**INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING**
***Int. J. Optim. Civil Eng., 2015; 5(2):205-225***

# STRUCTURAL RELIABILITY ASSESSMENT UTILIZING FOUR METAHEURISTIC ALGORITHMS

A. Kaveh[*, †] and M. Ilchi Ghazaan
*Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran, P.O. Box 16846-13114, Iran*

## ABSTRACT

The failure probability of the structures is one of the challenging problems in structural engineering. To obtain the reliability index introduced by Hasofer and Lind, one needs to solve a nonlinear equality constrained optimization problem. In this study, four of the most recent metaheuristic algorithms are utilized for finding the design point and the failure probability of problems with continuous random variables. These algorithms consist of Improved Ray Optimization, Democratic Particle Swarm Optimization, Colliding Bodies Optimization, and Enhanced Colliding Bodies Optimization. The performance of these algorithms is tested on nineteen engineering optimization problems.

Received: 5 January 2015; Accepted: 10 March 2015

KEY WORDS: structural reliability; reliability index; improved ray optimization (IRO); democratic particle swarm optimization (DPSO); colliding bodies optimization (CBO); enhanced colliding bodies optimization (ECBO).

## 1. INTRODUCTION

The evaluation of the safety of structures has been one of the subjects of interest for engineers. The safety of a structure depends on the resistance of the structure and the actions on the structure that are functions of random variables. Due to the presence of inherent uncertainties in mechanical and geometrical properties of structural systems and external loads achievement of absolute safety is impossible [1-3].

[*]Corresponding author: Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran, P.O. Box 16846-13114, Iran
[†]E-mail address: alikaveh@iust.ac.ir (A. Kaveh)

The methods of probability analysis for the sake of structural reliability can generally be classified into three categories: Moment methods [4-6], Simulation methods [7-11] and Heuristic methods [12-17]. In this paper, structural reliability assessment is studied based on metaheuristic algorithms. In this category, the shortest distance of the limit state function from the origin of the standard normal coordinate system is considered as a fitness function and is obtained by heuristic methods.

This paper proposes a reliability model based on metaheuristic algorithms, which are population-based methods to find the optimum result. These algorithms consist of Improved Ray Optimization (IRO) [18], Democratic Particle Swarm Optimization (DPSO) [19], Colliding Bodies Optimization (CBO) [20] and Enhanced Colliding Bodies Optimization (ECBO) [21]. The IRO is the enhanced version of the ray optimization [22] which was conceptualized using the relationship between the angles of incidence and fraction based on Snell's law. Each agent is modeled as a ray of light that moves in the search space to find the global or near-global optimum solution. Particle Swarm Optimization (PSO) has become one of the most popular optimization techniques that simulates the social interaction behavior of birds flocking and fish schooling [23]. The main emphasis in DPSO is placed upon addressing the problem of premature convergence which is believed to be one of the PSO's main drawbacks. In CBO, each agent is modeled as a body with a specified mass and velocity. A collision occurs between pairs of objects and the new positions of the colliding bodies are updated based on the collision laws. ECBO is proposed to improve the exploration capability of the CBO and thus to remove the problem of premature convergence in some problems.

The rest of this paper is organized as follows. In Section 2, brief presentation of the concepts of structural reliability is provided. In Section 3, IRO, DPSO, CBO and ECBO methods are briefly presented. Nineteen benchmark functions of the literature are studied in Section 4, and conclusions are derived in Section 5.

## 2. FAILURE PROBABILITY ASSESSMENT

The structure failure probability, $P_f$, is given by the following formula:

$$p_f = p(G(X) \le 0) = \int_{G(X)\le 0} f(X)dX \tag{1}$$

where $X=[x_1, x_{2,} \ldots, x_n]^T$ represents a vector of stochastic variables of the reliability problem, the superscript $T$ denotes the transpose, and $f(X)$ represents a joint probability density function in $X$-space. The operation of each structure can be expressed by a function of basic random variables of that structure called limit state function ($G(X)=0$) so that $G(X)>0$ indicates a safe state and $G(X)<0$ indicates a failure state. Therefore the integral represents the volume of the joint probability density function in the failure domain. The evaluation of this multi-fold probability integral is a fundamental problem in structural reliability theory because direct calculation of this integral is very difficult, especially for real structures.

Hasofer and Lind [24] proposed the concept of design point which is the most probable failure point. Based on this definition, the reliability index ($\beta$) is the shortest distance between the limit state function and the center of the standard normal space (the distance between the origin and the design point).

The independent normal distribution stochastic variables $x$ can be standardized by

$$u_i = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

where $x_i$ is the $i$th component of $x_i$, $\mu_i$ and $\sigma_i$ are the mean value and the standard variance of $x_i$ , respectively.

Finding the Hasofer-Lind's reliability index can be treated as an optimization problem. In other words, the objective function is supposed to determine the coordinates of a specific point on failure surface which is closest to the origin in the U-space (independent standard normal distribution). The mathematical formulation of these problems can be expressed as follows:

$$\begin{aligned} &\text{Find} && \{U\} = [u_1, u_2, .., u_n] \\ &\text{to minimize} && \beta(U) = \sqrt{u^T u} \\ &\text{subjected to:} && \mathrm{G(U)} = 0 \end{aligned} \tag{3}$$

In order to handle the constraints, the penalty approach is employed. Thus, the objective function is redefined as:

$$\beta(U) = (1 + \varepsilon_1 . \upsilon)^{\varepsilon_2} \times \beta(U), \qquad \upsilon = \sum_{j=1}^{nc} | G(u) | \tag{4}$$

where υ denotes the sum of the violations of the constraints and $nc$ is the number of the constraints. The constants $\varepsilon_1$ and $\varepsilon_2$ are selected considering the exploration and the exploitation rates of the search space. In this study, υ is considered closed to 0 if its absolute value is lower than $10^{-4}$ [13].

## 3. OPTIMIZATION ALGORITHMS

### *3.1 Improved Ray Optimization (IRO)*

Ray Optimization (RO), which is based on the transition of ray from one medium to another from physics, was introduced by Kaveh and Khayatazad [22]. As the light passes through a surface of two certain transparent materials, its path is changed slightly. This phenomenon was formulated by the Snell's law. In RO by utilizing Snell's refraction law and a number of random terms, each agent moves in the search space to find the global or near-global optimum solution. Kaveh et al. [18] developed Improved Ray Optimization (IRO) and it is proved to be

competent in structural optimization problems. The formulation of generating solution vectors and returning violated agents to feasible search space is changed in IRO. This algorithm consists of the following steps:

**Step 1:** Initialization

The initial positions of the agents are determined randomly in the search space and the objective function is evaluated for each agent. A memory which saves all the best positions as local best memory (***LBM***) is considered, and the position of the best agent is saved as the global best (***GB***).

**Step 2:** Initial velocity

The initial movement vectors of agents are stated as:

$$\boldsymbol{V}_{ij} = -1 + 2.rand \qquad j=1,2,\ldots,n \tag{5}$$

where $\boldsymbol{V}_{ij}$ is the initial movement value of the $j$th variable for the $i$th agent. New positions are obtained by adding the position of each agent with its movement vector.

**Step 3:** Regenerating

If any component of each agent violates a boundary, it must be regenerated by the following formula:

$$X_{ij}^{k+1} = X_{ij}^{k} + 0.9(Int_{ij} - X_{ij}^{k}) \tag{6}$$

where $X_{ij}^{k+1}$ and $X_{ij}^{k}$ are the refined component and component of the $j$th variable for the $i$th agent in $(k+1)$th and $k$th iteration, respectively. $Int_{ij}$ is the intersection point (in case an agent violates a boundary, it intersects the boundary at a specified point, because of having definite movement vector). After regenerating all violated components, the goal function is evaluated for each agent and the ***LBM*** and ***GB*** are updated.

**Step 4:** Evaluating the Origin

Consider the origin as a point which each agent wants to move toward it and specified by

$$\boldsymbol{O}_i^K = \frac{(ite+k).\boldsymbol{GB} + (ite-k).\boldsymbol{LB}}{2.ite} \tag{7}$$

where $\boldsymbol{O}_i^k$ is the origin of the $i$th agent for the $k$th iteration, and $ite$ is the total number of iterations for the optimization process. For each agent, ***LB*** is a solution vector which is selected randomly from local best memory (***LBM***). Target vector is defined as:

$$\boldsymbol{Tv_i} = O_i - X_i \tag{8}$$

where $\boldsymbol{Tv_i}$ and $X_i$ are target vector and current position of the $i$th agent, respectively.

**Step 5:** Evaluating the direction of movement vector

Direction of the new movement vector is defined by

$$\boldsymbol{V_i^{k+1}} = \alpha.Tv_i^{k} + \beta.V_i^{k} \tag{9}$$

$$\alpha = 1 + (\frac{k}{ite}) \tag{10}$$

$$\beta = 1 - 0.5(\frac{k}{ite}) \tag{11}$$

where $\boldsymbol{V}_i^{k+1}$and $\boldsymbol{V}_i^{k}$ are movement vectors for the *i*th agent in *(k+1)*th iteration and *k*th iteration, respectively. Finally all the $\boldsymbol{V}_i^{k+1}$ vectors should be normalized.

**Step 6:** Evaluating the magnitude of movement vector

Possibility like ***stoch*** is considered and the magnitude of movement vectors are defined as:

a. with probability like ***stoch***,

$$\boldsymbol{V}_{ij}^{k+1} = -1 + 2.rand \tag{12}$$

$$\boldsymbol{V}_i^{K+1} = \frac{\boldsymbol{V}_i^{k+1}}{norm(\boldsymbol{V}_i^{k+1})}.\frac{a}{d}.rand \tag{13}$$

$$a = \sqrt{\sum_{i=1}^{n}(\boldsymbol{X}_{j,\max} - \boldsymbol{X}_{j,\min})^2} \tag{14}$$

where *d* is a number that divides *a* into smaller length for effective search. $\boldsymbol{X}_{j,max}$ and $\boldsymbol{X}_{j,min}$ are the upper and lower bounds for the *j*th design variable, respectively. In this study *d* is set to 20.

b. with probability like ***(1*–*stoch)***,

If $norm(O_i^k - X_i^k)=0$, then

$$\boldsymbol{V}_i^{k+1} = \frac{\boldsymbol{V}_i^{k}}{norm(\boldsymbol{V}_i^{k})}.rand.0.001 \tag{15}$$

Otherwise,

$$\boldsymbol{V}_i^{k+1} = \boldsymbol{V}_i^{k+1}.norm(\boldsymbol{X}_i^{k} - \boldsymbol{O}_i^{k}) \tag{16}$$

**Step 7:** Terminating Criterion Control

The optimization process is terminated after a fixed number of iterations. If it is not fulfilled, each movement vector is added with its current position vector and the process of optimization is continued from Step 3.

### *3.2 Democratic Particle Swarm Optimization (DPSO)*

Particle Swarm Optimization (PSO) initially developed by Kennedy and Eberhart [25] is a metaheuristic algorithm which mimics the social behavior of certain species of animals like birds flocking and fishes schooling. PSO is one of the most widely used population-based technique having some advantages such as few parameters to adjust, easiness of implementation, global search capability in some problems and, in general, fast convergence.

However, despite having the above-mentioned benefits, the standard PSO is infamous of premature convergence. Democratic Particle Swarm Optimization (DPSO) is proposed by Kaveh and Zolghadr [19] to improve the exploration capabilities of the PSO and thus to address the problem of premature convergence. Each particle only leads by its historically best position and the historically best position of the entire swarm in standard PSO but in the DPSO all eligible particles have the right to be involved in decision making in order to overcome the problem of premature convergence. The steps of the DPSO are outlined in the following.

**Step 1:** Initialization

The positions of all particles are randomly set within predefined ranges and their associated velocities are set to 0. The objective function is evaluated for each particle and $\boldsymbol{xlbest_i}$ (historically best position of *i*th particle) and ***xgbest*** (historically best position of the entire swarm) are stored.

**Step 2:** Velocity and position updating

The velocity and position update rules are given by:

$$v_{i,j}^{k+1} = \chi\left(\omega v_{i,j}^{k} + \mathrm{c}_1.\mathrm{r}_1.(\mathrm{xlbest}_{\mathrm{i,j}}^{\mathrm{k}} - \mathrm{x}_{\mathrm{i,j}}^{\mathrm{k}}) + \mathrm{c}_2.\mathrm{r}_2.(\mathrm{xgbest}_{\mathrm{j}}^{\mathrm{k}} - \mathrm{x}_{\mathrm{i,j}}^{\mathrm{k}}) + \mathrm{c}_3.\mathrm{r}_3.\mathrm{d}_{\mathrm{i,j}}^{\mathrm{k}}\right) \tag{17}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{18}$$

where $v_{i,j}^{\ k}$ is the velocity or the amount of change of the design variable *j* of particle *i* in *k*th iteration, $x_{i,j}^{\ k}$ is the current value of the *j*th design variable of the *i*th particle. $r_1$, $r_2$ and $r_3$ are three random numbers uniformly distributed in the range (1, 0), $c_1$, $c_2$ and $c_3$ are three constant parameters. $\chi$ parameter is used to avoid divergence behavior. In this study, $c_1$, $c_2$, $c_3$ and $\chi$ are set to 2, 2, 4 and 0.5, respectively.

$\omega$ is the inertia weight for the previous iteration's velocity and it can be set in order to control the exploration of the algorithm. This parameter is defined as [26]:

$$\omega = 0.4[1 + \min(\mathrm{cov}, 0.6)] \tag{19}$$

where *cov* is the coefficient of variation of the swarm's objective function.

In Eq. (17), $d_{i,j}^{\ k}$ is the *j*th variable of the democratic vector *i*, $D_i$, in the *k*th iteration. The vector *D* represents the democratic effect of the other particles of the swarm on the movement of the *i*th particle is evaluated by:

$$D_i = \sum_{k=1}^{n} Q_{ij}(x_k - x_i) \tag{20}$$

$$Q_{ik} = \frac{E_{ik}\dfrac{obj_{best}}{obj(k)}}{\displaystyle\sum_{j=1}^{n} E_{ij}\frac{obj_{best}}{obj(j)}} \tag{21}$$

$$E_{ik} = \begin{cases} 1 & \dfrac{obj(k) - obj(i)}{obj_{worst} - obj_{best}} > rand \vee obj(k) < obj(i) \\ 0 & else \end{cases} \tag{22}$$

Where $n$ is the number of particles and the $\vee$ symbol stands for union. $obj_{worst}$ and $obj_{best}$ are the values of the objective function for the worst and the best particles in the current iteration, respectively.

**Step 3:** Updating ***xlbest*** and ***xgbest***

The objective function is evaluated for each particle and ***xlbest*** and ***xgbest*** are updated.

**Step 4:** Terminating criterion controlling

Repeat the optimization process until a fixed number of iterations is completed. Otherwise, go to Step 2 for a new round of iteration.

### *3.3 Colliding Bodies Optimization (CBO) and its Enhanced version (ECBO)*

#### *3.3.1 CBO*

Colliding Bodies Optimization (CBO) is a population-based meta-heuristic optimization algorithm introduced by Kaveh and Mahdavi [20], which is based on the governing laws of one dimensional collision between two bodies from the physics. One object collides with other object and they move toward minimum energy level. The CBO is simple in concept, shows fast-converging behavior and depends on no internal parameters. The following steps are developed to introduce the details of this method.

**Step 1:** Initialization

The initial positions of all Colliding Bodies (CBs) are determined randomly in search space.

**Step 2:** Defining mass

Each CB has a specified mass defined as:

$$m_k = \frac{\dfrac{1}{fit(k)}}{\dfrac{1}{\sum_{i=1}^{n} \dfrac{1}{fit(i)}}}, \qquad k = 1,2,...,n \tag{23}$$

Where *fit(i)* represents the objective function value of the *i*th CB and *n* is the number of colliding bodies. CBs are sorted according to their objective function values in an increasing order. To select the pairs of CBs for collision, they are divided into two equal groups: (i) stationary group, (ii) moving group.

**Step 3:** Criteria before the collision

The velocity of stationary bodies before collision is zero and moving objects move toward stationary objects:

$$v_i = 0, \qquad i = 1,2,...,\frac{n}{2} \tag{24}$$

$$v_i = x_{i-\frac{n}{2}} - x_i, \qquad i = \frac{n}{2}+1, \frac{n}{2}+2, \ldots, n \tag{25}$$

**Step 4:** Criteria after the collision

The velocities of stationary and moving bodies after the collision are evaluated by:

$$v_i^{'} = \frac{(m_{i+\frac{n}{2}} + \varepsilon m_{i+\frac{n}{2}})v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}} \qquad i = 1,2,\ldots,\frac{n}{2} \tag{26}$$

$$v_i^{'} = \frac{(m_i - \varepsilon m_{i-\frac{n}{2}})v_i}{m_i + m_{i-\frac{n}{2}}} \qquad i = \frac{n}{2}+1, \frac{n}{2}+2, \ldots, n \tag{27}$$

$\varepsilon$ is the coefficient of restitution (COR) that decreases linearly from unit to zero. Thus, it is stated as

$$\varepsilon = 1 - \frac{iter}{iter_{\max}} \tag{28}$$

where *iter* is the current iteration number and $iter_{max}$ is the total number of iteration for optimization process.

**Step 5:** Updating CBs

The new position of each stationary CB is:

$$x_i^{new} = x_i + rand \circ v_i^{'}, \qquad i = 1,2,\ldots,\frac{n}{2} \tag{29}$$

where $x_i^{new}$, $x_i$ and $v'_i$ are the new position, previous position and the velocity after the collision of the *i*th CB, respectively. *rand* is a random vector uniformly distributed in the range of [-1,1] and the sign ‘‘∘’’ denotes an element-by-element multiplication. The new position of each moving CB is calculated by:

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \circ v_i^{'}, \qquad i = \frac{n}{2}+1, \frac{n}{2}+2, \ldots, n \tag{30}$$

**Step 6:** Terminal condition check

The optimization process is terminated after a fixed number of iterations. If this criterion is not satisfied go to Step 2 for a new round of iteration.

### *3.3.2 ECBO*

Enhanced Colliding Bodies Optimization (ECBO) was introduced by Kaveh and Ilchi

Ghazaan [21] in order to improve the performance of CBO yet preserving some above mentioned strength points of the standard formulation. Colliding Memory (CM) is considered to save some historically best CB vectors and their related mass and objective function values to improve the performance of the CBO and reduce the computational cost. In each iteration, the solution vectors saved in CM are added to the population, and the same numbers of current worst CBs are deleted. To prevent premature convergence, a parameter like ***Pro*** within (0, 1) is introduced and it is specified whether a component of each CB must be changed or not. For each colliding body ***Pro*** is compared with $rn_i$ (i=1,2,…,n) which is a random number uniformly distributed within (0, 1). If $rn_i <$ ***pro***, one dimension of the *i*th CB is selected randomly and its value is regenerated as follows:

$$x_{ij} = x_{j,\min} + random.(x_{j,\max} - x_{j,\min}) \tag{31}$$

where $x_{ij}$ is the *j*th variable of the *i*th CB. $x_{j,min}$ and $x_{j,max}$ respectively, are the lower and upper bounds of the *j*th variable. In order to protect the structures of CBs, only one dimension is changed. In this paper, the value of ***Pro*** set to 0.35 and the size of the CM is taken as *n*/10 (*n* is the number of colliding bodies).

## 4. NUMERICAL EXAMPLES

In this section the IRO, DPSO, CBO and ECBO algorithms are applied to nineteen widely used test problems in literature, representing a broad range of possible limit states that can occur in practice. The final results are compared to the solutions of some other methods to demonstrate the validity and effectiveness of these algorithms.

For all examples a population of 20 agents is utilized. In order to investigate the effect of the initial solution on the final result and because of the stochastic nature of the meta-heuristic algorithms, each example is independently solved for 20 times with random initial designs. Afterwards the best results are reported for performance evaluation of proposed algorithms. In all the algorithms, the predefined maximum iteration is considered as 200. The total number of limit state function evaluations is the same for all of the algorithms (population size × maximum number of iterations).

The constants $\varepsilon_1$ of penalty function is set to unity for all examples except $G_5$ from Table 1, $G_3$ from Table 5 and $G_4$ from Table 9. For these functions $\varepsilon_1$ is set to 2, 3 and 3, respectively. In all examples, in the first steps of the search process, $\varepsilon_2$ is set to 1.5 and ultimately increased linearly to 3.

Table 1: Limit state function (the components are independent standard normal distribution)

| |
|---|
| $G_1(x) = 5 - 0.5(x_1 - 0.1)^2 - x_2$ |
| $G_2(x) = \exp(0.4(x_1 + 2) + 6.2) - \exp(0.3x_2 + 5) - 200$ |
| $G_3(x) = \exp(0.2x_1 + 1.4) - x_2$ |
| $G_4(x) = 2 + \sum_{i=1}^{9} x_i^2 - x_{10}$ |

| |
|---|
| $G_5(x) = 3 - x_2 + (4x_1)^4$ |
| $G_6(x) = 2 - x_2 - 0.1x_1^2 + 0.06x_1^3$ |
| $G_7(x) = 0.1(x_1 - x_2)^2 - \frac{(x_1 + x_2)}{\sqrt{2}} + 2.5$ |
| $G_8(x) = -0.5(x_1 - x_2)^2 - \frac{(x_1 + x_2)}{\sqrt{2}} + 3$ |

Eight benchmark limit state functions with independent standard normal random variables are summarized in Table 1. The optimal reliability index and corresponding probability of failure obtained by crude Monte-Carlo [27], PSO [13], CSS [17] and proposed algorithms are shown in Table 2. As can be observed, the results achieved by different methods are approximately identical. The optimal design points obtained by proposed methods are given in Table 3. Statistical information is presented in Table 4. Convergence history diagrams of average optimal reliability index for the first limit state function from Table 1 are depicted in Fig. 1.

Table 2: Comparison of failure probability and reliability index obtained by different methods

| | | Grooteman [27] | Elegbede [13] | Kaveh et al. [17] | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|---|---|---|
| $G_1$ | $P_f$ | - | 0.00183 | 0.00183 | 0.001831 | 0.001832 | 0.001832 | 0.001832 |
| | β | - | 2.9056 | 2.906 | 2.9058 | 2.9056 | 2.9057 | 2.9056 |
| $G_2$ | $P_f$ | - | 0.00337 | 0.003365 | 0.003334 | 0.003361 | 0.003354 | 0.003314 |
| | β | - | 2.7099 | 2.71 | 2.7129 | 2.7103 | 2.7109 | 2.7149 |
| $G_3$ | $P_f$ | - | 0.000405 | 0.000404 | 0.000404 | 0.000404 | 0.000404 | 0.000404 |
| | β | - | 3.4971 | 3.35 | 3.3496 | 3.3496 | 3.3496 | 3.3496 |
| $G_4$ | $P_f$ | 0.00534 | - | 0.0227498 | 0.022464 | 0.022746 | 0.022752 | 0.022747 |
| | B | 2.103 | - | 2 | 2.005333 | 2.0000 | 1.9999 | 2.0000 |
| $G_5$ | $P_f$ | 0.00018 | - | 0.0013498 | 0.001350 | 0.001350 | 0.001350 | 0.001350 |
| | β | 2.925 | - | 3 | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| $G_6$ | $P_f$ | 0.0347 | - | 0.022750 | 0.022756 | 0.022756 | 0.022756 | 0.022755 |
| | β | 1.996 | - | 2 | 1.9999 | 1.9999 | 1.9999 | 1.9999 |
| $G_7$ | $P_f$ | 0.00416 | - | - | 0.006211 | 0.006210 | 0.00621 | 0.006211 |
| | β | 2.481 | - | - | 2.4999 | 2.4999 | 2.4999 | 2.4999 |
| $G_8$ | $P_f$ | 0.105 | - | - | 0.04863 | 0.048629 | 0.048629 | 0.048628 |
| | β | 1.625 | - | - | 1.6582 | 1.6582 | 1.6582 | 1.6583 |

Table 3: Design points obtained by algorithms

| | Design point | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| $G_1$ | $x_1$ | -2.75145 | -2.74074 | -2.74499 | -2.74014 |
| | $x_2$ | 0.93450 | 0.96498 | 0.95296 | 0.96677 |
| $G_2$ | $x_1$ | -2.5872 | -2.52197 | -2.56748 | -2.47706 |
| | $x_2$ | 0.81629 | 0.99268 | 0.87017 | 1.11142 |
| $G_3$ | $x_1$ | -1.67975 | -1.68462 | -1.66811 | -1.68013 |
| | $x_2$ | 2.89798 | 2.89521 | 2.9047 | 2.89777 |
| $G_4$ | $x_1$ | 0.04490 | -0.01055 | 0.00574 | -0.00449 |

| | | | | | |
|---|---|---|---|---|---|
| | $x_2$ | -0.03938 | 0.00725 | 0.00347 | -0.00461 |
| | $x_3$ | 0.04674 | -0.0095 | 0.00452 | -0.00867 |
| | $x_4$ | 0.06378 | 0.00037 | -0.00460 | -0.01044 |
| | $x_5$ | -0.0192 | 0.01194 | -0.0019 | -0.01201 |
| | $x_6$ | 0.01470 | -0.0067 | -0.01274 | 0.00206 |
| | $x_7$ | -0.07475 | -0.00161 | -0.00243 | 0.00320 |
| | $x_8$ | -0.06726 | 0.00509 | 0.00127 | 0.00357 |
| | $x_9$ | -0.00213 | -0.01465 | 0.00012 | 0.01327 |
| | $x_{10}$ | 2.00020 | 1.99991 | 1.99990 | 1.99991 |
| $G_5$ | $x_1$ | -0.00325 | -0.00042 | -2.91e-008 | -0.00625 |
| | $x_2$ | 2.99990 | 2.9999 | 2.9999 | 2.99991 |
| $G_6$ | $x_1$ | -0.00174 | 0.00102 | -2.35e-008 | 0.00132 |
| | $x_2$ | 1.99989 | 1.9998 | 1.9999 | 1.99990 |
| $G_7$ | $x_1$ | 1.76760 | 1.76209 | 1.75802 | 1.76847 |
| | $x_2$ | 1.76778 | 1.7733 | 1.77742 | 1.7669 |
| $G_8$ | $x_1$ | 1.4699 | 1.46880 | 1.47398 | -0.76116 |
| | $x_2$ | -0.76747 | -0.7697 | -0.75980 | 1.47329 |

Table 4: Statistical optimization results obtained by algorithms

| | | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| | Best β | 2.9058 | 2.9056 | 2.9057 | 2.9056 |
| | Worst β | 3.1025 | 3.1430 | 3.5352 | 3.0951 |
| | Avg β | 2.9946 | 2.9881 | 3.0307 | 2.9274 |
| $G_1$ | Std β | 0.0957 | 0.0990 | 0.1768 | 0.0573 |
| | Iteration | 100 | 52 | 38 | 190 |
| | Avg Iteration | 105.8 | 121.25 | 39.75 | 118.85 |
| | Std Iteration | 27.2968 | 36.5180 | 14.0295 | 50.7805 |
| | Best β | 2.7129 | 2.7103 | 2.7109 | 2.7149 |
| | Worst β | 6.3233 | 5.4257 | 3.9674 | 5.6122 |
| | Avg β | 3.5772 | 3.7728 | 3.0350 | 3.2572 |
| $G_2$ | Std β | 1.1430 | 0.8326 | 0.3724 | 0.6505 |
| | Iteration | 162 | 88 | 47 | 118 |
| | Avg Iteration | 132.35 | 127.7 | 51.55 | 118.3 |
| | Std Iteration | 24.2036 | 40.9712 | 12.2193 | 37.9502 |
| | Best β | 3.3496 | 3.3496 | 3.3496 | 3.3496 |
| | Worst β | 3.3519 | 3.3653 | 3.5560 | 3.3620 |
| | Avg β | 3.3500 | 3.3522 | 3.3670 | 3.3513 |
| $G_3$ | Std β | 0.0005 | 0.0041 | 0.0479 | 0.0029 |
| | Iteration | 90 | 109 | 30 | 54 |
| | Avg Iteration | 82.5 | 108.75 | 27.4 | 99.25 |
| | Std Iteration | 19.1874 | 53.0698 | 10.049 | 52.3529 |
| | Best β | 2.0053 | 2.0000 | 1.9999 | 2.0000 |
| $G_4$ | Worst β | 2.1244 | 2.0045 | 6.9571 | 6.2084 |
| | Avg β | 2.0293 | 2.0009 | 3.0399 | 2.2107 |

| | | | | | |
|---|---|---|---|---|---|
| | Std β | 0.0271 | 0.0010 | 1.5502 | 0.9409 |
| | Iteration | 185 | 111 | 128 | 121 |
| | Avg Iteration | 174.15 | 137.55 | 157.95 | 132 |
| | Std Iteration | 20.3812 | 16.8693 | 30.8075 | 25.0620 |
| | Best β | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| | Worst β | 3.0005 | 2.9999 | 3.0053 | 3.0237 |
| | Avg β | 2.9999 | 2.9999 | 3.0001 | 3.0020 |
| $G_5$ | Std β | 0.0001 | 5.17e-007 | 0.0012 | 0.0057 |
| | Iteration | 66 | 36 | 20 | 149 |
| | Avg Iteration | 71.3 | 40.6 | 27.15 | 127.15 |
| | Std Iteration | 13.4363 | 7.8028 | 7.6796 | 47.8476 |
| | Best β | 1.9999 | 1.9999 | 1.9999 | 1.9999 |
| | Worst β | 2.0059 | 2.0000 | 2.0004 | 2.1130 |
| | Avg β | 2.0006 | 1.9999 | 1.9999 | 2.0056 |
| $G_6$ | Std β | 0.0016 | 3.12e-005 | 0.000126 | 0.0252 |
| | Iteration | 47 | 63 | 26 | 43 |
| | Avg Iteration | 93.2 | 60.8 | 27.7 | 68.65 |
| | Std Iteration | 34.1984 | 22.4232 | 8.4859 | 37.4858 |
| | Best β | 2.4999 | 2.4999 | 2.4999 | 2.499 |
| | Worst β | 2.5013 | 2.5012 | 2.5256 | 2.5149 |
| | Avg β | 2.5000 | 2.5001 | 2.5031 | 2.5009 |
| $G_7$ | Std β | 0.0003 | 0.0002 | 0.0057 | 0.0033 |
| | Iteration | 67 | 32 | 39 | 19 |
| | Avg Iteration | 78.05 | 64.25 | 27.05 | 55.1 |
| | Std Iteration | 28.9945 | 31.6525 | 7.5077 | 61.3170 |
| | Best β | 1.6582 | 1.6582 | 1.6582 | 1.6583 |
| | Worst β | 1.6679 | 1.6617 | 1.7081 | 1.6609 |
| | Avg β | 1.6595 | 1.6589 | 1.6663 | 1.6588 |
| $G_8$ | Std β | 0.0022 | 0.0010 | 0.0122 | 0.0005 |
| | Iteration | 98 | 42 | 20 | 73 |
| | Avg Iteration | 96.25 | 84.8 | 36.05 | 78.35 |
| | Std Iteration | 16.9049 | 41.6092 | 9.5832 | 47.2966 |

Avg β = average optimized β; Std β = standard deviation on optimized β; Iteration = required number of iteration for the best run; Avg Iteration = average required number of iteration; Std Iteration = standard deviation on required number of iteration.

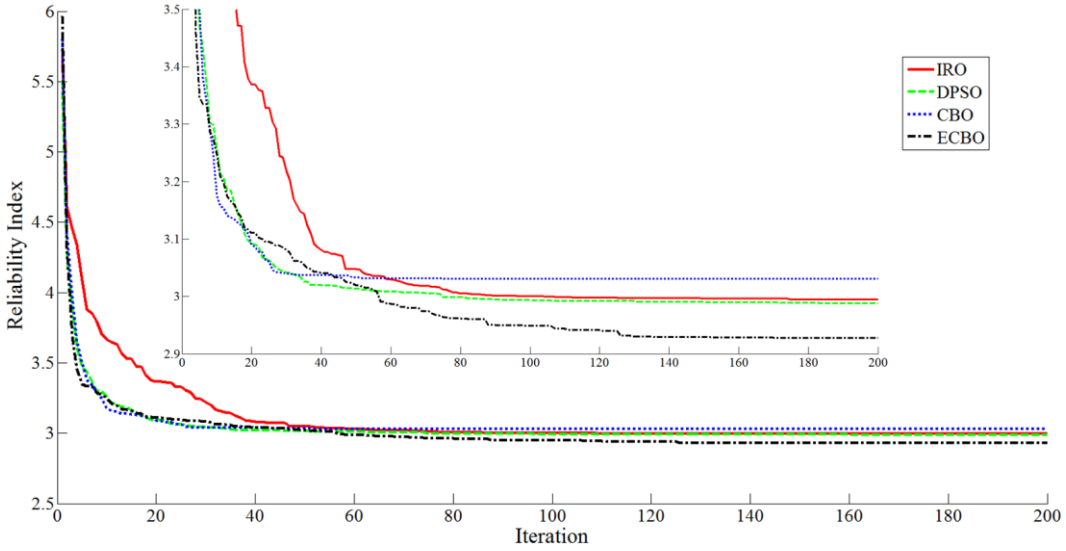


Figure 1. Comparison of average values of reliability index recorded in the optimization history for the different algorithms ($G_1$ from Table 1)

Table 5 provides five limit state functions with independent normal random variables. Column 2 gives the mean and standard deviation values of the variables. Table 6 lists the optimal reliability index and corresponding probability of failure obtained by different methods. It can be seen that the optimal β values achieved by various techniques are approximately the same. The optimal design points in normal and standard normal spaces are shown in Table 7. In Table 8, the statistical results of independent runs are summarized. The average convergence curves of proposed algorithms for the first limit state function from Table 5 are compared in Fig. 2.

Table 5: Limit state function (the components are independent normal distribution)

| Limit state function | Stochastic variables |
|---|---|
| $G_1(x) = 18.46154 - 7.476923\times10^{10}\,\frac{x_1}{x_2}$ | $x_1$: $N$(0.001, 0.0002)<br>$x_2$: $N$(250, 37.5) |
| $G_2(x) = x_1 - \frac{x_2}{x_3}$ | $x_1$: $N$(600, 3)<br>$x_2$: $N$(1000, 33)<br>$x_3$: $N$(2, 0.1) |
| $G_3(x) = 1 - \frac{\sqrt{3(1-\mu^2)}}{\pi x_1 x_2^2 \cos^2 x_3}\left(\frac{x_6}{2\gamma} + \frac{x_5}{\eta x_4}\right)$ | $x_1$: $N$(70000, 3500)<br>$x_2$: $N$(0.0025, 0.000125)<br>$x_3$: $N$(0.524, 0.01048)<br>$x_4$: $N$(0.9, 0.0225)<br>$x_5$: $N$(80000, 6400)<br>$x_6$: $N$(70000, 5600) |
| $G_4(x) = x_1 x_2 - 146.14$ | $x_1$: $N$(78064.4, 11709.7)<br>$x_2$: $N$(0.0104, 0.00156) |
| $G_5(x) = 2.5 - 0.2357\,(x_1 - x_2) + 0.00463\,(x_1 + x_2 - 20)^4$ | $x_1$: $N$(10, 3)<br>$x_2$: $N$(10, 3) |

Table 6: Comparison of failure probability and reliability index obtained by different methods

| | | Grooteman [27] | Elegbede [13] | Kaveh et al. [17] | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|---|---|---|
| $G_1$ | $P_f$ | - | 0.009879 | 0.009878 | 0.009879 | 0.009878 | 0.009878 | 0.009875 |
| | $\beta$ | - | 2.3309 | 2.33092 | 2.3309 | 2.3309 | 2.3309 | 2.3310 |
| $G_2$ | $P_f$ | - | 0.01161 | 0.01161 | 0.011394 | 0.009608 | 0.0116 | 0.008621 |
| | $\beta$ | - | 2.2697 | 2.26969 | 2.2769 | 2.3413 | 2.2701 | 2.3814 |
| $G_3$ | $P_f$ | - | 5.232e-7 | 8.07e-7 | 8.07e-007 | 8.05e-007 | 7.95e-007 | 7.97e-007 |
| | $\beta$ | - | 4.883 | 4.79654 | 4.796378 | 4.79704 | 4.7995 | 4.798937 |
| $G_4$ | $P_f$ | 1.46e-07 | - | 4.82e-8 | 4.72e-008 | 4.35e-008 | 4.53e-008 | 4.67e-008 |
| | $\beta$ | 5.443 | - | 5.33329 | 5.33702 | 5.3516 | 5.3442 | 5.3387 |
| $G_5$ | $P_f$ | 0.00286 | - | - | 0.006211 | 0.006210 | 0.006211 | 0.006210 |
| | $\beta$ | 2.431 | - | - | 2.4999 | 2.4999 | 2.4999 | 2.4999 |

Table 7: Design points obtained by algorithms

| | Design point | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| $G_1$ | $x_1$ | 0.00111 | 0.00111 | 0.00111 | 0.00111 |
| | $x_2$ | 165.48 | 165.48 | 165.53 | 165.24 |
| | $u_1$ | 0.59394 | 0.5944 | 0.59895 | 0.57012 |
| | $u_2$ | -2.25397 | -2.2538 | -2.2526 | -2.2602 |
| $G_2$ | $x_1$ | 559.71 | 565.22 | 555.71 | 540.56 |
| | $x_2$ | 1032.3 | 1044.8 | 1030.3 | 1012.9 |
| | $x_3$ | 1.8444 | 1.8484 | 1.854 | 1.8738 |
| | $u_1$ | -1.34300 | -1.15916 | -1.47619 | -1.9812 |
| | $u_2$ | 0.98002 | 1.35638 | 0.91719 | 0.39036 |
| | $u_3$ | -1.55578 | -1.5160 | -1.46048 | -1.26244 |
| $G_3$ | $x_1$ | 6.38e+010 | 6.39e+010 | 6.42e+010 | 6.39e+010 |
| | $x_2$ | 0.00199 | 0.00199 | 0.00199 | 0.00199 |
| | $x_3$ | 0.52789 | 0.52788 | 0.52867 | 0.52888 |
| | $x_4$ | 0.88833 | 0.8858 | 0.8862 | 0.88624 |
| | $x_5$ | 90364 | 90136 | 90752 | 90883 |
| | $x_6$ | 74181 | 74424 | 74333 | 73990 |
| | $u_1$ | -1.75342 | -1.73945 | -1.65693 | -1.71645 |
| | $u_2$ | -4.04282 | -4.0395 | -4.03658 | -4.01175 |
| | $u_3$ | 0.37142 | 0.37056 | 0.44558 | 0.46518 |
| | $u_4$ | -0.51851 | -0.63105 | -0.61343 | -0.61147 |
| | $u_5$ | 1.61930 | 1.58382 | 1.67996 | 1.70040 |
| | $u_6$ | 0.74657 | 0.79006 | 0.77378 | 0.71258 |
| $G_4$ | $x_1$ | 19516 | 51622 | 20470 | 55456 |
| | $x_2$ | 0.00748 | 0.00283 | 0.00713 | 0.00263 |
| | $u_1$ | -4.9999 | -2.25816 | -4.91856 | -1.93070 |
| | $u_2$ | -1.86650 | -4.8519 | -2.0901 | -4.97742 |
| $G_5$ | $x_1$ | 15.304 | 15.314 | 15.303 | 15.319 |
| | $x_2$ | 4.6974 | 4.708 | 4.6965 | 4.7125 |

| | | | | | |
|---|---|---|---|---|---|
| | $u_1$ | 1.76788 | 1.77142 | 1.76760 | 1.77295 |
| | $u_2$ | -1.76753 | -1.7640 | -1.76781 | -1.7625 |

Table 8: Statistical optimization results obtained by algorithms

| | | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| | Best β | 2.3309 | 2.3309 | 2.3309 | 2.3310 |
| | Worst β | 2.4651 | 2.4613 | 2.6532 | 2.4026 |
| | Avg β | 2.3546 | 2.3497 | 2.3614 | 2.3447 |
| $G_1$ | Std β | 0.0320 | 0.0346 | 0.0813 | 0.0193 |
| | Iteration | 85 | 67 | 33 | 76 |
| | Avg Iteration | 126.15 | 139.35 | 38.3 | 117.2 |
| | Std Iteration | 30.9060 | 49.9634 | 13.5805 | 38.7292 |
| | Best β | 2.2769 | 2.3413 | 2.2701 | 2.3814 |
| | Worst β | 5.1388 | 5.9199 | 4.6905 | 5.0723 |
| | Avg β | 3.2534 | 3.6394 | 2.5852 | 3.2628 |
| $G_2$ | Std β | 0.9307 | 1.0863 | 0.5750 | 0.8835 |
| | Iteration | 170 | 31 | 43 | 88 |
| | Avg Iteration | 163.05 | 125.75 | 60.6 | 121.8 |
| | Std Iteration | 18.7714 | 46.2133 | 11.4542 | 40.2852 |
| | Best β | 4.7963 | 4.7970 | 4.7995 | 4.7989 |
| | Worst β | 4.8038 | 4.8100 | 5.1449 | 4.8434 |
| | Avg β | 4.7986 | 4.8001 | 4.8548 | 4.8139 |
| $G_3$ | Std β | 0.0021 | 0.0033 | 0.0805 | 0.0119 |
| | Iteration | 85 | 152 | 77 | 141 |
| | Avg Iteration | 133.1 | 123.35 | 91.1 | 161.45 |
| | Std Iteration | 38.7147 | 48.3030 | 23.5011 | 23.7807 |
| | Best β | 5.3370 | 5.3516 | 5.3442 | 5.3387 |
| | Worst β | 5.4181 | 5.4278 | 5.4277 | 5.4280 |
| | Avg β | 5.3568 | 5.3946 | 5.4008 | 5.3878 |
| $G_4$ | Std β | 0.0261 | 0.0256 | 0.0247 | 0.0317 |
| | Iteration | 57 | 117 | 46 | 53 |
| | Avg Iteration | 90.8 | 110 | 55.6 | 104.9 |
| | Std Iteration | 42.3712 | 46.6713 | 14.4637 | 42.4932 |
| | Best β | 2.4999 | 2.4999 | 2.4999 | 2.4999 |
| | Worst β | 2.5005 | 2.5009 | 2.5149 | 2.5038 |
| | Avg β | 2.5000 | 2.5001 | 2.5015 | 2.5009 |
| $G_5$ | Std β | 0.0001 | 0.0003 | 0.0035 | 0.0011 |
| | Iteration | 60 | 38 | 23 | 30 |
| | Avg Iteration | 67.65 | 54.95 | 25.3 | 63.7 |
| | Std Iteration | 16.6647 | 24.1997 | 6.5862 | 55.2821 |

Avg β = average optimized β; Std β = standard deviation on optimized β; Iteration = required number of iteration for the best run; Avg Iteration = average required number of iteration; Std Iteration = standard deviation on required number of iteration.

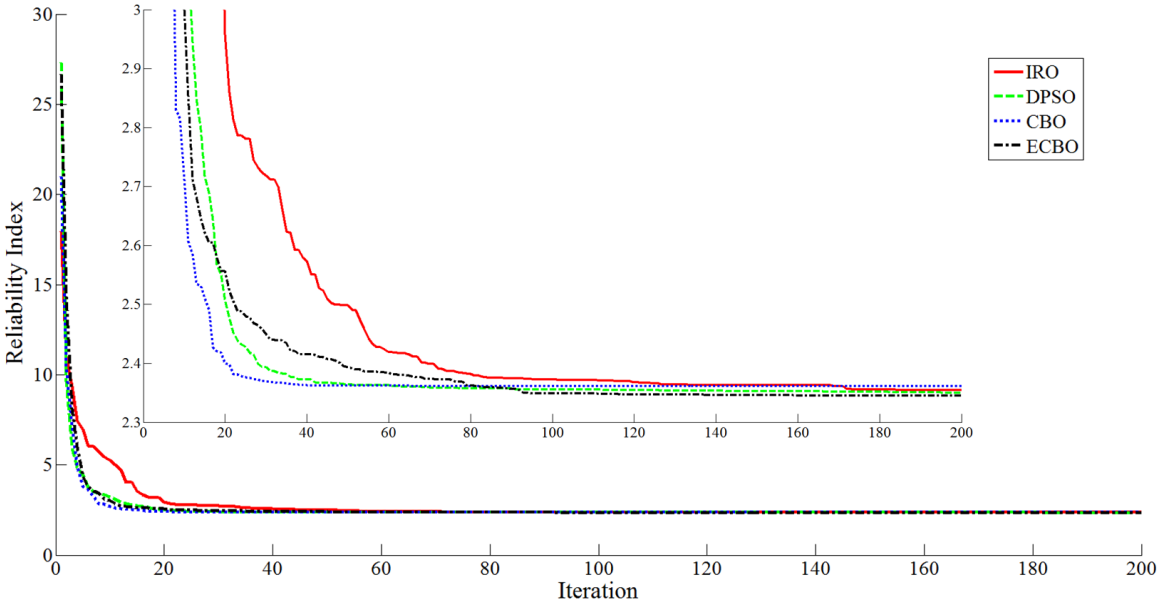


Figure 2. Comparison of average values of reliability index recorded in the optimization history for the different algorithms ($G_1$ from Table 5)

Six limit state functions are listed in Table 9. All of the random variables are independent standard normal. Table 10 summarizes the optimal reliability index and the corresponding probability of failure obtained by crude Monte-Carlo and proposed algorithms. Again the results achieved by different methods are approximately identical. The optimal design points are shown in Table 11 and statistical results are provided in Table 12. Fig. 3 depicts the average convergence history for the optimal reliability index of the first limit state function from Table 9.

Table 9: Limit state function (the components are independent standard normal distribution)

| Limit state function | Description |
|---|---|
| $g_1 = 2.677 - x_1 - x_2$ | |
| $g_2 = 2.500 - x_2 - x_3$ | |
| $g_3 = 2.323 - x_3 - x_4$ | Parallel system |
| $g_1 = 2.250 - x_4 - x_5$ | |
| $G_1(x) = \max(g_1, g_2, g_3, g_4)$ | |
| $g_1 = -x_1 - x_2 - x_3 + 3\sqrt{3}$ | |
| $g_2 = -x_3 + 3$ | Series system |
| $G_2(x) = \min(g_1, g_2)$ | |
| $g_1 = -x_1 - x_2 - x_3 + 3\sqrt{3}$ | |
| $g_2 = -x_3 + 3$ | Parallel system |
| $G_3(x) = \max(g_1, g_2)$ | |

| | |
|---|---|
| $g_1 = 2 - x_2 + \exp(-0.1x_1^2) + (0.2x_1)^4$ | |
| $g_2 = 4.5 - x_1 x_2$ | Series system |
| $G_4(x) = \min(g_1, g_2)$ | |
| $g_1 = 2 - x_2 + \exp(-0.1x_1^2) + (0.2x_1)^4$ | |
| $g_2 = 4.5 - x_1 x_2$ | Parallel system |
| $G_5(x) = \max(g_1, g_2)$ | |
| $g_1 = 0.1(x_1 - x_2)^2 - \frac{(x_1 + x_2)}{\sqrt{2}} + 3$ | |
| $g_2 = 0.1(x_1 - x_2)^2 + \frac{(x_1 + x_2)}{\sqrt{2}} + 3$ | Series system |
| $g_3 = x_1 - x_1 x_2 + 3.5\sqrt{2}$ | |
| $g_1 = -x_1 + x_1 x_2 + 3.5\sqrt{2}$ | |
| $G_6(x) = \max(g_1, g_2, g_3, g_4)$ | |

Table 10: Comparison of failure probability and reliability index obtained by different methods

| | | Grooteman [27] | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|---|
| $G_1$ | $P_f$ | 2.11e-04 | 0.003529 | 0.003442 | 0.003439 | 0.003363 |
| | β | 2.738 | 2.6940 | 2.7023 | 2.7026 | 2.7100 |
| $G_2$ | $P_f$ | 2.57e-03 | 0.00135 | 0.001350 | 0.001350 | 0.001350 |
| | β | 2.953 | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| $G_3$ | $P_f$ | 1.23e-04 | 0.000365 | 0.000364 | 0.000365 | 0.000364 |
| | β | 3.434 | 3.3780 | 3.3781 | 3.3780 | 3.3782 |
| $G_4$ | $P_f$ | 3.54e-03 | 0.001350 | 0.001350 | 0.00135 | 0.001350 |
| | β | 2.925 | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| $G_5$ | $P_f$ | 2.50e-04 | 0.000647 | 0.000647 | 0.000647 | 0.000647 |
| | β | 3.219 | 3.2171 | 3.2172 | 3.2171 | 3.2172 |
| $G_6$ | $P_f$ | 2.18e-03 | 0.001350 | 0.001350 | 0.001350 | 0.001350 |
| | β | 2.925 | 2.9999 | 2.9999 | 2.9999 | 2.9999 |

Table 11: Design points obtained by algorithms

| | Design point | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| | $x_1$ | 1.08776 | 1.03139 | 0.99898 | 1.18811 |
| | $x_2$ | 1.58933 | 1.64823 | 1.67792 | 1.49069 |
| $G_1$ | $x_3$ | 0.93877 | 0.86184 | 0.82198 | 1.0100 |
| | $x_4$ | 1.38594 | 1.4611 | 1.5020 | 1.40707 |
| | $x_5$ | 0.8640 | 0.80269 | 0.74781 | 0.84296 |
| | $x_1$ | -0.00010 | 0.00337 | -2.03e-008 | 0.00751 |
| $G_2$ | $x_2$ | -0.02114 | 0.00657 | 1.56e-008 | -0.00311 |
| | $x_3$ | 2.99990 | 2.9999 | 2.9999 | 2.9999 |
| $G_3$ | $x_1$ | 1.10801 | 1.08304 | 1.10151 | 1.11241 |

| | | | | | |
|---|---|---|---|---|---|
| | $x_2$ | 1.08813 | 1.11315 | 1.09463 | 1.08349 |
| | $x_3$ | 2.99990 | 2.99995 | 2.9999 | 3.00017 |
| $G_4$ | $x_1$ | 0.00839 | 0.00211 | -5.56e-008 | -5.43e-005 |
| | $x_2$ | 2.99989 | 2.99989 | 2.9999 | 2.99990 |
| $G_5$ | $x_1$ | 1.61840 | 1.61854 | 1.61840 | 1.61856 |
| | $x_2$ | 2.78044 | 2.78047 | 2.78044 | 2.78045 |
| $G_6$ | $x_1$ | -2.12052 | -2.1243 | -2.12428 | -2.12016 |
| | $x_2$ | -2.12197 | -2.1182 | -2.11821 | -2.1223 |

Table 12: Statistical optimization results obtained by algorithms

| | | IRO | DPSO | CBO | ECBO |
|---|---|---|---|---|---|
| | Best β | 2.6940 | 2.7023 | 2.7026 | 2.7100 |
| | Worst β | 2.8248 | 2.8696 | 4.7878 | 2.8324 |
| | Avg β | 2.7277 | 2.7413 | 2.9720 | 2.7592 |
| $G_1$ | Std β | 0.0420 | 0.0446 | 0.4713 | 0.0350 |
| | Iteration | 177 | 128 | 104 | 157 |
| | Avg Iteration | 160.55 | 146.4 | 110.4 | 138.05 |
| | Std Iteration | 21.3380 | 42.7285 | 22.2365 | 49.5160 |
| | Best β | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| | Worst β | 3.0089 | 3.0007 | 2.9999 | 3.0412 |
| | Avg β | 3.0022 | 3.0000 | 2.9999 | 3.0070 |
| $G_2$ | Std β | 0.0026 | 0.0001 | 1.45e-012 | 0.0104 |
| | Iteration | 76 | 61 | 29 | 52 |
| | Avg Iteration | 116.05 | 84.15 | 31.35 | 113.35 |
| | Std Iteration | 22.4768 | 34.6581 | 5.1837 | 58.6481 |
| | Best β | 3.3780 | 3.3781 | 3.3780 | 3.3782 |
| | Worst β | 3.4034 | 3.3869 | 3.3955 | 3.3967 |
| | Avg β | 3.3840 | 3.3811 | 3.3808 | 3.3823 |
| $G_3$ | Std β | 0.0070 | 0.0023 | 0.0040 | 0.0047 |
| | Iteration | 111 | 172 | 56 | 98 |
| | Avg Iteration | 127.3 | 127.2 | 50.05 | 116.35 |
| | Std Iteration | 25.1732 | 44.2880 | 14.5872 | 39.3864 |
| | Best β | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| | Worst β | 3.0508 | 3.0077 | 3.0388 | 3.0083 |
| | Avg β | 3.0081 | 3.0003 | 3.0035 | 3.0003 |
| $G_4$ | Std β | 0.0131 | 0.0017 | 0.0093 | 0.0018 |
| | Iteration | 106 | 34 | 28 | 90 |
| | Avg Iteration | 112.65 | 64.85 | 28.25 | 81.35 |
| | Std Iteration | 47.0668 | 38.7777 | 5.9105 | 41.3995 |
| | Best β | 3.2171 | 3.2172 | 3.2171 | 3.2172 |
| | Worst β | 3.2221 | 3.2209 | 3.3381 | 3.2194 |
| $G_5$ | Avg β | 3.2178 | 3.2182 | 3.2304 | 3.2180 |
| | Std β | 0.0013 | 0.0009 | 0.0297 | 0.0006 |
| | Iteration | 84 | 72 | 54 | 36 |

| | | | | | |
|---|---|---|---|---|---|
| | Avg Iteration | 119.05 | 100.75 | 62.5 | 108.75 |
| | Std Iteration | 24.9071 | 42.2970 | 24.3688 | 45.5653 |
| | Best β | 2.9999 | 2.9999 | 2.9999 | 2.9999 |
| | Worst β | 3.0014 | 3.0027 | 3.0070 | 3.0038 |
| | Avg β | 3.0001 | 3.0005 | 3.0008 | 3.0006 |
| $G_6$ | Std β | 0.0004 | 0.0007 | 0.0016 | 0.0011 |
| | Iteration | 74 | 108 | 21 | 95 |
| | Avg Iteration | 81.05 | 76.7 | 27.95 | 83.95 |
| | Std Iteration | 12.9917 | 32.5383 | 8.5129 | 56.0549 |

Avg β = average optimized β; Std β = standard deviation on optimized β; Iteration = required number of iteration for the best run; Avg Iteration = average required number of iteration; Std Iteration = standard deviation on required number of iteration.

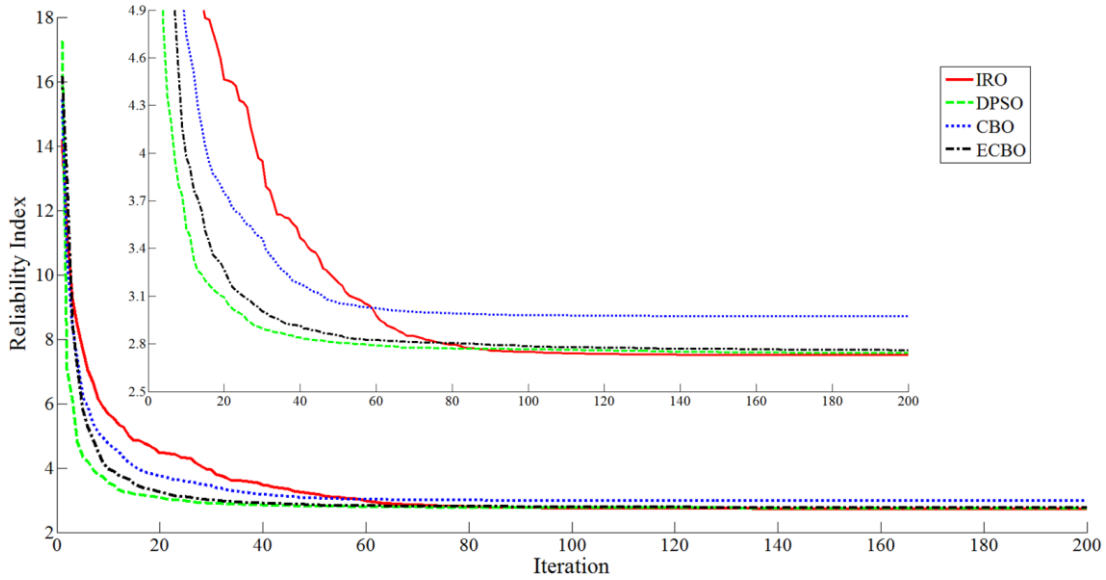


Figure 3. Comparison of average values of reliability index recorded in the optimization history for the different algorithms ($G_1$ from Table 9)

## 5. CONCLUSION

Structural reliability analysis utilizing the recently developed metaheuristic algorithms consisting of Improved Ray Optimization, Democratic Particle Swarm Optimization, Colliding Bodies Optimization, and Enhanced Colliding Bodies Optimization is studied in this paper. The various problems serve to demonstrate the efficiency and robustness of these optimization methods. The optimal reliability indexes yielded by the proposed algorithms are approximately identical with respect to those results reported in literature such as crude Monte-Carlo that can be obtained a near exact solution by evaluating many times of the limit state function. Therefore the proposed methods always converge to the optimal values of reliability index and their accuracies are proved. Also, a little standard deviation from the mean value of the twenty

independent runs and the low computing time of these zero order algorithms show the reliability of search and the efficiency of proposed methods, respectively. Generally, comparison of the results proves the suitability and efficiency of the proposed algorithms.

## REFERENCES


1. Jiang SH, Li DQ, Zhou CB, Zhang LM. Capabilities of stochastic response surface method and response surface method in reliability analysis, *Struct Eng Mech* 2014; **49**(1): 111-28.
2. Abbasnia R, Shayanfar M, Khodam A. Reliability-based design optimization of structural systems using a hybrid genetic algorithm, *Struct Eng Mech* 2014; **52**(6): 1099-1120.
3. Meng Z, Li G, Wang BP, Hao P. A hybrid chaos control approach of the performance measure functions for reliability-based design optimization, *Comput Struct* 2015; **146**: 32-43.
4. Zhao YG, Ono T. A general procedure for first/second-order reliability method (FORM/SORM), *Struct Safety* 1999; **21**: 95-112.
5. Zhao YG, Ono T. Moment methods for structural reliability, *Struct Safety* 2001; **23**: 47-75.
6. Lee SH, Kwak BM. Response surface augmented moment method for efficient reliability analysis, *Struct Safety* 2006; **28:** 261-72.
7. Harbitz A. An efficient sampling method for probability of failure calculation, *Struct Safety* 1986; **3**(2): 109-15.
8. Yonezawa M, Okuda S, Kobayashi H. Structural reliability estimation based on quasi ideal importance sampling simulation, *Struct Eng Mech* 2009; **32**(1): 55-69.
9. Ghohani Arab H, Ghasemi M R, Rashki M, Miri M. Enhancing weighted uniform simulation for structural reliability analysis, *Int J Optim Civil Eng* 2013; **3**(4): 635-51.
10. Jahani E, Shayanfar MA, Barkhordari A. A New Adaptive Importance Sampling Monte Carlo Method for Structural Reliability, *KSCE J Civil Eng* 2013; **17**(1): 210-15.
11. Angelis MD, Patelli E, Beer M. Advanced Line Sampling for efficient robust reliability analysis, *Struct Safety* 2015; **52**: 170-82.
12. Deng L, Ghosn M, Shao S. Shredding genetic algorithm for reliability analysis of structural systems, *J Struct Safety* 2004; **27**(4): 113-31.
13. Elegbede C. Structural reliability assessment based on particles swarm optimization, *Struct Safety* 2005; **27**: 171-86.
14. Zou D, Goa L, Wu J, Li S, Li Y. A novel global harmony search algorithm for reliability problems, *Comput Indust Eng* 2010; **58**: 307-16.
15. Zou D, Goa L, Li S, Wu J. An effective global harmony search algorithm for reliability problems, *Expert Systems with Applications* 2011; **38**: 4642-8.
16. Valian E, Tavakoli S, Mohanna S, Haghi A. Improved cuckoo search for reliability optimization problems, *Comput Indust Eng* 2013; **64**: 459-68.
17. Kaveh A, Massoudi MS, Ghanooni Bagha M. Structural reliability analysis using charged system search algorithm, *IJST, Trans Civil Eng* 2014; 38(C2): 439-48.

18. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T. An improved ray optimization algorithm for design of truss structures, *Periodica Polytechnica* 2013; **57**(2): 1-15.
19. Kaveh A, Zolghadr A. Democratic PSO for truss layout and size optimization with frequency Constraints, *Comput Struct* 2014; **130**: 10-21.
20. Kaveh A, Mahdavai VR. Colliding bodies optimization: A novel meta-heuristic method. *Comput Struct* 2014; **139**: 18-27.
21. Kaveh A, Ilchi Ghazaan M. Enhanced colliding bodies optimization for design problems with continuous and discrete variables, *Adv Eng Softw* 2014; **77**: 66-75.
22. Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization, *Comput Struct* 2012; **112-113**: 283–94.
23. Eberhart RC, Kennedy J. A new optimizer using particle swarm theory, *In Proc 6th Int Symp Micromach Hum Sci* 1995; pp. 39–43.
24. Hasofer AM, Lind NC. An exact and invariant first order reliability format, *J Eng Mech*, ASCE 1974; **100**(1): 111-21.
25. Kennedy J, Eberhart RC. Particle swarm optimization, *In Proc IEEE Int Conf Neural Networks* 1995; pp. 1942-1948.
26. Gomes A. Truss optimization with dynamic constraints using a particle swarm algorithm, *Expert Syst Applic* 2011; **38**: 957-68.
27. Grooteman F. Adaptive radial-based importance sampling method for structural reliability, *Struct Safety* 2008; **30**: 533-42.