and also the pulse emission rate $r \in (0, 1)$ is adjusted, depending on the distance of their prey (target).
c) The loudness is assumed to be changeable from a high positive value A_0 to a low constant value A_{min} when finding its prey (although other researches could assume variations in many ways).

During the solution process, the position and velocity for each bat (*i*) in the search space should be well-defined and as such updated during subsequent iterations. The equations for calculating the new position and velocity at time step *t* are:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (8)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)f_i \quad (9)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (10)$$

where, β is in the range [0, 1] which is a random vector from a uniform distribution. x^* is the current global best solution, which is found after the comparison between solutions among all the *n* bats of the whole population is done. x_i is the location of the i^{th} bat in the solution space.

For carrying out the bat algorithm, it's widely assumed that, f_{min} = 0 and f_{max} = 100, depending on the dimension of the current problem. At the beginning, a frequency, that is chosen uniformly from [f_{min}, f_{max}], is randomly assigned to each bat [11]. Also, one of the basic roles in the process is the determination of the pulse emission rate, denoted by $r_i \in$ [0, 1]. This is done through the automatic zooming mechanism that the algorithm has. At each iteration, a random number is generated and compared to with r_i, and if it is greater than r_i, a better solution is chosen from among the current best solutions and a local search strategy, called random walk, is applied to generate a local new solution for each bat using Eq.(11). By this mechanism, it implements the search process and assuring both the global and local search, protecting the search process from stucking into local optima.

$$x_{new} = x_{old} + \varepsilon A^t \quad (11)$$

where ε is a random number from [-1, 1], and A^t is the average loudness of all bats at this current time *i*, $A^t = \bar{A}_i^t$.

As soon as a bat finds its target or prey, the loudness decreases and the pulse emission rates increases. As such, the loudness could be chosen as any value of suitability. A_0 = 1 and A_{min} = 0 could be used, for simplicity. For A_{min} = 0, indicates that a bat has just hit its prey and as a result stop producing any sound.

On the other hand, the pulse rate should be in the interval [0, 1] where 0 means no pulse and 1 means maximum rate of pulse emission. Therefore, at each iteration of the algorithm, the loudness A_i and the pulse rate emission r_i have to be updated as follows using Eqs. (12) and (13):

$$A_i^{t+1} = \alpha A_i^t \quad (12)$$

$$r_i^{t+1} = r_i^0[\,1 - \exp(-\gamma t)] \quad (13)$$

where α and γ are constants [12]. For any $0 < \alpha, \gamma < 1$: $A_i^t \to 0, \quad r_i^t \to r_i^0 \ as\ t \to \infty$.

For those proposals, $\alpha = \gamma = 0.9$ (used in most Yang's experiments for simplicity). However, at the beginning of the algorithm, the loudness A_i^0 and the pulse rate r_i^0 for each bat should be different and randomly chosen. They are updated only when new better solutions are found, which means that these bats are moving towards the optimal solution [13].

For a multi-objective BA, a weighted sum function could be used to combine all objectives into a single objective. The weights are randomly generated from a uniform distribution but it could be different to guarantee the diversity of the points on the Pareto front [14].

6. Stochastic multi-objective bat algorithm (SMOBA)

The basic rules of the BA are used to solve unconstrained optimization problems. However, for implementation point of view, nonlinear uncertain

constrains should be handled. In this section, a stochastic multi-objective BA is proposed in which the Monte Carlo simulation approach is used for constraint handling. The proposed algorithm is capable of handling any constraints where the random variables involved in the parameters follow an arbitrary continuous distribution with known probability density functions. The stochastic constraints are represented as chance constraints with some determined level of probability. The feasibility of the generated solutions is checked by applying the technique of the stochastic Monte Carlo simulation discussed in section 2.

The basic steps for the proposed stochastic multi-objective bat algorithm (shown in Algorithm 1) to solve chance constrained stochastic programming problems are as follows:

Step 1: The algorithm begins by initializing the values parameters of the bat algorithm, the minimum frequency f_{min}, maximum frequency f_{max}, pop. size n, the loudness constant α, pulse emission rate constant γ, the initial loudness A_0, the minimum loudness A_{min}, the initial pulse emission rate r^0, the maximum number of iterations, and the counter for Monte Carlo simulation L'_m.

Step 2: The initial population is randomly generated by finding the initial position x_0 and the initial velocity v_0 for each solution in the population and assigning its initial frequency f_0.

Step 3: Generate K weights (no. of objective functions) so that $\sum_{k=1}^{K} w_k = 1$.

Step 4: The initial population is evaluated by calculating the value of the objective function for each solution. The values of pulse emission rate r_i and the loudness A_i are initialized.

Step 5: New solutions are generated to form the new population by adjusting the position x_i, velocity v_i and frequency f_i for each solution using Eq. 2, 3, and 4.

Checking the feasibility of each constraint

Step 6: For each constraint, generate a random number from the probability density function of the stochastic random vector component.

Step 7: Estimate the value of the constraint, and check if it is satisfied, if so, increase the counter L'_m by one

Step 8: Repeat steps 6 and 7 L times.

Step 9: Find the value of L'_m/L. If it is greater than the pre-determined level of probability α_m, then the constraint is feasible.

Step 10: Evaluate the new population by calculating the value of the objective function for each solution, and select best solution from the population

Step 11: The local search method is applied using Eq. (11) to make fine tuning to reach best found solution.

Step 12: The new solution is generated, and hence the pulse emission rate is increased and the loudness is decrease using Eqs. (12) and (13).

Step 13: The new population is generated and ranked to select the best solution.

In the following, the pseudo code for the above SMOBA algorithm is shown:

Algorithm 1 Stochastic Multi-Objective Bat Algorithm (SMOBA)

Objective functions: $f_1(x), \dots, f_K(x), x = (x_1, \dots, x_d)^T$
Constraints: $Prob\,[g_m(x, r_m) \leq 0] \geq \alpha_m)$
Set initial values f_{min}, f_{max}, α, γ, A_0, A_{min}, r^0, n.
Initialize the bat population $x_i, i = (1, 2, \dots, n)$ and v_i
Define pulse frequency f_i at x_i
Initialize pulse rates r_i and the loudness A_i
Initialize $L'_m = 0,\ m = (1, 2, \dots, M)$
For j = 1 to J (points on Pareto fronts)
  Generate K weights $w_k \geq 0$ so that $\sum_{k=1}^{K} w_k = 1$
  Form a single objective $f = \sum_{k=1}^{K} w_k f_k$
  While (t < Max number of iterations)
    Get new solutions by fine-tuning frequency, updating velocities and positions according to Eq. 2, 3, 4.
    For h =1 to L
      Generate a random number according to the density function of the continuous random variables r_m
      Check the value of $g_m(x, r_m)$
      If ($g_m(x, r_m)$ is satisfied)
        Increase L'_m by one
      End If
    End

Find the value of L_i'/L, and if it is greater than α_m, then the constraint is feasible
Evaluate the new population by calculating the value of the objective function for each solution
If (rand > r_i)
Randomly, select a solution from among the best solutions
Generate a local solution around the found best solution using a Random walk
End If
If (rand < A_i & $f(x_i) < f(x^*)$)
Accept the new solutions,
and increase r_i & reduce A_i
End If
Evaluate the new population by calculating the value of the objective function for each solution
Sort the bats and find the current best x^*
End While
Record x^* as a non-dominated solution
End
Post process results

7. Numerical example

In this section, a test example is presented that was previously solved using basic DE, PSO and GA, proposed in details in [15], and the results are compared to that obtained using the proposed SMOBA. The test problem invloves random variables that follow different types of probability distributions.

$$\textbf{Max } z_1 = 5x_1 + 6x_2 + 3x_3$$
$$\textbf{Max } z_2 = 6x_1 + 3x_2 + 5x_3$$
$$\textbf{Max } z_3 = 2x_1 + 5x_2 + 8x_3$$
Subject to
$$P(3x_1 + 2x_2 + 2x_3 \le b_1) \ge 0.9,$$
$$P(2x_1 + 8x_2 + 5x_3 \le b_2) \ge 0.98$$
$$P(5x_1 + 3x_2 + 2x_3 \le b_3) \ge 0.95,$$
$$P(0.5x_1 + 0.5x_2 + 0.25x_3 \le b_4) \ge 0.9$$
$$P(8x_1 + 3x_2 + 4x_3 \le b_5) \ge 0.99$$
$$x_1, x_2, x_3 \ge 0 \qquad (14)$$

Where, b_1 follows Power function distribution with parameter $\lambda = 10$ and $a = 5$;
b_2 follows Pareto distribution with parameter $\lambda = 8$ and $a = 2$;
b_3 follows Beta distribution with parameter $\lambda = 15$ and $a = 10$;
b_4 follows Weibull distribution with parameter $\theta = \frac{1}{5}$ and $a = 10$;
b_5 follws Burr type XII distribution with parameter $\lambda = \frac{1}{10}$, $\theta = \frac{1}{15}$ and $a = \frac{1}{5}$.

Using the concept of weighting characterization of the objective functions, the deterministic model for the above problem as per Charles et al. [15] is:

$$\textbf{Max } z = w_1(5x_1 + 6x_2 + 3x_3) + w_2(6x_1 + 3x_2 + 5x_3) + w_3(2x_1 + 5x_2 + 8x_3)$$
Subject to
$$3x_1 + 2x_2 + 2x_3 \le 6.3096,$$
$$2x_1 + 8x_2 + 5x_3 \le 8.0812$$
$$5x_1 + 3x_2 + 2x_3 \le 4.7115,$$
$$0.5x_1 + 0.5x_2 + 0.25x_3 \le 0.9379$$
$$8x_1 + 3x_2 + 4x_3 \le 10.0321$$
$$w_1 + w_2 + w_3 = 1$$
$$w_1, w_2, w_3 \ge 0,\ x_1, x_2, x_3 \ge 0 \qquad (15)$$

This deterministic problem was solved using the basic DE, GA, and PSO and the results are shown in Table 1.

Using the proposed SMOBA algorithm we solved the problem directly without converting it to the deterministic equivalent, in its stochastic form. It was run with 1000 runs of 20 bats, and hence performed 20,000 function evaluations with population size n = 20 and $\alpha = \gamma = 0.9$ (as recommended from the literature) for the loudness and the pulse emission rates constants.
Comparing the results of DE, GA, and PSO with SMOBA in terms of best objective function value and optimal decision variables values, the results are shown in Table 1.

It could be seen from Table 1 that SMOBA gives a higher value for objective function as global optimum, i.e. better results than all three algorithms. It is noticed that the global optimum for the SMOBA is slightly higher than that for PSO which is a logical justification since both algorithms have similar solution mechanisms, but SMOBA has the

Table 1. Results of DE, GA, PSO and SMOBA

	DE	**GA**	**PSO**	**SMOBA**
z	9.48978	8.5089	12.9299	12.9311
z_1	6.18688	6.4834	4.84872	6.35469
z_2	9.48978	8.3125	8.0812	9.88835
z_3	12.5073	10.514	12.9299	13.53224
x_1	0.35214	0.3727	0	0.3012
x_2	2.12E-07	0.2319	0	0
x_3	1.47538	1.0761	1.61624	1.61623
St.Dev.	2.06789	NA	3.91715	0.00173
Avg. FE	14,691	30,720	20,573	11,430

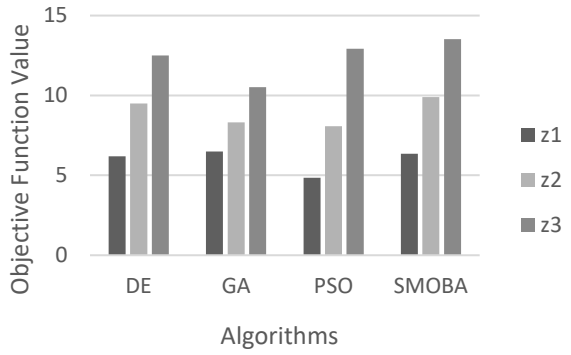

Figure. 1 Performance of DE, GA, PSO and SMOBA

added advantage of frequency tuning (a function of the BA). Nevertheless, as evident from the table, SMOBA results in a much lower standard deviation, which indicates a more efficient and robust solution when compared to the others. To show the convergence speed of the algorithms, the average number of functions evaluations (Avg. FE) was considered. In addition to efficiency and robustness, the percentage improved by the proposed algorithm in terms of avg. FE in comparison with DE, GA, and PSO is 77.8%, 37.2%, and 55.5% respectively.

All this is done by SMOBA while considering the stochastic problem in its original stochastic form without the need to convert it to its deterministic equivalent one as needed by the other algorithms.

Fig. 1 shows the performance of DE, GA, PSO, and SMOBA in terms of objective function values for the three objectives.

8. Conclusions and future work

In this study, the multi-objective constrained BA algorithm is modified to solve stochastic constraints optimization problems. In the proposed SMOBA, the feasibility of the chance constrains was checked using the Monte Carlo simulation approach. The main advantage of the proposed algorithm is that it does not require the deterministic equivalents of the chance constrains which are difficult to achieve in many situations. It is applicable for any stochastic random variables involved in the constraints. The numerical results showed that the proposed algorithm is effective in finding a robust and uniformly spread approximation of the Pareto optimal set of the multi-objective problem, when compared to the tested problem. Future directions of research are to accommodate stochastic objective functions rather than deterministic ones and incorporating more sophisticated approach to ensure dominance of solutions to further improve the convergence of the algorithm.

Compliance with Ethical Standards

Human and animal participation: This is a theoretical research therefore there is no human nor animal subjects is involved.

Disclosure of potential conflicts of interest: The authors declare that they have no conflict of interest.

Informed consent: Informed consent was obtained from all individual participants included in the study.

References

[1] A. Charnes and W. Cooper, "Chance-constrained programming", *Management Science*, Vol. 6, No. 1, pp.73-79, 1959.

[2] D. Kalyanmoy, "Multi objective optimization using evolutionary algorithms", *John Wiley and Sons*, pp. 124-124, 2001.

[3] C. Coello, G. Lamont, and D. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Vol. 5, Springer, New York, N.Y.2007.

[4] H. Mete and Z. Zabinsky, "Multiobjective interacting particle algorithm for global optimization", *INFORMS Journal on Computing*, Vol. 26, No. 3, pp. 500-513, 2014.

[5] G. Said, A. Mahmoud, and E. El-Horbaty, "A comparative study of meta-heuristic algorithms for solving quadratic assignment problem", *International Journal of Advanced Computer Science and Applications*, Vol. 5, No. 1, pp. 1-6, 2014.

[6] M. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comprehensive review of swarm optimization algorithms", *PLoS One*, Vol. 10, No. 5, 2015.

[7] X.S. Yang, "A new metaheuristic bat-inspired algorithm", *Nature Inspired Cooperative Strategies for Optimization*, Vol. 284, pp. 65-74, 2010.

[8] X.S. Yang and X. He, "Bat algorithm: literature review and applications", *International Journal of Bio-Inspired Computation*, Vol. 5, No. 3, pp. 141-149, 2013.

[9] R.K. Jana and M.P. Biswal, "Stochastic simulation-based genetic algorithm for chance constraint programming problems with continuous random variables", *International Journal of Computer Mathematics*, Vol.81, No. 9, pp.1069-1076, 2004.

[10] A.H. Gandomi, X.S. Yang, A.H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks", *Neural Computing and Applications*, Vol. 22, No. 6, pp.1239-1255, 2013.
[11] X.S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization", *Engineering Computations*, Vol. 29, No. 5, pp. 464-483, 2012.
[12] J. Senthilnath, S. Kulkarni, J.A. Benediktsson, and X.S. Yang, "A novel approach for multispectral satellite image classification based on the bat algorithm", *IEEE Geoscience and Remote Sensing Letters*, Vol. 13, No. 4, pp. 599-603, 2016.
[13] T. Jayabarathi, T. Raghunathan, and A.H. Gandomi, "The Bat Algorithm, Variants and Some Practical Engineering Applications: A Review", *Nature-Inspired Algorithms and Applied Optimization*, pp. 313-330, 2018.
[14] X.S. Yang, "Bat algorithm for multi-objective optimization", *International Journal of Bio-Inspired Computation*, Vol. 3, No. 5, pp. 267-274, 2011.
[15] V. Charles, S.I. Ansari, and M.M. Khalid, "Multi-objective stochastic linear programming with general form of distributions", *Int. J. Oper. Res. Optim.*, Vol. 2, No. 2, pp.261-278, 2011.