Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

An improved particle swarm optimizer for mechanical design optimization problems

S. He , E. Prempain & Q. H. Wu
[a] Department of Electrical Engineering and Electronics, The University of Liverpool, Liverpool, L69 3GJ, UK
Version of record first published: 25 Jan 2007.

To cite this article: S. He , E. Prempain & Q. H. Wu (2004): An improved particle swarm optimizer for mechanical design optimization problems, Engineering Optimization, 36:5, 585-605

To link to this article: http://dx.doi.org/10.1080/03052150410001704854

Engineering Optimization
Vol. 36, No. 5, October 2004, 585–605

Taylor & Francis
Taylor & Francis Group

AN IMPROVED PARTICLE SWARM OPTIMIZER FOR MECHANICAL DESIGN OPTIMIZATION PROBLEMS

S. HE, E. PREMPAIN and Q. H. WU*

Department of Electrical Engineering and Electronics, The University of Liverpool, Liverpool L69 3GJ, UK

(Received 10 September 2003; Revised 12 January 2004; In final form 24 March 2004)

This paper presents an improved particle swarm optimizer (PSO) for solving mechanical design optimization problems involving problem-specific constraints and mixed variables such as integer, discrete and continuous variables. A constraint handling method called the 'fly-back mechanism' is introduced to maintain a feasible population. The standard PSO algorithm is also extended to handle mixed variables using a simple scheme. Five benchmark problems commonly used in the literature of engineering optimization and nonlinear programming are successfully solved by the proposed algorithm. The proposed algorithm is easy to implement, and the results and the convergence performance of the proposed algorithm are better than other techniques.

Keywords: Evolutionary algorithms; Particle swarm optimization; Constrained optimization; Mechanical design

1 INTRODUCTION

In the past few decades, many optimization algorithms have been applied to solve mechanical design optimization problems. Among them, evolutionary algorithms (EAs) such as genetic algorithms (GAs), evolutionary programming (EP) and evolution strategies (ES) are attractive because they do not apply mathematical assumptions to the optimization problems and have better global search abilities over conventional optimization algorithms [1]. Many successful applications of EAs have been reported to solve engineering problems such as power system dispatch [2, 3] and mechanical optimal design problems [4, 5]. Recently a new EA called particle swarm optimization (PSO) has been proposed [6]. PSO is a population-based optimization algorithm which was inspired by the social behaviour of animals such as fish schooling and birds flocking. Similar to other EAs, it can solve a variety of hard optimization problems but with a faster convergence rate [7]. Another advantage is that it requires only few parameters to be tuned making it attractive from an implementation viewpoint.

Most mechanical optimal design problems are hard to solve for both conventional optimization algorithms and EAs, because they involve problem-specific constraints. To handle these constraints, many different approaches have been proposed. The most common approach in the EA community is to make use of penalty functions. However, the major drawback of

* Corresponding author. E-mail: qhwu@liv.ac.uk

Engineering Optimization
ISSN 0305-215X print; ISSN 1029-0273 online
http://www.tandf.co.uk/journals
DOI: 10.1080/03052150410001704854

using penalty functions is that they require additional tuning parameters. In particular, the penalty coefficients have to be fine tuned in order to balance the objective and penalty functions. Inappropriate penalty coefficients will make the optimization problem intractable [8, 9]. Other approaches to handle constraints, according to Ref. [10], include rejection of infeasible individuals, maintaining a feasible population, repair of infeasible individuals, separation of individuals and constraints, replacement of individuals by their repaired versions and use of decoders. Standard PSO is usually applied to solve unconstrained optimization problems. In this paper, the standard PSO algorithm is extended to solve constrained mechanical design optimization problems using methods which preserve a feasible population.

Mechanical optimal design problems may contain integer, discrete and continuous variables, which are referred to as mixed-variable nonlinear optimization problems. To solve them, Sandgren [11] and Hajela and Shih [12] have proposed nonlinear branch and bound algorithms based on integer programming. Cao and Wu developed mixed-variable evolutionary programming (MVEP) [4] with different mutation operators associated with different types of variables. Deb and Goyal [5] presented a combined genetic search technique (GeneAS) which combined binary and real-coded GAs to handle mixed variables. Originally PSO was proposed to handle continuous optimization problems. Recently, PSO had been applied to Integer Programming by Parsopoulos and Vrahatis [13] by simply truncating the real values to integers, which does not affect significantly the search performance. In this paper, the standard PSO is extended to handle mixed-variable nonlinear optimization problems more effectively.

This paper is organized as follows: Section 2 introduces the formulation of mechanical design optimization problems. The standard PSO is presented in Section 3. Section 4 proposes a modified version of the PSO algorithm to handle constraints with mixed variables. The proposed PSO has been tested on five examples which are commonly used in the mechanical design optimization literature. Experimental results and discussions are given in Section 5. The paper is concluded in Section 6.

2 FORMULATION OF MECHANICAL DESIGN OPTIMIZATION PROBLEMS

Mechanical design optimization problems can be formulated as a nonlinear programming (NLP) problem. Unlike generic NLP problems which only contain continuous or integer variables, mechanical design optimizations usually involve continuous, binary, discrete and integer variables. The binary variables are usually involved in the formulation of the design problem to select alternative options. The discrete variables are used to represent standardization constraints such as the diameters of standard sized bolts. Integer variables usually occur when the numbers of objects are design variables, such as the number of gear teeth. Considering the mixed variables, the formulation can be expressed as follows:

$$\min f(X) \tag{1}$$

subject to:

$$h_i(X) = 0 \quad i = 1, 2, \ldots, m$$
$$g_i(X) \geq 0 \quad i = m+1, \ldots, p$$

where $f(X)$ is the scalar objective function, $h_i(X)$ and $g_i(X)$ are the equality and inequality constraints, respectively.

The variables vector $X \in \mathbb{R}^N$ represents a set of design variables which can be written as:

$$X = \begin{pmatrix} X^{\mathrm{C}} \\ X^{\mathrm{B}} \\ X^{\mathrm{I}} \\ X^{\mathrm{D}} \end{pmatrix} = [x_1^{\mathrm{C}}, \ldots, x_{n_{\mathrm{C}}}^{\mathrm{C}}, x_1^{\mathrm{B}}, \ldots, x_{n_{\mathrm{B}}}^{\mathrm{B}}, x_1^{\mathrm{I}}, \ldots, x_{n_{\mathrm{I}}}^{\mathrm{I}}, x_1^{\mathrm{D}}, \ldots, x_{n_{\mathrm{D}}}^{\mathrm{D}}]^{\mathrm{T}}$$

where

$$\begin{aligned} x_i^{\mathrm{Cl}} \le x_i^{\mathrm{C}} \le x_i^{\mathrm{Cu}}, \quad & i = 1, 2, \ldots, n_{\mathrm{C}} \\ x_i^{\mathrm{B}} \in \{x_i^{\mathrm{Bl}}, x_i^{\mathrm{Bu}}\}, \quad & i = 1, 2, \ldots, n_{\mathrm{B}} \\ x_i^{\mathrm{Il}} \le x_i^{\mathrm{I}} \le x_i^{\mathrm{Iu}}, \quad & i = 1, 2, \ldots, n_{\mathrm{I}} \\ x_i^{\mathrm{Dl}} \le x_i^{\mathrm{D}} \le x_i^{\mathrm{Du}}, \quad & i = 1, 2, \ldots, n_{\mathrm{D}} \end{aligned} \tag{2}$$

where $X^{\mathrm{C}} \in \mathbb{R}^{n_{\mathrm{C}}}$, $X^{\mathrm{B}} \in \mathbb{R}^{n_{\mathrm{B}}}$, $X^{\mathrm{I}} \in \mathbb{R}^{n_{\mathrm{I}}}$ and $X^{\mathrm{D}} \in \mathbb{R}^{n_{\mathrm{D}}}$ denote feasible subsets of comprising continuous, binary, integer and discrete variables, respectively. x_i^{Cl}, x_i^{Bl}, x_i^{Il} and x_i^{Dl} are the lower bounds of the ith variables of X^{C}, X^{B}, X^{I} and X^{D}, respectively. x_i^{Cu}, x_i^{Zu}, x_i^{Iu} and x_i^{Du} are the upper bounds of the ith variables of X^{C}, X^{B}, X^{I} and X^{D}, respectively. n_{C}, n_{B}, n_{I} and n_{D} are the numbers of continuous, binary, integer and discrete variables, respectively. The total number of variables is $N = n_{\mathrm{C}} + n_{\mathrm{B}} + n_{\mathrm{I}} + n_{\mathrm{D}}$.

3 PARTICLE SWARM OPTIMIZER

The PSO is a population-based optimization algorithm. Its population is called a *swarm* and each individual is called a *particle*. Each particle flies through the problem space to search for optima. The ith particle at iteration k has the following two attributes:

1) A current position in an N-dimensional search space which represents a potential solution: $X_i^k = (x_{i,1}^k, \ldots, x_{i,n}^k, \ldots, x_{i,N}^k)$, where $x_{i,n}^k \in [l_n, u_n]$ is the nth dimensional variable, $1 \le n \le N$, l_n and u_n are the lower and upper bounds for the nth dimension, respectively.
2) A current velocity $V_i^k = (v_{i,1}^k, \ldots, v_{i,n}^k, \ldots, v_{i,N}^k)$, which controls its fly speed and direction. V_i^k is restricted to a maximum velocity $V_{\max}^k = (v_{\max,1}^k, \ldots, v_{\max,n}^k, \ldots, v_{\max,N}^k)$.

At each iteration, the swarm is updated by the following equations:

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (P_i^k - X_i^k) + c_2 r_2 (P_g^k - X_i^k) \tag{3}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{4}$$

where P_i is the best previous position of the ith particle (also known as *pbest*) and P_g is the global best position among all the particles in the swarm (also known as *gbest*). They are given by the following equations:

$$P_i = \begin{cases} P_i \colon f(X_i) \ge P_i \\ X_i \colon f(X_i) < P_i \end{cases} \tag{5}$$

$$P_{\mathrm{g}} \in \{P_0, P_1, \ldots, P_M\} | f(P_{\mathrm{g}}) = \min(f(P_0), f(P_1), \ldots, f(P_M)) \tag{6}$$

where f is the objective function, M is the total number of particles, r_1 and r_2 are the elements generated from two uniform random sequences on the interval [0, 1]: $r_1 \sim U(0, 1)$; $r_2 \sim U(0, 1)$ and ω is an inertia weight [14] which is typically chosen in the range of [0,1]. A larger inertia weight facilitates global exploration and a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area [15]. Therefore the inertia weight ω is critical for the PSO's convergence behaviour. A suitable value for the inertia weight ω usually provides balance between global and local exploration abilities and consequently results in a better optimum solution. Initially, the inertia weight was kept constant. However, Ref. [16] indicated that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions. c_1 and c_2 are acceleration constants [16] which also control how far a particle will move in a single iteration. The maximum velocity $V_{\max}$ is set to be half of the length of the search space in one dimension.

4 IMPROVED PARTICLE SWARM OPTIMIZER

As mentioned in the introduction, the difficulties in using EAs to solve mechanical optimization problems come from problem-specific constraints and mixed variables. Little work has been done for solving constrained mixed-variable optimization problems with PSO. In this section, PSO techniques to handle mixed variables and constraints are proposed.

4.1 Mixed-Variable Handling Methods

Originally, most of the EAs were proposed to handle continuous variables. In the last decade, GAs [17] and ESs [18] and EPs [4] have been extended to handle mixed variables.

In its basic form, the PSO can only handle continuous variables. To handle integer variables, simply truncating the real values to integers to calculate fitness value will not effect the search performance significantly [13]. The truncation is only performed in evaluating the fitness function. That is, the swarm will 'fly' in a continuous search space regardless of the variable type. Binary variables, since they can be regarded as integer variables within the range of [0, 1], are not considered separately.

For discrete variables of the ith particle X_i, the most straightforward way is to use the indices of the set of discrete variables with n_{D} elements:

$$X_i^{\mathrm{D}} = [x_{i,1}^{\mathrm{D}}, \ldots, x_{i,n_{\mathrm{D}}}^{\mathrm{D}}]$$

For particle i, the index value j of the discrete variable $x_{i,j}^{D}$ is then optimized instead of the discrete value of the variable directly. In the population, the indices of the discrete variables of the ith particle should be the float point variables before truncation. That is, $j \in [1, n_{\mathrm{D}} + 1)$, n_{D} is the number of discrete variables. Hence, the fitness function of the ith particle X_i can be expressed as follows:

$$f(X_i) \qquad i = 1, \ldots, M \tag{7}$$

where

$$X_i = \begin{cases} x_{i,j}\colon x_{i,j} \in X_i^{\mathrm{C}} & j = 1, \ldots, n_{\mathrm{C}} \\ \mathrm{INT}(x_{i,j})\colon x_{i,j} \in X_i^{\mathrm{I}} \cup X_i^{\mathrm{B}} & j = 1, \ldots, n_{\mathrm{I}} + n_{\mathrm{B}} \\ x_{i,\mathrm{INT}(j)}\colon x_{i,\mathrm{INT}(j)} \in X_i^{\mathrm{D}} & j \in [1, n_{\mathrm{D}} + 1) \end{cases} \tag{8}$$

where $X_i^{\mathrm{C}} \in \mathbb{R}^{n_{\mathrm{C}}}$, $X_i^{\mathrm{B}} \in \mathbb{R}^{n_{\mathrm{B}}}$, $X_i^{\mathrm{I}} \in \mathbb{R}^{n_{\mathrm{I}}}$ and $X_i^{\mathrm{D}} \in \mathbb{R}^{n_{\mathrm{D}}}$ denote the feasible subsets of continuous, binary, integer and discrete variables of particle X_i, respectively. $\mathrm{INT}(x)$ denotes the greatest integer less than the real value x.

4.2 Constraint Handling Methods

EAs are heuristic optimization techniques which have been successfully applied to various optimization problems. However, they are not able to handle constrained optimization problems directly [19]. In the past few years, much work has been done to improve EAs performance to deal with constrained optimization problems. Penalty functions are commonly used to incorporate constraints into the fitness function. Other techniques developed to handle the constraints, reported in Refs. [4] and [10], include rejection of infeasible individuals, maintaining a feasible population, repair of infeasible individuals, and multi-objective optimization techniques.

PSO algorithms have been applied to constrained optimization problems. EI-Gallad *et al*. [20] proposed a constraint handling technique based on maintaining a feasible population. However, our experimental results indicate that such a technique will lower the efficiency of the standard PSO. Their technique reset the infeasible particles to their previous best positions *pbest* which will sometimes prevent the search reaching a global minimum. Hu and Eberhart [21] also proposed a constraint handling technique based on preserving a feasible population. The algorithm starts from a feasible initial population. During the search process, only feasible particles are counted when calculating the value of the previous best position *pbest* and global best position *gbest*. Parsopoulos and Vrahatis [22] incorporated a non-stationary multi-stage assignment penalty function into PSO. In their paper, a set of six benchmark functions were tested. However some of their solutions are not feasible. Other attempts include applying a multi-objective optimization technique to handle constraints [23].

In this study, the technique of maintaining a feasible population is investigated. The technique starts from a feasible initial population. A closed set of operators is used to maintain the feasibility of the solutions. Therefore, the subsequent solutions generated at each iteration are also feasible. Algorithms based on this technique are much more reliable than those based on a penalty approach [10]. For mechanical design problems, reliability is crucial since most of the constraints need to be satisfied. The concept of maintaining a feasible population is suitable for incorporation into the standard PSO algorithm for solving mechanical design problems.

For the PSO algorithm, the intuitive idea to maintain a feasible population is for a particle to fly back to its previous position when it is outside the feasible region. This is the so called 'fly-back mechanism'. Since the population is initialized in the feasible region, flying back to a previous position will guarantee the solution to be feasible. From our experience, the global minima of mechanical optimal design problems are usually close to the boundaries of the feasible space, as shown in Figure 1. Flying back to its previous position when a particle

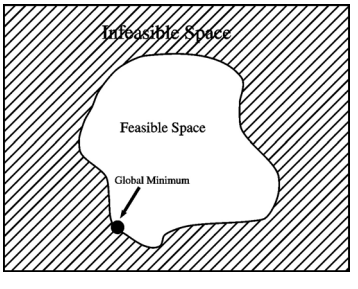

FIGURE 1 Global minimum in the feasible space.

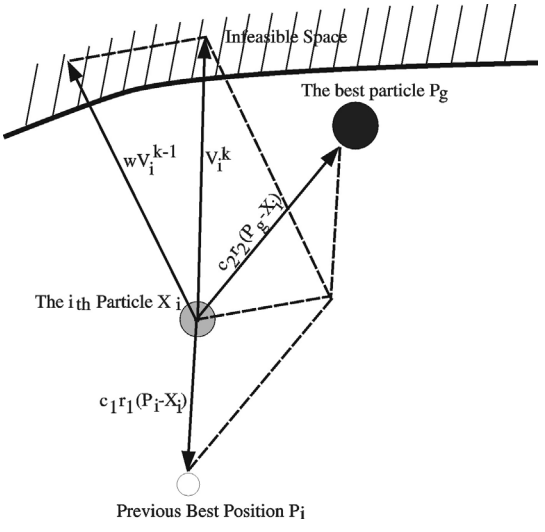

FIGURE 2 X_i at iteration k would fly outside the feasible search space.

violates the constraints will allow a new search closer to the boundaries. Figures 2 and 3 illustrate the search process of the 'fly-back mechanism'. In Figure 2, the ith particle would fly into the infeasible search space at the kth iteration. At the next iteration as shown in Figure 3, this particle is set back to its previous position X_i^{k-1} and starts a new search. Assuming that the global best particle P_g stays in the same position, the direction of the new velocity V_i^{k+1} will still point to the boundary but closer to P_g. Since P_g is inside the feasible space and ωV_i^k is smaller than V_i^k, the chance of particle X_i flying outside the boundaries at the next iteration will be decreased. This property makes the particles more likely to explore the feasible search space near the boundaries. Therefore, such a 'fly-back mechanism' is suitable for mechanical design problems. Moreover, our experimental results show that this technique can find better minima with fewer iterations compared with other techniques.

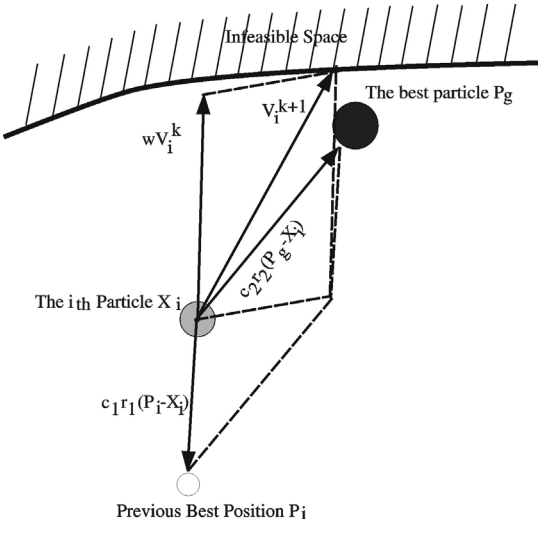

FIGURE 3 X_i flies back to its previous position and starts a new search.

TABLE I Pseudo code for the improved PSO algorithm.

```
Set k = 1;
Randomly initialize positions and velocities of all particles;
  FOR (each particle i in the initial population)
  WHILE (the constraints are violated)
    Randomly re-initialize current particle X_i
  END WHILE
END FOR
WHILE (the termination conditions are not met)
  FOR (each particle i in the swarm)
    Check feasibility: Check the feasibility of the current particle. If X_i^k
                       is outside the feasible region, then reset X_i^k to the
                       previous position X_i^{k-1};
    Calculate fitness: Calculate the fitness value f(X_i^k) of current particle
                       using Eq. (8);
    Update pbest:      Compare the fitness value of pbest with f(X_i^k). If
                       f(X_i^k) is better than the fitness value of pbest, then
                       set pbest to the current position X_i^k;
    Update gbest:      Find the global best position of the swarm. If the
                       f(X_i^k) is better than the fitness value of gbest, then
                       gbest is set to the position of the current particle
                       X_i^k;
    Update velocities: Calculate velocities V_i^k using Eq. (3);
    Update positions:  Calculate positions X_i^k using Eq. (4);
  END FOR
  Set k = k + 1;
END WHILE
```

4.3 Improved Particle Swarm Optimizer Algorithm

Regarding the proposed constraint handling technique described in Section 4.2, the improved PSO requires a feasible initial population to guarantee that the solutions of successive generations are feasible. To do so, an extra loop at the beginning of the algorithm is required to keep randomly re-initializing infeasible particles to ensure that they stay inside the feasible search space. Our experience indicates that this simple method is sufficiently good for most mechanical design problems since their feasible search spaces are usually large and feasible particles can be easily generated. Small size populations are preferred to minimize the time to find a feasible initial population.

The improved PSO algorithm is given in Table I.

5 NUMERICAL EXAMPLES

In this section, five numerical examples have been used to test our new PSO algorithm. The first example is a classical benchmark problem in nonlinear constrained optimization. Four other examples are taken from the mechanical design optimization literature [24]. All these problems have linear and nonlinear constraints and have been investigated by various EAs or traditional techniques.

For all problems a population of 30 individuals is used. Although a time decreasing inertia weight was suggested to be better than a fixed one [16], the experimental results suggested that for these five examples, a fixed inertia weight $\omega = 0.8$ can produce better results. The default value of acceleration constants c_1, c_2 typically are set to 2.0. However with a setting of $c_1 = c_2 = 0.5$ better results were obtained. For each problem, 100 independent runs were

carried out. The proposed algorithm was implemented in MATLAB 6.5 and executed on a Pentium 4, 2-GHz machine.

5.1 Example 1: Himmelblau's Function

This problem, proposed by Himmelblau [25], is a common benchmark function for nonlinear constrained optimization problems. We adopted this problem to test our PSO algorithm which has an improved constraint handling capability. The problem, which has five design variables and six nonlinear constraints, is as follows:

Minimize

$$f(X) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{9}$$

subject to:

$$0 \le g_1(X) \le 92 \tag{10}$$

$$90 \le g_2(X) \le 110 \tag{11}$$

$$20 \le g_3(X) \le 25 \tag{12}$$

where

$$g_1(X) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \tag{13}$$

$$g_2(X) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \tag{14}$$

$$g_3(X) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \tag{15}$$

and

$$78 \le x_1 \le 102, \;\; 33 \le x_2 \le 45, \;\; 27 \le x_3 \le 45, \;\; 27 \le x_4 \le 45 \;\text{ and }\; 27 \le x_5 \le 45$$

Himmelblau [25] used the generalized reduced gradient (GRG) method to solve this problem. This problem was also tackled by Gen and Cheng [26] using a GA based on both local and global references. Runarsson and Yao [27] proposed an ES with stochastic ranking to solve this problem.

For Himmelblau's function, all the results obtained from the methods mentioned above are listed in Table II and are compared against those obtained with the proposed PSO. Other researchers have also proposed different approaches to solve this problem and produced good results. For example, Koziel and Michalewicz [28] proposed a new approach to solve this problems based on incorporating a homomorphous mapping between an n-dimensional cube and a feasible search space. The best result they obtained was -30664.5. Parsopoulos and Vrahatis [22] reported a best result of -31528.289, which is not feasible. The best solution reported by Hu and Eberhart [21] was -30665.5. Since the design variables were not included in their papers, we could not list their solutions in Table II.

The maximum number of generations, used in the proposed PSO, was 3000 with 90,000 function evaluations. The average execution time required for finding a feasible initial population and 90,000 function evaluations was 36.5 s of CPU time. From Table II it can be seen that the proposed PSO has found the same optimum. The mean value for 100 independent runs is -30643.989 with a standard deviation of 70.043, which is worse than the mean value of 30665.539 reported by Runarsson and Yao [27]. However, it is worth mentioning that the

TABLE II Optimal solution of Himmelblau's function.

Design variables	*Best solution found*			
	PSO	*Runarsson [27]*	*GRG [25]*	*Gen [26]*
x_1	78.0000	78.0000	78.6200	81.490
x_2	33.0000	33.0000	33.4400	34.0900
x_3	29.995256025682	29.995256025682	31.0700	31.2400
x_4	45.0000	45.0000	44.1800	42.2000
x_5	36.775812905789	36.775812905788	35.2200	34.3700
$g_1(X)$	92.0000	92.0000	91.7927	91.7819
$g_2(X)$	98.8405	98.8405	98.8929	99.3188
$g_3(X)$	20.0000	20.0000	20.1316	20.0604
$f(X)$	**−30665.539**	**−30665.539**	**−30373.949**	**−30183.576**

number of function evaluations of their stochastic ranking technique was 350,000. The proposed PSO has a much faster performance.

5.2 Example 2: Spring Design

In this section we will investigate two cases of a compression spring design problem. They both have three design variables: the wire diameter $d = x_1$, the mean coil diameter $D = x_2$ and the number of active coils $N = x_3$ as shown in Figure 4. The data type of design variables, objective function and constraints of these two cases are different.

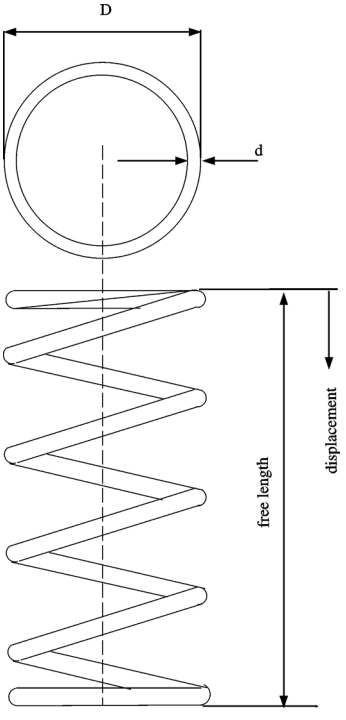

FIGURE 4 Spring design.

5.2.1 Case 1

Case 1 is a real-world optimization problem which involves discrete, integer and continuous design variables. It is aimed at minimizing the volume of a compression spring under static loading. The three design variables are mixed: D is continuous, N is an integer and d is a discrete variable having 42 possible value as shown in Table III. The problem is formulated as follows:

Minimize

$$f(X) = \frac{\pi^2 x_2 x_1^2 (x_3 + 2)}{4} \tag{16}$$

subject to:

$$g_1(X) = \frac{8C_f F_{\max} x_2}{\pi x_1^3} - S \leq 0 \tag{17}$$

$$g_2(X) = l_f - l_{\max} \leq 0 \tag{18}$$

$$g_3(X) = d_{\min} - x_1 \leq 0 \tag{19}$$

$$g_4(X) = x_2 - D_{\max} \leq 0 \tag{20}$$

$$g_5(X) = 3.0 - \frac{x_2}{x_1} \leq 0 \tag{21}$$

$$g_6(X) = \sigma_{\mathrm{p}} - \sigma_{\mathrm{pm}} \leq 0 \tag{22}$$

$$g_7(X) = \sigma_{\mathrm{p}} + \frac{(F_{\max} - F_{\mathrm{p}})}{K} + 1.05(x_3 + 2)x_1 - l_f \leq 0 \tag{23}$$

$$g_8(X) = \sigma_w - \frac{(F_{\max} - F_{\mathrm{p}})}{K} \leq 0 \tag{24}$$

where

$$C_f = \frac{4(x_2/x_1) - 1}{4(x_2/x_1) - 4} + \frac{0.615 x_1}{x_2} \tag{25}$$

$$K = \frac{G x_1^4}{8 x_3 x_2^3} \tag{26}$$

$$\sigma_{\mathrm{p}} = \frac{F_{\mathrm{p}}}{K} \tag{27}$$

$$l_f = \frac{F_{\max}}{K} + 1.05(x_3 + 2)x_1 \tag{28}$$

TABLE III Possible spring steel wire diameters.

Wire diameters (in.)						
0.009	0.0095	0.0104	0.0118	0.0128	0.0132	0.014
0.015	0.0162	0.0173	0.018	0.020	0.023	0.025
0.028	0.032	0.035	0.041	0.047	0.054	0.063
0.072	0.080	0.092	0.105	0.120	0.135	0.148
0.162	0.177	0.192	0.207	0.225	0.244	0.263
0.283	0.307	0.331	0.362	0.394	0.4375	0.500

Other specifications are: the maximum work load $F_{\max} = 1000.0$ lb; the maximum free length $l_{\max} = 14.0$ in.; the minimum wire diameter $d_{\min} = 0.2$ in.; the allowable maximum shear stress $S = 189000.0$ psi; the maximum outside diameter of the spring $D_{\max} = 3.0$ in.; the preload compression force $F_{\mathrm{p}} = 300.0$ lp; the allowable maximum deflection under preload $\sigma_{\mathrm{pm}} = 6.0$ in.; the deflection from preload position to maximum load position $\sigma_w = 1.25$ in.; the shear modulus of the material $G = 11.5 \times 10^6$ psi;

The design variables are limited as follows:

$$0.2 \le x_1 \le 1, \quad 0.6 \le x_2 \le 3, \quad 1 \le x_3 \le 70$$

This problem was investigated by Sandgren [11]. Deb and Goyal [5] applied genetic adaptive search (GeneAS) to solve this problem. Other attempts included a mixed-variable differential evolution (DE) algorithm [29].

The maximum number of generations, used in the proposed PSO, was fixed to 500 with 15,000 function evaluations. The best solution for 100 runs is listed and it is compared to the results obtained by the other techniques mentioned above, which are listed in Table IV. It can be seen that PSO found the same global optimum as DE. It is worth mentioning that the maximum number of generations of DE was 650, corresponding to 26,000 function evaluations [29].

The mean value for the 100 runs performed was 2.738024 with a standard deviation of 0.107061. The average time required for a single run was 5.8 s of CPU time.

5.2.2 Case 2

This problem was first investigated by Belegundu [30] and Arora [31] and aims to minimize the weight of a tension/compression spring. All three design variables are continuous. There are four constraints which relate to minimum deflection, shear stress, surge frequency, and limits on outside diameter and design variables [31]. The mathematical model of the problem can be expressed as follows:

Minimize

$$f(X) = (x_3 + 2)x_2x_1^2 \tag{29}$$

TABLE IV Optimal solution of spring design for Case 1.

Design variables	*Best solution found*			
	PSO	*Sandgren [11]*	*GeneAS [5]*	*DE [29]*
$x_1(d)$	0.283	0.283	0.283	0.283
$x_2(D)$	1.223041010	1.180701	1.226	1.223041010
$x_3(N)$	9	10	9	9
$g_1(X)$	−1008.8114	−54309	−713.510	−1008.8114
$g_2(X)$	−8.9456	−8.8187	−8.933	−8.9456
$g_3(X)$	−0.083	−0.08298	−0.083	−0.083
$g_4(X)$	−1.777	−1.8193	−1.491	−1.777
$g_5(X)$	−1.3217	−1.1723	−1.337	−1.3217
$g_6(X)$	−5.4643	−5.4643	−5.461	−5.4643
$g_7(X)$	0.0000	0.0000	0.0000	0.0000
$g_8(X)$	0.0000	0.0000	−0.009	0.0000
$f(X)$	**2.65856**	**2.7995**	**2.665**	**2.65856**

subject to:

$$g_1(X) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \tag{30}$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \tag{31}$$

$$g_3(X) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \tag{32}$$

$$g_4(X) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \tag{33}$$

The boundaries of the design variables are as follows:

$$0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15$$

Arora [31] proposed an optimization technique called constraint correction at constant cost to deal with this problem. Coello Coello [32] investigated this problem with a GA with a self-adaptive penalty approach to handle constraints. This problem was also tackled by Ray and Liew using an EA inspired by a formal society and the civilization model [33].

The maximum number of generation was 500 corresponding to 15,000 fitness function evaluations. The average execution time required for a single run was 5.2 s of CPU time. Table V lists the best solutions for 100 runs of our PSO and the techniques mentioned above. From Table V, it can be noticed that Arora's technique is not applicable because the first constraint is violated. It can also be seen that our proposed approach was able to find the best solution.

The mean value for the 100 runs performed was 0.01270233 with a standard deviation of 4.124390×10^{-5}. Ray and Liew [33] reported a mean from 50 runs of 0.012922669 which is worse than that obtained by our proposed technique. The number of fitness function evaluations of Ray's algorithm was 25,167.

5.3 Example 3: Pressure Vessel Design

The pressure vessel design problem, shown in Figure 5, was introduced by Sandgren [11]. The objective of this problem is to minimize the total cost of materials, forming and welding of

TABLE V Optimal solution of spring design for Case 2.

Design variables	*Best solution found*			
	PSO	*Ray [33]*	*SAPA [32]*	*Arora [31]*
$x_1(d)$	0.05169040	0.0521602	0.051480	0.053396
$x_2(D)$	0.35674999	0.36815870	0.351661	0.399180
$x_3(N)$	11.28712599	10.64844226	11.632201	9.185400
$g_1(X)$	−0.00000449	−0.00000001	−0.002080	0.000019
$g_2(X)$	0.00000000	−0.00000000	−0.000110	−0.000018
$g_3(X)$	−4.05382661	−4.075805	−4.026318	−4.123832
$g_4(X)$	−0.72770641	−0.71978739	−4.02638	−0.698283
$f(X)$	**0.0126652812**	**0.0126692493**	**0.0127047834**	**0.0127302737**

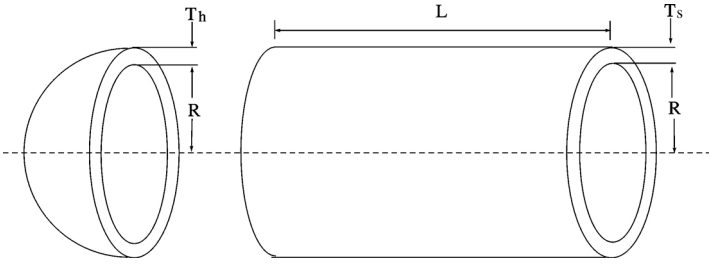

FIGURE 5 Pressure vessel design.

the pressure vessel. There are four design variables: the shell thickness $T_s = x_1$, the thickness of the head $T_h = x_2$, the inner radius $R = x_3$ and the length of the cylindrical section of the vessel $L = x_4$. T_s and T_h are discrete values which are integer multiples 0.0625 in., in accordance with the available thickness of rolled steel plates; R and L are continuous. The optimization problem can be expressed as follows:

Minimize

$$f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{34}$$

subject to:

$$g_1(X) = 0.0193x_3 - x_1 \leq 0 \tag{35}$$

$$g_2(X) = 0.00954x_3 - x_2 \leq 0 \tag{36}$$

$$g_3(X) = 1{,}296{,}000 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leq 0 \tag{37}$$

$$g_4(X) = x_4 - 240 \leq 0 \tag{38}$$

where the design variables have to be in the following ranges:

$$0.0625 \leq x_1 \leq 6.1875, \quad 0.0625 \leq x_2 \leq 6.1875, \quad 10 \leq x_3 \leq 200, \quad 10 \leq x_4 \leq 200.$$

This problem was dealt with by Coello Coello [34] using GA with a dominance-based tournament selection scheme (GADTS) to handle constraints. This problem was also investigated previously by Deb using GeneAS [35]. It has also been tackled by Cao and Wu [4] using MVEP.

The maximum number of generations of the proposed PSO was set to 1000, corresponding to 30,000 fitness function evaluations. The algorithm undertook 100 runs and the best result is listed in Table VI. The average CPU time required was 8.2 s for a single run. Table VI also lists the best results produced by the other methods. Clearly, the new PSO gives better results than the other techniques.

The mean fitness value was $f(x) = 6289.92881$ with a standard deviation of 305.78, which is worse than the mean value of 6177.253268 produced by GADTS [34]. However, it is worth mentioning that the number of fitness function evaluations of GADTS was 80,000.

5.4 Example 4: Welded Beam Design

As shown in Figure 6, a rectangular beam is designed as a cantilever beam to carry a certain load with minimum overall cost of fabrication. The problem involves four design variables: the

TABLE VI Optimal solution of pressure vessel design.

Design variables	*Best solution found*			
	PSO	*GADTS [34]*	*GeneAS [35]*	*MVEP [4]*
$x_1(T_s)$	0.81250000	0.8125	0.9345	1.000
$x_2(T_h)$	0.43750000	0.4375	0.5000	0.625
$x_3(R)$	42.09844560	40.097398	48.3290	51.1958
$x_4(L)$	176.63659584	176.654047	112.6790	90.7821
$g_1(X)$	0.00000000	−0.000020	−0.004750	−0.0119
$g_2(X)$	−0.03588083	−0.035891	−0.038941	−0.1366
$g_3(X)$	0.00000000	−27.886075	−3652.876838	−13584.5631
$g_4(X)$	−63.36340416	−63.345953	−127.321000	−149.2179
$f(X)$	**6059.7143**	**6059.946341**	**6410.3811**	**7108.6160**

thickness of the weld $h = x_1$, the length of the welded joint $l = x_2$, the width of the beam $t = x_3$ and the thickness of the beam $b = x_4$. The values of x_1 and x_2 are coded with integer multiples of 0.0065. There are seven constraints, which involve shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), deflection of the beam (δ) and side constraints [24]. The welded beam problem is stated as follows:

Minimize

$$f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{39}$$

subject to:

$$g_1(X) = \tau(X) - \tau_{\max} \leq 0 \tag{40}$$

$$g_2(X) = \sigma(X) - \sigma_{\max} \leq 0 \tag{41}$$

$$g_3(X) = x_1 - x_4 \leq 0 \tag{42}$$

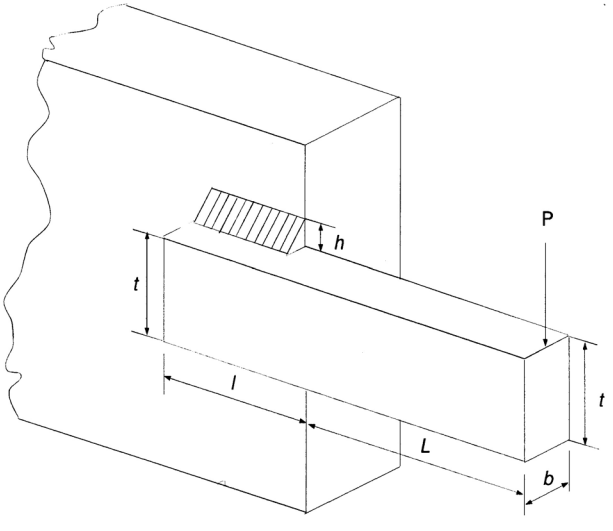

FIGURE 6 Welded beam design.

$$g_4(X) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5 \le 0 \tag{43}$$

$$g_5(X) = 0.125 - x_1 \le 0 \tag{44}$$

$$g_6(X) = \delta(X) - \delta_{\max} \le 0 \tag{45}$$

$$g_7(X) = P - P_c(X) \le 0 \tag{46}$$

where

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{47}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \tag{48}$$

$$\tau'' = \frac{MR}{J}, \quad M = P\left(L + \frac{x_2}{2}\right) \tag{49}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{50}$$

$$J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{51}$$

$$\delta(X) = \frac{4PL^3}{Ex_3^3x_4}, \quad \sigma(X) = \frac{6PL}{x_4x_3^2} \tag{52}$$

$$P_c(X) = \frac{4.013\sqrt{(EGx_3^2x_4^6)/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{53}$$

$$P = 6000\,\text{lb}, \quad L = 14\,\text{in.}, \quad E = 30 \times 10^6\,\text{psi}, \quad G = 12 \times 10^6\,\text{psi} \tag{54}$$

$$\tau_{\max} = 13{,}600\,\text{psi}, \quad \sigma_{\max} = 30{,}000\,\text{psi}, \quad \delta_{\max} = 0.25\,\text{in.} \tag{55}$$

The ranges for the design variables are given as follows:

$$0.1 \le x_1 \le 2.0, \quad 0.1 \le x_2 \le 10, \quad 0.1 \le x_3 \le 10, \quad 0.1 \le x_4 \le 2.0.$$

This problem was investigated by Ragsdell and Phillips [36] using geometric programming. Deb [37] proposed a simple GA with binary representation and a traditional penalty function to solve this problem. The best-known result was also obtained by Deb using real parameter GA [38]. Ray and Liew tackled this problem using a society and civilization algorithm [33].

The best solution for 100 runs of the proposed PSO and those produced by the methods mentioned above are listed in Table VII. However, we could not list the best-known result of 2.38119 in this table, because the design variables were not presented in Ref. [38]. We can see that the new PSO algorithm provides even better results, which were obtained with the maximum number of generations set to 1000 and the total number of fitness function evaluations performed set to 30,000. The average CPU time required for one execution of the proposed algorithm was 10.2 s.

The mean value of the objective function obtained from 100 runs was 2.381932, with a standard deviation 5.239371×10^{-3}. The number of fitness function evaluations of Deb's technique was 40,080.

TABLE VII Optimal solution of welded beam design.

Design variables	*Best solution found*			
	PSO	*Ray [33]*	*Ragsdell [36]*	*Deb [37]*
$x_1(h)$	0.24436898	0.244438276	0.2455	0.2489
$x_2(l)$	6.21751974	6.237967234	6.1960	6.1730
$x_3(t)$	8.29147139	8.288576143	8.2730	8.1789
$x_4(b)$	0.24436898	0.244566182	0.2455	0.2533
$g_1(X)$	−5741.17693313	−5760.11047125	−5743.826517	−5758.603777
$g_2(X)$	−0.00000067	−3.24542756	−4.71509720	−255.576901
$g_3(X)$	0.00000000	−0.00012790	0.00000000	−0.004400
$g_4(X)$	−3.02295458	−3.02005520	−3.02028858	−2.982866
$g_5(X)$	−0.11936898	−0.11943827	−0.12050000	−0.123900
$g_6(X)$	−0.23424083	−0.23423703	−0.23420813	−0.234160
$g_7(X)$	−0.00030900	−13.07930496	−74.27685602	−618.81849251
$f(X)$	**2.3809565827**	**2.3854347**	**2.385937**	**2.433116**

5.5 Example 5: Hydrostatic Thrust Bearing Design

The thrust bearing design problem was proposed by Siddall [39]. This problem aims to minimize power loss associated with the bearing shown in Figure 7 while satisfying several constraints. Four design variables are used: the bearing step radius R, recess radius R_0, oil viscosity μ and flow rate Q. There are seven constraints which limit load-carrying capacity, inlet oil pressure, oil temperature rise, oil film thickness and some physical requirements. The optimization problem can be formulated as follows:

Minimize

$$F(X) = \frac{QP_0}{0.7} + E_f \tag{56}$$

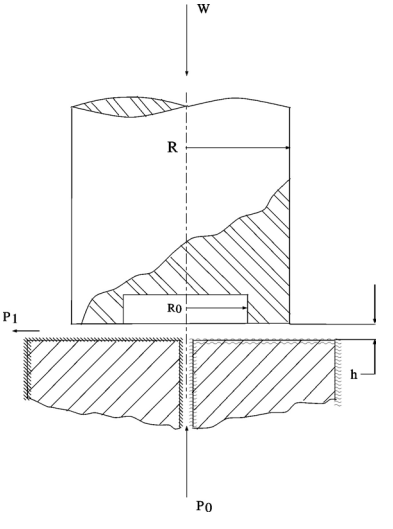

FIGURE 7 Thrust bearing design.

subject to:

$$g_1(X) = W - W_s \leq 0 \tag{57}$$

$$g_2(X) = P_{\max} - P_0 \leq 0 \tag{58}$$

$$g_3(X) = \triangle T_{\max} - P_0 \leq 0 \tag{59}$$

$$g_4(X) = h - h_{\min} \leq 0 \tag{60}$$

$$g_5(X) = R - R_0 \leq 0 \tag{61}$$

$$g_6(X) = 0.001 - \frac{\gamma}{g P_0}\left(\frac{Q}{2\pi R h}\right) \leq 0 \tag{62}$$

$$g_7(X) = 5000 - \frac{W}{\pi(R^2 - R_0^2)} \leq 0 \tag{63}$$

where W is the load carrying capacity which is given by:

$$W = \frac{\pi P_0}{2}\frac{R^2 - R_0^2}{\ln(R/R_0)} \tag{64}$$

and P_0 is the inlet pressure which can be defined as:

$$P_0 = \frac{6\mu Q}{\pi h^3}\ln\frac{R}{R_0} \tag{65}$$

and E_f is the friction loss:

$$E_f = 9336 Q \gamma C \triangle T \tag{66}$$

where $\gamma = 0.0307\,\text{lb/in}^3$ is the weight density of oil and specific heat of oil $C = 0.5\,\text{Btu/lb}\,^\circ\text{F}$. And $\triangle T$ is the temperature which can be estimated by

$$\triangle T = 2(10^P - 559.7) \tag{67}$$

where

$$P = \frac{\log_{10}\log_{10}(8.122 \times 10^6 \mu + 0.8) - C_1}{n} \tag{68}$$

and n and C_1 are constants for a given oil. Table VIII gives n and C_1 for various grades of oil. In this example, SAE 20 grade oil is chosen. Therefore, $n = 10.04$ and $C_1 = -3.55$. The film thickness can be calculated from the friction loss E_f from following equation:

$$h = \left(\frac{2\pi N}{60}\right)^2 \frac{2\pi\mu}{E_f}\left(\frac{R^4}{4} - \frac{R_0^4}{4}\right) \tag{69}$$

Other specification of design are: weight of generator, $W_s = 101000\,\text{lb}$ (45804.99 kg); maximum pressure available, $P_{\max} = 1000\,\text{psi}$ ($6.89655 \times 10^6\,\text{Pa}$); maximum temperature rise $\triangle T_{\max} = 50\,^\circ\text{F}$ (10 °C); minimum oil thickness $h_{\min} = 0.001$ in. (0.00254 cm); $g = 32.3 \times 12 = 386.4\,\text{in./seg}^2$ ($981.465\,\text{cm/seg}^2$) and angular speed of shaft $N = 750$ RPM.

The following ranges were used for the design variables:

$$1.000 \leq R \leq 16.000, \quad 1.000 \leq R_0 \leq 16.000,$$

$$1.0 \times 10^{-6} \leq \mu \leq 16 \times 10^{-6}, \quad 1.000 \leq Q \leq 16.000.$$

TABLE VIII Values of n and C_1 for various grades of oil.

Oil	C_1	n
SAE 5	10.85	−3.91
SAE 10	10.45	−3.72
SAE 20	10.04	−3.55
SAE 30	9.88	−3.48
SAE 40	9.83	−3.46
SAE 50	9.82	−3.44

This problem was tackled by Siddall [39] using ADRANS (Gall's adaptive random search with a penalty function). Deb and Goyal [5] used GeneAS to deal with this problem. Coello Coello [40] proposed a novel constraint handling technique to solve this problem: GASO, which treats constraints as objective functions and solves them with a multi-objective technique.

It is worth noting that there are several discrepancies of units and design specifications between Deb and Coello's papers [5, 40] and Siddall's book [39]. The first one is the absolute temperature (°F °Rankine) of ambient. Deb and Coello used 560.0 while Siddall used 559.7 in Eq. (67). In Siddall's book, the fourth constraint (g_4) and the sixth one (g_6) are multiplied by 10^8, and the fifth constraint and the third one are multiplied by 10^5 and 2000, respectively. The unit of fitness value from Deb and Coello's papers is foot-pounds per second while Siddall used inches-pounds per second. Due to these differences, we adopted two experiments: Case 1 and Case 2, with different units and design specifications. The results are compared against those of Deb and Coello and Siddall, respectively. Each experiment was performed 100 runs. The best solutions for Case 1 and for Deb and Coello's papers are listed in Table IX. The best solutions for Case 2 and Siddall's book are listed in Table X.

The maximum numbers of generations for both cases were set to 3000 with 90,000 evaluations of the fitness function. The average execution time required for both cases were 52.7 and 48.8 s of CPU time, respectively. The average fitness value from the proposed PSO for Case 1 is 1757.376840 with a standard deviation of 316.851024 which is better than most of the best results reported by other techniques depicted in Table IX. The average fitness value for Case 2

TABLE IX Optimal solution of thrust bearing design for Case 1, Coello and Deb's papers.

Design variables	*Best solution found*			
	PSO	*GASO [40]*	*GeneAS [5]*	*BGA [5]*
$x_1(R)$	5.956868685	6.271	6.778	7.7077
$x_2(R_0)$	5.389175395	12.901	6.234	6.549
$x_3(\mu) \times 10^{-6}$	5.40213310	5.605	6.096	6.619
x_4 (Q)	2.30154678	2.938	3.809	4.849
$g_1(X)$	22.01094912	2126.86734	8329.7681	1440.6013
$g_2(X)$	0.00000000	68.0396	177.3527	297.1495
$g_3(X)$	0.58406092	3.705191	10.684543	17.353800
$g_4(X)$	0.00033480	0.000559	0.000652	0.000891
$g_5(X)$	0.56769329	0.666000	0.544000	0.528000
$g_6(X)$	0.00083138	0.000805	0.000717	0.000624
$g_7(X)$	7.61684431	849.718683	83.618221	467.686527
$f(X)$	**1632.2149**	**1950.2860**	**2161.4215**	**2296.2119**

TABLE X Optimal solution of thrust bearing design for Case 2 and Siddall's book.

Design variables	*Best solution found*	
	PSO	*Siddall [39]*
$x_1(R)$	5.956048839021	7.1550805
$x_2(R_0)$	5.388766560465	6.6886822
$x_3(\mu) \times 10^{-6}$	6.001637904878	8.3207655
x_4 (Q)	2.778703032216	9.1684614
$g_1(X)$	0.00000074	71.040915
$g_2(X)$	0.00234129	328.27277
$g_3(X)$	17605.54663647	68912.380
$g_4(X)$	47175.71419475	144524.15
$g_5(X)$	56728.22785557	46639.822
$g_6(X)$	79777.35289299	17724.808
$g_7(X)$	4.54730416	17.287484
$f(X)$	**20374.684**	**29221.321**

is 22874.674800 with a standard deviation of 3140.292915, which is better than the best result reported by Siddall [39].

In order to further illustrate the superiority of our algorithm, both in terms of accuracy and convergence rate, Case 1 of Example 5 is used to compare the proposed algorithm with the modified PSO algorithm of El-Gallad *et al*. [20] and a standard PSO with a static penalty given in Ref. [41]. The average solutions of the three algorithms were obtained after 100 runs where the maximum generation was set to 3000. The major drawback of Ref. [41] is that the static penalty coefficient r_g requires to be fine tuned in order to generate an acceptable result. For El-Gallad's PSO and the standard PSO, the average solutions were 1877.195620 and 2939.070620, respectively, which are worse than the average result of 1757.37684 found by the proposed algorithm. The search processes of these three algorithms are shown in Figure 8. Clearly, from this figure one can see that our algorithm converges more quickly than the algorithms given in Refs. [20, 41].

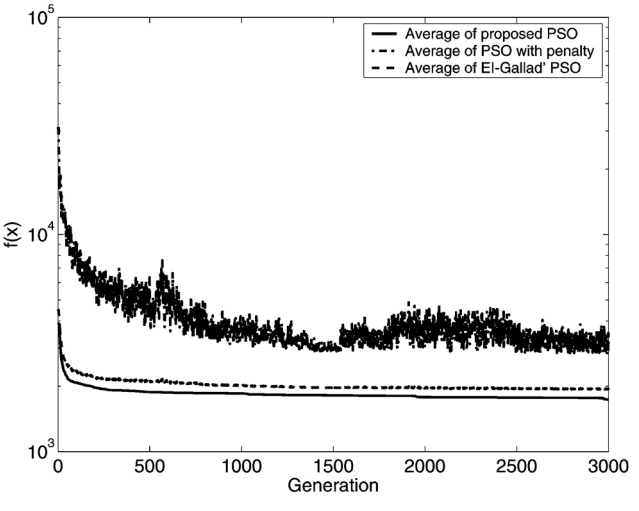

FIGURE 8 Search processes of three algorithms for thrust bearing design Case 1.

6 CONCLUSIONS

In this paper, the standard PSO algorithm has been extended to handle mixed variables and constraints. The proposed method is relatively simple and easy to implement. A 'fly-back mechanism' is proposed to preserve feasible individuals. Compared to other constraint handling techniques based on penalty functions, this method is simpler, faster and provides more reliable solutions without any violation of the constraints.

The proposed PSO algorithm has been applied to solve a mathematical benchmark function and four mechanical design optimization problems. The numerical results obtained by the proposed algorithm are better than or equal to other existing methods. Moreover, for most of our numerical examples, the PSO algorithm with 'fly-back mechanism' converges to the global minima within a few hundred iterations and its computational time is far less than the other PSO algorithms.

A drawback of the proposed PSO is that the constraint handling method requires a feasible initial population. For some problems, finding a feasible solution is NP-hard [42], and even impossible for the problems with conflicting constraints. Future work should extend the proposed PSO to tackle the initial population problem.

Acknowledgement

The authors would like to acknowledge Dr. Carlos Coello for his helpful discussions.

References

[1] Coello Coello, C. A. (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, **191**(11–12), 1245–1287.
[2] Wu, Q. H., Cao, Y. J. and Wen, J. Y. (1998) Optimal reactive power dispatch using an adaptive genetic algorithm. *Electrical Power and Energy Systems*, **20**(8), 563–569.
[3] Wu, Q. H. and Ma, J. T. (1995) Power system optimal reactive power dispatch using evolutionary programming. *IEEE Transactions on Power Systems*, **10**(3), 1243–1249.
[4] Cao, Y. J. and Wu, Q. H. (1999) A mixed variable evolutionary programming for optimisation of mechanical design. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, **7**(2), 77–82.
[5] Deb, K. and Goyal, M. (1997) Optimizing engineering designs using a combined genetic search. In: *Seventh International Conference on Genetic Algorithms*, Ed. I. T. Back, pp. 512–528.
[6] Kennedy, J. and Eberhart, R. (1995) Particle swarm optimization. *IEEE International Conference on Neural Networks*, Vol. 4, IEEE Press, pp. 1942–1948.
[7] Kennedy, J. and Eberhart, R. C. (2001) *Swarm Intelligence*. Morgan Kaufmann Publishers.
[8] Davis, L. (1987) *Genetic Algorithms and Simulated Annealing*. Pitman, London.
[9] Le Riche, R. G., Knopf-Lenoir, C. and Haftka, R. T. (1995) A segregated genetic algorithm for constrained structural optimization. In: *Sixth International Conference on Genetic Algorithms*, University of Pittsburgh, Morgan Kaufmann, pp. 558–565.
[10] Michalewicz, Z. and Schoenauer, M. (1996) Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, **4**(1), 1–32.
[11] Sandgren, E. (1990) Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, **112**, 223–229.
[12] Hajela, P. and Shih, C. (1989) Multiobjective optimum design in mixed-integer and discrete design variable problems. *AIAA Journal*, **28**(4), 670–675.
[13] Parsopoulos, K. E. and Vrahatis, M. N. (2002) Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, **1**, 235–306.
[14] Shi, Y. and Eberhart, R. C. (1997) A modified particle swarm optimiser. *Proc. IEEE Inc. Conf. on Evolutionary Computation*, pp. 303–308.
[15] Shi, Y. and Eberhart, R. C. (1998) Parameter selection in particle swarm optimization. *Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447*, Springer, pp. 591–600.
[16] Eberhart, R. C. and Shi, Y. (2001) Particle swarm optimization: developments, applications and resources. *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp. 81–86.

[17] Chen, J. L. and Tsao, Y. C. (1993) Optimal design of machine elements using genetic algorithms. *Journal of the Chinese Society of Mechanical Engineering*, **12**(2), 193–199.
[18] Thierauf, G. and Cai, J. (1997) Evolution strategies – parallelisation and application in engineering optimization. In: *Parallel and Distributed Processing for Computational Mechanics*, Ed. B. H. V. Topping, Saxe-Coburg Publications.
[19] Tahk, M. and Sun, B. C. (2000) Co-evolutionary augmented Lagrangian methods for constrained optimization. *IEEE Transactions on Evolutionary Computation*, **4**(2), 114–124.
[20] El-Gallad, A. I., El-Hawary, M. E. and Sallam, A. A. (2001) Swarming of intelligent particle for solving the nonlinear constrainted optimization problem. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, **9**(3), 155–163.
[21] Hu, X. and Eberhart, R. C. (2002) Solving constrained nonlinear optimization problems with particle swarm optimization. *Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002)*. Orlando, USA.
[22] Parsopoulos, K. and Vrahatis, M. N. (2002) Particle swarm optimization method for constrained optimization problems. In: *Intelligent Technologies – Theory and Applications: New Trends in Intelligent Technologies, Frontiers in Artificial Intelligence and Applications*, Vol. 76, Ed. V. Sincak, J. Vascak, IOS Press, pp. 214–220.
[23] Ray, T. and Liew, K. M. (2002) A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, **32**(2), 141–153.
[24] Rao, S. S. (1996) *Engineering Optimization*. John Wiley and Sons.
[25] Himmelblau, D. M. (1972) *Applied Nonlinear Programming*. McGraw-Hill, New York.
[26] Gen, M. and Cheng, R. (1997) *Genetic Algorithms and Engineering Design*. John Wiley and Sons.
[27] Runarsson, T. P. and Yao, X. (2000) Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, **4**(3), 284–294.
[28] Koziel, S. and Michalewicz, Z. (1999) Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, **7**(1), 19–44.
[29] Lampinen, J. and Zelinka, I. (1999) Mixed integer-discrete-continuous optimization by differential evolution. In: *Proceedings of the 5th International Conference on Soft Computing*, pp. 71–76.
[30] Belegundu, A. D. (1982) A study of mathematical programming methods for structural optimization. *Technical report*, University of Iowa.
[31] Arora, J. S. (1989) *Introduction to Optimun Design*. McGraw-Hill, New York.
[32] Coello Coello, C. A. (2000) Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, **41**(2), 113–127.
[33] Ray, T. and Liew, K. M. (2003) Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, **7**(4), 386–396.
[34] Coello Coello, C. A. and Mezura Montes, E. (2001) Use of dominance-based tournament selection to handle constraints in genetic algorithms. In: *Intelligent Engineering Systems through Artificial Neural Networks (ANNIE'2001)*, Vol. 11, ASME Press, St. Louis, Missouri, pp. 177–182.
[35] Deb, K. (1997) Geneas: a robust optimal design technique for mechanical component design. In: *Evolutionary Algorithms in Engineering Applications*, Ed. D. Dasgupta and Z. Michalewicz, Springer-Verlag, pp. 497–514.
[36] Ragsdell, K. M. and Phillips, D. T. (1976) Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industries*, **98**(3), 1021–1025.
[37] Deb, K. (1991) Optimal design of a welded beam via genetic algorithms. *AIAA Journal*, **29**(11), 2013–2015.
[38] Deb, K. (2000) An efficient constraint handling method for genetic algorithms. *Computer Methods Applied Mechanics and Engineering*, **186**(2–4), 311–338.
[39] Siddall, J. N. (1982) *Optimal Engineering Design*. Marcel Dekker.
[40] Coello Coello, C. A. (2000) Treating constraints as objectives for single-objective evolutionary optimization. *Eng. Optimization*, **32**(3), 275–308.
[41] Fiacco, A. V. and McCormick, G. P. (1968) *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Wiley, New-York.
[42] Smith, A. E. and Coit, D. W. (1997) Constraint handling techniques – penalty functions. *Handbook of Evolutionary Computation*, Ed. T. Back, D. B. Fogel and Z. Michalewicz, Oxford University Press and Institute of Physics Publishing.