Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100207

A Hybrid PSO and DE Algorithm for Solving Engineering Optimization Problems

Ahmed F. Ali[1,2] *and Mohamed A. Tawhid*[2,3,*]

[1] Department of Computer Science, Faculty of Computers & Informatics, Suez Canal University, Ismailia, Egypt
[2] Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC, V2C 0C8, Canada
[3] Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Moharam Bey 21511, Alexandria, Egypt

Received: 7 Aug. 2015, Revised: 25 Oct. 2015, Accepted: 26 Oct. 2015
Published online: 1 Mar. 2016

Abstract: In this paper, we present a new hybrid swarm optimization and differential evolution algorithm for solving constrained and engineering optimization problems. The proposed algorithm is called hybrid particle swarm optimization and differential evolution with population size reduction (HPSODEPSR). The powerful performance of any metaheuristics algorithm is measured by its capability to balance between the exploration and exploitation process. In the beginning of the search, the algorithm needs to explore the search space with a large number of solutions in the population then during the search the need of the exploration process is reduces while the need of the exposition process increases. From this point, we propose a population size reduction mechanism (PSRM), in PSRM, the proposed algorithm starts with a large number of solutions in the population and during the search the number of these solutions decreases after applying the greedy selection operator in order to remove the worst solutions from the population. Also, we propose a new automatic termination criterion which is called a progress vector V. V is a $(1\times n)$ zero vector, where n equal to the number of population partitions and contains of a number of subsets equal to the number of population reduction steps (partitions), when the population reduced, the corresponding subset value in V converted to one. The algorithm terminates the search when all subsets values in the progress vector become ones. Moreover, we test the proposed algorithm on eleven benchmark functions and five engineering optimization problems. We compare our proposed algorithm against seven algorithms in order to investigate the general performance of it. The numerical experiments show that the proposed algorithm is a promising algorithm and can reach to the optimal or near optimal solution faster than the other comparative algorithms.

Keywords: Particle swarm optimization, differential evolution, constrained/engineering optimization problems, population size reduction, global optimization

1 Introduction

In 1995, James Kennedy and Russell C. Eberhart [17] proposed the particle swarm optimization (PSO) algorithm for optimization problems. PSO is a population-based search algorithm based on the simulation of the social behavior of birds within a flock.

Due to its simplicity, easy implementation, and efficiency, PSO has attracted much attention by many researchers and been successfully applied in a variety of fields [3], [4], [23], [36], [37]. In the original version of PSO, particles fly through the search space influenced by two factors in order to find global optima: each individual's best position ever found (pbest) or local best solution (lbest) and the group's best position ever found (global best solution denoted by gbest).

In gbest, all particles share information with each other and move to global best position. However, gbest has drawback, because it is easy to trapped in local optima. In lbest, a specific number of particles are neighbors to one particle, but it also has drawback, which is the slow of convergence. In order to overcome these drawbacks, many researchers tried to improve the performance of PSO by combining it with other algorithms, for example genetic algorithm (GA).

Robinson et al. [29] proposed two hybrid PSO and GA algorithms for solving a particular electromagnetic application of profiled corrugated horned antenna. The

[*] Corresponding author e-mail: Mtawhid@tru.ca

first algorithm is called GA-PSO, while the second algorithm is called PSO-GA. In PSO-GA algorithm, PSO generates an initial population for GA, while in GA-PSO algorithm, the initial population is generated by using GA for PSO. The final results showed that the PSOGA hybrid algorithm outperforms the GA-PSO version as well as simple PSO and GA versions.

Also, in [20], Krink and Lvbjerg hybridized PSO with the GA and Hill Climbing approaches and was applied for solving unconstrained global optimization problems. The authors showed that the proposed algorithm can be applied to a different sub-population of individuals in which each individual is dynamically assigned according to some pre-designed rules.

Grimaldi et al. [10] proposed a hybrid technique combining GA and PSO called genetical swarm optimization (GSO) for solving combinatorial optimization problems. They applied GSO algorithm for solving an electromagnetic optimization problem. In GSO, the population is divided into two parts and is evolved with the GA and PSO algorithms at each iteration. The populations are then recombined in the updated population, and divided again randomly into two parts in the next iteration for another run of genetic or particle swarm operators. In GSO, a new parameter HC (or Hybridization Constant) has been defined that expresses the percentage of population that is evolved with GA in every iteration.

Gandelli et al. [9] proposed several hybridization strategies (static, dynamic, alternate, self adaptive etc.) for GSO algorithm and validated them with some multimodal benchmark problems.

There are some applications of GSO that [7], [8] and [9]. Juang [15] proposed another hybridization strategy of PSO and GA (HGAPSO) to neural/fuzzy network design, where the upper half of the best performing individuals in a population is regarded as elite solutions. Before using GA operators, the algorithm was first enhanced by means of PSO, instead of being reproduced directly to the next generation.

In [30], Settles and Soule proposed a breeding swarm (BS) algorithm for solving four unconstrained optimization problems with different dimensions by hybrid GA and PSO. The BS algorithm combines the standard velocity and position update rules of PSOs with the GAs selection, crossover and mutation.

In [14], Jian and Chen introduced a PSO hybrid with the GA recombination operator and dynamic linkage discovery called PSO-RDL for solving 25 unconstrained test problems with varying degrees of complexities. They assumed that the relation between different dimensions is dynamically changed along with the search process and updated the linkage configuration according to the fitness feedback.

Mohammadi and Jazaeri [26] proposed a hybrid algorithm for solving an IEEE 68 bus system in which the initial population for GA is generated by PSO. Esmin et al. [6] proposed a HPSOM algorithm for solving unconstrained global optimization problems by combining PSO algorithm with a GA mutation operator only.

Kim [18] proposed an improved GA, called GA-PSO for obtaining the local and global optima of Foxhole function by using PSO and the concept of Euclidean distance. In GA-PSO, the performance of GA was improved by using PSO and Euclidean data distance on mutation procedure of GA. In the PSO-GA, Yang et al. [35] proposed a hybrid evolutionary algorithm (HEA) for solving 3 unconstrained as well as 3 constrained optimization problems by dividing the evolution into two stages. The first stage is similar to the standard PSO algorithm. The second stage is similar to GA where genetic operators of selection, reproduction, crossover, and mutation are exerted on particles at predetermined probability. They used a single point crossover, Gaussian mutation and Roulette wheel for selection process.

In the last few years, many researchers have applied PSO to solve constrained optimization problems (COPs) because of its success in solving unconstrained optimization problems. For example, Liang and Suganthan [24] proposed a new dynamic multi-swarm PSO with a novel constraint to solve COPs and Krohling and Coelho [21] proposed a co-evolutionary PSO based on Gaussian distribution for solving constrained optimization by generating the acceleration coefficients using Gaussian probability distribution.

He and Wang [11] proposed another co-evolution PSO algorithm by coping it with both decision variables and constraints. In this algorithm, the swarm is divided in two parts: searching good solutions and optimizing appropriate penalty factor, respectively. He and Wang [12] proposed a hybrid PSO with feasibility-based rule [5] by implementing the feasibility based rule to update the personal best of each particle in the swarm and the applied simulated annealing algorithm in order to avoid premature convergence.

Pulido and Coello [28] proposed a simple mechanism to handle constraints with PSO. In their proposed mechanism if the particles compared are infeasible, the best particle is the particle with the lower value in its normalized violation of constraints.

Wang et al. [33] proposed a hybrid multi-swam particle swarm optimization (HMPSO) for solving constrained optimization problems by splitting the swarm into several sub-swarms and each sub-swarm evolves independently. HMPSO uses the feasibility based rule to compare particles in the swarm.

Inspired by the paper in [33], we propose a new hybrid particle swarm optimization and deferential evolution algorithm to solve COPs. The proposed algorithm is called hybrid particle swarm optimization and differential evolution with population size reduction (HPSODEPSR).

In the proposed algorithm, the particle swarm optimization algorithm is used to explore the search space while the DE mutation is used to update the best personal particle at each iteration. Also, we propose a new population size reduction mechanism to control the number of the particles during the search process. In the beginning of the search, the algorithm needs to make exploration process with large number of particles (solutions) then during the search, the need of the exploration is reduced and the need of the exploitation increases which needs lower number of solutions. In order to monitor the search process and control the population reduction process, we propose a progress vector V which is a zero vector and each subset corresponds to population reduction stage (partition). When the algorithm reaches to the stage of getting the desired value, then the value of the corresponding subset converted from zero to one and the population size is reduced by applying the greedy selection operator. Furthermore, we test the proposed algorithm on eleven benchmark functions and compare against seven algorithms. The experimental results show that the proposed algorithm is a promising algorithm and can obtain the optimal or near optimal solution in reasonable time.

The rest of this paper is organized as follows. In Section 2, we present the definition of the constrained optimization problem. We summarize the main concepts of the particle swarm optimization and the differential evolution algorithm in Section 3 and 4, respectively. In Section 5, we describe the population size reduction mechanism. In Section 6, we highlight the proposed algorithm and its main structure. In Section 7, we report the experimental results and finally, the conclusion makes up Section 8.

2 Constrained optimization problems

The constrained optimization problems and constraint handling is one of the most challenging in many applications. A general form of a constrained optimization is defined as follows:

$$\begin{aligned}&\text{Minimize}\quad f(x), x=(x_1,x_2,\cdots,x_n)^T, \qquad (1)\\&\text{Subject to}\\&\qquad g_i(x)\le 0, i=1,\cdots,m\\&\qquad h_j(x)=0, j=1,\cdots,l\\&\qquad x_l\le x_i\le x_u\end{aligned}$$

Where $f(x)$ is the objective function, x is the vector of n variables, $g_i(x)\le 0$ are inequality constraints, $h_j(x)=0$ are equality constraints, x_l, x_u are variables bounds. Many techniques were proposed in order to handle constraints. Michalewicz [27] grouped them into the following categories: penalty function technique; rejection of infeasible solutions technique; repair algorithms technique; specialized operators technique; and behavior memory technique.

2.0.1 The Penalty function method

The penalty function method is used to transform the constrained optimization problems to unconstrained optimization problem by penalizing the constraints and forming a new objective function as follows:

$$f(x)=\begin{cases}f(x) & \text{if } x\in \text{feasible region}\\ f(x)+penalty(x) & x\notin \text{feasible region.}\end{cases} \qquad (2)$$

Where,

$$\text{penalty(x)}=\begin{cases}0 & \text{if no constraint is violated}\\ 1 & \text{otherwise.}\end{cases}$$

There are two different types of points in the search space of the constrained optimization problems (COP), namely, feasible points which satisfy all constraints and infeasible points which violate at least one of the constraints. At the feasible points, the penalty function value is equal the value of objective function, but at the infeasible points the penalty function value is equal to a high value as in (2). In this paper, a non stationary penalty function has been used, which the values of the penalty function are dynamically changed during the search process. A general form of the penalty function is defined in [34] as the following:

$$F(x)=f(x)+h(k)H(x),\quad x\in S\subset\mathbb{R}^n, \qquad (3)$$

where $f(x)$ is the objective function, $h(k)$ is a non stationary (dynamically modified) penalty function, k is the current iteration number and $H(x)$ is a penalty factor.

3 Particle swarm optimization

In the following subsection, we will give the main concepts and structure of the particle swarm optimization algorithm.

3.1 Main concepts

Particle swarm optimization (PSO) is a population based method that inspired from the behavior (information exchange) of the birds in a swarm [16]. In PSO the population is called a swarm and the individuals are called particles. In the search space, each particle moves with a velocity. The particle adapt his velocity due to the information exchange between it and other neighbors. At each iteration, the particle uses a memory in order to save its best position and the overall best particles positions. The best particle position is saved as a best local position, which assigned to a neighborhood particles, while the overall best particles position is saved as a best global position, which assigned to all particles in the swarm.

3.2 Particle movement and velocity

Each particle is represented by a D dimensional vectors,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in S. \tag{4}$$

The velocity of the initial population is generated randomly and each particle has the following initial velocity:

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{iD}). \tag{5}$$

The best local and global positions are assigned, where the best local position encounter by each particle is defined as

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{iD}) \in S. \tag{6}$$

At each iteration, the particle adjust it's personal position according to the best local position (pbest) and the overall (global) best position (gbest) among particles in its neighborhood as fellow.

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}, \quad i = 1, \ldots, P \tag{7}$$

$$\begin{aligned} v_i^{(t+1)} &= v_i^{(t)} + c_1 r_{i1} \times (pbest_i^{(t)} - x_i^{(t)}) \\ &+ c_2 r_{i2} \times (gbest - x_i^{(t)}). \end{aligned} \tag{8}$$

where c_1, c_2 are two acceleration constants called cognitive and social parameters, r_1, r_2 are random vector $\in [0,1]$.

3.3 Particle swarm optimization algorithm

We can summarize the main steps of the PSO algorithm as follows.

- **–Step 1.** The algorithm starts with the initial values of swarm size P, acceleration constants c_1 and c_2.
- **–Step 2.** The initial position and velocity of each solution (particle) in the population (swarm) are randomly generated as in (4) and (5).
- **–Step 3.** Each solution in the population is evaluated by calculating its corresponding fitness value $f(x_i)$.
- **–Step 4.** The best personal solution *pbest* and the best global solution *gbest* are assigned.
- **–Step 5.** The following steps are repeated until the termination criterion is satisfied

  Step 5.1. At each iteration t, the position of each particle x_i^t is justified in (7), while the velocity of each particle v_i^t is justified in (8).

  Step 5.2. Each solution in the population is evaluated $f(x_i)$ and the new best personal solution *pbest* and best global solution *gbest* are assigned.

  Step 5.3. The operation is repeated until the termination criteria are satisfied.
- **–Step 6.** Produce the best found solution so far.

Algorithm 1 Particle swarm optimization algorithm

```
1: Set the initial value of the swarm size SS, acceleration constants $c_1$ and $c_2$.
2: Set $t := 0$.
3: Generate random $x_i^{(t)}$ and $v_i^{(t)} \in [L,U]$ where $i = 1,\ldots,SS$. {$SS$ is the population (swarm) size}.
4: Evaluate the fitness function $f(x_i^{(t)})$.
5: Set $gbest^{(t)}$. {$gbest$ is the best global solution in the swarm}.
6: Set $pbest_i^{(t)}$. {$pbest_i^{(t)}$ is the best local solution in the swarm}.
7: repeat
8:    $v_i^{(t+1)} = v_i^{(t)} + c_1 r_{i1} \times (pbest_i^{(t)} - x_i^{(t)}) + c_2 r_{i2} \times (gbest - x_i^{(t)})$. {$r_1$ and $r_2$ are random vectors $\in [0,1]$}.
9:    $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$, $i = 1,\ldots,SS$. {Update particles positions}.
10:   Evaluate the fitness function $f(x_i^{(t+1)})$, $i = 1,\ldots,SS$.
11:   if $f(x_i^{(t+1)}) \le f(pbest_i^{(t)})$ then
12:      $pbest_i^{(t+1)} = x_i^{(t+1)}$.
13:   else
14:      $pbest_i^{(t+1)} = pbest_i^{(t)}$.
15:   end if
16:   if $x_i^{(t+1)} \le f(gbest^{(t)})$ then
17:      $gbest^{(t+1)} = x_i^{(t+1)}$.
18:   else
19:      $gbest^{(t+1)} = gbest^{(t)}$.
20:   end if
21:   Set $t = t+1$. {Iteration counter increasing}.
22: until Termination criteria are satisfied.
23: Produce the best particle.
```

4 Differential evolution algorithm

Differential evolution algorithm (DE) was proposed by Stron and Price in 1997 [31]. In DE, the initial population consists of number of individuals, which is called a population size P. Each individual in the population size is a vector consists of D dimensional variables and can be defined as follows:

$$\mathbf{x}_i^{(t)} = \{x_{i,1}^{(t)}, x_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}\}, \quad i = 1,2,\ldots,P \tag{9}$$

where t is a generation number, D is a problem dimensional number and P is a population size. DE employs mutation and crossover operators in order to generate a trail vectors, then the selection operator starts to select the individuals in new generation *t+1*. We present in details the overall process as follows:

4.1 Mutation operator

Each vector $\mathbf{x}_i$ in the population size creates a trail mutant vector $\mathbf{v}_i$ as follows.

$$\mathbf{v}_i^{(t)} = \{v_{i,1}^{(t)}, v_{i,2}^{(t)}, \ldots, x_{i,D}^{(t)}\}, \quad i = 1,2,\ldots,P. \tag{10}$$

DE applies different strategies to generate a mutant vector as follows:

$$DE/rand/1: \mathbf{v}_i^{(t)} = \mathbf{x}_{r_1}^{(t)} + F \cdot (\mathbf{x}_{r_2} + \mathbf{x}_{r_3}) \tag{11}$$

$$DE/best/1: \mathbf{v}_i^{(t)} = \mathbf{x}_{best}^{(t)} + F \cdot (\mathbf{x}_{r_1} + \mathbf{x}_{r_2}) \tag{12}$$

$$DE/currenttobest/1: \mathbf{v}_i^{(t)} = \mathbf{x}_i^{(t)} + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \tag{13}$$

$$DE/best/2: \mathbf{v}_i^{(t)} = \mathbf{x}_{best}^{(t)} + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) + F \cdot (\mathbf{x}_{r_3} - \mathbf{x}_{r_4}) \tag{14}$$

$$DE/rand/2: \mathbf{v}_i^{(t)} = \mathbf{x}_{r_1}^{(t)} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) + F \cdot (\mathbf{x}_{r_4} - \mathbf{x}_{r_5}) \tag{15}$$

where the indexes r_d, $d = 1, 2, \ldots, 5$ represent the random and mutually different integers generated within the range $[1, P]$ and and also different from index i. F is a mutation scale factor within the range $[0, 2]$. $\mathbf{x}_{best}^{(t)}$ is the best vector in the population in the current generation t.

4.2 Crossover operator

A crossover operator starts after mutation in order to generate a trail vector according to target vector $\mathbf{x}_i$ and mutant vector $\mathbf{v}_i$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise.} \end{cases} \tag{16}$$

where CR is a crossover control parameter or factor within the range $[0, 1]$ and presents the probability of creating parameters for a trial vector from the mutant vector. Index j_{rand} is a randomly chosen integer within the range $[1, P]$.

4.3 Selection operator

The DE algorithm applies greedy selection. The selection operator selects between the trails and targets vectors. The selected individual (solution) is the best vector with the better fitness value. We present the description of the selection operator as follows.

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{u}_i^{(t)}, & \text{if } f(\mathbf{u}_i^{(t)}) \le f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i, & \text{otherwise.} \end{cases} \tag{17}$$

Algorithm 2 Differential evolution algorithm

```
1:  Set the generation counter $t := 0$.
2:  Set the initial value of $F$ and $CR$.
3:  Generate randomly an initial population $Pop^0$.
4:  Evaluate the fitness function of all individuals in $Pop^0$.
5:  repeat
6:    Set $t = t + 1$. {Generation counter increasing}.
7:    for $i = 0; i < P; i++$ do
8:      Select random indexes $r_1, r_2, r_3$, where $r_1 \neq r_2 \neq r_3 \neq i$.
9:      $\mathbf{v}_i^{(t)} = \mathbf{x}_{r_1}^{(t)} + F \times (\mathbf{x}_{r_2}^{(t)} - \mathbf{x}_{r_3}^{(t)})$. {Mutation operator}.
10:     $j = rand(1, D)$
11:     for $(k = 0; k < D; k++)$ do
12:       if $(rand(0,1) \le CR$ or $k = j$ then
13:         $u_{ik}^{(t)} = v_{ik}^{(t)}$ {Crossover operator}
14:       else
15:         $u_{ik}^{(t)} = x_{ik}^{(t)}$
16:       end if
17:     end for
18:     if $(f(\mathbf{u}_i^{(t)}) \le f(\mathbf{x}_i^{(t)}))$ then
19:       $\mathbf{x}_i^{(t+1)} = \mathbf{u}_i^{(t)}$ {Greedy selection}.
20:     else
21:       $\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)}$
22:     end if
23:   end for
24: until Termination criteria are satisfied.
```

4.4 Differential evolution algorithm

In this subsection, we highlight the main steps of the DE algorithm which are depicted in Algorithm 2.

–**Step 1.** The algorithm starts by setting the initial values of the iteration parameter t, mutation scale factor F and crossover factor. **Lines(1,2)**

–**Step 2.** The initial population is generated and each solution in the population is evaluated by calculating its fitness function. **Lines(3,4)**

–**Step 3.** The algorithm starts its main loop by selecting three different solutions x_{ri} from the population, where $r_1 \neq r_2 \neq r_3$ in order to apply mutation operator at each solution in the population as in (11) and (15).

–**Step 4.** The trail solutions are randomly generated after applying the crossover operator by comparing each variable in the trail solution (after applying mutation operator) and its corresponding variable in the original solution, then the new solution is generated with new variables if its fitness function is better than the corresponding variables in the original solution. **Lines (11-17)**.

–**Step 5.** The trail solutions that generated from the crossover operator are accepted to participate in the next iteration if their fitness function are better than the original solutions in the population, this kind of selection is called greedy selection. **Lines (18-22)**.

–**Step 6.** The overall process is repeated until the termination criteria are satisfied. **Line (24)**.

5 Population size reduction mechanism

The proposed population size reduction mechanism (PSRM) helps the proposed algorithm to accelerate the search and explore the search space. In the first stage, iterations of the algorithm with large number of solutions in the population will be reduced during the search since the need of the exploration process is decreased while the need of exploitation process is increased.The PSRM depends on the following two techniques

–balancing between the exploration and exploitation during the search process

–Apply automatic termination criterion by using a zero's vector V to monitor the progress of the algorithm and terminate the search when all subset in V Convert from zero's to one's.

The main steps of the proposed PSRM mechanism are presented in Algorithm 3 and Figure 1.

Algorithm 3 Population size reduction mechanism

```
1: Set the values of $part_{no}$, $popsize$,
2: Set the reduction counter $K := 0$.
3: Set the initial value of reduction vector $V$, where $V = zero(1, part_{no})$.
4: Evaluate all individuals in the population $P$ and assign the global best solution $Gbest$ in the population.
5: Set $PRP = gbest$.
6: Set $popred = popsize/part_{no}$
7: Set $RF = (Gbest - optsol)/part_{no}$.
8: for ($i = 1; i \leq part_{no}; i++$) do
9:     Set $PRP = PRP - RF$
10:    Set $List(i) = PRP$
11: end for
12: repeat
13:    Set $K = 1$
14:    if ($Gbest \leq List(K)$) then
15:        Evaluate and sort the population $P$
16:        Remove the worst $popred$ solutions from the population
17:        Set $popsize = popsize - popred$ {Greedy selection}
18:        Update the population $P$
19:        Set $V(1,K) = 1$
20:        Set $K = K + 1$
21:    end if
22: until $popsize < popred$.
```

We can describe the main steps in Algorithm 3 as follows.

–**Step 1.** The PSRM algorithm starts by setting the values of $part_{no}$, $popsize$, and reduction counter K. **Lines(1,2)**.

–**Step 2.** The PSRM algorithm uses a special zero vector with size $(1, part_{no})$, called progress vector V, in order to monitor the progress of the search and terminate the search as soon as all subsets converted from zeros to ones. **Line 3**.

–**Step 3.** The fitness function of each solution in the population is calculated and the best overall solution value is assigned to $gbest$. **Lines (4,5).**

–**Step 4.** The $popred$ variable is calculated and determined the number of the worst solutions which are removed from the population as soon as the subset with zero value in the progress vector converted to one. **Line 6**

–**Step 5.** The reduction scale is assigned by using a reduction factor RF, which is calculated by dividing the subtracting of the initial $Gbest$ value from the optimal solution (given with each tested function) by the number of partition numbers $part_{no}$. **Line 7**.

–**Step 6.** After assigning the scale of reduction, the population reduction points PRP are assigned and saved in a list in order to know at which function value the algorithm has to reduce the number of the population and update the subset in the progress vector from zero to one. **Lines (8-11)**

–**Step 7.** The algorithm starts to monitor the progress of the search, checks the $gbest$ with the current PRP in the list, and applies the greedy selection on the population in order to remove the worst solutions from the population. The progress vector monitors the search progress and terminates the search when its all subsets values converted from zeros to ones and the population is updated. **Lines (12-22)**

In Figure 1, we give an example when $part_{no} = 5$ to explain the PSRM algorithm. There are 5 stages in order to reduce the population size P from P_1 to P_5, where P_1 is the full population size and P_5 is the minimum population size after removing all worst solutions. As shown in Figure 1, the progress vector is initialized with zeros subsets and each subset value convert to one when the population size reduced. The overall process is repeated until all the progress vector subsets converted to ones, then the algorithm is terminated and search is stopped. Finally, the algorithm produces the best solution "Best".

6 The proposed HPSODEPSR algorithm

In this section, we present in details the main steps of the proposed HPSODEPSR algorithm as shown in Algorithm 4 and the Flowchart in Figure 2. Before we give a description of the main steps of the proposed algorithm, we highlight on how the proposed algorithm can handle the constraints when the variables violate the constraints.

6.1 Constraints handling

All particles positions and velocities in the population are updated when we use the PSO algorithm. At each particle if the variable value $x_{ij}^{(t+1)}$ violates the boundary constraints the violated variable value reflected back from

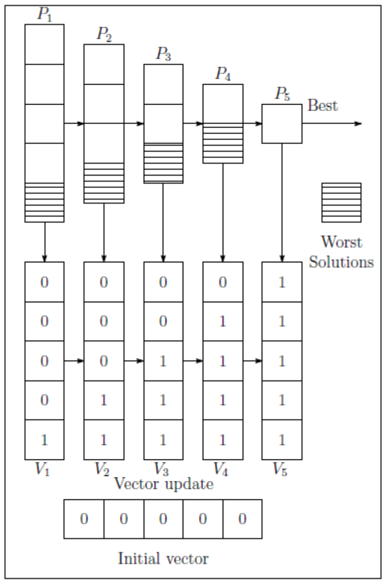

Fig. 1: Population size reduction mechanism

the violated boundary as follow [33].

$$x_{i,j}^{(t+1)} = \begin{cases} 0.5(x_{(i,j)}^t + L_j, & \text{if } x_{(i,j)}^t < L_j, \\ 0.5(x_{(i,j)}^t + U_j, & \text{if } x_{(i,j)}^t > U_j, \end{cases} \tag{18}$$

where L,U are the lower and upper bound for each function.

At each iteration, the best personal particles *pbest* are saved in a list, then the DE mutation operator starts to update these particles in order to improve their positions (values) by generating a trail solutions $Z_{(i,j)}^t$. If the variable in each *pbest* particle (solution) violates the boundary constraints, then the violated variable value reflected back from the violated boundary as follows [33]

$$z_{i,j}^{(t)} = \begin{cases} L_j & \text{if } (r \le 0.5) \wedge (z_{(i,j)}^t < L_j), \\ U_j & \text{if } (r \le 0.5) \wedge (z_{(i,j)}^t > U_j), \\ 2L_j - z_{(i,j)}^t & \text{if } (r > 0.5) \wedge (z_{(i,j)}^t < L_j), \\ 2U_j - z_{(i,j)}^t & \text{if } (r > 0.5) \wedge (z_{(i,j)}^t > U_j), \end{cases} \tag{19}$$

where r is a uniform distributed random number and $r \in [0,1]$.

6.2 The proposed HPSODEPSR algorithm

In the following subsection, we describe the proposed algorithm in more details as follows.

–**Step 1.** HPSODEPSR algorithm starts with the initial values of the acceleration constants c_1 and c_2, mutation scale factor F, partitions number pop_{no}, and the initial value of the iteration counter t. **Lines (1,2)**

–**Step 2.** The initial position and velocity for each solution (particle) in the population (swarm) is generated randomly as in (4) and (5). **Lines (3,4)**

–**Step 3.** Each solution in the population is evaluated by calculating its corresponding fitness value $f(x_i)$. **Line 5**

–**Step 4.** The best personal solution *pbest* and the best global solution *gbest* are assigned. **Lines (6,7)**

–**Step 5.** The following steps are repeated until the termination criterion satisfied

Step 5.1. At each iteration t, the position of each particle $x_i^{(t+1)}$ is justified as in (7), while the velocity of each particle $v_i^{(t+1)}$ is justified as in (8). **Lines (10,11)**

Step 5.2. At each variable in solution $x_i^{(t+1)}$, if the variable violate the constraints, then the violated variables are reflected back as in (18). **Line 12**

Step 5.3. The DE mutation is applied on each best personal solution *pbest* in the population and the trail solutions are generated. **Lines (13, 14)**

Step 5.4. The new trail solution is assigned to the personal best solation *pbest* if its value is better than the current *pbest*. **Lines (15-19)**

–**Step 6.** The variables on each *pbest* in the personal best solution list are reflected back as in (19) if their values violate the constraints. **Line 20**

–**Step 7.** The population size reduction mechanism PSRM as shown in Algorithm 3 is applied in order to reduce the population size when the proposed algorithm produce progress during the search. **Line 22**

–**Step 8.** if all subsets in the progress vector V converted to ones instead of zeros, the algorithm terminates the

Algorithm 4 Hybrid particle swarm optimization and deferential evolution with population size reduction algorithm

```
1: Set the values of the acceleration constants $c_1$ and $c_2$, amplification factor $F$ and partitions number $pop_{no}$.
2: Set $t := 0$.
3: Generate the initial population (swarm) $P$ and $x_i^{(t)}$ particles (solutions).
4: Generate (randomly) the velocity $v_i^{(t)}$ of each particle in the population, where $x_i, v_i \in [L,U]$ and $i = 1,\dots,P$. {$P$ is the population (swarm) size}.
5: Evaluate the fitness function $f(x_i^{(t)})$.
6: Set $gbest^{(t)}$. {$gbest$ is the best global solution in the swarm}.
7: Set $pbest_i^{(t)}$. {$pbest_i^{(t)}$ is the best local solution in the swarm}.
8: repeat
9:   for ($i = 0; i < P; i{+}{+}$) do
10:    $v_i^{(t+1)} = v_i^{(t)} + c_1 r_{i1} \times (pbest_i^{(t)} - x_i^{(t)}) + c_2 r_{i2} \times (gbest - x_i^{(t)})$. {$r_1, r_2$ are random vectors $\in [0,1]$}.
11:    $x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$, $i = 1,\dots,P$. {Update particles positions}.
12:    Check variables violation as shown in Equation 18
13:    Apply a DE mutation operator on $Pbest_i^{(t+1)}$ for each particle $x_i^{(t+1)}$ on the population.
14:    Set the trail vector $u_i^{(t+1)}$ equal to the DE mutation output
15:    if $u_i^{(t+1)} < pbest_i^{(t+1)}$ then
16:      Set $Pbest_i^{(t+1)} = u_i^{(t+1)}$
17:    else
18:      Set $Pbest_i^{(t+1)} = pbest_i^{(t+1)}$
19:    end if
20:    Check the variable violation on each variable on the $pbest_i^{(t+1)}$ as in (19).
21:  end for
22:  Apply the population size reduction mechanism as shown in Algorithm 3.
23:  Set $t = t + 1$. {Iteration counter increasing}.
24: until Termination criteria are satisfied.
25: Produce the best particle.
```

search and the best solution is presented. Otherwise the algorithm terminates the search when it reaches to the standard termination criterion which is the maximum number of iterations.

7 Numerical experiments

In order to investigate the efficiency of the HPSODEPSR, we present the general performance of it by applying HPSODEPSR on 5 engineering optimization problems and 11 benchmark functions. The results of the proposed algorithm have been compared against 7 algorithms. HPSODEPSR was programmed by MATLAB, the results of the comparative algorithms are taken from their original papers. In the following subsections, the parameter setting of the proposed algorithm in more details and the properties of the applied test functions have been reported. Also, the performance analysis of the proposed algorithm is presented with the comparative results against the other algorithms.

7.1 Parameter setting

The parameters of the HPSODEPSR algorithm have been summarized with their assigned values in Table 1.

Table 1: Parameter setting.

Parameters	Definitions	Values
P	population size	60
$Part_{no}$	partition numbers	4
PRS	population reduction scale	$P/part_{no}$
c_1	acceleration constant for cognition part	0.5
c_2	acceleration constant for social part	1.5
F	amplification factor	0.7
δ	equality constraint constant	0.00001
max_{itr}	maximum iteration number	3000

–**Population size** P. The experimental tests show that the best population size is $P = 60$¿ Note that increasing this number, it will increase the evaluation function values without any improvement in the obtained results.

–**Partitions number** $Part_{no}$.$Part_{no}$ is the maximum number of partitions that the proposed algorithm applied on the population. The experimental results show that the best number of the applied partitions is 4. The general performance of the proposed algorithm with different population partitioning number is reported in Table 3.

–**Population reduction scale** PRS. The proposed algorithm applies the greedy selection where the worst solutions in the population are discarded while the best solutions will be remained to the next iterations. The population reduction scale PRS controls the number of the of discarded solutions while it is equal to the number of the population size/the number of partitions.

–**Acceleration constant** c_1 **and** c_2. The parameters c_1 and c_2 are acceleration constants, they are a weighting stochastic acceleration, which pull each particle towards personal best and global best positions. We set the values of c_1 and c_2 to 0.5 and 1.5, respectively.

–**Mutation scale factor F** is an mutation factor in the mutation operator of the deferential evolution algorithm.

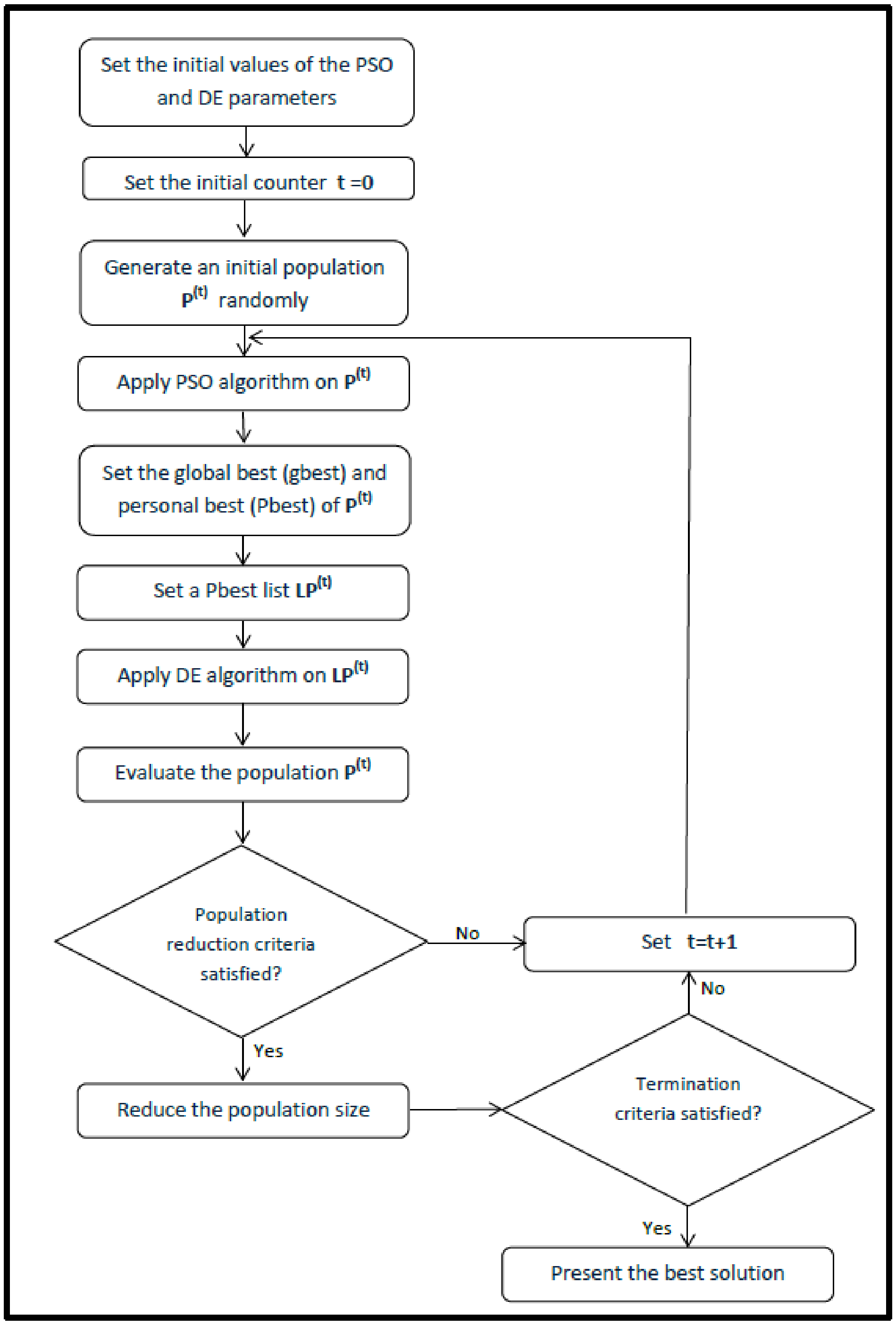
Set the initial values of the PSO and DE parameters
Set the initial counter t =0
Generate an initial population P(t) randomly
Apply PSO algorithm on P(t)
Set the global best (gbest) and personal best (Pbest) of P(t)
Set a Pbest list LP(t)
Apply DE algorithm on LP(t)
Evaluate the population P(t)
Population reduction criteria satisfied?
No
Set t=t+1
No
Yes
Reduce the population size
Termination criteria satisfied?
Yes
Present the best solution

Fig. 2: HPSODEPSR flowchart

–**Equality constraint constant** δ All equality constraints have been converted into inequality by using the degree of violation $\delta = 0.0001$.
–**Maximum iteration number** max_{itr}. HPSOAC terminates the search by applying two termination criteria. The first one is when the population vector converted from zeros to ones while the second termination criterion is the maximum number of iterations reaches to 3000.

7.2 Test constrained optimization problems

HPSODEPSR algorithm has been tested on 11 constrained optimization functions. These functions are listed as the following.
Test problem 1. This problem is defined by

$$\begin{aligned} &Minimize \quad f_1(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\ &Subject \quad to \\ &\quad g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \\ &\quad g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0 \\ &\quad g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \\ &\quad g_4(x) = -8x_1 + x_{10} \le 0 \\ &\quad g_5(x) = -8x_2 + x_{11} \le 0 \\ &\quad g_6(x) = -8x_3 + x_{12} \le 0 \\ &\quad g_7(x) = -2x_4 - x_5 + x_{10} \le 0 \\ &\quad g_8(x) = -2x_6 - x_7 + x_{11} \le 0 \\ &\quad g_9(x) = -2x_8 - x_9 + x_{12} \le 0 \end{aligned}$$

where $0 \le x_i \le 1$ when $i = 1, \ldots, 9$, $0 \le x_i \le 100$ when $i = 10, 11, 12$ and $0 \le x_{13} \le 1$.
The global minimum is $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$ where $f(x^*) = -15$.

Test problem 2. This problem is defined by

$$\begin{aligned} &Maximize \quad f_2(x) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \\ &Subject \quad to \\ &\quad g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \le 0 \\ &\quad g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0 \end{aligned}$$

where $n = 20$ and $0 \le x_i \le 10$ whne $i = 1, \ldots, n$.
The global maximum is unknown; the best reported solution is $f(x^*) = 0.803619$.

Test problem 3. This problem is defined by

$$\begin{aligned} &Maximize \quad f_3(x) = (\sum n)^2 \prod_{i=1}^{n} x_i \\ &Subject \quad to \\ &\quad h(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \end{aligned}$$

where $n = 10$ and $0 \le x_i \le 10$ when $i = 1, \ldots, n$.
The global maximum is $f(x^*) = 1$.

Test problem 4. This problem is defined by

$$\begin{aligned} &Minimize \quad f_4(x) = 5.358547x_3^2 + 0.8356891x_1x_5 \\ &\qquad\qquad + 37.293239x_1 - 40792.141 \\ &Subject \quad to \\ &\quad g_1(x) = 85.334407 + 0.0056858x_2x_5 \\ &\qquad + 0.0006262x_1x_4 + 0.0022053x_3x_6 \\ &\qquad \le 92 \\ &\quad g_2(x) = -85.334407 + 0.0056858x_2x_5 \\ &\qquad - 0.0006262x_1x_4 + 0.0022053x_3x_6 \\ &\qquad \le 0 \\ &\quad g_3(x) = 80.51249 + 0.0071317x_2x_5 \\ &\qquad + 0.0029955x_1x_2 + 0.0021813x_3^2 \\ &\qquad - 110 \le 0 \\ &\quad g_4(x) = -80.51249 - 0.0071317x_2x_5 \\ &\qquad - 0.0029955x_1x_2 - 0.0021813x_3^2 \\ &\qquad + 90 \le 0 \\ &\quad g_5(x) = 9.300961 + 0.0047026x_3x_5 \\ &\qquad + 0.0012547x_1x_3 + 0.0019085x_3x_4 \\ &\qquad - 25 \le 0 \\ &\quad g_6(x) = -9.300961 - 0.0047026x_3x_5 \\ &\qquad - 0.0012547x_1x_3 - 0.0019085x_3x_4 \\ &\qquad + 20 \le 0 \end{aligned}$$

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$ and $27 \le x_i \le 45$ when $i = 3, 4, 5$.
The global minimum is $f(x^*) = -30665.539$.

Test problem 6. This problem is defined by

$$\begin{aligned} &Minimize \quad f_6(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \\ &Subject \quad to \\ &\quad g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0 \\ &\quad g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0 \end{aligned}$$

where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The global minimum is $f(x^*) = -6961.81388$.

Test problem 7. This problem is defined by

$$\begin{aligned} Minimize \quad f_7(x) = & x_1^2+x_2^2+x_1x_2-14x_1-16x_2 \\ & +(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2 \\ & +2(x_6-1)^2+5x_7^2+7(x_8-11)^2 \\ & +2(x_9-10)^2+(x_{10}-7)^2+45. \end{aligned}$$

$Subject \quad to$

$$\begin{aligned} g_1(x) &= -105+4x_1+5x_2-3x_7+9x_8 \le 0 \\ g_2(x) &= 10x_1-8x_2-17x_7+2x_8 \le 0 \\ g_3(x) &= -8x_1+2x_2+5x_9-2x_{10}-12 \le 0 \\ g_4(x) &= 3(x_1-2)^2+4(x_2-3)^2+2x_3^2 \\ & \quad -7x_4-120 \le 0 \\ g_5(x) &= 5x_1^2+8x_2+(x_3-6)^2-2x_4-40 \le 0 \\ g_6(x) &= x_1^2+2(x_2-2)^2-2x_1x_2+14x_5 \\ & \quad -6x_6 \le 0 \\ g_7(x) &= 0.5(x_1-8)^2+2(x_2-4)^2+3x_5^2 \\ & \quad -x_6-30 \le 0 \\ g_8(x) &= -3x_1+6x_2+12(x_9-8)^2-7x_{10} \le 0 \end{aligned}$$

where $-10 \le x_i \le 10$ where $i = 1,2,\dots,10$.
The global minimum is $f(x^*) = 24.3062091$.

Test problem 8. This problem is defined by

$$Minimize \quad f_8(x) = \frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1+x_2)}.$$

$Subject \quad to$

$$\begin{aligned} g_1(x) &= x_1^2-x_2+1 \le 0 \\ g_2(x) &= 1-x_1+(x_2-4)^2 \le 0 \end{aligned}$$

where $-10 \le x_1 \le 10$ and $0 \le x_2 \le 10$.
The global minimum $f(x^*) = 0.095825$.

Test problem 9. This problem is defined by

$$\begin{aligned} Minimize \quad f_9(x) = & (x_1-10)^2+5(x_2-12)^2 \\ & +x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2. \\ & +x_7^4-4x_6x_7-10x_6-8x_7 \end{aligned}$$

$Subject \quad to$

$$\begin{aligned} g_1(x) &= -127+2x_1^2+3x_2^4+x_3 \\ & \quad +4x_4^2+5x_5 \le 0 \\ g_2(x) &= -282+7x_1+3x_2+10x_3 \\ & \quad +x_4-x_5 \le 0 \\ g_3(x) &= -196+23x_1+x_2^2+6x_6^2 \\ & \quad -8x_7 \le 0 \\ g_4(x) &= 4x_1^2x_2^2-3x_1x_2+2x_3^2 \\ & \quad +5x_6-11x_7 \le 0 \end{aligned}$$

where $-10 \le x_i \le 10$ for $i = 1,2,\dots,7$.
The global minimum $f(x^*) = 680.6300573$.

Test problem 10. This problem is defined by

$$Minimize \quad f_{10}(x) = x_1+x_2+x_3$$

$Subject \quad to$

$$\begin{aligned} g_1(x) &= -1+0.0025(x_4+x_6) \le 0 \\ g_2(x) &= -1+0.0025(x_5+x_7-x_4) \le 0 \\ g_3(x) &= -1+0.01(x_8-x_5) \le 0 \\ g_4(x) &= -x_1x_6+833.33252x_4+100x_1 \\ & \quad -83333.333 \le 0 \\ g_5(x) &= -x_2x_7+1250x_5+x_2x_4-1250x_4 \le 0 \\ g_6(x) &= -x_3x_8+1250000+x_3x_5-2500x_5 \le 0 \end{aligned}$$

where $-100 \le x_1 \le 10,000$ and $1000 \le x_i \le 10,000$ when $i = 2,3,\dots,8$.
The global minimum $f(x^*) = 7049.248021$.

Test problem 11. This problem is defined by

$$Minimize \quad f_{11}(x) = x_1^2+(x_2-1)^2$$

$Subject \quad to$

$$h(x) = \quad x_2-x_1^2 = 0$$

Where $-1 \le x_1 \le 1$ and $-1 \le x_2 \le 1$.
The global minimum $f(x^*) = 0.75$.

Test problem 12. This problem is defined by

$$Minimize \quad f_{12}(x) = \frac{100-(x_1-5)^2-(x_2-5)^2-(x_3-5)^2}{100}$$

$Subject \quad to$

$$g(x) = \quad (x_1-p)^2+(x_2-q)^2+(x_3-w)^2-0.0625 \le 0$$

where $0 \le x_i \le 10$ when $i = 1,2,3$ and $p,q,w = 1,2,\dots = 1,2,\dots,9$.
The global minimum $f(x^*) = 1$.

7.3 The general performance of the proposed HPSODEPSR algorithm with constrained optimization problems

The general performance of the proposed HPSOAC algorithm has been investigated by testing it on 11 test benchmark functions which are reported in Table 2. The best, average (mean), worst, standard deviation (SD) of the function values and the average of function evaluation values (FFEs) are reported over 100 runs. The general performance of the proposed algorithm with 6 benchmark function (randomly picked) are shown in Figure 3 by plotting the values of function values versus the number of iterations. The results in Table 2, show that the proposed algorithm can obtain the optimal or near optimal solution within small number of function evaluations values when compared with other algorithms as shown in the next subsection. Figure 3 shows the general performance of HPSODEPSR algorithm for functions f_1,

Table 2: The general performance of the proposed HPSODEPSR algorithm with constrained optimization problems

Function	Best	Mean	Worst	SD	FFEs
f_1	-15	-15	-15	$9.59e^{-9}$	35,040
f_2	-0.80361598	-0.8036149	-0.788415456	$1.23e^{-2}$	90,195
f_3	-1.0050	-1.0050100	-0.9999109	$3.41e^{-5}$	90,285
f_4	-30665.5387	-30665.5387	-30665.53887	$2.14e^{-12}$	12,180
f_6	-6961.81388	-6961.81388	-6961.81386	$1.31e^{-9}$	91,635
f_7	24.3062091	24.3062109	24.30620111	$1.15e^{-6}$	90,495
f_8	-0.095825041	-0.095825037	-0.095825041	$1.32e^{-8}$	1540
f_9	680.6300574	680.6300574	680.63005739	$2.75e^{-9}$	57,660
f_{10}	7049.248146	7049.248021	7049.248221	$2.83e^{-8}$	90,195
f_{11}	0.749999	0.749999	0.749999	$3.15e^{-7}$	16,440
f_{12}	1	1	1	0.00	3180

f_2, f_4, f_6, f_{11}, f_{12}. We can conclude from Figure 3 that the function values of the proposed HPSODEPSR rapidly decrease as the number of iterations increases and the hybridization between the particle swarm optimization algorithm and the deferential evolution mutation operator with the population size reduction mechanism can accelerate the search and helps the algorithm to obtain the optimal or near optimal solution in reasonable time.

7.4 The efficiency of the population size reduction mechanism

The idea of the population size reduction mechanism was inspired from the cooling schedule and temperature reduction value in the simulated annealing (SA) algorithm [19]. In SA algorithm, there is a direct relation between the quality of the generated trail solutions and the speed of the cooling schedule. If the temperature is slowly decreased, better solutions are obtained, but high computation time is obtained, on the other side, a fast decrement rule causes increasing the probability of being trapped in a local minimum. The HPSODEPSR starts the search with the whole population size in order to make a wide exploration which needs big number of solutions to search in the search space. During the search, the exploration process decreases and the exploitation process increases by focus the search around the promising solutions and the needs of the big number of solutions decreases. The balancing between the exploration and exploitation process is very important and depends on the number of applied solutions in the population. The experimental results show that the best number of applied partitions is 4, which means there are 4 stages (calling) for applying population size reduction mechanism to reduce the population size as shown in Table 3. In Table 3, the mean (average) and standard deviation are reported over 100 runs. Increasing the number of partitions more than 4 means the number of the applied solution in the exploitation process are small and not enough to focus the search around the promising solutions. On the other side, decreasing the number of the partitions numbers means reducing the number of the solution in early stage of the algorithm which effect on the exploration process that needs big number of solutions in the first stages of the algorithm. The best value is reported with **bold face** text. The results in Table 3 show that the best partition size is equal to 4.

7.5 HPSODEPSR and other algorithms

In order to investigate the performance of the proposed HPSODEPSR algorithm, we compare it with seven benchmark algorithms. Before discussing the comparison results of all algorithms, we present a brief description about the comparative seven algorithms as the following.

–**PSO-DE** [22]. An algorithm that integrates particle swarm optimization (PSO) with differential evolution (DE) to solve constrained numerical and engineering optimization problems. DE is incorporated into update the previous best positions of particles to force PSO jump out of stagnation, because of its strong searching ability. The hybrid algorithm speeds up the convergence and improves the algorithm's performance.

–**CRGA** [1].Changing Range-based GA (CRGA): This algorithm adaptively shifts and shrinks the size of the search space of the feasible region by employing feasible and infeasible solutions in the population to reach the global optimum.

–**SAPF** [32]. Self-Adaptive Penalty Function (SAPF) is an algorithm for solving constrained optimization problems using genetic algorithms. In SAPF, a new

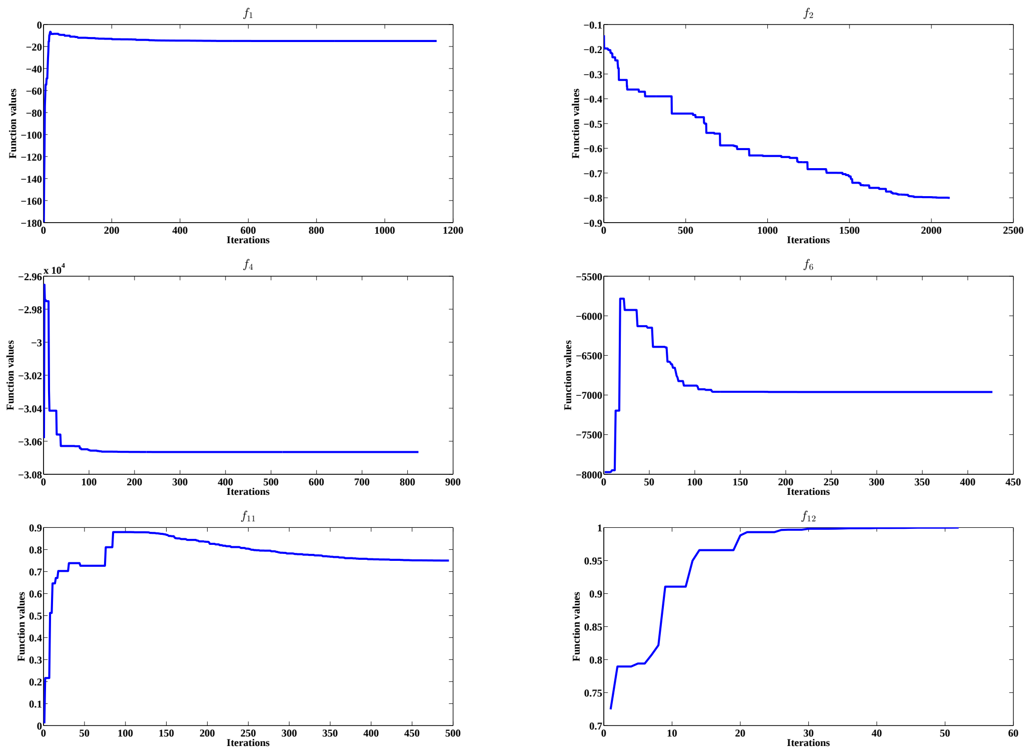

Fig. 3: The general performance of the proposed HPSODEPSR algorithm with constrained optimization problems.

fitness value, called distance value, in the normalized fitness-constraint violation space, and two penalty values are applied to infeasible individuals so that the algorithm would be able to identify the best infeasible individuals in the current population.

–**CDE** [13]. A differential evolution algorithm based on a co-evolution mechanism, is proposed to solve the constrained problems. First, a special penalty function is designed to handle the constraints. Second, a co-evolution model is presented and differential evolution (DE) is employed to perform evolutionary search in spaces of both solutions and penalty factors.

–**CULDE** [2]. A cultural algorithm with a differential evolution population is proposed in this paper. This cultural algorithm uses different knowledge sources to influence the variation operator of the differential evolution algorithm, in order to reduce the number of fitness function evaluations required to obtain competitive results

–**CPSO-GD** [21]. An algorithm based on co-evolutionary particle swarm optimization to solve constrained optimization problems formulated as min-max problems. In standard particle swarm optimization (PSO), a uniform probability distribution is used to generate random numbers for the accelerating coefficients of the local and global s. The proposed algorithm, uses a Gaussian probability distribution to generate the accelerating coefficients of PSO.

–**SMES** [25]. This algorithm uses a simple diversity mechanism based on allowing infeasible solutions to remain in the population instead of using a penalty function. It uses a simple diversity mechanism based on allowing infeasible solutions to remain in the population. This technique helps the algorithm to find the global optimum despite reaching reasonably fast the feasible region of the search space.

7.5.1 Comparison between PSO-DE,CRGA,SAPF,CDE, CULDE, CPSO-GD,SMES and HPSODEPSR

In order to verify the efficiency of the HPSODEPSR algorithm, we compare it against seven algorithms. The best, mean, worst computational cost (number of function evaluations FFEs) are reported over 100 runs in Table 4. The proposed algorithm requires from 1540 to 91,635 FFEs to obtain the reported results in Table 4 as shown in Table 2, while the other reported results for the PSO-DE require from 10,600 to 140,100 FFEs, 500,00 FFEs by SAPF algorithm, 248,000 FFES by CDE, 100,100 FFEs by CULDE and 240,000 FFEs by SMES algorithm. We can conclude from the results in Table 4, that the HPSODEPSR can obtain the optimal or near optimal solution faster than the other algorithms.

Table 3: The efficiency of the population size reduction mechanism

	$Part_{no}=2$		$Part_{no}=3$		$Part_{no}=4$		$Part_{no}=5$	
f	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
f_1	-14.79999145	0.6	-14.9999999	$5.94e^{-8}$	**-15**	$9.59e^{-9}$	-15	$7.2e^{-9}$
f_2	-0.802721064	$2.7e^{-3}$	-0.80353757	$4.9e^{-4}$	**-0.8036149**	$1.23e^{-2}$	-0.790663954	$1.72e^{-2}$
f_3	-0.962237027	$6.96e^{-2}$	-0.992436478	$2.11e^{-2}$	**-1.0050100**	$3.41e^{-5}$	-0.863857	$2.9e^{-2}$
f_4	-30665.53867	0.00	-30665.53867	$2.13e^{-9}$	**-30665.5387**	$2.14e^{-12}$	-30665.5386	$2.15e^{-5}$
f_6	-6961.813872	$1.03e^{-6}$	-6961.813874	$1.42e^{-6}$	**-6961.81388**	$1.31e^{-9}$	-6961.81372	$1.7e^{-6}$
f_7	24.30620907	$9.46e^{-10}$	24.30620907	$1.06e^{-5}$	**24.3062109**	$1.1e^{-6}$	24.30622538	$5.16e^{-5}$
f_8	-0.095820081	$3.23e^{-6}$	-0.095824866	$3.29e^{-6}$	**-0.095825037**	$1.32e^{-8}$	-0.09582068	$2.64e^{-6}$
f_9	680.6300574	$4.04e^{-9}$	680.6300574	$7.63e^{-9}$	**680.6300574**	$2.75e^{-9}$	680.6300574	$3.22e^{-9}$
f_{10}	7049.248025	$6.02e^{-6}$	7049.248021	$1.5e^{-3}$	**7049.248021**	$2.4e^{-8}$	7049.335286	$1.15e^{-2}$
f_{11}	0.749928102	$4.53e^{-5}$	0.749999	$4.54e^{-5}$	**0.749999**	$3.15e^{-7}$	0.749999	$4.08e^{-5}$
f_{12}	0.9999992081	$3.11e^{-9}$	0.99999911	$7.3695e^{-6}$	**1**	0.00	0.99999907	$5.11e^{-7}$

Table 4: Comparison results of HPSOAC and other PSO-based algorithms for problems $f_1 - f_{12}$

Function		PSO-DE	CRGA	SAPF	CDE	CULDE	CPSO-GD	SMES	HPSODEPSR
f_1	Best	-15.000000	-14.9977	-15.000	-15.0000	-15.000000	-15.0	-15.000	-15
	Mean	-15.000000	-14.9850	-14.552	-15.0000	-14.999996	-14.997	-15.000	-15
	Worst	-15.000000	-14.9467	-13.097	-15.0000	-14.999993	-14.994	-15.000	-15
f_2	Best	-0.8036145	-0.802959	-0.803202	-0.794669	-0.803619	NA	-0.803601	-0.80361598
	Mean	-0.756678	-0.764494	-0.755798	-0.785480	-0.724886	NA	-0.785238	-0.8036149
	Worst	-0.6367995	-0.722109	-0.745712	-0.779837	-0.590908	NA	-0.751322	-0.788415456
f_3	Best	-1.0050100	-0.9997	-1.000	NA	-0.995413	NA	-1.000	-1.0050100
	Mean	-1.0050100	-0.9972	-0.964	NA	-0.788635	NA	-1.000	-1.0050100
	Worst	-1.0050100	-0.9931	-0.887	NA	-0.639920	NA	-1.000	-0.9999109
f_4	Best	-30665.539	-30665.520	-30665.401	-30665.539	-30665.539	NA	-30665.539	-30665.5387
	Mean	-30665.539	-30664.398	-30665.922	-30665.536	-30665.539	NA	-30665.539	-30665.5387
	Worst	-30665.539	-30660.313	-30656.471	-30665.509	-30665.539	NA	-6952.482	-30665.53887
f_6	Best	-6961.8139	-6956.251	-6961.046	-6961.814	-6961.8139	NA	-6961.284	-6961.81388
	Mean	-6961.8139	-6740.288	-6953.061	-6960.603	-6961.8139	NA	-6961.284	-6961.81388
	Worst	-6961.8139	-6077.123	-6943.304	-6901.285	-6961.8139	NA	-6952.482	-6961.81386
f_7	Best	24.306210	25.746	27.328	NA	24.306210	25.709	24.475	24.3062091
	Mean	24.306210	25.746	27.328	NA	24.306210	25.709	24.475	24.3062109
	Worst	24.30622	27.381	33.095	NA	24.3062	27.166	24.843	24.30620111
f_8	Best	-0.095826	-0.095825	-0.095825	NA	-0.095825	NA	-0.095825	-0.095825041
	Mean	-0.0958259	-0.095819	-0.095635	NA	-0.095825	NA	-0.095825	-0.095825037
	Worst	-0.0958259	-0.095808	-0.092697	NA	-0.095825	NA	-0.095825	-0.095825041
f_9	Best	680.63006	680.726	680.773	680.771	680.63006	680.678	680.632	680.6300574
	Mean	680.63006	681.347	681.246	681.503	680.63006	680.7810	680.643	680.6300574
	Worst	680.6301	682.965	682.081	685.144	680.6301	681.371	680.719	680.63005739
f_{10}	Best	7049.2480	7114.743	7069.981	NA	7049.2481	7055.6	7051.903	7049.248146
	Mean	7049.2480	8785.149	7238.964	NA	7049.2483	8464.2	7253.04	7049.248021
	Worst	7049.2482	10826.09	7489.406	NA	7049.2485	11,458	7638.366	7049.248221
f_{11}	Best	0.749999	0.75	0.749	NA	0.749900	NA	0.75	0.749999
	Mean	0.749999	0.752	0.751	NA	0.757995	NA	0.75	0.749999
	Worst	0.750001	0.757	0.757	NA	0.796455	NA	0.75	0.749999
f_{12}	Best	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	NA	-1.000	1
	Mean	-1.000000	-1.000000	-0.99994	-1.000000	-1.000000	NA	-1.000	1
	Worst	-1.000000	-1.000000	-0.999548	-1.000000	-1.000000	NA	-1.000	1

7.6 Engineering optimization problems

In order to investigate the general performance of HPSODEPSR on real world engineering optimization problems, five well studied engineering design examples have been selected and solved by the proposed algorithm. HPSODEPSR has been applied with the engineering optimization problems with the same parameter settings which applied with the other 11 test problems. In the following subsections, we highlight the five engineering optimization problems.

7.6.1 Welded beam design problem

The aim of a welded beam design problem is to minimize the cost of the welded beam structure which consists of a beam A and the required weld to hold it to member B subject to constraints on shear stress τ, beam bending stress θ, buckling load on the bar P_c, beam end deflection δ. The four design variables $h(x_1), l(x_2), t(x_3)$ and $b(x_4)$ are shown in Figure 4.

$$Minimize \quad f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

$$Subject \quad to$$

$$\begin{aligned}
g_1(x) &= \tau(x) - \tau_{max} \le 0 \\
g_2(x) &= \sigma(x) - \sigma_{max} \le 0 \\
g_3(x) &= x_1 - x_4 \le 0 \\
g_4(x) &= 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) \\
&\quad -0.5 \le 0 \\
g_5(x) &= 0.125 - x_1 \le 0 \\
g_6(x) &= \delta(x) - \delta_{max} \le 0 \\
g_7(x) &= P - P_c(x) \le 0
\end{aligned}$$

where the other parameters are defined as the following.

$$\tau(x) = \sqrt{((\tau')^2 + (\tau'')^2 + \frac{2\tau'\tau''x_2}{2R}}, \quad \tau' = \frac{p}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}, \quad M = P(L + \frac{x_2}{2}), \quad R = \sqrt{\left(\frac{x_1 + x_3}{2}\right)^2 + \frac{x_2^2}{4}}$$

$$J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \quad \sigma(x) = \frac{6PL}{x_4x_3^2}$$

$$\delta(x) = \frac{4PL^3}{Ex_4x_3^3}, \quad P_c(x) = \frac{4.013\sqrt{EGx_3^2x_4^6/36}}{L^2}\left(1 - \frac{x^3}{2L}\sqrt{\frac{E}{4G}}\right)$$

where $P = 6000$lb, $L = 14$, $\delta_{max} = 0.25$in, $E = 30,106$ psi, $G = 12,106$ psi, $\tau_{max} = 13,600$psi, $\sigma_{max} = 30,000$ psi and $0.1 \le x_i \le 10.0$ when $i = 1,2,3,4$.

7.6.2 Pressure vessel design problem

In this problem, a cylindrical vessel is capped at both ends by hemispherical heads and the objective is to minimize the total cost of material, forming and welding.

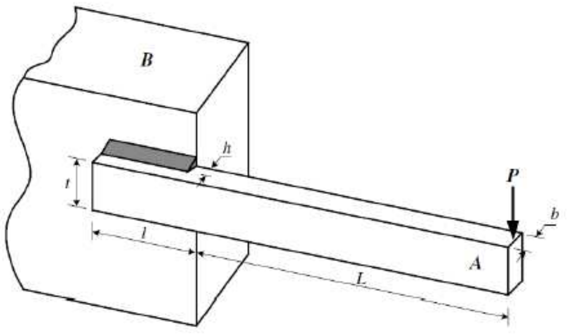

Fig. 4: Welded beam design problem

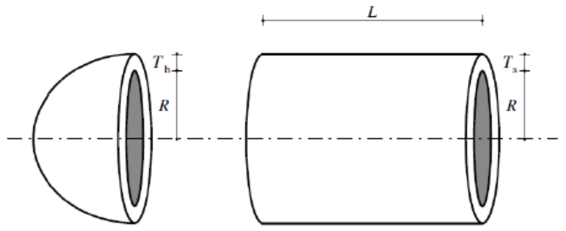

Fig. 5: Pressure vessel design problem

There are four design variables as shown in Figure 5, T_s (x_1, thickness of the shell), T_h (x_2, thickness of the head), R (x_3, inner radius) and L (x_4, length of the cylindrical section of the vessel without the head). T_s and T_h are integer multiples of 0.0625in, R and L are continuous variables.

$$\begin{aligned}
f(x) = &\ 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\
&+ 19.84x_1^2x_3
\end{aligned}$$

$$Subject \quad to$$

$$\begin{aligned}
g_1(x) &= -x_1 + 0.0193x_3 \\
g_2(x) &= -x_2 + 0.00954x_3 \le 0 \\
g_3(x) &= -\pi x_3^2x_4 - 4/3\pi x_3^3 + 1296000 \le 0 \\
g_4(x) &= x_4 - 240 \le 0
\end{aligned}$$

where $1 \le x_1 \le 99$, $1 \le x_2 \le 99$, $10 \le x_3 \le 200$ and $10 \le x_4 \le 200$.

7.6.3 Speed reducer design problem

The objective of a speed reducer problem is to minimize a gear box volume and weight subject to several constrains as shown in Figure 6. There are seven design variables $x_1 - x_7$, which describe as follow. x_1 is a width of the gear face (cm), x_2 teeth module (cm), x_3 number of pinion teeth, x_4 shaft 1 length between bearings (cm), x_5 shaft 2

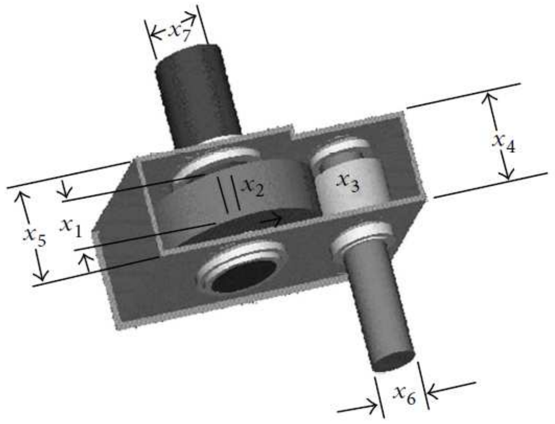

Fig. 6: Speed reducer design problem

length between bearing (cm), x_6 diameter of shaft 1 (cm) and x_7 diameter of shaft 2 (cm).

$$
\begin{aligned}
f(x) = {} & 0.7854x_1x_2^2(3.3333x_3^2+14.9334x_3\\
& -43.0934)\\
& -1.508x_1(x_6^2+x_7^2)+7.4777(x_6^3+x_7^3)\\
& +0.7854(x_4x_6^2+x_5x_7^2)
\end{aligned}
$$

Subject to

$$
\begin{aligned}
g_1(x) &= \frac{27}{x_1x_2^2x_3}-1\le 0\\
g_2(x) &= \frac{397.5}{x_1x_2^2x_3^2}-1\le 0\\
g_3(x) &= \frac{1.93x_4^3}{x_2x_6^4x_3}-1\le 0\\
g_4(x) &= \frac{1.93x_5^3}{x_2x_7^4x_3}-1\le 0\\
g_5(x) &= \frac{[(745x_4/x_2x_3)^2+16.9\times10^6]^0.5}{110.0x_6^3}-1\le 0\\
g_6(x) &= \frac{[(745x_5/x_2x_3)^2+157.5.9\times10^6]^0.5}{85.0x_7^3}-1\le 0\\
g_7(x) &= \frac{x_2x_3}{40}-1\le 0\\
g_8(x) &= \frac{5x_2}{x_1}-1\le 0\\
g_9(x) &= \frac{x_1}{12x_2}-1\le 0\\
g_{10}(x) &= \frac{1.5x_6+1.9}{x_4}-1\le 0\\
g_{11}(x) &= \frac{1.1x_7+1.9}{x_5}-1\le 0 \qquad (20)
\end{aligned}
$$

where $2.6\le x_1\le 3.6$, $0.7\le x_2\le 0.8$, $17\le x_3\le 28$, $7.3\le x_4\le 8.3$, $7.3\le x_5\le 8.3$, $2.9\le x_6\le 3.9$, and $5\le x_7\le 5.5$

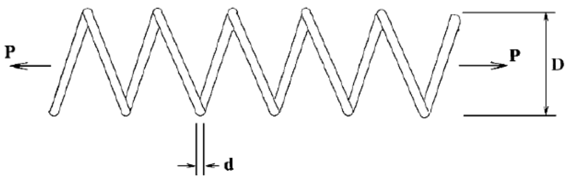

Fig. 7: Tension compression spring problem

7.6.4 Three-bar truss design problem

The objective of this application is to minimize the weight $f(x)$ of three bar truss subject to some constrains as follows.

$$f(x) = (2\sqrt{2}x_1+x_2)\times l$$

Subject to

$$
\begin{aligned}
g_1(x) &= \frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0\\
g_2(x) &= \frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0\\
g_3(x) &= \frac{1}{\sqrt{2}x_2+x_1}P-\sigma\le 0
\end{aligned}
$$

where $0\le x_1\le 1$ and $0\le x_2\le 1$, $l=100$ cm, $P=2kN/cm^2$ and $\sigma=2kN/cm^2$. The minimum weighted structure should be achieved by determining the optimal cross-sectional areas x_1, and x_2.

7.7 Tension compression spring problem

In this problem, we need to minimize the weight $f(x)$ of a tension compression spring design subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variable. The mean coil diameter $D(x_2)$, the wire diameter $d(x_1)$ and the number of active coils $P(x_3$ are the design variables as shown in Figure 7.

$$f(x) = (x_3+2)x_2x_1^2$$

Subject to

$$
\begin{aligned}
g_1(x) &= 1-\frac{x_2^3x_3}{71785x_1^4}\le 0\\
g_2(x) &= \frac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\frac{1}{5108x_1^2}-1\le 0\\
g_3(x) &= 1-\frac{140.45x_1}{x_2^2x_3}\le 0
\end{aligned}
$$

where $0.05\le x_1\le 2$, $0.25\le x_2\le 1.3$ and $2\le x_3\le 15$.

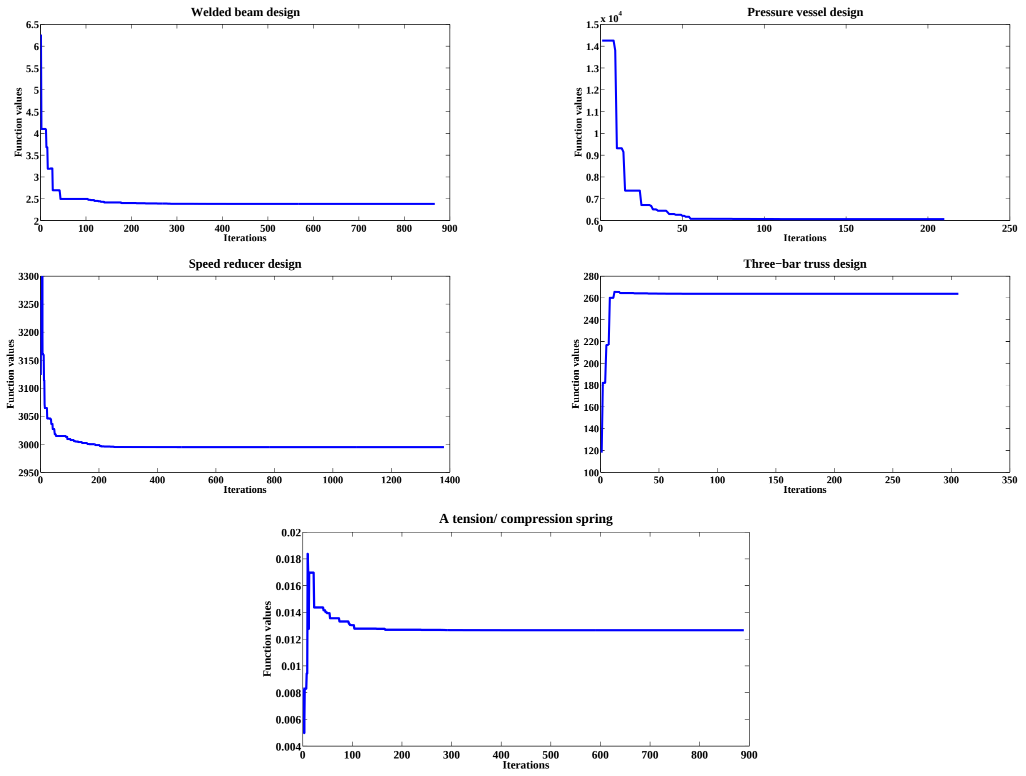

Fig. 8: The general performance of the proposed HPSODEPSR algorithm with engineering optimization problems

Table 5: The general performance (function evaluations) of HPSODEPSR with engineering optimization problems

Design problem	Best	Mean	Worst	Std
Welded beam design	24,690	26,062.5	28,110	1452.111
Pressure vessel design	11,385	14,591.25	16,620	2242.70
Speed reducer design	91,335	90,195	92,055	806.4738
Three-bar truss design	9390	10,062	10,500	410,5728
A tension/compression spring design	20,640	24,174	26,220	2321.795

Table 6: The general performance (function values) of HPSODEPSR with engineering optimization problems

Design problem	Best	Mean	Worst	Std
Welded beam design	2.380957153	2.380957168	2.380957192	$1.82257e^{-8}$
Pressure vessel design	6059.714335	6059.714335	6059.714335	$4.5423e^{-12}$
Speed reducer design	2.99447106614	2.99447106614	2.99447106614	0.00
Three-bar truss design	263.89584338	263.89584338	263.89584338	$3.5427e^{-11}$
A tension/compression spring design	0.012665233	0.012665233	0.012665233	$5.77594e^{-12}$

7.7.1 The general performance of the proposed HPSODEPSR algorithm with engineering optimization problems

The general performance of the proposed HPSODEPSR algorithm is tested on 5 engineering optimization problems shown in Figure 8 by plotting the number of iterations versus the number of function values. The best mean worst and standard deviation of the function evaluations and function values are reported over 100 runs in Tables 5 and 6 respectively. We can conclude from Figure 8, Tables 5 and 6 that the proposed algorithm can obtain optimal or near optimal solutions in reasonable time.

8 Conclusion

In this paper, we propose a novel hybrid particle swarm optimization and differential evolution algorithm in order to solve constrained optimization problems and a real life engineering optimization problems. The proposed algorithm applies a new population size reduction mechanism (PSRM) in order to balance between the exploration and exploitation process by starting the search in early stage with large number of solutions in the population and during the search the number of theses solutions decreases. Also in the proposed algorithm, we tried to avoid terminating the search with the traditional termination criterion like number of iterations, but we proposed the progress vector V as a new automatic termination criterion instead of letting the algorithm running with more iterations without any improvement in the results. In order to investigate the general performance of the proposed algorithm, we tested the proposed algorithm on eleven benchmark functions and five real life engineering optimization problems and compared against seven algorithms. Our numerical experimental results show that the proposed algorithm is a promising algorithm and can solve the constrained optimization problems with high performance in reasonable time.

Acknowledgments

The research of the 1st author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The postdoctoral fellowship of the 2nd author is supported by NSERC.

References

[1] A. Amirjanov, The development of a changing range genetic algorithm, Computer Methods in Applied Mechanics and Engineering, Vol. 195, pp. 2495–2508, 2006.
[2] R. L. Becerra and C. A. C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering, Vol. 195, pp. 4303–4322, 2006.
[3] W.D. Chang, PID control for Chaotic synchronization using particle swarm optimization. Chaos, Solitons and Fractals, Vol. 39, No. 2, pp. 910-917, 2009.
[4] C. A. C. Coello, G. T. Pulido and M. S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transaction on Evolutionary Computation, Vol. 8, No. 3, pp. 256–279, 2004.
[5] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, Vol. 18, No. 2-4, pp. 311–338, 2008.
[6] A. A. Esmin, G. Lambert-Torres and G.B. Alvarenga, Hybrid evolutionary algorithm based on PSO and GA mutation, Proceedings of 6th International Conference on Hybrid Intelligent Systems, pp. 57–62, 2006.
[7] E. A. Gandelli, F. Grimaccia, M. Mussetta, P. Pirinoli and R. E. Zich, Genetical swarm optimization: a new hybrid evolutionary algorithm for electromagnetic application, in: Proceedings of the 18th International Conference on Applied Electromagnetics, ICECcom 2005, ICECom 2005. 18th International Conference on, pages 1–4, 2005.
[8] E. A. Gandelli, F. Grimaccia, M. Mussetta, P. Pirinoli and R. E. Zich, Genetical swarm optimization: an evolutionary algorithm for antenna design, Journal of AUTOMATIKA, Vol. 47, (3-4), pp. 105–112, 2006.
[9] E. A. Gandelli, F. Grimaccia, M. Mussetta, P. Pirinoli, R. E. Zich, Development and validation of different hybridization strategies between GA and PSO, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2782–2787, 2007.
[10] E. A. Grimaldi, F. Grimacia, M. Mussetta, P. Pirinoli, R.E. Zich, A new hybrid genetical swarm algorithm for electromagnetic optimization, Proceedings of International Conference on Computational Electromagnetics and its Applications, Beijing, China, pp. 157–160, 2004.
[11] Q. He and L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Engineering Application of Artificial Intelligence, Vol. 20, No. 1, pp.89–99, 2007.
[12] Q. He and L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation, Vol. 186, No. 2, pp. 1407–1422, 2007.
[13] F.Z. Huang, L. Wang and Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics and Computation, Vol. 186, No. 1, pp. 340–356, 2007.
[14] M. Jian and Y. Chen, Introducing recombination with dynamic linkage discovery to particle swarm optimization, Proceedings of the Genetic and Evolutionary Computation Conference, pp. 85–86, 2006.
[15] C. F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics, Vol. 34, pp. 997–1006, 2004.
[16] J. Kennedy and R. C. Eberhart, Swarm intelligence. San Mateo: Morgan Kaufmann, 2001.
[17] J. Kennedy, RC. Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks, Vol 4, pp 1942–1948, 1995.
[18] H. Kim, Improvement of genetic algorithm using PSO and Euclidean data distance, International Journal of Information Technology, Vol. 12, pp. 142–148, 2006.
[19] S. Kirkpatrick, CD. Gelatt, MP. Vecchi. Optimization by simulated annealing. Science, Vol. 220, No. 4598, pp. 671–680, 1983.
[20] T. Krink and M. Lvbjerg, The lifecycle model: combining particle swarm optimization, genetic algorithms and hill climbers, Proceedings of the Parallel Problem Solving From Nature, pp. 621–630, 2002.
[21] R. A. Krohling, L. S. Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics, 36(6): 1407–1416, 2006.
[22] H. Liu, Z. Cai and Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Applied Soft Computing, Vol. 10, pp. 629–640, 2010.
[23] M. D. Lei, A Pareto archive particle swarm optimization for multiobjective job shop scheduling, Computers & Industrial Engineering, Vol. 54, No. 4, pp. 960–971, 2008.

[24] J. J. Liang and P. N. Suganthan, Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006). IEEE Press, pp. 9–16, 2006.
[25] E. Mezura-Montes and C.A.C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation, Vol. 9, No. 1, pp. 1–17, 2005.
[26] A. Mohammadi and M. Jazaeri, A hybrid particle swarm optimization-genetic algorithm for optimal location of SVC devices in power system planning, Proceedings of 42nd International Universities Power Engineering Conference, pp.1175–1181, 2007.
[27] Z. Michalewicz, A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, Evolutionary Programming, Vol.4, pp. 135, 1995.
[28] G. T. Pulido and C. A. C. Coello, A constraint-handling mechanism for particle swarm optimization, Proceedings of 2004 Congress on Evolutionary Computation (CEC 2004). IEEE Press, pp. 1396–1403, 2004.
[29] J. Robinson, S. Sinton, and Y.R. Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna, Proceedings of the IEEE International Symposium in Antennas and Propagation Society, pp. 314–317, 2002.
[30] M. Settles and T. Soule, Breeding swarms: a GA/PSO hybrid, Proceedings of Genetic and Evolutionary Computation Conference, pp. 161-168, 2005.
[31] R. Storn and K. Price, Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces, J Glob Optim , Vol. 11, pp. 341–359, 1997.
[32] B. Tessema and G. Yen, A self-adaptive penalty function based algorithm for constrained optimization, Proceedings 2006 IEEE Congress on Evolutionary Computation, pp. 246–253, 2006.
[33] Y. Wang and Z. Cai, A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems, Frontiers of Computer Science in China 3.1, pp. 38–52, 2009.
[34] J. M. Yang, Y.P. Chen, J.T. Horng and C.Y. Kao. Applying family competition to evolution strategies for constrained optimization. In Lecture Notes in Mathematics Vol. 1213, pp. 201–211, New York, Springer, 1997.
[35] B. Yang, Y. P. Chen, and Z. L. Zhao, A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems, Proceedings of the IEEE International Conference on Control and Automation, pp. 166–170, 2007.
[36] K. Zielinski, P. Weitkemper and R. Laur, Optimization of power allocation for interference cancellation with particle swarm optimization, IEEE Transactions on Evolutionary Computation, Vol. 13, No. 1, pp. 128–150, 2009.
[37] J. D. Zhang, D. L. Jia, and K. Li, FIR digital filters design based on Chaotic mutation particle swarm optimization, Proceedings of the IEEE international conference on audio, language and image processing, pp. 418–422, 2008.

Ahmed F. Ali Received the B.Sc., M.Sc. and Ph.D. degrees in computer science from the Assiut University in 1998, 2006 and 2011, respectively. Currently, he is a Postdoctoral Fellow at Thompson Rivers University, Kamloops, BC Canada. In addition, he is an Assistant Professor at the Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt. He served as a member of Computer Science Department Council from 2014-2015. He worked as director of digital library unit at Suez Canal University; he is a member in SRGE (Scientific Research Group in Egypt). He also served as a technical program committee member and reviewer in worldwide conferences. Dr. Ali research has been focused on meta-heuristics and their applications, global optimization, machine learning, data mining, web mining, bioinformatics and parallel programming. He has published many papers in international journals and conferences and he has uploaded some meta-heuristics lectures in slidshare website.

Mohamed A. Tawhid got his PhD in Applied Mathematics from the University of Maryland Baltimore County, Maryland, USA. From 2000 to 2002, he was a Postdoctoral Fellow at the Faculty of Management, McGill University, Montreal, Quebec, Canada. Currently, he is a full professor at Thompson Rivers University. His research interests include nonlinear/stochastic/heuristic optimization, operations research, modelling and simulation, data analysis, and wireless sensor network. He has published in journals such as Computational Optimization and Applications, J. Optimization and Engineering, Journal of Optimization Theory and Applications, European Journal of Operational Research, Journal of Industrial and Management Optimization, Journal Applied Mathematics and Computation, etc. Mohamed Tawhid published more than 40 referred papers and edited 5 special issues in J. Optimization and Engineering (Springer), J. Abstract and Applied Analysis, J. Advanced Modeling and Optimization, and International Journal of Distributed Sensor Networks. Also, he has served on editorial board several journals. Also, he has worked on several industrial projects in BC, Canada.