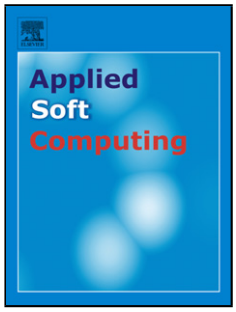


Title: Improved Accelerated PSO Algorithm for Mechanical Engineering Optimization Problems

Author: Najeh Ben.Guedria

PII: S1568-4946(15)00690-0
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2015.10.048
Reference: ASOC 3291

To appear in: *Applied Soft Computing*

Received date: 10-5-2015
Revised date: 14-10-2015
Accepted date: 19-10-2015

Please cite this article as: N. Ben.Guedria, Improved Accelerated PSO Algorithm for Mechanical Engineering Optimization Problems, *Applied Soft Computing Journal* (2015), http://dx.doi.org/10.1016/j.asoc.2015.10.048

List of authors:

Najeh Ben.Guedria

Higher Institute of Transport and Logistics, University of Sousse, Tunisia.

Corresponding Author

Najeh Ben.Guedria

Address: P.O.BOX 4023, Riadh City, Sousse-Tunisia.

Phone : (216) 54650040

Emails : najeh.benguedria@istls.rnu.tn

najehbenguedria@gmail.com

Title: Improved Accelerated PSO Algorithm for Mechanical Engineering Optimization Problems

Highlights

- A new Improved Accelerated Particle Swarm Optimization algorithm is proposed (IAPSO)
- Individual particles memories are incorporated in order to increase swarm diversity.
- Balance between exploration and exploitation is controlled through two selected functions.
- IAPSO outperforms several recent meta-heuristic algorithms, in terms of accuracy and convergence speed.
- New optimal solutions, for some benchmark engineering problems, are obtained.

1. Introduction

A large number of optimization problems out coming from numerous engineering and scientific fields are generally formulated as nonlinear constrained optimization problems. Frequently, the objective function and constraints are neither smooth nor continuous. Hence, deterministic methods such as the steepest descent method, the Newton method and the quasi Newton method, are inefficient, inaccurate and unstable when used to solve these complex optimization problems. During the last decades, a huge number of meta-heuristic algorithms have been proposed in order to overcome these defects. Most of these algorithms are inspired form physical and natural phenomena and based on a combination of several rules and randomness. Such algorithms are Simulated Annealing (SA) algorithm [1], Genetic Algorithm (GA) [2,3], Particle Swarm Optimization (PSO) [4,5], Ant Colony Optimization (ACO) [6], Harmony Search (HS) [7], Artificial Bee Colony (ABC) [8], Firefly Algorithm (FA) [9], etc. and more recently League Championship Algorithm (LCA) [10], Water Cycle Algorithm (WCA) [11] and Mine Blast Algorithm (MBA) [12].

Among all these meta-heuristics, PSO algorithm has attracted scientific researchers due to its efficiency to solve complex optimization problems, arising in various fields, and ease of implementation. PSO algorithm was for the first time introduced by Eberhart and Kennedy [4, 5] and presented as a population based algorithm which mimics the behavior of a swarm of birds or a school of fish. In PSO algorithm, each member of the swarm, called particle, is characterized by a position and a velocity vectors. A particle adjusts, during the optimization process, its velocity with respect to its own experience and the experience of the whole swarm. The particle position is then updated according to its previous position and its new velocity. However, despite its advantages, in some complex and intricate optimization problems, PSO can be trapped in local optima and can in a later period of the iterative process have a weak convergence rate. In order to overcome these limitations, a great effort of scientific researchers has therefore been devoted to the development of diverse approaches and strategies to improve the PSO algorithm. These based PSO algorithms have therefore been applied on diverse scientific and engineering problems. Early modifications of the original PSO, are conducted by its own creators and their collaborators, leading to two principal models which are the inertia weight model [13, 14] and the constriction factor model [15]. The main goal of both approaches is to balance exploration and exploitation, through searching process, and then improving convergence rate and solution quality.

Next, several other based PSO algorithms have been developed and applied on benchmark and real life optimization problems. Angel *et al* [16] proposed a modified PSO algorithm, called particle evolutionary swarm optimization (PESO). Two perturbation operators are successively added to the standard PSO in order to prevent premature convergence and improving diversity. He and Wang [17] presented a Co-evolutionary PSO, named CPSO, for solving constrained engineering optimization problems. Two kinds of swarms are employed and evolved simultaneously using the standard PSO algorithm. The first kind of multiple swarms is used to evolve decision solutions, while the second one of single swarm is used to adapt penalty factors for solution evaluation. A modified co-evolutionary particle swarm optimization using Gaussian distribution (CPSO–GD) was developed by Krohling and Coelho [18]. The authors suggested a Gaussian probability distribution to generate the accelerating coefficients of the PSO with constriction factor [15]. Aguirre *et al.* [19] extended the PESO algorithm [16] by introducing a new ring neighborhood structure and a technique based on feasibility and sum of constraints violation to handle constraints. The resulting algorithm is named: constrained optimization via PSO algorithm: COPSO. He and Wang [20] presented a hybrid PSO algorithm (HPSO) with a feasibility-based rule to solve constrained optimization problem. In this work, the PSO algorithm is hybridized with the simulated annealing algorithm (SA) to avoid premature convergence. Worasucheep [21] introduced a Constrained Particle Swarm Optimization with Stagnation Detection and Dispersion (PSO-DD) based on the standard PSO and using some techniques to handle stagnation and constraints. Cagnina *et al.* [22] used the standard PSO algorithm with a simple technique to handle constrains. The algorithm is called: Simple Constrained PSO (SiC-PSO). Wang and Yin [23] proposed a Ranking Selection-based PSO (RSPSO) to solve engineering design problems with mixed variables. The authors have integrated a ranking selection scheme into the standard PSO to control the search behavior of a swarm in different search phases and on categorical variables. Yildiz [24] developed a hybrid approach based on PSO and receptor editing property of immune system (PSRE). The authors of [25] presented a modified particle swarm optimization (MPSO) which included the heuristic crossover derived from genetic algorithm to compete with standard PSO in order to improve local search. Zahara and Kao [26] hybridized PSO with a well-known local search namely Nelder-Mead Simplex method and applied it for solving constrained optimization problems. This hybrid version was named as NM-PSO. Liu *et al.* [27] proposed a hybrid algorithm called PSO-DE, which incorporates standard PSO with differential evolution (DE) to solve constrained numerical and engineering optimization problems. The use of DE is mainly to fight stagnation problem. Coelho [28]

introduced an improved quantum-behaved PSO (QPSO) approaches using mutation operator with Gaussian probability distribution. The algorithm is called G-QPSO. Gaussian mutation operator is used instead of random sequences in QPSO in order to prevent premature convergence to local optima. For solving real life engineering problems, recently, Davoodi *et al.* [29] derived a hybrid algorithm combining Improved Quantum-behaved Particle Swarm Optimization (IQPSO) with Simplex algorithms: (IQPSOS) for solving load flow problems. The Simplex method is used as a local search to fine tune the obtained solution from QPSO. Similarly, in [30], PSO and DE are used for the parametric identification of seismic isolators. A complete survey of the state of the art in particle swarm optimization can be found in [31 - 33].

All the above developed variants of the PSO algorithm propose to modify acceleration coefficients, add some operators or combine other existing algorithms with PSO to produce new hybrid algorithms. The goal of these variants is mainly to improve diversity, prevent premature convergence and increase convergence rate. However, the majority of these algorithms are time-consuming, due to integration of complex procedures, and need a large number of iterations to reach the optimum or a near-optimum solution. In order to reduce computation effort and improve the convergence of the PSO, Yang [34] has proposed a simplified based PSO model called accelerated particle swarm optimization (APSO). The main modification is the removal of particles velocities vectors from the standard PSO model, hence particles of the swarm are characterized only by their position vectors. In addition, Yang has proposed to replace the contribution of particles personal best positions, which are used to increase diversity, by some randomness. However, despite the successful test of this algorithm on some benchmark functions and its high convergence speed, the APSO suffers from premature convergence and weak diversity, particularly when handling highly nonlinear optimization problems. Motivated by these disadvantages, in the present study we propose some modifications on the APSO in order to alleviate the problem of premature convergence and to improve swarm diversity. These modifications are the incorporation of the individual particles' memories, in order to increase swarm diversity, and the introduction of two selected functions to control balance between exploration and exploitation during search process. The resulting algorithm is called improved accelerated particle swarm optimization (IAPSO).

The remainder of this paper is organized as follows. Section 2, presents a brief review of the standard PSO algorithm. In section 3, the APSO [34] is presented and discussed. The proposed IAPSO algorithm and related formulations are detailed in Section 4. Section 5 of the

paper covers performance evaluation of the proposed algorithm through six benchmark mechanical engineering optimization problems. The last section provides a clear conclusion of the study.

2. Basic Particle Swarm Optimization (PSO)

Particle swarm optimization is a population based algorithm developed by Eberhart and Kennedy in 1995 [4, 5]. The PSO algorithm is inspired by social behavior of bird flocking, or fish schooling. Meanwhile, it gained popularity among scientific researches thanks to its ability to quickly converge to a reasonably good solution and its simplicity of implementation.

Mathematically, PSO can be described as follows: Consider a search space $D \in \Re^d$ and f an objective function. d is the number of design variables. As mentioned above, PSO is a population-based algorithm. The population is called the swarm and its individuals are called the particles. The swarm is defined as a set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}^T$ of N particles. Each particle is assigned a position vector, defined as $\mathbf{x}_i = \{x_{i1}, x_{i2}, ..., x_{id}\} \in D$, $i = 1, 2, ..., N$, and a velocity vector denoted as $\mathbf{v}_i = \{v_{i1}, v_{i2}, ..., v_{id}\}$.

In order to allow particles visiting all search space, each particle, updates its velocity according to the memory gained so far. In fact, PSO maintains a memory set $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}^T$ where $\mathbf{p}_i$ is the best position ever visited by the i^{th} particle. In addition, as PSO is based on information exchange allowing particles to mutually communicate their experiences, the best position ever visited by all particles is shared. Let g be the index of the global best position in $\mathbf{P}$ then the global best position is defined as $\mathbf{p}_g = \arg\min_i (\mathbf{p}_i)$.

At each iteration step t, the particle velocity and position are updated successively using respectively the two following equations:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha\, r_1 \left(\mathbf{p}_i^t - \mathbf{x}_i^t\right) + \beta\, r_2 \left(\mathbf{p}_g - \mathbf{x}_i^t\right) \tag{1}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{2}$$

where r_1 and r_2 are two uniformly random numbers independently distributed in the interval $[0,1]$ which are used to maintain the diversity of the swarm. α and β are respectively the cognitive and the social parameters (or acceleration parameters) which drive particles

towards local and global best positions, usually set as $\alpha \approx \beta \approx 2$. However, some researchers report that we might even better choose a larger cognitive parameter α than a social parameter β but respecting $\alpha + \beta \leq 4$ [15].

3. Accelerated Particle Swarm Optimization

The accelerated particle swarm optimization (APSO) is a simplified version of the standard PSO developed by Yang [34]. While PSO algorithm uses both the particle best $\mathbf{p}_i$ and the global best $\mathbf{p}_g$ to update the velocity $\mathbf{v}_i$, as shown in Eq. (1); the APSO algorithm uses only the global best position. According to Yang, this simplification is due to the fact that individual best is used to increase the diversity in the quality solutions and hence, can be replaced simply by some randomness [34]. Consequently, the velocity of the i^{th} particle is updated as follows:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha\,\boldsymbol{\varepsilon} + \beta\left(\mathbf{p}_g - \mathbf{x}_i^t\right) \tag{4}$$

where $\boldsymbol{\varepsilon}$ is a random vector uniformly distributed in the range [0,1]. The position of the i^{th} particle is updated using Eq. (2). Moreover, in order to increase the convergence even further, the update of the location is simply written in a single step as follows:

$$\mathbf{x}_i^{t+1} = (1-\beta)\mathbf{x}_i^t + \beta\,\mathbf{p}_g + \alpha\,\boldsymbol{\varepsilon} \tag{5}$$

Typically, the parameter $\alpha = 0.1L \sim 0.5L$ where L is the scale of each variable, while $\beta = 0.1 \sim 0.7$ is sufficient for most applications [35].

As can be seen, the APSO is described only by a single equation (Eq. (5)). Thus, there is no need to deal with velocity. In addition, Yang has suggested a further improvement of the APSO by reducing the randomness as iterations proceed. This is done by introducing the parameter α as a monotonically decreasing function such as:

$$\alpha = \alpha_0\, e^{-\gamma t} \tag{6}$$

or

$$\alpha = \alpha_0\, \gamma^t, \qquad (0 < \gamma < 1) \tag{7}$$

where $\alpha_0 = 0.5 \sim 1$ is the initial value of the randomness parameter and γ is a control parameter.

4. The proposed Improved APSO algorithm (IAPSO)

According to [34], in the case of highly nonlinear and multimodal optimization problems, the APSO algorithm suffers from premature convergence due to its weak diversity. Moreover, the APSO algorithm lacks memory since the update of particle position does not consider the memory gained by each particle during the search process. To overcome these disadvantages, three modifications will be substituted to the APSO algorithm.

The first modification is to replace the current position $\mathbf{x}_i$ by the best position $\mathbf{p}_i$ in Eq. (5), providing a memory to the APSO algorithm. We can never underestimate the significance of using local memory. Indeed, using personal bests of all particles in the population, which represent the memory of the swarm, will enhance diversity of the swarm. The particles current positions, in this case, are regarded as a set of explorers whose purpose is the exploration of search space. This will allow particles to explore broadly the whole of the search space, mainly promising sub-regions, and alleviating premature convergence.

Similarly, the second change is also included to improve diversity of the swarm. In fact, the random vector $\boldsymbol{\varepsilon}$, Eq. (5), is substituted by the line vector $\mathbf{R}_i$ of the matrix $\mathbf{R} = \left[\mathbf{R}_1 \quad \mathbf{R}_2 \quad ... \quad \mathbf{R}_N\right]^T$. The matrix $\mathbf{R}$ is a ($N \times d$) normally distributed random matrix computed at each iteration t. Each j^{th} column $\left(j = 1,...,d\right)$ of $\mathbf{R}$ is randomly generated with a mean zero and a standard deviation σ_j, where σ_j is the standard deviation of the j^{th} design parameter of all the particles best positions $\mathbf{P} = \left\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\right\}^T$. Fig. 1 shows schematically the computation of σ_j for N particles best positions. The approach of taking the standard deviation for the normal distribution is used in [36]. The goals of using $\mathbf{R}_i$ is to improve diversity of the swarm and to take part of the collective memory of all particles since it is computed based on particles best positions.

The standard APSO uses two constant parameters α and β, Eq. (5), as described in the above section. Whereas, in the present work, we propose as a third modification to use these two parameters as two monotonically functions with respect to the iteration counter t.

The function $\alpha(t)$ is used to control particles exploration of the search space. It is considered as a decreasing step function updated during the iterative process repeatedly after each *s* iterations. The mathematical expression of $\alpha(t)$ is as follows:

$$\alpha(t) = \alpha_{\max} - \left(\frac{\alpha_{\max} - \alpha_{\min}}{t_{\max}} \right) t \tag{8}$$

where $t_{\max}$ is the maximum iteration number. As can be noted, during the iterative process $\alpha(t)$ decreases between the upper and lower limits $\alpha_{\max}$ and $\alpha_{\min}$, respectively. Indeed, a large value of α favor global exploration, while a small value tends to enable exploitation. The use of $\alpha(t)$ as a step function allows the diversity of the swarm to be preserved during the *s* iterations and hence, helps to avoid premature convergence. Furthermore, contrarily to the APSO, where α must be scaled with respect to design parameters, in IAPSO, there is no need to scale it.

In the same way, the function $\beta(t)$ is considered as an increasing function expressed as follows:

$$\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \sin\left(\frac{\pi}{2} \frac{t}{t_{\max}} \right) \tag{9}$$

where $\beta_{\min}$ and $\beta_{\max}$ are respectively minimal and maximal values of $\beta(t)$ at first and last iterations. The function $\beta(t)$ is employed to control balance between global and local explorations during the search process. In fact, a small value of β gives height impact to personal best $\mathbf{p}_i$ $(i = 1, \ldots, N)$ on the swarm allowing global exploration, whereas, a large value gives more influence of the global best $\mathbf{p}_g$ on the new particles positions, which favors solutions refinement. It is worth pointing that the function $\beta(t)$ has been chosen after numerical simulations showing its superiority compared to several other tested functions.

Taking into account of the above modifications, mathematically the fundamental equation of the proposed algorithm is as follows:

$$\mathbf{x}_i^{t+1} = \left(1 - \beta(t)\right) \mathbf{p}_i^t + \beta(t) \mathbf{p}_g^t + \alpha(t) \mathbf{R}_i^t \tag{10}$$

As can be noted, the IAPSO algorithm uses a simple equation (Eq. 10) to update the current position of each particle in the swarm. The position vector is updated based on the memory

gained by each particle as well as the knowledge gained by the swarm as a whole, both are controlled by the function β. In addition, some randomness, generated based on the current particles best positions, is substituted to enhance diversity of the population and also controlled by the function α. Thus, based on the personal memory and the social behavior of the swarm, each particle position is updated accordingly, allowing particle to adapt to its environment by returning to promising regions of the search space previously discovered, and searching for better positions over time.

4.1 Constraint handling for the IAPSO algorithm

Similar to other stochastic optimization algorithms, the IAPSO is defined for unconstrained problems. However, during search process, particles may violate either the limits of the design variables or the problem specific constraints. For any particle with a position $\mathbf{x}_i$ exceeding the boundary of the variable range, its position is reset to the upper (or lower) boundary value. On the other hand, to take into account constraints violation in this work, the penalty function method [37] is adopted thanks to its simple principle and easy implementation, especially for continuous constrained problems.

Generally, a constrained optimization problem is described as follows:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{Subject to:} \quad g_k(\mathbf{x}) \le 0, \quad k = 1,...,m \end{aligned} \tag{11}$$

where f is the objective function and g_k is the k^{th} inequality constraint.

Integration of penalty functions into the objective function will transform the above constrained problem to an unconstrained one. The penalized objective function f_p is then written as follows:

$$f_p(\mathbf{x}) = f(\mathbf{x}) + \lambda \sum_{k=1}^{m} \delta_k \left[g_k(\mathbf{x}) \right]^2 \tag{12}$$

where $\lambda > 0$ (e.g. $\lambda = 10^{15}$) is a penalty factor and $\begin{cases} \delta_k = 1 & \text{if constraint } g_k \text{ is violated} \\ \delta_k = 0 & \text{if constraint } g_k \text{ is satisfied} \end{cases}$.

4.2 Implementation of the IAPSO algorithm

The step-wise procedure for the implementation of the proposed IAPSO algorithm is given as follows:

Step 1: Define the optimization problem and initialize the optimization parameters.

- The problem is defined as shown in Eq (11) and transformed as illustrated in Eq (12), number of design variables (d), and limits of design variables (U_b, L_b).
- Initialize the population size (N), maximum number of iterations (t_{max}), the control functions $\alpha = \alpha_{\max}$, $\beta = \beta_{\min}$ and the iteration counter $t = 0$.

Step 2: Initialize the population.

- Generate a random population $\mathbf{X}^0 = \left\{\mathbf{x}_1^0, \mathbf{x}_2^0, ..., \mathbf{x}_N^0\right\}^T$ according to the population size and number of design variables. For the IAPSO, initial location of the i^{th} particle is given as:

$$\mathbf{x}_i^0 = L_b + rand\left(U_b - L_b\right) \tag{13}$$

  where $rand$ is a uniformly distributed random number between 0 and 1.

- Set initial particles best positions as initial population: $\mathbf{p}_i^0 = \mathbf{x}_i^0$ for $i = 1, \ldots, N$.
- Find the global best position $\mathbf{p}_g$ such that $f_p\left(\mathbf{p}_g\right) = \min_i\left(f_p\left(p_i^0\right)\right)$.

Step 3: Repeat until the stopping criteria is met: $\left(t = t_{\max}\right)$.

- Update iteration counter $\left(t = t + 1\right)$, control functions α and β using respectively Eq (8) and Eq (9). Calculate $\mathbf{R}^t$ as illustrated in Fig (2).
- Update particle position $\mathbf{x}_i^t$ using Eq (10) and calculate its fitness $f_p\left(\mathbf{x}_i^t\right)$.
- Update the best particle position $\mathbf{p}_i^t$ at current iteration (t) and global best particle position $\mathbf{p}_g$.

5. Parameter settings and analysis of results

In this section, a set of six benchmark mechanical engineering design optimization problems, frequently used by the specialized literature, have been chosen to test the performance of the proposed IAPSO algorithm. The considered benchmark problems include objective functions and constraints of various types and nature (quadratic, cubic, polynomial and nonlinear) with

several numbers and types of design variables (continuous, mixed, discrete and integer). The mathematical formulations of the test problems can be consulted in the appendix of this paper.

The obtained optimization results have been compared with other well-known optimizers particularly the APSO algorithm [34]. It is worth pointing that numerical results obtained by APSO algorithm are carried out during this study. The APSO parameters are tuned according to those advised in [34]. Results were compared in terms of statistical results and number of function evaluations (NFEs). In this paper, the computational cost which is considered as the best NFEs corresponding to the obtained best solution, is calculated by the product of the number of swarm's particles and the maximum number of iterations (i.e. NFEs = $N \times t_{max}$).

The proposed algorithm was coded in MATLAB programming software and the simulations and numerical solutions were run on a T5870 @ 2 GHz Intel Core™ 2 Duo process with 4 GB Random Access Memory (RAM). The task of optimizing each mechanical design problems was carried out by using 25 independent runs. The user parameters of IAPSO, for considered engineering optimization problems, are presented in Table1. These parameters are empirically selected after numerous experiments and fine-tuned using the following guidelines:

- The control function β is defined such as $\beta_{\min} \le \beta \le \beta_{\max}$. Typically $\beta_{\min} \approx 0.1$ to 0.3 and $\beta_{\max} \approx 0.5$ to 0.9. A lower value of β at first iterations allows exploration and then gradually increases during iterations to a higher value for more exploitation. In other word, β controls the influence of the global best particle and personal best on the current solution.
- The decreasing function α, which controls diversification of the swarm, is bounded such as $\alpha_{\min} \le \alpha \le \alpha_{\max}$. Typically, $\alpha_{\max} \approx 0.5$ to 2 and $\alpha_{\min} \approx 0.2$ to 0.6. A large value of α, at first iterations, allows more exploration of the search space and a lower value is used to refine solutions quality.
- To maintain a constant level of exploration, during optimization process, the parameter s is used. Typically $1 \le s \le 5$.
- The size of the swarm N and the maximum number of iterations t_{max} depend on the complexity of the optimization problems. In fact, for highly complex problems, a choice of 20-50 particles and 200-400 iterations may be sufficient. However, for

simple to medium complex problem a reasonable choice is to set 10-25 particles and 10-50 iterations.

Table1
User parameters used for IAPSO for engineering problems.

Problems	N	t_{max}	β_{max}	β_{min}	α_{max}	α_{min}	s
Welded beam	50	250	0.6	0.1	0.9	0.6	3
Pressure Vessel	25	300	0.6	0.1	1.5	0.5	5
Speed Reducer case 1&2	30	200	0.7	0.1	0.9	0.8	1
Spring Design	10	200	0.5	0.2	1.0	0.6	5
Gear train	20	40	0.9	0.2	0.6	0.2	1
Multiple disk clutch brake	40	10	0.9	0.2	1.6	0.6	2

5.1 Welded beam design optimization problem

This problem aims to design a welded beam for minimum cost, subject to some constraints [38]. Fig. 2 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The objective is to find the minimum fabrication cost, considering four design variables: $x_1,...,x_4$ and constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, and end deflection on the beam δ.

The algorithms earlier used to optimize this problem include: genetic algorithm (GA) based co-evolution model GA1 [39], GA through the use of dominance-based tour tournament selection GA2 [40], hybrid genetic algorithm (HGA) [41], CPSO-GD [18], HPSO [20], NM–PSO [26], cultural algorithms with evolutionary programming CAEP [42], society and civilization algorithm SC [43], differential evolution DE [44] and more recently LCA [10], WCA [11] and MBA [12].

Table 2 shows the comparison of the best solutions for previously reported studies and the proposed algorithm in terms of design variables, function values and constraints accuracy. The statistical optimization results for reported algorithms are given in Table 3. From Table 3, the proposed algorithm stably detects the best known solution recently obtained by LCA and strongly dominates the standard APSO algorithm [34]. Moreover, in terms of statistical results, IAPSO offered better results using fewer number of function evaluations (NFEs) compared to all other considered algorithms. However, the best solution is obtained with a standard deviation value (SD) greater than that given by LCA. It is worth pointing that the best solution obtained by NM-PSO violates the third constraint and then cannot be compared to other best solutions.

Table 2
Comparison of the best solution obtained from various algorithms for the welded beam problem.

DV	GA2[40]	CPSO[17]	CAEP[42]	HGA[41]	NM-PSO[26]	WCA[11]	MBA[12]	APSO[34]*	IAPSO
x_1	*0.205986*	*0.202369*	*0.205700*	*0.205700*	*0.205830*	0.205728	0.205729	0.202701	0.2057296
x_2	3.471328	3.544214	3.470500	3.470500	3.468338	3.470522	3.470493	3.574272	3.47048866
x_3	9.020224	9.048210	9.036600	9.036600	9.036624	9.036620	9.036626	9.040209	9.03662391
x_4	0.206480	0.205723	0.205700	0.205700	0.20573	0.205729	0.205729	0.2059215	0.20572964
$g_1(x)$	-0.103049	-13.655547	1.988676	1.988676	-0.02525	-0.034128	-0.001614	-117.467062	-1.05E-10
$g_2(x)$	-0.231747	-78.814077	4.481548	4.481548	-0.053122	-3.49E - 05	-0.016911	-51.712981	-6.91E-10
$g_3(x)$	-5e-04	-3.35E-03	0.000000	0.000000	0.000100	-1.19e-06	-2.40E-07	-0.003221	-7.66E-15
$g_4(x)$	-3.430044	-3.424572	-3.433213	-3.433213	-3.433169	-3.432980	-3.432982	-3.421741	-3.43298378
$g_5(x)$	-0.080986	-0.077369	-0.080700	-0.080700	-0.080830	-0.080728	-0.080729	-0.077701	-0.080729639
$g_6(x)$	-0.235514	-0.235595	-0.235538	-0.235538	-0.235540	-0.235540	-0.235540	-0.235571	-0.235540323
$g_7(x)$	-58.646888	-4.472858	2.603347	2.603347	-0.031555	-0.013503	-0.001464	-18.367012	-5.80E-10
$f(x)$	1.728226	1.728024	1.724852	1.724852	1.724717	1.724856	1.724853	1.736193	**1.7248523**

(*) numerical results obtained by APSO algorithm are carried out during this study.

Table 3
Comparison of the statistical results obtained from different optimization engines for the welded beam problem.

Method	**Worst**	**Mean**	**Best**	**SD**	**NFEs**
GA1[39]	1.785835	1.771973	1.748309	1.12E-02	900,000
GA2[40]	1.993408	1.792654	1.728226	7.47E-02	80,000
CAEP[42]	3.179709	1.971809	1.724852	4.43E-01	50,020
CPSO[17]	1.782143	1.748831	1.728024	1.29E-02	240,000
HPSO[20]	1.814295	1.749040	1.724852	4.01E-02	81,000
PSO-DE[27]	1.724852	1.724852	1.724852	6.70E-16	66,600
NM-PSO[26]	1.733393	1.726373	1.724717	3.50E-03	80,000
SC[43]	6.399678	3.002588	2.385434	9.6E-01	33,095
DE[44]	1.824105	1.768158	1.733461	2.21E-02	204,800
WCA[11]	1.744697	1.726427	1.724856	4.29E-03	46,450
LCA[10]	1.7248523	1.7248523	1.7248523	7.11E-15	15,000
MBA[12]	1.724853	1.724853	1.724853	6.94E-19	47,340
APSO[32]*	1.993999	1.877851	1.736193	0.076118	50,000
IAPSO	**1.7248624**	**1.7248528**	**1.7248523**	**2.02E-06**	**12,500**

(NFEs) and (SD) stand for number of function evaluations and standard deviation, respectively.

5.2 Pressure Vessel design optimization problem

As a second example, the design optimization of the cylindrical pressure vessel capped at both ends by hemispherical heads (Fig. 4) is considered [45, 46]. The vessel will be designed for a working pressure of 3000 psi and a minimum volume of 750 ft^3 regarding the provisions of ASME boiler and pressure vessel code. The objective is to minimize the total manufacturing cost of the vessel including materials, forming and welding costs. There are four design variables: Thickness of the shell (x_1), thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4). The variables x_1 and x_2 are discrete values which are integer multiples of 0.0625 and x_3 and x_4 are continuous values.

To solve this mixed nonlinear problem we round the value of x_1 and x_2 to their integer part and multiply them by 0.0625. The proposed IAPSO algorithm was applied to the pressure vessel optimization problem and the optimal results were compared to earlier methods as shown in Table 4. This problem has been solved previously using GA1, GA2, CPSO, HPSO, NM–PSO, G-QPSO, QPSO, PSO-DE, co-evolutionary differential evolution CDE [47], LCA, WCA and MBA and compared with the proposed IAPSO as given in Table 5.

As can be seen from Table 4, results indicate that IAPSO is superior to other optimizers in term of the best solution. Furthermore, the proposed algorithm is the most stable algorithm reporting always the smallest value of the standard deviation (SD), the most dependable algorithm in terms of the mean and the worst performance as shown in Table 5. Moreover, IAPSO converge to the optimal solution with the lowest NFEs compared to all other methods. It is worth pointing that solutions obtained respectively by NM-PSO, WCA and MBA are infeasible (see Table 4) since the design variables x_1 and x_2 haven't discrete values. Furthermore, the best solutions reported by NM-PSO and HPSO violate constraints g_1, g_2 and g_3 respectively. To conclude, we can say that IAPSO is the most efficient algorithm for optimizing the design features of a pressure vessel. Fig. 5 illustrates the function values with respect to the number of iterations for the pressure vessel design optimization problem. As before, IAPSO shows a good converging behavior, thanks to the efficiency of search model, despite the infeasibility of the initial population.

Table 4
Comparison of the best solution obtained from various studies for the pressure vessel problem.

DV	CDE[47]	GA1[39]	CPSO[17]	HPSO[20]	NM-PSO[26]	G-QPSO[28]	WCA[11]	MBA[12]	APSO[34]*	IAPSO
x_1	0.8125	0.8125	0.8125	0.8125	0.8036	0.8125	0.7781	0.7802	0.8125	**0.8125**
x_2	0.4375	0.4375	0.4375	0.4375	0.3972	0.4375	0.3846	0.3856	0.4375	**0.4375**
x_3	42.0984	42.0974	42.0913	42.0984	41.6392	42.0984	40.3196	40.4292	42.0984	**42.0984**
x_4	176.6376	176.6540	176.7465	176.6366	182.4120	176.6372	200.0000	198.4964	176.6374	**176.6366**
$g_1(x)$	- 6.67E- 07	-2.01E- 03	- 1.37E - 06	-8.80E -07	3.65E-05	-8.79E- 07	-2.95E-11	0	-9.54E-7	**-4.09E-13**
$g_2(x)$	-3.58E- 02	-3.58E- 02	-3.59E-04	-3.58E- 02	3.79E-05	-3.58E-02	-7.15E-11	0	-3.59E-2	**-3.58E-2**
$g_3(x)$	-3.705123	-24.7593	-118.7687	3.1226	-1.5914	- 0.2179	-1.35E -06	-86.3645	-63.3626	**-1.39E-07**
$g_4(x)$	-63.3623	- 63.3460	-63.2535	-63.3634	-57.5879	-63.3628	-40.0000	-41.5035	-0.9111	**-63.3634**
$f(x)$	6059.7340	6059.9463	6061.0777	6059.7143	5930.3137	6059.7208	5885.3327	5889.3216	6059.72418	**6059.71433**

Table 5
Comparison of statistical results given by different optimizers for the pressure vessel problem.

Method	Worst	Mean	Best	SD	NFEs
GA1[39]	6308.4970	6293.8432	6288.7445	7.4133	900,000
GA2[40]	6469.3220	6177.2533	6059.9463	130.9297	80,000
CPSO[17]	6363.8041	6147.1332	6061.0777	86.4500	240,000
HPSO[20]	6288.6770	6099.9323	6059.7143	86.2000	81,000
NM-PSO[26]	5960.0557	5946.7901	5930.3137	9.1610	80,000
G-QPSO[28]	7544.4925	6440.3786	6059.7208	448.4711	8,000
QPSO[28]	8017.2816	6440.3786	6059.7209	479.2671	8,000
PSO[12]	14076.3240	8756.6803	6693.7212	1492.5670	8,000
CDE[47]	6371.0455	6085.2303	6059.7340	43.0130	204,800
WCA[11]	6590.2129	6198.6172	5885.3327	213.0490	27,500
LCA[10]	6090.6114	6070.5884	6059.8553	11.37534	24,000
MBA[12]	6392.5062	6200.64765	5889.3216	160.34	70,650
APSO[34]*	7544.49272	6470.71568	6059.7242	326.9688	200,000
IAPSO	**6090.5314**	**6068.7539**	**6059.7143**	**14.0057**	**7,500**

5.3 Speed Reducer design optimization problem

The weight of the speed reducer [48] as shown in Fig. 6 is to be minimized subject to constraints on bending stress of the gear teeth, surfaces stress, transverse deflections of the shafts and stresses in the shafts. The design variables $(x_1,...,x_7)$ are respectively the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings and the diameter of the first and second shafts. The third variable is integer, the rest of them are continuous. This is an example of a mixed integer optimization problem. The mathematical formulation of this complex problem [49] having 11 constraints makes the search of feasible solutions very hard. Again two cases of the problem are investigated where the only difference between them is in the allowable range for variable x_5.

The speed reducer problem (case 1) previously was optimized using COPSO, SiC-PSO, LCA, modified bacterial foraging optimization algorithm MBFOA [50], methods based on differential evolution strategies DES [51] and the simple evolutionary algorithm SES [52]. The statistical results of these algorithms were compared with the proposed IAPSO and are given in Table 6. As we can see, IAPSO algorithm exhibits the best performance among all rivals. Indeed, after only 6,000 function evaluations, IAPSO stably converge to a new optimal solution better than all known solutions so far, which is:

$\mathbf{x}^* = (3.49999999999760,\ 0.7,\ 17,\ 7.3,\ 7.8,\ 3.35021466609630,\ 5.28668322975692)$

and $f(\mathbf{x}^*) = 2996.34816496772$, with the lowest standard deviation value compared to those of other optimizers. Moreover, during simulation, despite the IAPSO starts with initial infeasible solutions, as shown in Fig. 7, it converges rapidly to the neighborhood of the optimal solution, thanks to its effectiveness and the perfect tuning of the exploration and exploitation parameters.

Table 6
Comparison of statistical results given by different optimizers for the speed reducer design optimization problem (case 1).

Method	Worst	Mean	Best	SD	NFEs
SiC-PSO[22]	NA	2996.4085	2996.34816	0.0000	24,000
COPSO [19]	NA	2996.408525	2996.372448	2.867E−02	30,000
MBFOA [50]	NA	3014.759	2999.264	11.0	30,000
DES[51]	NA	2996.348	2996.348	7.54E−06	36,000
SES[52]	3226.248291	3088.777816	3025.005127	47.36189	36,000
LCA[10]	2996.34816497	2996.34816497	2996.34816497	2.63E−12	24000
APSO[34]*	4677.005187	3855.581557	3177.530771	473.767	30,000

IAPSO	**2996.34816497**	**2996.34816497**	**2996.34816497**	**6,88E-13**	**6000**

(N.A) means not available.

Similarly, the speed reducer problem (case 2) previously was optimized using ABC, LCA, WCA, MBA, PSO–DE, SC, differential evolution with level comparison DELC [53], differential evolution with dynamic stochastic selection DEDS [54], hybrid evolutionary algorithm and adaptive constraint handling technique HEAA [55], (μ + λ)-ES [56], modified differential evolution (MDE) [57,58]. The comparison of the best solutions of reported methods is presented in Table 7. Note that IAPSO converge once again to a new solution better than all known solutions so far. Moreover, the IAPSO outperformed other considered optimization engines as shown in Table 8. In fact, in terms of computation effort and stability, it reaches the best solution faster than other reported algorithms using only 6,000 function evaluations with a very lower standard deviation value. Fig. 8 depicts the reduction of function values versus the number of iterations for speed reducer design problem (case 2), where IAPSO shows the same convergence behavior as the previous case.

Table 7
Comparison of the best solution obtained from various studies for the speed reducer design optimization problem (case 2).

DV	DEDS[54]	DELC[53]	HEAA[55]	MDE[57]	PSO-DE[27]	WCA[11]	MBA [12]	LCA[10]	APSO[34]*	IAPSO
x_1	3.5	3.5	3.500022	3.500010	3.5	3.5	3.5	3.5	3.501313	**3,5**
x_2	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	**0,7**
x_3	17	17	17.000012	17	17	17	17	17	18	**17**
x_4	7.3	7.3	7.300427	7.300156	7.3	7.3	7.300033	7.3	8.127814	**7,3**
x_5	7.715319	7.715319	7.715377	7.800027	7.800000	7.715319	7.715772	7.8	8.042121	**7,7153199**
x_6	3.350214	3.350214	3.350230	3.350221	3.350214	3.350214	3.350218	3.350214666096	3.352446	**3,350214666096**
x_7	5.286654	5.286654	5.286663	5.286685	5.2866832	5.286654	5.286654	5.286683229758	5.287076	**5,286654464979**
f(x)	2994.471066	2994.471066	2994.499107	2996.356689	2996.348167	2994.471066	2994.482453	2994.4710661468	3187.630486	**2994,4710661459**

Table 8
Comparison of statistical results given by different optimizers for the speed reducer design optimization problem (case 2).

Method	Worst	Mean	Best	SD	NFSs
SC[43]	3009.964736	3001.758264	2994.744241	4.0	54,456
PSO-DE[27]	2996.348204	2996.348174	2996.348167	6.4E -06	54,350
DELC[53]	2994.471066	2994.471066	2994.471066	1.9E -12	30,000
DEDS[54]	2994.471066	2994.471066	2994.471066	3.6E -12	30,000
HEAA[55]	2994.752311	2994.613368	2994.499107	7.0E -02	40,000
MDE[57]	NA	2996.367220	2996.356689	8.2E -03	24,000
(μ + λ)-ES[56]	NA	2996.348000	2996.348000	0.0	30,000
ABC[8]	NA	2997.058000	2997.058000	0.0	30,000
WCA[11]	2994.505578	2994.474392	2994.471066	7.4E -03	15,150
LCA[10]	2994.471066146825	2994.47106614682	2994.47106614682	2.66E -12	24000
MBA[12]	2999.652444	2996.769019	2994.482453	1.56	6300
APSO[34]*	4443.017639	3822.640624	3187.630486	366.146	30,000
IAPSO	**2994,47106615489**	**2994,47106614777**	**2994,47106614598**	**2,65E -09**	**6000**

5.4 Tension/compression spring design optimization problem

The weight of a tension/compression spring [59,60], as shown in Fig. 9, is to be minimized subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables (see appendix). The design variables x_1, x_2, and x_3 are respectively the wire diameter, the mean coil diameter and the number of active coils.

This problem was solved by several researchers using different optimization algorithms such as GA1, GA2, CAEP, CPSO, HPSO, NM–PSO, G-QPSO, QPSO, PSO–DE, PSO, DELC, DEDS, HEAA, SC, DE, ABC, and (μ + λ)-ES, etc. Table 9 shows the comparison for the best solution obtained by the proposed IAPSO and above cited algorithms. In terms of solution quality, it is clear that the best solution obtained by IAPSO is similar to the best-known solution obtained by the LCA. It is worth pointing that the best solution reported by NM-PSO is infeasible since the first two constraints are violated. Moreover, as displayed in table 10, the proposed IAPSO converge to the best solution with much less computing effort than LCA method and a satisfactory standard deviation value. Fig. 10 depicts the reduction of function values versus the number of iterations for tension/compression spring design problem. Note that during numerical simulation, in early iterations, all particles positions was in the infeasible region, however, after only fewer iterations, particles are directed to the feasible region and the optimal solution is reached. This may be owing to the effectiveness of the searching criteria and exploration ability of the IAPSO to locate promoting regions of problem domain.

Table 9
Comparison of the best solutions obtained from various studies for the Tension/compression spring design optimization problem.

DV	G-QPSO[28]	DEDS[54]	HEAA[55]	NM–PSO[26]	DELC[53]	WCA[11]	LCA[10]	MBA[12]	APSO[34]*	IAPSO
x_1	0.051515	0.051689	0.051689	0.051620	0.051689	0.051680	0.051689	0.051656	0.052588	**0,051685**
x_2	0.352529	0.356717	0.356729	0.355498	0.356717	0.356522	0.356718	0.355940	0.378343	**0,356629**
x_3	11.538862	11.288965	11.288293	11.333272	11.288965	11.300410	11.28896	11.344665	10.138862	**11,294175**
$g_1(x)$	-4.83E-05	1.45E -09	3.96E -10	1.01E -03	-3.40E- 09	-1.65E -13	NA	0	-1.549E-4	**-1,97E-10**
$g_2(x)$	-3.57E-05	-1.19E -09	-3.59E-10	9.94E -04	2.44E -09	-7.9E- 14	NA	0	-8.328E-4	**-4,64E-10**
$g_3(x)$	−4.0455	-4.053785	-4.053808	- 4.061859	-4.053785	-4.053399	NA	-4.052248	-4.089171	**-4,053610**
$g_4(x)$	−0.73064	-0.727728	-0.727720	-0.728588	-0.727728	-0.727864	NA	-0.728268	-1.069069	**-1,091686**
$f(x)$	0.012665	0.012665	0.012665	0.012630	0.012665	0.012665	0.01266523	0.012665	0.012700	**0.01266523**

Table 10
Comparison of statistical results given by different optimizers for the Tension/compression spring design optimization problem.

Method	Worst	Mean	Best	SD	NFEs
GA1[39]	0.012822	0.012769	0.012704	3.94E -05	900,000
GA2[40]	0.012973	0.012742	0.012681	5.90E - 05	80,000
CAEP [42]	0.015116	0.013568	0.012721	8.42E -04	50,020
CPSO [17]	0.012924	0.012730	0.012674	5.20E - 04	240,000
HPSO [20]	0.012719	0.012707	0.012665	1.58E -05	81,000
NM–PSO [26]	0.012633	0.012631	0.012630	8.47E - 07	80,000
G-QPSO[28]	0.017759	0.013524	0.012665	0.001268	2000
QPSO [28]	0.018127	0.013854	0.012669	0.001341	2000
PSO [12]	0.071802	0.019555	0.012857	0.011662	2000
DE [44]	0.012790	0.012703	0.012670	2.7E -05	204,800
DELC [53]	0.012665	0.012665	0.012665	1.3E -07	20,000
DEDS [54]	0.012738	0.012669	0.012665	1.3E - 05	24,000
HEAA [55]	0.012665	0.012665	0.012665	1.4E -09	24,000
PSO–DE [27]	0.012665	0.012665	0.012665	1.2E -08	24,950
SC [43]	0.016717	0.012922	0.012669	5.9E -04	25,167
(μ + λ)-ES [56]	NA	0.013165	0.012689	3.9E -04	30,000
ABC [8]	NA	0.012709	0.012665	12.813E-3	30,000
LCA[10]	0.01266667	0.01266541	0.01266523	3.88E-07	15,000
WCA[11]	0.012952	0.012746	0.012665	8.06E-05	11,750
MBA[12]	0.012900	0.012713	0.012665	6.30E−05	7650
APSO[34]*	0.014937	0.013297	0.012700	6.85e-4	120,000
IAPSO	**0,01782864**	**0,013676527**	**0.01266523**	**1.573E-3**	**2000**

5.5 Gear train design problem

Gear train design problem [61] aims to minimize the cost of the gear ratio of the gear train as shown in Fig.11. The decision variables of the problem are x_1, x_2, x_3 and x_4 which are respectively the number of teeth for gear A, B, D and F. The constraints are only limits on design variables (side constraints). Design variables to be optimized are in integer form since each gear has to have an integer number of teeth. It is well known that constrained problems with discrete variables may increase the complexity of the problem. For the present problem, the lower and upper bounds of integer design variables are 12 and 60, respectively. The mathematical formulation of the problem is reported in appendix.

Table 11 shows the comparison for the best solution of cuckoo search (CS) algorithm [62], the unified particle swarm optimization (UPSO) algorithm [63] and MBA in terms of the value of design variables and function value. As can be seen, the IAPSO algorithm converges to the best known solution similar to the three other optimization techniques. Statistical results illustrated in table 12 show that IAPSO exceeded all optimizers with respect to the computation effort (NFEs) and displays once again a stable behavior by reporting a very low standard deviation value. Fig. 12 demonstrates the reduction of function values with respect to the number of iterations. Whereas, IAPSO algorithm starts with infeasible solutions, the particles were able to quickly reach the optimal solution thanks to the efficiency of the searching criteria and the balance between exploration and exploitation.

Table 11
Comparison of the best solution obtained from various studies for the gear train design optimization problem.

DV	CS[62]	MBA[12]	APSO[34]*	IAPSO
x_1	43	43	43	**43**
x_2	16	16	16	**16**
x_3	19	19	19	**19**
x_4	49	49	49	**49**
$f(x^*)$	2.70097E-12	2.700857E-12	2.700857E-12	**2.700857E-12**

Table 12
Comparison of statistical results given by different optimizers for the gear train design optimization problem.

Method	Worst	Mean	Best	SD	NFSs
MBA[12]	2.062904E−08	2.471635E−09	2.700857E−12	3.94E−09	1120
UPSO[63]	N.A	3.80562E−08	2.700857E−12	1.09E−07	100,000
CS[62]	2.3576E-9	1.9841E-9	2.7009E-12	3.5546E-9	5,000
APSO[34]*	7.072678E-06	4.781676E-07	2.700857E-12	1.44E-06	8,000
IAPSO	**1.827380E-08**	**5.492477E-09**	**2.700857E-12**	**6.36E-09**	**800**

5.6 Multiple disk clutch brake design problem

The mass of a multiple disc clutch brake [64], as shown in Fig. 13, is to be minimized. The design variables $x_1,...,x_5$ are respectively inner radius, outer radius, thickness of the disc, actuating force and number of friction surfaces. All design variables are discrete and restricted to take the following values:

$x_1 = 60, 61,\ldots, 80$; $x_2 = 90, 91,.., 110$; $x_3 = 1, 1.5,\ldots, 3$; $x_4 = 600, 610,\ldots,1000$ and $x_5 = 2, 3,\ldots, 9$.

This discrete optimization problem (see Appendix) previously was optimized using ABC, WCA, NSGA-II [65] and TLBO [66]. The comparison for the best solution and the statistical optimization results, given by such algorithms, are presented in Tables 13 and 14, respectively. As it can be seen from table 13, IAPSO, TLBO and WCA algorithms converge to the same optimal solution. However, in term of statistical results, the IAPSO shows superiority compared with other optimization algorithms in terms of computation effort (NFEs) and stability (SD). Fig. 14 depicts the reduction of function values versus the number of iterations for the multiple disk clutch brake design problem. It is worth noting that after only a few iterations, the population was converged to the neighborhood of the optimal solution and due to the searching criteria, particles are prompt efficiently conducted to reach the optimal solution.

Table 13
Comparison of the best solution obtained from various studies for the multiple disk clutch brake design optimization problem.

DV	NSGA-II[65]	TLBO[66]	WCA[11]	APSO[34]*	IAPSO
x_1	70	70	70	76	70
x_2	90	90	90	96	90
x_3	1.5	1.0	1.0	1	1.0
x_4	1000	810	910	840	900
x_5	3	3	3	3	3
$g_1(x)$	0.0	0.0	0.0	0.0	0.0
$g_2(x)$	22.0	24.0	24	24	24
$g_3(x)$	0.9005	0.919427	0.909480	0.922273167327214	0.910475344510809
$g_4(x)$	9.7906	9830.3710	9.809429	9.82421128537948	9.8115234375
$g_5(x)$	7.8947	7894.696500	7.894696	7.73837800183432	7.89469658978184
$g_6(x)$	3.3527	0.702013	2.231421	1.3966105059236	1.35977138761092
$g_7(x)$	60.6250	37706.25000	49.768749	48.8483720930233	48.5625
$g_8(x)$	11.6473	14.297986	12.768578	13.6033894940764	13.6402286123891
$f(x)$	0.4704	0.313656	0.313656	0.337181	0.313656

Table 14
Comparison of statistical results given by different optimizers for the multiple disk clutch brake design optimization problem.

Method	Worst	Mean	Best	SD	NFSs
ABC[8]	0.352864	0.324751	0.313657	NA	>900
TLBO[66]	0.392071	0.327166	0.313657	NA	>900
WCA[11]	0.313656	0.313656	0.313656	1.69E-16	500
APSO[34]*	0.716313	0.506829	0.337181	0.09767	2000
IAPSO	**0.313656**	**0.313656**	**0.313656**	**1.13E-16**	**400**

6. Conclusion

In this paper, an improved accelerated particle swarm optimization algorithm (IAPSO) is proposed as a simple and efficient optimization technique for handling nonlinear constrained optimization problems with continuous, mixed, discrete and integer design variables. The proposed approach uses the penalty function method as a constraint handling technique. The IAPSO operates with a simple equation to update particles positions, where there is no need to deal with particles velocities; so easy to apprehend and to implement. Position vector is updated based on the memory gained by each particle, as well as the knowledge gained by the swarm as a whole, allowing particles to discover and preserve promising regions of the search space. Furthermore, fine tuning of integrated control functions, perfectly balance exploration and exploitation of the design space, permitting the IAPSO algorithm to avoid premature convergence and to rapidly converge to the optimum solution. Six benchmark mechanical design optimization problems are considered to demonstrate the effectiveness of the IAPSO compared to other meta-heuristic optimization methods. The obtained results show that the IAPSO algorithm offers better or similar optimal solutions compared to other optimizer methods for all considered engineering problems. In addition, statistical results demonstrate the superiority of the IAPSO to others algorithms in terms of mean and standard deviation and moreover it performs the lowest computation effort for all considered optimization problems.

References

[1] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing., Science. 220 (1983) 671–
680.

[2] J. Holland, Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press, Ann Arbor, 1975.

[3] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 1989.

[4] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, MHS'95. Proc. Sixth Int. Symp. Micro Mach. Hum. Sci. (1995) 39–43.

[5] J. Kennedy, R. Eberhart, Particle swarm optimization, Neural Networks, 1995. Proceedings., IEEE Int. Conf. 4 (1995) 1942–1948 vol.4.

[6] M. Dorigo, M. Birattari, T. Stützle, Ant colony optimization artificial ants as a computational intelligence
technique, IEEE Comput. Intell. Mag. 1 (2006) 28–39.

[7] K.S. Lee, Z.W. Geem, A new structural optimization method based on the harmony search algorithm, Comput. Struct. 82 (2004) 781–798. doi:10.1016/j.compstruc.2004.01.002.

[8] D. Karaboga, B. Basturk, Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization, Lnai 4529. (2007) 789–798.

[9] X.S. Yang, Firefly algorithms for multimodal optimization, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 5792 LNCS (2009) 169–178.

[10] A. Husseinzadeh Kashan, An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA), Comput. Des. 43 (2011) 1769–1792.

[11] H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems, Comput. Struct. 110-111 (2012) 151–166.

[12] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. J. 13 (2013) 2592–2612. doi:10.1016/j.asoc.2012.11.026.

[13] Y. Shi, R. Eberhart, A modified particle swarm optimizer, 1998 IEEE Int. Conf. Evol. Comput. Proceedings. IEEE World Congr. Comput. Intell. (Cat. No.98TH8360). (1998) 69–73.

[14] Y. Shi, R. Eberhart, Empirical study of particle swarm optimization, Proc. 1999 Congr. Evol. Comput. (1999) 1945–1950.

[15] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002) 58–73.

[16] A.E.M. Zavala, A.H. Aguirre, E.R. Villa Diharce, Particle evolutionary swarm optimization algorithm (PESO), Proc. Mex. Int. Conf. Comput. Sci. 2005 (2005) 282–289.

[17] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99.

[18] R. a Krohling, L.D.S. Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems., IEEE Trans. Syst. Man. Cybern. B. Cybern. 36 (2006) 1407–1416.

[19] A.H. Aguirre, A.E.M. Zavala, E. Villa, A. Hern, A.E. Mu, COPSO : Constrained Optimization via PSO algorithm, Statistics (Ber). 2007 (2007).

[20] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2007) 1407–1422.

[21] C. Worasucheep, Solving Constrained Engineering Optimization Problems by the Constrained PSO-DD, (2008) 8–11.

[22] L.C. Cagnina, S.C. Esquivel, C. A. Coello Coello, Solving engineering optimization problems with the simple constrained particle swarm optimizer, Inform. 32 (2008) 319–326.

[23] J. Wang, Z. Yin, A ranking selection-based particle swarm optimizer for engineering design optimization problems, Struct. Multidiscip. Optim. 37 (2008) 131–147.

[24] A.R. Yıldız, A novel particle swarm optimization approach for product design and manufacturing, Int. J. Adv. Manuf. Technol. 40 (2009) 617–628.

[25] L.J.L. Jianjun, L.J.L. Jian, A Modified Particle Swarm Optimization for Practical Engineering Optimization, 2009 Fifth Int. Conf. Nat. Comput. 3 (2009).

[26] E. Zahara, Y.T. Kao, Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems, Expert Syst. Appl. 36 (2009) 3880–3886.

[27] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. J. 10 (2010) 629–640.

[28] L.D.S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2010) 1676–1683.

[29] Davoodi, E., Hagh, M. T., & Zadeh, S. G. (2014). A hybrid Improved Quantum-behaved Particle Swarm Optimization–Simplex method (IQPSOS) to solve power system load flow problems. Applied Soft Computing, 21, 171–179. http://doi.org/10.1016/j.asoc.2014.03.004.

[30] Quaranta, G., Marano, G. C., Greco, R., & Monti, G. (2014). Parametric identification of seismic isolators using differential evolution and particle swarm optimization. Applied Soft Computing, 22, 458–464. http://doi.org/10.1016/j.asoc.2014.04.039.

[31] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference - Imprint of: IGI Publishing (2010).

[32] S.Y. Own, N. Me, A Survey of the State of the Art in Particle Swarm Optimization, Research Journal of Applied Sciences, Engineering and Technology 4.9 (2012) 1181-1197.

[33] N. Nouaouria, M. Boukadoum, R. Proulx, Particle swarm classification: A survey and positioning, Pattern Recognit. 46 (2013) 2028–2044.

[34] X. S. Yang, Engineering Optimization: an Introduction with Metaheuristic Applications, John Wiley & Sons, Inc., Hoboken, New Jersey, 2010.

[35] X.-S. Yang, S. Deb, S. Fong, Accelerated Particle Swarm Optimization and Support Vector Machine for Business Optimization and Applications, Networked Digit. Technol. (2012) 12.

[36] O. Hasançebi, S. Azad, An efficient metaheuristic algorithm for engineering optimization: SPOT, Int. J. Optim. Civ. Eng. 2 (2012) 479–487.

[37] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, in: Proceedings of the 4th Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 135–155.

[38] K. Ragsdell, D. Phillips, Optimal Design of a Class of Welded Structures using Geometric Programming, J. Eng. Ind. 98(3) (1976) 1021–1025.

[39] C. A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.

[40] C. A. Coello Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Informatics. 16 (2002) 193–203.

[41] Q. Yuan, F. Qian, A hybrid genetic algorithm for twice continuously differentiable NLP problems, Comput. Chem. Eng. 34 (2010) 36–41.

[42] C.A. Coello Coello, R.L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, Eng. Optim. 36 (2004) 219–236.

[43] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, IEEE Trans. Evol. Comput. 7 (2003) 386–396.

[44] J. Lampinen, A constraint handling approach for the differential evolution algorithm, in: Proc. 2002 Congr. Evol. Comput. CEC'02 (Cat. No.02TH8600), IEEE, 2002: pp. 1468–1473.

[45] E. Sandgren, Nonlinear Integer and Discrete Programming in Mechanical Design Optimization, J. Mech. Des. 112 (1990) 223-229.

[46] B.K. Kannan, S.N. Kramer, An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design, J. Mech. Des. 116 (1994) 405-411.

[47] F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (2007) 340–356.

[48] J. Golinski, An adaptive optimization system applied to machine synthesis, Mech. Mach. Theory. 8 (1973) 419–436.

[49] Ku, K. J., Rao, S. S., & Chen, L. (1998). Taguchi-Aided Search Method for Design Optimization of Engineering Systems. Engineering Optimization, 30(1), 1–23. http://doi.org/10.1080/03052159808941235

[50] E. Mezura-Montes, B. Hernández-Ocaña, Modified Bacterial Foraging Optimization for Engineering Design, (2009). Na9se

[51] H.-K.K.H.-K. Kim, J.-K.C.J.-K. Chong, K.-Y.P.K.-Y. Park, D. a. Lowther, Differential Evolution Strategy for Constrained Global Optimization and Application to Practical Engineering Problems, IEEE Trans. Magn. 43 (2007) 1565–1568.

[52] C. A. Coello. Coello, R. Landa-becerra, Engineering Optimization Using a Simple Evolutionary Algorithm, (2003). 149-156.

[53] L. Wang, L.P. Li, An effective differential evolution with level comparison for constrained engineering design, Struct. Multidiscip. Optim. 41 (2010) 947–963.

[54] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Inf. Sci. (Ny). 178 (2008) 3043–3074.

[55] Y. Wang, Z. Cai, Y. Zhou, Z. Fan, Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique, Struct. Multidiscip. Optim. 37 (2009) 395–413.

[56] E. Mezura-Montes, C. A. Coello Coello, Useful infeasible solutions in engineering optimization with evolutionary algorithms, eds., MICAI 2005: : Lect. Notes Artif. Int. 3789 (2005) 652–662.

Fig. 3 illustrates the function values with respect to the number of iterations for the welded beam design problem. Note that the proposed IAPSO shows a good converging behavior thanks to the efficiency of the searching model and the perfect balance between exploration and exploitation during optimization process.

[57] E. Mezura-Montes, J. Velazquez-Reyes, C.A. Coello Coello, Modified Differential Evolution for Constrained Optimization, in: 2006 IEEE Int. Conf. Evol. Comput., IEEE, n.d.: pp. 25–32.

[58] E. Mezura-Montes, C. A. Coello Coello, J. Velázquez-Reyes, Increasing Successful Offspring and Diversity in Differential Evolution for Engineering Design, Proc. Adaptive Computing in Design and Manufcature (ACDM 2006), Bristol, UK (2006). 131–139.

[59] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.

[60] J.S. Arora, Introduction to Optimum Design, Elsevier, 2012.

[61] E. Sandgren, Nonlinear Integer and Discrete Programming in Mechanical Design Optimization, J. Mech. Des. 112 (1990) 223-229.

[62] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (2013) 17–35.

[63] L. Wang, K. Chen, Y.S. Ong, Unified particle swarm optimization for solving constrained engineering optimization problems, Advances in Natural Computation, Lecture Notes in Computer Science Volume, 3612 (2005) 582-591.

[64] A. Osyczka, Evolutionary algorithms for single and multicriteria design optimization: studies in fuzzyness and soft computing. Physica-Verlag, Heidelberg (2002) 218.

[65] K. Deb, A. Srinivasan, Innovization, in: Proc. 8th Annu. Conf. Genet. Evol. Comput. - GECCO '06, ACM Press, New York, New York, USA, (2006) 1629-1636.

[66] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput. Des. 43 (2011) 303–315.

Appendix

A1. ***Welded beam design problem***

Minimize $f(X) = 1.10471 x_1 x_2^2 + 0.04811 x_3 x_4 (14 + x_2)$

subject to :

$$g_1(X) = \tau(X) - \tau_{\max} \le 0$$
$$g_2(X) = \sigma(X) - \sigma_{\max} \le 0$$
$$g_3(X) = x_1 - x_4 \le 0$$
$$g_4(X) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0$$
$$g_5(X) = 0.125 - x_1 \le 0$$
$$g_6(X) = \delta(X) - 0.25 \le 0$$
$$g_7(X) = P - P_c(X) \le 0$$
$$0.1 \le x_1, x_4 \le 2$$
$$0.1 \le x_2, x_3 \le 10$$

Where $\tau(X) = \sqrt{\tau'^2 + 2\tau'\tau'' x_2 / 2R + \tau''^2}$

$$\tau' = \frac{P}{\sqrt{2} x_1 x_2}, \tau'' = \frac{MR}{J}, M = P\left(14 + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, J = 2\left\{\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(X) = \frac{504000}{x_4 x_3^2}, \delta(X) = \frac{65856000}{(30 \times 10^6) x_4 x_3^3},$$

$$P_c = \frac{4.013 \times (30 \times 10^6)}{196} \sqrt{\frac{x_3^2 x_4^6}{36}} \times \left[1 - \left(x_3 \sqrt{\frac{30 \times 10^6}{4 \times (12 \times 10^6)}} \Big/ 28\right)\right],$$

$$P = 6000 lb,\ L = 14 in,\ E = 30 \times 10^6\ psi,\ G = 12 \times 10^6\ psi,$$
$$\tau_{\max} = 13,600 psi,\ \sigma_{\max} = 30,000 psi,\ \delta_{\max} = 0.25 in$$

A2. ***Pressure Vessel design optimization problem***

Minimize $f(X) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3$

subject to :

$$g_1(X) = -x_1 + 0.0193 x_3 \le 0$$
$$g_2(X) = -x + 0.00954 x_3 \le 0$$
$$g_3(X) = -\pi x_3^2 x_4^2 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0$$
$$g_4(X) = x_4 - 240 \le 0$$

with $1 \times 0.0625 \le x_1, x_2 \le 99 \times 0.0625,\ 10.0 \le x_3$ and $x_4 \le 200.0$

A3. **Speed reducer design optimization problem**

Case I

$$\text{Minimize } f(X) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right)$$

subject to:

$$g_1(X) = \frac{27}{x_1x_2^2x_3} - 1 \le 0$$

$$g_2(X) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$$

$$g_3(X) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0$$

$$g_4(X) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0$$

$$g_5(X) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9\times10^6} - 1 \le 0$$

$$g_6(X) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5\times10^6} - 1 \le 0$$

$$g_7(X) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(X) = \frac{5x_2}{x_1} - 1 \le 0$$

$$g_9(X) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(X) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(X) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

with $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3,$
$7.8 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9$ and $5.0 \le x_7 \le 5.5$

Case II

Only the lower bound on the design variable x_5 changes as follows:

$7.3 \le x_5 \le 8.3.$

A4. **Tension/compression spring design optimization problem**

Minimize $f(X) = (x_3 + 2) x_2 x_1^2$

subject to:

$$g_1(X) = 1 - \frac{x_2^3 x_3}{7.1785 x_1^4} \le 0$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12.566\left(x_2 x_1^3\right) - x_1^4} + \frac{1}{5.108 x_1^2} - 1 \le 0$$

$$g_3(X) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

with $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$, and $2.0 \le x_3 \le 15.0$.

A5. **Gear train design optimization problem**

Minimize $f(X) = \left(\left(\frac{1}{6.931}\right) - \left(\frac{x_2 x_3}{x_1 x_4}\right)\right)^2$

subject to:

$12 \le x_i \le 60$

A6. **Multiple disk clutch brake design optimization problem**

Minimize $f(X) = \pi\left(r_0^2 - r_i^2\right) t (Z+1) \rho$

subject to:

$$g_1(X) = r_0 - r_i - \Delta r \ge 0$$
$$g_2(X) = l_{\max} - (Z+1)(t+\delta) \ge 0$$
$$g_3(X) = p_{\max} - p_{rz} \ge 0$$
$$g_4(X) = p_{\max} v_{sr\max} - p_{rz} v_{sr} \ge 0$$
$$g_5(X) = v_{sr\max} - v_{sr} \ge 0$$
$$g_6(X) = T_{\max} - T \ge 0$$
$$g_7(X) = M_h - s M_s \ge 0$$
$$g_8(X) = T \ge 0$$

where,

$$M_h = \frac{2}{3}\mu F Z \frac{r_0^3 - r_i^3}{r_0^2 - r_i^2}, \quad p_{rz} = \frac{F}{\pi\left(r_0^2 - r_i^2\right)},$$

$$v_{rz} = \frac{2\pi n\left(r_0^3 - r_i^3\right)}{90\left(r_0^2 - r_i^2\right)}, \quad T = \frac{I_z \pi n}{30\left(M_h - M_f\right)}$$

$\Delta r = 20\,\text{mm},\ I_z = 55\,\text{kgmm}^2,\ p_{\max} = 1\,\text{MPa},\ F_{\max} = 1000\,\text{N},$
$T_{\max} = 15\,\text{s},\ \mu = 0.5,\ s = 1.5,\ M_s = 40\,\text{Nm},\ M_f = 3\,\text{Nm},\ n = 250\,\text{rpm},$
$v_{sr\max} = 10\,\text{m/s},\ l_{\max} = 30\,\text{mm},\ r_{i\min} = 60,\ r_{i\max} = 80,\ r_{0\min} = 90,\ r_{0\max} = 110,$
$t_{\min} = 1.5,\ t_{\max} = 3,\ F_{\min} = 600,\ F_{\max} = 1000,\ Z_{\min} = 2,\ Z_{\max} = 9.$

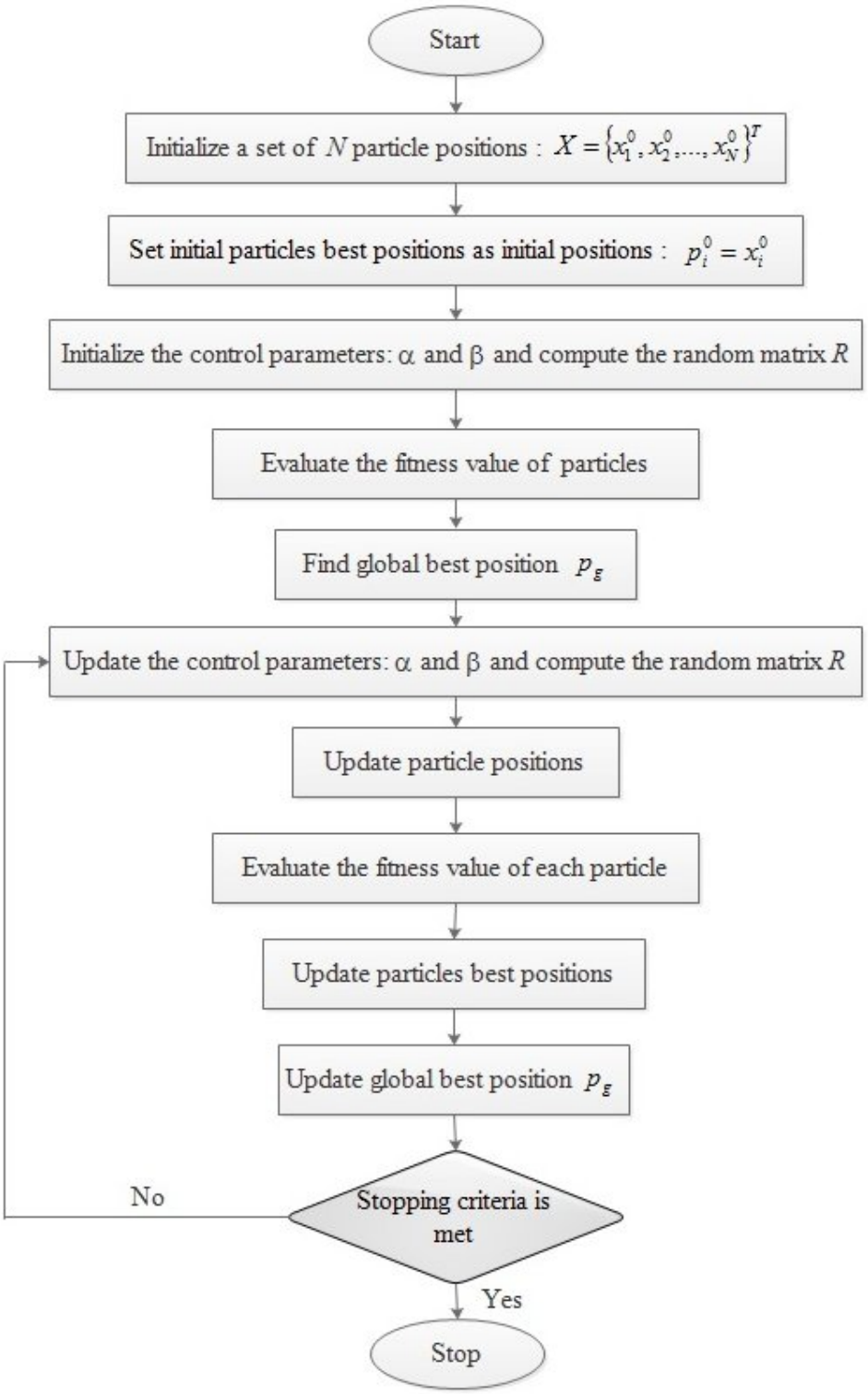
Biography

Dr. Najeh Ben.Guedria received his PhD in Mechanical Engineering from the National School of Engineering of Tunis, Tunisia. Presently he work at Higher Institute of Transport and Logistics of Sousse, as an associate professor. In addition to his interests in mechanical vibration, structural dynamic modification, rotor dynamics optimization and logistics network optimization, he is also enthusiastic about metaheuristic algorithms and their application to mechanical engineering problems.

Corresponding author

Dr. Najeh Ben.Guedria

Address: P.O.BOX 4023, Riadh City, Sousse-Tunisia.
Phone : (216) 54650040
Emails : najeh.benguedria@istls.rnu.tn
najehbenguedria@gmail.com



Flowchart of the Improved Accelerated Particle Swarm Optimization