Available online at www.sciencedirect.com
SCIENCE DIRECT®
Structural Safety 27 (2005) 171–186
STRUCTURAL SAFETY
www.elsevier.com/locate/strusafe

Structural reliability assessment based on particles swarm optimization

Charles Elegbede *

Department of RAMS and Nuclear Safety Analysis, EADS SPACE Transportation, Avenue du Général Niox, 33165 Saint Médard en Jalles, France

Received 6 February 2004; received in revised form 8 May 2004; accepted 26 October 2004

Abstract

The PSO algorithm is very efficient to solve global optimization problems with continuous variables. Its use in the structural reliability field presents not only the advantage of its facility of implementation, but also the possibility to obtain the design point and the failure probability with a good accuracy. Several examples of the literature studied in this paper show that the results yielded by PSO are quasi-exact with respect to those yielded by MC and response surface methods. The low computing time of this zero order algorithm is also a great advantage to solve optimization problem. Therefore, this algorithm can be considered as an additional efficient algorithm to those existing in the literature based on gradient method.

Keywords: Structural reliability; Particles swarm optimization; Evolutionary algorithms

1. Introduction

Structural time invariant reliability assessment consists in modeling every design variable on which there are uncertainties by using random vector $\boldsymbol{X}$. After that, a failure criterion is defined by a limit state function $G(\boldsymbol{X})$ that defines the failure domain D [1]. To assess the structure

* Tel.: +33 5 56 57 20 85; fax: +33 5 56 57 38 10.
E-mail address: charles.elegbede@space.eads.net.

0167-4730/$ - see front matter
doi:10.1016/j.strusafe.2004.10.003

reliability, one needs to know the joint density probability function, $f_X(x)$, of vector $\boldsymbol{X}$. The failure probability, P_f, will then be obtained by integrating $f_X(x)$ over the failure domain D.

For designers, it is not only important to obtain the failure probability but also to know the design point of the structure. Intensive researches have been carried out on these two points during the two last decades. The design point gives these two information, because it enables to calculate the failure probability by means of FORM approximation. Even if Monte-Carlo simulation is an efficient approach when the limit state function is known, it is well-known that a large number of simulations is necessary to assess small failure probabilities [2,9] (10^{-8}–10^{-4} for example). This is a weakness of classical Monte-Carlo simulations in the assessment of failure probability. Modern Monte-Carlo simulation enables to fill this gap, but does not enable to obtain the design point. Thus, it is useful to develop methods based on global optimization to determine the design point and therefore, the failure probability.

Hasofer and Lind introduced a notion of reliability index that enables to obtain the structure reliability [3]. To obtain the reliability index, one needs to solve a constrained optimization problem. Some methods are proposed in the literature based on gradient method such as Abdo–Rackwitz algorithm [3]. The aim of this paper is to propose an evolutionary algorithm that enables to solve global search optimization problem: Particles Swarm Optimization [5,6]. This algorithm is not competing with existing methods, it is introduced because of its ability to solve global optimization problem efficiently.

The remainder of the paper is organized as follows: after Section 2 which presents the main notations used in the paper, Section 3 presents the statement of reliability problem and some existing methods to solve them. Section 4 presents the PSO optimization framework and their operating rules. Section 5 studies some limit states functions reported in the literature. Section 6 provides the main conclusions and perspectives.

2. Notations

β	reliability index
ξ	penalty function
λ	penalty coefficient
lsf	limit state function
P_f	failure probability
r.v	random variable
$\boldsymbol{\Lambda}$	space where variables are independent and standard Gaussian
$\boldsymbol{\Psi}$	space of physical variables
$\boldsymbol{u}^*$	design point in space $\boldsymbol{\Lambda}$
$\boldsymbol{x}^*$	design point in space $\boldsymbol{\Psi}$
T	transformation enabling to change space $\boldsymbol{\Psi}$ into space $\boldsymbol{\Lambda}$
G	a given lsf
D	failure domain of the structure
$\delta(D)$	border of the failure domain D
COV	Coefficient Of Variation
PDF	Probability Density Function
FORM	First Order Reliability Method

SORM Second Order Reliability Method
PSO Particles Swarm Optimization.

3. Failure probability assessment

In this section, the problem of reliability assessment is stated and the reliability index is introduced for reliability assessment.

3.1. Statement of the problem

Let $\boldsymbol{x}$ be the vector of design variables on which there are some uncertainties and n its dimension. This vector may be modeled by random vector $\boldsymbol{X}$, and $f_X(x)$ its PDF. A structure failure state is defined by a limit state function (lsf) G, and conventionally in the literature [1] with the following properties:

$$\begin{aligned} G: \quad & \Re^n \to \Re \\ & X \mapsto G(X), \end{aligned} \tag{0}$$

such that:

- if $G(X) < 0$ the system is in a failure state,
- if $G(X) = 0$ the system is in a limit state (this state is considered as a failure state),
- if $G(X) > 0$ the system is in a safe state.

The random variable defined by $Z = G(\boldsymbol{X})$ is also called the performance of the structure or the safety margin of the structure. According to the definition of G, when the structure margin is lower or equal to zero, the system fails. Therefore, the failure domain is defined by $D = \{\mathbf{x} \in \Re^n, G(\boldsymbol{x}) \leqslant 0\}$ (considering that the border $\delta(D)$ is also a failure set). The structure failure probability, P_f, is given by the formula

$$P_f = \int_D f_X(x)\, \mathrm{d}x. \tag{1}$$

Generally, it is difficult to solve (1) because of one of the two following reasons:

- the border $\delta(D)$ is non linear,
- the vector $\boldsymbol{X}$ is not a linear function of standard Gaussian vector (i.e., vector $\boldsymbol{X}$ components are not Gaussian or they are correlated).

In the two last decades, intensive researches have been carried out to provide methods to solve (1), a (non exhaustive probably) list of main approaches to solve (1) is:

- direct integration of PDF on the failure domain,
- Monte-Carlo simulations [7] (including subsets simulations [8]),
- FORM/SORM using response surface method or the reliability index [1,8],
- discrete approximation [10].

The recent state of the art on structural reliability analysis gives a good insight of most of these methods [11].

3.2. Reliability index

Two main reliability indexes may be used as measures of the reliability of a structure.

3.2.1. Cornell reliability index β_C

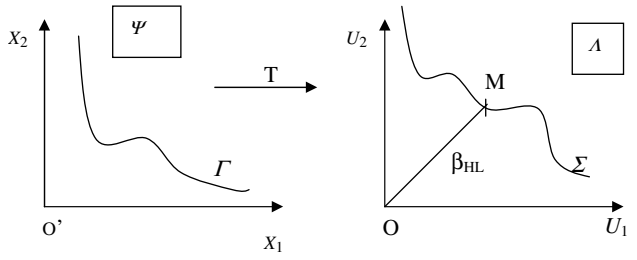
With the notations used in Section 3 Cornell index is defined by

$$\beta_{\mathrm{C}} = \frac{E(Z)}{\sigma_Z}. \tag{2}$$

In the physical variables space, Cornell reliability index is not interesting for it strongly depends on the mathematical expansion of limit state function $\boldsymbol{G}$ (two mathematical formulations of the same failure mode lead to different value of Cornell index). It is not easy to link Cornell index to failure probability.

3.2.2. Hasofer–Lind reliability index

Hasofer–Lind reliability index is proposed in a space, $\boldsymbol{\Lambda}$, where the vector components are Gaussian standard (uncorrelated). Let $\boldsymbol{U}$ be any vector of this space, $\boldsymbol{\Gamma}$ the n-dimensional surface defined by the limit state function $\boldsymbol{G}$ in the physical variables space $\boldsymbol{\Psi}$, and $\boldsymbol{\Sigma} = T(\boldsymbol{\Gamma})$ its image in the standard Gaussian space $\boldsymbol{\Lambda}$. The Hasofer–Lind reliability index, β_{HL} is defined as $\beta_{\mathrm{HL}} =$ Min-(d(O,M)) where O is the center (origin) of space $\boldsymbol{\Lambda}$, and M evolves on the surface $\boldsymbol{\Sigma}$. This index enables to have a first order approximation (FORM) of the reliability by the relation $P_{\mathrm{f}} \approx \Phi(-\beta_{\mathrm{HL}})$. This relation is exact when the lsf is linear in space $\boldsymbol{\Lambda}$: $P_{\mathrm{f}} = \Phi(-\beta_{\mathrm{HL}})$.



There are three main transformations $\boldsymbol{T}$ which enable to change a vector $\boldsymbol{X}$ of the physical variables space ($\boldsymbol{\Psi}$) into a vector $\boldsymbol{U}$ of the standard Gaussian space ($\boldsymbol{\Lambda}$).

Rosemblatt transformation: used when the PDF of $\boldsymbol{X}$ is known, and when $\boldsymbol{X}$ components are correlated. The components of vector$\boldsymbol{X}$ are not necessary Gaussian. More details may be seen in [1]. A variant of this transformation is also proposed in [12].

Nataf transformation: used when PDF of $\boldsymbol{X}$ is not known but the correlation matrix of $\boldsymbol{X}$ is known. More details may be seen in [13].

Plain linear transformation: used when the vector $\boldsymbol{X}$ is Gaussian without any correlation between its components. Therefore, if m_i and σ_i ($\sigma_i \neq 0$, $i = 1,\ldots,n$) are, respectively, the mean value and the

standard deviation of component X_i of vector $\boldsymbol{X}$, the transformation $\boldsymbol{T}(\boldsymbol{X}) = \boldsymbol{U}$, may be obtained by $T_i(\boldsymbol{X}) = U_i\ (i = 1,\ldots,n)$ with $U_i = (X_i - m_i)/\sigma_i$. From this, one can obtain the surface $\boldsymbol{\Sigma} = \boldsymbol{T}(\boldsymbol{\Gamma})$.

The aim of this paper is to calculate β_{HL} in the space $\boldsymbol{\Lambda}$ using an evolutionary algorithm named PSO. To achieve this, one has to solve a constrained optimization problem that is

$$\begin{cases} \underset{\boldsymbol{u}}{\text{Minimize}} \sum_{i=1}^{n} u_i^2 \\ \text{Subject to} \\ G\left(T^{-1}(\boldsymbol{u})\right) = 0. \end{cases} \tag{3}$$

Solve Eq. (3) is equivalent to solve the relaxed form obtained by penalty method

$$\underset{\boldsymbol{u}}{\text{Minimize}} \sum_{i=1}^{n} u_i^2 + \lambda\xi\left(G\left(T^{-1}(\boldsymbol{u})\right)\right), \tag{4}$$

where ξ is the penalty function and λ is the penalty coefficient (strictly positive).

The solution $\boldsymbol{u}^*$ of Eq. (3) or (4) is called the design point and enables to calculate the reliability index as

$$\beta_{\mathrm{HL}} = \|\mathbf{u}^*\|. \tag{5}$$

The penalty method used in Eq. (4) is called exterior penalty [15,16]. The search is performed from the exterior of the solutions space. For function ξ, generally one of the two following functions is used: (i) quadratic function: $x \mapsto x^2$; (ii) exact penalty function $x \mapsto |x|$. This last penalty function is the one used in the present paper. In case of inequality constraint, the penalty function is equal to 0 when the constraint is satisfied, and equal to ξ otherwise.

The choice of the penalty coefficient λ in Eq. (4) is crucial for the convergence of the search toward the solution of Eq. (3). In the case of inequality constraints, λ should be positive and large enough to converge [14,15]. Therefore, to determine this coefficient, an iterative approach is suggested consisting in choosing a sequence of λ_i, such that $\lambda_{i+1} = 10\lambda_i$, and $\lambda_0 = 1$ [14]. Generally, the convergence is reached quickly, i.e., after a few number of iterations. In case of equality constraints as it is addressed in this paper, the penalty coefficient will be searched by an iterative process from a low value because the search space is a hyper-surface. According to our experience, an appropriate sequence of λ is λ_i, such that $\lambda_{i+1} = 2\ \lambda_i$, and $\lambda_0 = 0.1$. The value of λ will be considered suitable when the quantity $\xi(G(T^{-1}(u)))$ in Eq.(4) is enough small($<10^{-4}$ for example). For each problem solved in this paper, the corresponding λ and the control parameter $\xi(G(T^{-1}(u)))$ are supplied.

The main inconvenient of penalty method is that when the penalty coefficient is too large, it leads to ill conditioning of the problem [14,17]. But, the sequences quoted above to determine penalty coefficient enable to overcome this difficulty [14,15].

4. Optimization engine

Assume now that one wants to solve the following problem:

$$\underset{x\in\Omega}{\text{Maximize}}\, H(x), \tag{6}$$

where Ω is a subset of $\Re^n$.

It is always possible to turn any optimization problem into a maximization one. In the following section, it is proposed to solve such a problem by means of PSO. The problem stated in Eq. (6) is a global optimization problem. The difficulty in global optimization is that solution found may not be a global optimum but a local optimum. Therefore, it is necessary to have algorithms that have the ability to find the global optimum. It is well known that the traditional algorithms like those based on gradient method can only provide a local optimum depending on the guess of the search [14].

Unlike classical methods, evolutionary algorithms are able to find a global optimum or a good approximation of the solution [18], generally without a theoretical proof. This is due to their ability to explore the search domain with "jump" from a local solution to another local solution and therefore step by step reach the global optimum.

4.1. General scope

Particles swarm optimization is first introduced by Kennedy and Eberhart in 1995 [4] to solve continuous global optimization problem with non linear objective. The method is based on simulating social behavior of bird flock and school fish. The key idea is that in the flock, or school fish, any agent of the group can profit from the discoveries and previous experiences of all members of the school during search of food. This advantage can become decisive, outweighing the inconvenient of competition for food items, whenever the resource is unpredictably distributed in patches. That means information are socially shared by the bird flock or fish school and give an evolutionary advantage. The main idea is to generate particles randomly and assign to them a motion law by the use of velocity notion.

The algorithm proposed by Kennedy and Eberhart is called Particles Swarm Optimization in accordance to the analogy with the Swarm theory of Millonas and the Particles theory of Reeves recalled in [5]. Five principles govern the swarm theory: (i) the proximity principle: the population should be able to carry out simple space and time computation; (ii) the quality principle: the population should be able to respond to quality factors in environment; (iii) the principle of diverse responses: the population should not commit its activities along excessively narrow channels; (iv) the principle of stability: the population should not change its mode of behavior every time the environment changes; (v) the principle of adaptability: the population must be able to change its behavior mode when it is worth the computational price. In [5], it is stated that the paradigm PSO fulfils all these five conditions. Reeves defines particles systems as clouds of primitive particles as models of diffuse objects such as clouds, fire and smoke [5]. In PSO, population consists of points which are mass-less and volume-less, it is why Eberhart and Kennedy [5] chose the term particles.

4.2. Generating the initial population

The optimization based on particles swarm starts with an initial population like in Genetic algorithms. In fact, the population is equivalent to the locations (positions) of a group of birds or fishes in search for food for example. The position of each bird of the group will be considered as a potential solution of the problem addressed. In genetic algorithms, the individuals are reproduced according to the selection, the cross over and mutation, whereas in PSO, the notion of reproduction does not exist. The individuals are supposed to be the same throughout the search

and only their positions in the space change according to their velocity as described in Section 4.4. In the remainder of this paper, the word "individual" is used to denote a potential solution, a particle, or the position of a bird or a fish. At the beginning of the optimization, the initial population is generated randomly according to an uniform law in the search space Ω. It is admitted that the search space is the hypercube $\Omega = \prod_{j=1}^{n}[x_j^l, x_j^u]$, and x_j^l, x_j^u are, respectively, the lower and upper bounds of variable x_j ($j = 1,\ldots,n$), these bounds may increase to infinity. The initial population consists of T_{pop} samples of vector $\boldsymbol{x}$ stored in a matrix I_{ij}, with I_{ij} = uniform random (x_j^l, x_j^u), ($i = 1,\ldots,T_{\text{pop}}$). There is no general rule to choose the size of the population. But it can be admitted that it must be chosen such that the computing time is minimized. A large size of population involves a large computing time and a small size of population may conduct to a fail in the search. Therefore, a tradeoff must be found in the choice of the size. According to Shi and Eherhart [19] it is sufficient to consider the size of the population equal to 20. In this paper, the choice of the size will arbitrary be 70. With example # 5, some simulations are performed to show, briefly, the impact of the size on the results.

The problem addressed in this paper is an optimization of a function in the space $\boldsymbol{\Lambda}$ which is in fact $(-\infty, +\infty)^n$. As variables U_j for each dimension j of that space are standard normal random variables, the search space can be reduced to $[-10, +10]^n$ due to the fact that $P(U_j < -10) < 10^{-16}$. That means rare events with very low probability occurrence are not considered. This remains realistic, at least, in engineering point of view.

4.3. Evaluation procedure

At each iteration of the algorithm, the individuals are classified according to their fitness. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are two different individuals in the current population, $\boldsymbol{x}$ is better than $\boldsymbol{y}$ if and only if $H(\boldsymbol{x}) > H(\boldsymbol{y})$. In the PSO that means the position occupied by particle $\boldsymbol{x}$ is closer to the solution than $\boldsymbol{y}$ is. The function H is called fitness function and is problem-dependant. In case of unconstrained maximization problems, H is the objective function, and in case of constrained optimization problems, it is sufficient to turn the problem into an unconstrained optimization one by using penalties methods for example. It is even the approach used in this paper to transform the problem stated in Eq. (3), which is a constrained optimization, into Eq. (4) which is an unconstrained optimization.

4.4. Velocity of individuals and motion law

The velocity of a particle, i, is a vector having the same dimension. For each dimension, the velocity is calculated according to the following formula [19]:

$$V_{ij}^{(h+1)} = wV_{ij}^{(h)} + c_1 r_1\left(P_{ij}^{(h)} - I_{ij}^{(h)}\right) + c_2 r_2\left(P_{gj}^{(h)} - I_{ij}^{(h)}\right), \tag{7}$$

where: i, $i = 1,\ldots,T_{\text{pop}}$; j, $j = 1,\ldots,n$ the index of the search space dimension; h, the index of current iteration; g, the index of the best position met by the group from the beginning of the search; c_1, a deterministic parameter; r_1, a sample of a uniform (0,1) random variable at each iteration; c_2, a deterministic parameter; r_2, a sample of a uniform (0,1) random variable at each iteration; w, a deterministic parameter, which is an inertia parameter.

The algorithm starts with all velocities equal to 0 at the first iteration.

There are two main contributors to the velocity:

- the own experience of the individual taken into account by the term $(P_{ij}^{(h)} - I_{ij}^{(h)})$. This term depends on both the best position, $P_{ij}^{(h)}$, that has been occupied throughout the simulation and the current position, $I_{ij}^{(h)}$ occupied by the particle i.
- the group's experience taken into account by the term $(P_{gj}^{(h)} - I_{ij}^{(h)})$. This term depends on both the best position, $P_{gj}^{(h)}$ that has been reached by the group throughout the simulations, and the current position, $I_{ij}^{(h)}$ of the particle i.

The motion law which enables the particle to move from a position at iteration (h) to another at iteration ($h + 1$) in the space is governed by equation

$$I_{ij}^{(h+1)} = I_{ij}^{(h)} + V_{ij}^{(h+1)}. \tag{8}$$

To perform a local search, Eberhart et al. [5] proposed a local version of PSO, which consists in using for a given individual, i, the neighborhood composed of $i - 1$, i, $i + 1$ for neighborhood 2 for example. The maximum neighborhood used is 6. Therefore, instead of particles i using knowledge from the group, it only uses knowledge from his neighborhood. The velocity is also bounded at 6 for this version to avoid the particle to miss the local optimum.

For our purpose which is a global optimization, we use a notion that we call "come back to the center" to control the velocity. That means if the velocity is out of the search space, the particle is called back in its center.

At each iteration, the velocity and the position of each particle must be updated by means of Eqs. (7) and (8).

4.4.1. The inertia parameter

The notion of inertia parameter is introduced by Shi and Eberhart [19] to control the impact of the previous history of velocities on current velocity. This enables to influence the tradeoff between global and local exploration abilities of the particles. A larger inertia weight facilitates global optimization, while smaller inertia weight facilitates local optimization. In the same reference, it is suggested w to range, in a decreasing way, from $w = 1.4$ to $w = 0$ in an adaptive way. The optimum is the range [0.8, 0.9].

In the first version of the algorithm introduced by Kennedy and Eberhart [4], the inertia (even if they did not use this term) is equal to $w = 1$ and at the same time they try a version of the algorithm with $w = 0$. This latter was proved to be less appropriate for global search.

In this paper, $w = 0.75$ is chosen close to the optimal value known. It seems to be a compromise for the global search.

4.4.2. The parameters c_1 and c_2

In the first version of the algorithm PSO introduced by Kennedy and Eberhart [4], the constants c_1 and c_2 are chosen equal to 2 corresponding to the optimal value for the problem studied by the authors. They indicate in the same reference that the choice of these constants is problem dependent. In our opinion, these constants should be enough lower than the half of the maximum bound of the search space bound in each dimension. This may be explained by the fact that if c_1 and c_2 are too large with respect to the bound of the search space, the particle will regularly leave the search space.

In this paper $c_1 = 1.5$ and $c_2 = 1.2$ are used; other values like 2, 3, 4 enable the convergence of the algorithm in an equivalent number of simulations.

The impact of the these coefficients on the search is that if they are not optimal, the number of iterations necessary for the search will be large. An efficient way to determine the set of optimal parameters for the algorithm may be the use of an experience plan as done by Elegbede and Adjallah [20] to tune genetic algorithms parameters.

4.5. Optimization principle

As discussed in previous sections, the algorithm needs some parameters to be initialized:

- $w = 0.75$ in this paper, see Section 4.4.1,
- $c_1 = 1.5$ and $c_2 = 1.2$ in this paper see Section 4.4.2,
- $T_{\text{pop}} = 70$ in this paper see Section 4.2,
- $N_{\text{iter}} = 990$ the number of iterations, i.e., the number of runs to be performed ($h = 1,\ldots,N_{\text{iter}}$).

The PSO algorithm, which is a zero order algorithm (gradient free) can be summarized as follows:

Algorithm 1.

1. Generate randomly a set of T_{pop} individuals represented by (I_{ij}), $i = 1,\ldots,T_{\text{pop}}$, $j = 1,\ldots,n$.
2. Identify the individual having the best position of the group, let g be its subscript: I_{gj}, $j = 1,\ldots,n$.
3. Identify the best position (P_{ij}) taken by each individual until run hth, $h = 1,\ldots,T_{\text{pop}}$.
4. Calculate the velocity (V_{ij}) of each individual with information from steps 2 and 3.
5. Calculate the new position of each individual.
6. Stop if a criterion is met, else go to step 2.

5. Numerical examples

In this section, it is proposed to use some relevant examples drawn from the literature in order to compare the efficiency of this algorithm. These examples are performed in the literature using classical methods as: (i) Monte-Carlo simulation, (ii) response surface method or (iii) improved response surface method.

As it is stated in the previous section, the tuning of the penalty parameter is crucial, it is why its value used for the convergence is supplied in this paper for each addressed example. The value of $\xi(G(T^{-1}(u)))$ (cf. Eq. (4)) must be close to 0. It will be considered closed to 0 if its absolute value is lower than 10^{-4} (arbitrary choice but in accordance with the problem addressed because generally $\beta > 1$). This is a necessary condition to have the global optimum.

With example #5, a brief sensitivity analysis is performed on the parameters T_{pop} and N_{iter} of the algorithm. The parameters of the algorithm used in this paper are in Table 1.

In the following examples, when the vector X is defined in the space Ψ, x^* denotes the design point in the physical space and u^* denotes the design point in the space Λ. $x^* = T^{-1}(u^*)$, where T is the plain transformation presented in Section 3.2.2.

Table 1
Parameters of the algorithm

w	c_1	c_2	T_{pop}	N_{iter}
0.75	1.5	1.2	70	990

5.1. Example # 1

This example has been studied in [21]. The limit state function in consideration is

$$G(X_1, X_2) = 5 - 0.5(X_1 - 0.1)^2 - X_2. \tag{9}$$

The components of the vector X, X_1 and X_2 are independent standard normal distribution (with 0 mean and unit standard deviation). This problem has clearly two design points which are provided in [21], $u_1^* = (-2.741, 0.965)$ and $u_2^* = (2.916, 1.036)$ using a gradient-based algorithm. It is the first design point that is the global optimum. The solution obtained by the algorithm used in this paper is closed to the solution u_1^*, it is obtained with a good accuracy.

The penalty coefficient retained for this search is $\lambda = 4.0$, and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 1.128 \times 10^{-15}$.

Method	Design point in space Λ	β	P_f
Kiureghian and Dakessian [21]	u^* = (−2.741, 0.965)	2.906	0.00183
PSO	u^* = (−2.74163, 0.962577)	2.9056	0.00183

5.2. Example # 2

This example has been studied in [22]. The limit state function in consideration is

$$G(X_1, X_2) = \exp(0.4(X_1 + 2) + 6.2) - \exp(0.3X_2 + 5.0) - 200, \tag{10}$$

where X_1 and X_2 are two independent standard normal random variables.

The penalty coefficient retained for this search is $\lambda = 0.1$, and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 6.19 \times 10^{-16}$.

Method	Design point in space Λ	β	P_f
Kim and Na [22]	Not available	2.668	0.00382
MC [22]	–	2.685	0.00363
PSO	u^* = (−2.54776, 0.92355)	2.7099	0.00337

5.3. Example # 3

This example has been studied in [22,23] using a response surface method and Monte-Carlo simulations. The limit state function in consideration is

$$G(X_1, X_2) = \exp(0.2X_1 + 1.4) - X_2, \tag{11}$$

where X_1 and X_2 are two independent standard normal random variables.

The penalty coefficient retained for this search is $\lambda = 7.2$, and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 3.94 \times 10^{-16}$.

The results obtained are presented in the following table:

Method	Design point in space Λ	β	P_f
Kim and Na [22]	Not available	3.399	0.000338
MC [22]	–	3.382	0.00038
CQ2RS [23]	$u^* = (-1.686, 2.892)$	3.47	0.000409
PSO	$u^* = (-1.688257, 2.893157)$	3.4971	0.000405

5.4. Example # 4

This example has been studied in [23] using a modern response surface method called CQ2RS. The structure under consideration is a cantilever beam with rectangular cross-section and is subjected to an uniformly distributed load. The limit state function in consideration is

$$G(X_1, X_2) = 18.46154 - 7.476923 \times 10^{10} \frac{X_1}{X_2^3}, \tag{12}$$

where X_1 and X_2 are two independent normal random variables defined as:

Variables	Mean	Standard deviation	COV (%)
X_1	0.001	0.0002	20.0
X_2	250	37.5	15.0

The penalty coefficient retained for this search is $\lambda = 0.8$, and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 1.7 \times 10^{-15}$.

The results obtained are presented in the following table:

Method	Design point in space Λ	Design point in space Ψ	β	P_f
CQ2RS [23]	$u^* = (0.586, -2.243)$	$x^* = (0.0011172, 165.8875)$	2.318	0.010225
RYFES/COMREL [23]	$u^* = (0.592, -2.255)$	$x^* = (0.0011184, 165.4375)$	2.331	0.009877
PSO	$u^* = (0.594605, -2.252998)$	$x^* = (0.001118921, 165.43723)$	2.3309	0.009879

5.5. Example # 5

This example has been studied by Kim and Na [22] using response surface method. The limit state function in consideration is

$$G(X_1, X_2, X_3) = X_1 - \frac{X_2}{X_3}, \tag{13}$$

where X_1, X_2, and X_3 are three independent normal random variables defined as:

Variables	Mean	Standard deviation	COV (%)
X_1 (MPa)	600.00	30.00	5.0
X_2 (N)	1000.00	33.00	3.3
X_3 (mm^2)	2.00	0.1	5.0

The penalty coefficient retained in this paper for the search is $\lambda = 0.2$ and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 4.81 \times 10^{-16}$.

The results obtained are presented in the following table:

Method	Design point in space Λ	Design point in space Ψ	β	P_f
Kim and Na [22]	Not available		2.232	0.01281
MC [22]	–		2.249	0.01226
PSO	u^* = (−1.483109, 0.8773503, −1.47722539)	x^* = (555.5067, 1028.95925, 1.8522774)	2.2697	0.01161

Now, let us see the impact of some parameters on the optimization; the interactions between them are not studied here, it would require too many simulations. The aim is to obtain some general guidelines in the use of the algorithm.

5.5.1. Number of calls of the fitness function

The number of calls of the objective to be optimized is $O(N_{iter} \times T_{pop} \times N_\lambda)$. Indeed, N_{iter} iterations are performed during the search; at each iteration, T_{pop} particles fitness are calculated. The algorithm is repeated N_λ times according the sequence λ_i defined in Section 3.2.2 to choose the appropriate value of λ ensuring the convergence. Let us notice that generally $N_\lambda < 10$; N_{iter} and T_{pop} predefined by the algorithm user as described in Section 4.

5.5.2. Population size sensitivity

All the parameters of the algorithm defined in Table 1 are kept constant, except the population size on which a sensitivity analysis is performed.

T_{pop}	u_1	u_2	u_3	β	P_f	CPU (s)
10	−1.365585	0.7925202	−1.6441211	2.27948346	0.01131914	0
20	−1.503937	0.8451922	−1.4754929	2.27007834	0.01160138	0.05
30	−1.475559	0.9037779	−1.4690817	2.26986561	0.01160783	0.05
40	−1.491182	0.8914286	−1.4607799	2.26983405	0.01160879	0.11
60	−1.493282	0.8660177	−1.4737793	2.26978043	0.01161042	0.16
70	−1.4831096	0.8773503	−1.47722539	2.26970317	0.01161276	0.26

From this example, it can be seen that there is no relevant difference between the results yielded by the algorithm using $T_{pop} = 30, 40, 60, 70$ except that the computing time is increased with T_{pop}. For $T_{pop} = 10$ the result obtained in term of failure probability is close to the one obtained with $T_{pop} = 70$ if it is considered only three digits after the dots. But the design points are not exactly the same showing that some additional simulations were necessary to reach exactly the optimum. The results are in accordance with $T_{pop} = 20$ used in [19]. Thus, for a problem on which computing time must be optimized, the population size may be optimized by choosing a moderate size.

5.5.3. Number of iterations sensitivity

All the parameters of the algorithm defined in Table 1 are kept constant, except the number of iterations on which a sensitivity analysis is performed.

N_{iter}	u_1	u_2	u_3	β	P_f	CPU (s)
100	−1.4599322	0.89028492	−1.49244334	2.26966878	0.01161381	0.05
200	−1.48391709	0.8747991	−1.47787600	2.26966977	0.01161378	0.06
300	−1.4831819	0.8772677	−1.47720203	2.26970329	0.01161276	0.11
400	−1.48310947	0.87735149	−1.47722474	2.26970312	0.01161277	0.27
900	−1.48310960	0.87735033	−1.47722539	2.26970318	0.01161276	0.22

From the results obtained with this problem, 300 iterations are sufficient to reach the optimal solution. With 100 and 200 simulations the results obtained are also acceptable, they are close to the optimal solutions if one considers, respectively, one digit and two digits.

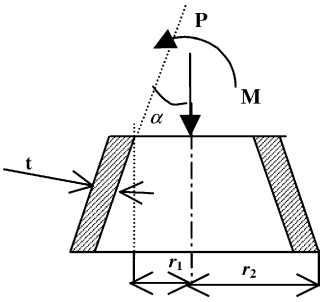
5.5.4. Partial conclusion

The computing time increases with T_{pop} and N_{iter}. If it is considered that the size, T_{pop}, of the population is chosen in the range [20; 100], which is quite sufficient, to optimize the computing time, the stop criteria can consist in stopping the search when one of the following conditions is met: (i) the maximum number of iterations is reached; (ii) after a given number of iterations there is no relevant improvement of the solution.

5.6. Example # 6

5.6.1. Description of the structure

Let us study a conical structure submitted to a compressive axial load P, and a bending moment M. The geometrical and mechanical features of the structure are presented in the following table, and are defined as independent normal variables:



	Variables	Mean	Deviation	COV (%)
X_1	E (MPa)	70000.00	3500.00	5.00
X_2	t (m)	0.00250	0.000125	5.00
X_3	α (rd)	0.524	0.010480	2.00
X_4	r_1 (m)	0.900	0.022500	2.50
X_5	M (N m)	80000.00	6400.00	8.00
X_6	P (N)	70000.00	5600.00	8.00

The main phenomena that can involve the failure of the structure are:

- the loss of strength of the structure,
- the buckling of the structure due to instability.

For the loss of strength, according to the large margin obtained in the analysis, this failure mode shall not be analyzed. It is only the buckling of the structure that will be analyzed under the combined solicitations. According to NASA space vehicles design rules [24] the buckling criterion is

$$\frac{P}{P_{\text{crit}}} + \frac{M}{M_{\text{crit}}} \geqslant 1, \tag{14}$$

P, M, respectively, axial load and bending moment loading the structure; P_{crit}, M_{crit}, respectively, critical axial load and bending moment (for buckling).

According to NASA [24]:

$$P_{\text{crit}} = \gamma \frac{2\pi E t^2 \cos^2\alpha}{\sqrt{3(1-\mu^2)}}, \tag{15}$$

$$M_{\text{crit}} = \eta \frac{\pi E t^2 r_1 \cos^2\alpha}{\sqrt{3(1-\mu^2)}}, \tag{16}$$

γ, η, enable to correlate theoretical results with experimental ones ($\gamma = 0.33$, $\eta = 0.41$ according to NASA [24]).

According to Eqs. (14)–(16), the limit state function of the structure is

$$G(X) = 1 - \frac{\sqrt{3(1-\mu^2)}}{\pi X_1 X_2^2 \cos^2 X_3}\left(\frac{X_6}{2\gamma} + \frac{X_5}{\eta X_4}\right). \tag{17}$$

5.6.2. Numerical results

The penalty coefficient retained in this paper for the search is $\lambda = 150.00$ and the control parameter is $|\lambda\xi(G(T^{-1}(u)))| = 7.9 \times 10^{-5}$.

i	1	2	3	4	5	6
u_i^*	−1.7983144	−4.13748603	0.043717421	−0.55844645	1.60862529	0.768704877
x_i^*	6.3706×10^{10}	0.0019828	0.524458159	0.88743495	90295.20	131496.39

The reliability index is $\beta = 4.883$ and the associated failure probability is $P_f = 5.232 \times 10^{-7}$ and the computing time is about 0.58 s.

6. Conclusion

The PSO algorithm is very efficient to solve global optimization problems with continuous variables. Its use in structural reliability field presents not only the advantage of its facility of implementation, but also the possibility to obtain the design point and the failure probability with a good accuracy. In addition, PSO is a zero order algorithm, for no derivative is necessary for its implementation. Several examples of the literature studied in this paper show that the results yielded are quasi-exact with respect to those yielded by MC and response surface methods. An application on a conical buckling is also performed. For a given problem, the computing time depends on the size of the initial population (70 in this paper) and the number of iterations (990 in this paper) performed to reach the results. The computing time is not too large and ranges form 0.05 to 0.6 s. for the examples studied in this paper on a PC 600 MHz with F77 code. Therefore, this algorithm can be considered as an additional efficient algorithm to those existing in the literature based on gradient method that do not insure the global optimum.

References

[1] Ditlevesen O, Madsen HO. In: Willey J, editor. Structural reliability methods. 1996. ISBN 0 471 96086 1.
[2] Melnik-Melnikov PG, Dekhtyaruk ES. Rare events probabilities estimation by "Russian Roulette and Splitting" simulation technique. Probabilist Eng Mech 2000;15:125–9.

[3] Abdo T, Rackwitz R. A new beta point algorithm for large time invariant reliability problems. In: Reliability and optimization of structures, Proceedings of the third WG 7.5 IFIP conference, 1990. p. 1–11.
[4] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks (Australia), 1995. p. 1942–8. ISBN 0-7803-2768-3/95.
[5] Eberhart R, Kennedy J. A new optimizer using particles swarm theory. In: Sixth international symposium on micro machine and human science, 1995. 0-7803-2676-8/95.
[6] Kennedy J. In: The particles swarm: social adaptation of knowledge. conference 0-7803-3949-5/97 IEEE; 1997. p. 303–8.
[7] Rubinstein RY. Simulation and Monte Carlo methods. New York: Wiley; 1981.
[8] Zhao Y-G, Ono AT. A general procedure for first/second order reliability methods (FORM/SORM). Struct Safety 1999;21:95–112.
[9] Au S-K, Beck JL. Estimation of small failure probabilities in high dimension by subset simulation. Probabilist Eng Mech 2001;16:263–77.
[10] Roy D, Dasgupta T. A discretizing approach for evaluating reliability of complex systems under stress-strength model. IEEE Trans Reliab 2001;50(2):145–50.
[11] Rackwitz R. Reliability analysis – a review. Struct Safety 2001;23:365–3951.
[12] Castillo E, Fernadez-Cantelli A, Minguez R. Computing failure probabilities. Application to reliability analysis. Reliab Eng System Safety 2002;77:131–41.
[13] Nataf A. Détermination des distributions dont les marges sont données. Compte Rendu académie des sciences, Paris, France (225); 1962.
[14] Minoux M. In: Dunod, editor. Programmation mathématique, vol. 1. Paris; 1983.
[15] Fiacco AV, McCormick GP. Non linear programming: sequential unconstrained minimization techniques. New York: Wiley; 1968.
[16] Ashgar B. Nonlinear constrained optimization using penalty function. In: Mathematica developer Conference proceedings; 1999.
[17] Coello CAC. An empirical study of evolutionary techniques for multi-objective optimization in engineering design. PhD dissertation, Tulane University; 1996.
[18] Schoenauer M, Michalewicz Z. Evolutionary computation: an introduction. Control Cybernet 1997;26(3):307–38.
[19] Shi Y, Eherhart RC. Parameter selection in particle swarm optimization. Author manuscript. Personal note of the author: www.engr.iup.edu/~shi/PSO.
[20] Elegbede C, Adjallah K. Availability allocation to repairable systems with genetic algorithms: a multi-objective formulation. Reliab Eng System Safety 2003;82:319–30.
[21] Kiureghian AD, Dakessian T. Multiple design points in structural reliability. Structural safety and reliability. Rotterdam: Balkema. ISBN 9054109785.
[22] Kim S-H, Na S-W. Response surface method using vector projected sampling point. Struct Safety 1997;19(1):3–19.
[23] Gayton N, Bourinet JM, Lemaire M. CQ2RS: a new statistical approach to the response surface method for reliability analysis. Struct Safety 2003;25(1):99–121.
[24] NASA SP-8019 Buckling of thin-walled truncated-cones. NASA space vehicle design criteria (structures), September, 1968.