Applied Mathematics and Computation 268 (2015) 246–269

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A novel differential search algorithm and applications for structure design

Jianjun Liu [a], Changzhi Wu [b,*], Guoning Wu [a], Xiangyu Wang [b,c]

[a] College of Science, China University of Petroleum, Beijing 102249, China
[b] Australasian Joint Research Centre for Building Information Modelling, School of Built Environment, Curtin University, Perth, WA 6845, Australia
[c] Department of Housing and Interior Design, Kyung Hee University, Seoul, Korea

ARTICLE INFO

Keywords:
Differential search
Constrained optimization
Dynamic penalty function
Structure design

ABSTRACT

Differential Search method is recently proposed to solve box constrained global optimization problems. In this paper, we will further extend this method to solve generalized constrained optimization problems, particularly for structure design optimization problems. To handle the constraints, we first propose a novel dynamic S-type soft-threshold penalty method. Then, the original constrained optimization problem is transformed into a sequence of unconstrained optimization problems. The proposed method is mainly comprised of two steps: parameter iteration and solution iteration. The parameter iteration is to update the dynamic penalty parameter through a soft-threshold scheme and the solution iteration is to implement Differential Search algorithm to solve an unconstrained optimization problem. Two benchmark sets, CEC2006 and CEC2010, and four engineering structure design optimization problems are solved by our proposed algorithm as well as many other swarm-based algorithms proposed in recent literatures. Numerical results show that our method can achieve better performance but with fewer function evaluations comparing with the existing algorithms.

1. Introduction

Many practical mechanical design problems can be cast as constrained optimization problems [1]. Unlike general nonlinear programming problems which only contain continuous or integer variables, mechanical design optimization problems usually involve both continuous and discrete variables. The continuous variables are used to represent standardization constraints such as the diameters of standard sized bolts, while the discrete variables are usually involved in the formulation of the design problem to select alternative options, such as gear teeth. Since discrete variables can be transformed into continuous variables through introducing equality constraints, we consider the following generalized optimization problem:

$$\begin{aligned} \min\ & y = f(x) \\ \text{s.t. } & g_j(x) \le 0, \ \text{for } j = 1, \ldots, q, \\ & g_j(x) = 0, \ \text{for } j = q+1, \ldots, m, \end{aligned}$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, $l_i \le x_i \le u_i$, $i = 1, \ldots, n$, and the objective function $f(x)$ is continuous and defined on a search space, which is defined as an n-dimensional rectangle region (domains of variables defined by their lower and upper bounds).

* Corresponding author. Tel.: +61 478018777.
E-mail address: c.wu@curtin.edu.au (C. Wu).

http://dx.doi.org/10.1016/j.amc.2015.06.036
0096-3003/

The feasible region set is defined by a set of m additional linear or nonlinear constraints $g_j(x)$. Let this problem be referred to as Problem (P).

To solve Problem (P), there are two main kinds of computational methods available: (i) mathematical programming methods that use approximation techniques to solve the optimization problem; and (ii) metaheuristic algorithms that mimic some natural phenomena including biology and evolution theory. Popular metaheuristic algorithms are swarm algorithms, including modified versions of Artificial Bee Colony (ABC) algorithm [2,3], Firefly Algorithm (FA) [4,5], Particle Swarm Optimization (PSO) [6–8], Social-Spider Optimization (SSO) algorithm [9], several variants of Differential Evolution (DE) algorithm [10–13], Backtracking Search Optimization Algorithm [14], Artificial Cooperative Search Algorithm [15] and some Evolutionary Strategy algorithms [16–19]. For general constrained optimization problems, mathematical programming methods are always stuck by the local minimizers. However, metaheuristic algorithms are able to escape from local minima [20]. Thus, how to introduce new efficient metaheuristic algorithms with higher potential and simpler usage is important.

To improve the performance of metaheuristic algorithms, many new techniques are introduced. In [21], a combination of Ant Colony Optimization with chaotic sequences are employed in continuous optimization problems of engineering design. PSO, Quantum-behaved PSO and Gaussian probability distribution are employed in continuous optimization problems of engineering design [22,23]. In [11,24,25], the hybrid DE to find global optimum in problems with complex design spaces and one hybrid algorithm enhances a basic DE with a local search operator, random walk with direction exploitation, to strengthen the exploitation ability, while the other adding a second metaheuristic, harmony search, to cooperate with DE algorithm so as to produce the desirable synergetic effect. A class of niche hybrid Cultural Algorithms for solving miscellaneous engineering optimization problems are studied in [26]. Although most algorithms mentioned above can be applied to structure design optimization problems, they are still computationally inefficient in some degree. In this paper, we will introduce a new modified Differential Search algorithm which can solve generalized constrained optimization problems and achieves better performances.

Differential Search (DS) algorithm is a new population-based metaheuristic optimization algorithm which is first introduced in [27] in 2012 to solve box constrained global optimization problems. Numerical experiments show that it achieves better performance than many other metaheuristic algorithms. For example, it is more robust against the choice of initial population and control parameters than PSO, ABC and others for continuous unconstrained optimization [27]. DS algorithm in [27] can only be used to solve box constrained optimization problems. Although application of DS to unconstrained optimization is well-established, so far few studies have focused on the application of DS to constrained optimization. In this paper, we will further develop this algorithm to solve generalized constrained optimization problems. To handle the constraints, the penalty function methods are always introduced. If the penalty parameter is a constant, this method is referred to as static penalty method. Otherwise, it is referred to as dynamic penalty method. In terms of static penalty, dynamic penalty is more desirable since the penalty parameter is adjusted dynamically. Different from the existing dynamic penalty method [28,29], we will introduce a novel S-type function to generate the dynamic penalty parameter. Comparing with the dynamic penalty method in [28], our proposed dynamic method achieves better numerical performances. Assorting to the penalty method, the original constrained optimization problem is approximated by a sequence of unconstrained optimization problems. Then, DS algorithm in [30] is applied to solve those unconstrained optimization problems. Thus, our proposed algorithm can be described as two main steps: penalty factor iteration and optimal solution iteration. In the penalty factor iteration, the penalty parameter is updated using an S-type threshold dynamic update scheme. While in the optimal solution iteration, a nonlinear unconstrained optimization with the penalty function is solved by DS. To show the performance of our proposed algorithm, two benchmark sets, CEC2006 and CEC2010, and four engineering design optimization problems have solved by our method as well as many swarm-based algorithms proposed in recent literatures. Numerical results show that our method can achieve better performance but with fewer function evaluations comparing with those algorithms.

2. Differential search algorithm

Differential Search (DS) algorithm is originally introduced to solve the problem of transforming the geocentric cartesian coordinates into geodetic coordinates in 2012. Comparison studies are carried out in [27] for continuous unconstrained optimization problems between the DS algorithm and eight widely used algorithms, including PSO, ABC, DE, and Gravitational Search Algorithm (GSA). The results show that DS algorithm in [27,31] is more powerful. DS algorithm also is successfully applied to circular antenna array design[32], needle roller bearing design[33] and multilevel color image thresholding [34]. DS algorithm simulates the Brownian-like random-walk movement carried out by an organism to migrate. In the migration movement, the migrating species of living beings constitute a superorganism, which contains a large number of individuals. Then, the superorganism starts to change its position by moving towards more fruitful areas. The movement of a superorganism can be described by a *Brownian-like random walk* model. Therefore, the simulation of the Brown motion is taken as a search strategy in DS.

2.1. Generation of the initial solution

In the process of solving optimization problem by DS, it is assumed that a population is made up of random outcomes of these artificial-superorganism migrations. An artificial-superorganism will migrate to the global minimum of the problem. During this migration, the artificial-superorganism examines whether some randomly selected positions are suitable temporary positions during the migration. If such a position is suitable to stopover temporarily during the migration, the members of the

Algorithm 1 Pseudo code: a Brownian-like random walk process searching for a stopover.

```
if rand_1 < rand_2 then
  if rand_3 < p_1 then
    r = rand(N, D)
    for Counter_1 = 1 : N do
      r(Counter_1, : ) = r(Counter_1, : ) < rand_4
    end for
  else
    r = ones(N, D)
    for Counter_2 = 1 : N do
      r(Counter_2, randi(D)) = r(Counter_2, randi(D)) < rand_5
    end for
  end if
else
  r = ones(N, D)
  for Counter_3 = 1 : N do do
    d = randi(D, 1, ⌈p_2 · D⌉)
    for Counter_4 = 1 : size(d) do
      r(Counter_3, d(Counter_4)) = 0
    end for
  end for
end if
individuals_IJ ← r_IJ | I ∈ i, J ∈ [1, D]
S(individuals_IJ) = Superorganism(individuals_IJ)
```

artificial-superorganism (i.e. artificial-organisms) that made such discovery immediately settle at the discovered position and continue their migration from this position.

Artificial-organisms (i.e., $X_i, i = 1, 2, \ldots, N$) making up an artificial-superorganism (i.e.,$S_g, g = 1, 2, \ldots, G$) contain elements (i.e., $x_{ij}, j = 1, 2, \ldots, D$) whose size is equal to the problem dimension. Here, N, G and D denote, respectively, the number of members in the superorganism, the maximum generation (or iteration) number, and the dimension of each problem. Each element of an artificial-organism in the initial position is defined as:

$$x_{ij} = l_j + r \cdot (u_j - l_j) \tag{1}$$

where r is a uniform-random number. Set $L = (l_1, l_2, \ldots, l_D)$ and $U = (u_1, u_2, \ldots, u_D)$ with that l_j and u_j are the lower and upper bounds of the jth dimension of the ith artificial-organism X_i, respectively. In such a case, an artificial-organism can be defined as $X_i = [x_{ij}]$.

2.2. Searching strategy of DS

The mechanism of DS for finding a stopover site in the remaining areas between the artificial-organisms can be described by a *Brownian-like random walk* model (see Algorithm 1). These individuals which are randomly selected in the artificial-organisms move towards the targets of

$$donor = X_{Random\_Shuffling(i)} \tag{2}$$

so as to discover stopover sites. The *Random_Shuffling* function randomly changes the order of the numbers of the elements in the set of $i = \{1, 2, \ldots, N\}$.

In the artificial-organisms, the scale of the positions change among their members is controlled by a scale factor γ . The scale value is generated from the Gamma distribution, with parameters a and b, denoted by

$$\gamma = 1/GamRnd(a, b) \tag{3}$$

Certainly, the way of generating scale value allows the respective artificial-superorganisms to change directions radically in the habitat.

A stopover site position, S, is produced according to the following equation:

$$S = X_i + \gamma \cdot (donor - X_i) \tag{4}$$

The members (i.e., *individuals*) X_i of the artificial-organisms among the superorganisms S_k to participate in the search process at stopover site are determined by a random process. This random process is given by Algorithm 1 in which the $rand_i$, $(i = 1, \ldots, 5)$ are the uniform random numbers and $Counter_i$, $(i = 1, \ldots, 4)$ denote four counters. Both control parameters p_1 and p_2 are given as $p_1 = 0.3$ and $p_2 = 0.3$. This process is important to ensure a successful migration to global optima.

If one of the elements x_{ij} of X_i is moving beyond the limits of the habitat (i.e. search space), it is randomly deferred to another position in the habitat. If the *individuals* of the artificial-organism discover a stopover site, which is more fertile than the sources owned by the artificial-organism, the artificial-organism moves to that stopover site. This greedy rule in DS can be modelled by the following formula

$$S^* = \begin{cases} S, & \text{if } y(S) < y(S^*); \\ S^*, & \text{otherwise.} \end{cases} \tag{5}$$

where $y(S)$ and $y(S^*)$ are the evaluations of the stopover site and current best optima, respectively. If the artificial-organisms change site, the superorganism containing the artificial organisms continues its migration towards the global optima.

A flowchart for the DS algorithm is presented as Fig. 1.

2.3. Balance mechanism of exploitation and exploration in DS

The main components of any metaheuristic algorithms are: intensification (or exploitation) and diversification (or exploration) [35]. Diversification means to generate diverse solutions so as to explore the search space on the global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region. They are combined together to select a better solution.

The exploration, in DS algorithm, is realized through *Brownian-like random walks* (see Algorithm 1) and the exploitation is realized through a greedy selection method shown (5). Despite the importance of a fine balance between the right amount of exploration and the right degree of exploitation, there is no practical guideline for this balance [36]. However, the numerical experimental results in Section 4 show that the DS holds a good balance between exploration and exploitation while keeping a simple searching strategy in solution space.

3. D-DS algorithm for constrained optimization

Most common techniques in the population-based algorithms to handle the constraints are assorting to penalty methods. The strengths and weaknesses between various penalty-based methods are discussed in [28]. Dynamic penalty function approach is adapted in this paper because of its numerical efficiency.

3.1. Dynamic penalty factor

Different from the dynamic methods in the literature [28], a new dynamic penalty function method is introduced in this paper. In the dynamic penalty method, an augmented Lagrangian function is appended to the objective function in a way that the penalty parameter vector $\mathbf{p}(gen) = (p_1, p_2, \ldots, p_m)$ is adjusted according to the generation number *gen*, i.e.,

$$F(x) = f(x) + H(\beta, x), \tag{6}$$

$H(\beta, x)$ in Eq. (6) is defined as

$$H(\beta, x) = \sum_{j=1}^{q} p_j P_j^{\beta}(x) + \sum_{j=q+1}^{m} p_j P_j(x). \tag{7}$$

where

$$P_j(x) = \begin{cases} 0, & g_j(x) \le 0, \\ |g_j(x)|, & \text{otherwise.} \end{cases} \quad (1 \le j \le q);$$

and

$$P_j(x) = \begin{cases} 0, & -\epsilon \le g_j(x) \le \epsilon, \\ |g_j(x)|, & \text{otherwise.} \end{cases} \quad (q+1 \le j \le m).$$

And the constant β is chosen as either $\beta = 1$ or $\beta = 2$, the element of penalty vector p_j varies in accordance with the generation number.

Thus, the choice of penalty parameter $\mathbf{p}(gen)$ can adjust the order of magnitude of $f(x)$ and the constraints violation $H(\beta, x)$. Here we will propose a novel S-type function to generate the penalty factor p_j as following:

$$p_j(gen) = \begin{cases} 10^{\theta_1}\left(1 + e^{\frac{\theta_2\left(\frac{G_{\max}}{2} - gen\right)}{G_{\max}}}\right)^{-1}, & v_j > \epsilon; \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where $G_{\max}$ is the maximum iteration number, v_j is the jth constraint violation and ϵ is the tolerance for the constraint violation.

If we assume $v_j > \epsilon$, the variation of p_j with respect to *gen* is shown by left graph in Fig. 2 where $\theta_1 = 3$ and $\theta_2 = 2$, 4 and 6, respectively. The right graph in Fig. 2 shows the variation of penalty factor with respect to θ_1 and θ_2 where $gen = 500$. From Fig. 2,

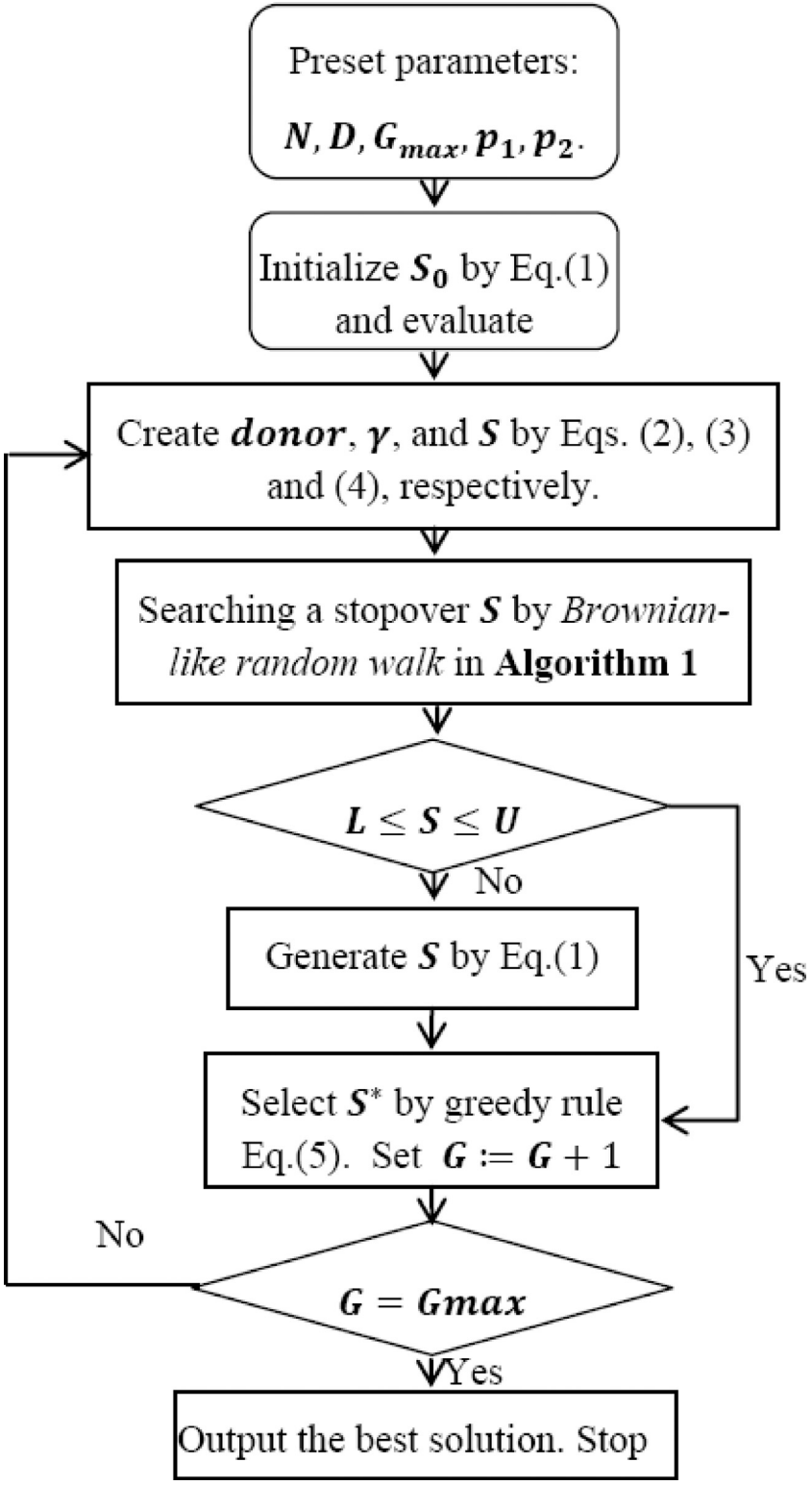

Fig. 1. Flowchart of DS.

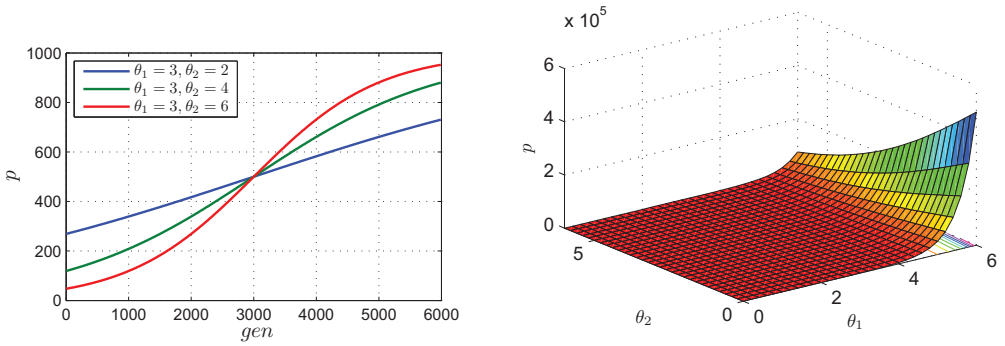

Fig. 2. Left: p versus gen; right: p versus θ_1 and θ_2 where $gen = 6000$.

Algorithm 2 DS with dynamic penalty function (D-DS) for constrained optimization problem.

Initialization:

Set the constraint violation ϵ, initial population $\{S_0\}$, population size $N \in \mathbb{R}^+$, maximum generation number $G_{\max} \in \mathbb{R}^+$, initial penalty parameter $\beta = 2$.

Iteration:

1: **while** Stop criteria are not met **do**
2: Transform the constrained optimization problem (P) into an unconstrained optimization problem (6) through the S-type penalty threshold (8).
3: Apply Algorithm 1 to solve unconstrained optimization problem (6).
4: **end while**
5: Output optimal solution x^*.

we can observe that the dynamic penalty parameter p_j starts from a smaller value. Then, the dynamic parameter p_j increases exponentially with the increase of gen. At initial stage, the smaller penalty parameter will help the superorganism to explore the search space, which, in return, enables the diversity of the population. This means that a certain percentage of infeasible individuals are maintained in the superorganism, which allows determining the global optimum in highly sparse feasible space. At later stage, a larger p_j will lead to the convergence of the algorithm to a feasible point. This means that a large penalty factor restricts the search inside the feasible region, for bidding any shortcut across the infeasible region, and thus eventually the iteration solutions converge to a potential optimum. Thus, the penalty factor $\mathbf{p}(gen)$ generated through (8) has a good property to trade off between exploitation and exploration during the optimization process.

After applying the dynamic penalty function method, the original constrained optimization problem has been transformed to solve a sequence of unconstrained optimization problems (6). Then, Differential Search Algorithm 1 is applied.

3.2. Dynamic penalty DS (D-DS)

Now the computational procedure of the D-DS algorithm for constrained optimization problem is described in Algorithm 2.

Some important features of the D-DS algorithm are worth of mention. The first is that different from ABC and PSO, D-DS algorithm has no inclination to move correctly towards the best possible solution of the problem [27]. This kind of strategy is more suitable for solution of multimodal functions. The second is that the penalty factor is dynamically tuned during the implementation process. At initial stage, the penalty factor is small. Then, it increases exponentially with the increase of the generations. This penalty method may have more chance to capture the global optima because the smaller penalty will be helpful for exploration while the larger penalty will maintain the feasibility of the solution at later stage. The third one is that the searching strategy in D-DS is based on Brownian random walk to simulate the individuals migration behaviors. This kind of searching strategy is simpler than those in the other population-based algorithms. Thus, it is faster and has more potential to solve large-scaled optimization problems. The last one is that D-DS can be implemented in parallel easily since it is a population-based algorithm.

4. Numerical experiments and comparison

In this section, the performance of the proposed D-DS is tested on two benchmark sets, CEC2006 and CEC2010, which can be found in [37] and [38], respectively. All programs were coded in Matlab and executed in a Lenovo PC with Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz.

Table 1
Summary of the 24 test problems in CEC2006 benchmark set.

Prob.	D	Type	ρ(%)	LI	NI	LE	NE	a
g01	13	Quadratic	0.0111	9	0	0	0	6
g02	20	Nonlinear	99.9971	0	2	0	0	1
g03	10	Polynomial	0.0000	0	0	0	1	1
g04	5	Quadratic	52.1230	0	6	0	0	2
g05	4	Cubic	0.0000	2	0	0	3	3
g06	2	Cubic	0.0066	0	2	0	0	2
g07	10	Quadratic	0.0003	3	5	0	0	6
g08	2	Nonlinear	0.8560	0	2	0	0	0
g09	7	Polynomial	0.5121	0	4	0	0	2
g10	8	Linear	0.0010	3	3	0	0	6
g11	2	Quadratic	0.0000	0	0	0	1	1
g12	3	Quadratic	4.7713	0	1	0	0	0
g13	5	Nonlinear	0.0000	0	0	0	3	3
g14	10	Nonlinear	0.0000	0	0	3	0	3
g15	3	Quadratic	0.0000	0	0	1	1	2
g16	5	Nonlinear	0.0204	4	34	0	0	4
g17	6	Nonlinear	0.0000	0	0	0	4	4
g18	9	Quadratic	0.0000	0	13	0	0	6
g19	15	Nonlinear	33.4761	0	5	0	0	0
g20	24	Linear	0.0000	0	6	2	12	16
g21	7	Linear	0.0000	0	1	0	5	6
g22	22	Linear	0.0000	0	1	8	11	19
g23	9	Linear	0.0000	0	2	3	1	6
g24	2	Linear	79.6556	0	2	0	0	2

4.1. Comparison with other constraint handling methods

4.1.1. Test problems in CEC2006 benchmark set

Types of objective functions, the number of variables (D), the numbers of linear equality (LE) constraints, nonlinear equality (NE) constraints, linear inequality (LI) constraints and nonlinear inequality (NI) constraints as well as dimensions of decision variables of the 24 test problem in CEC2006 benchmark set are listed in Table 1. The value of ρ, expressed as a percentage, is the ratio of the feasible region to the given box constrained area. Here ρ is calculated by generating 1,000,000 random points in the box-constrained area of each problem and a is the number of active constraints at optimum x^*. Note that test problems g02, g03, g08, and g12 are maximization, and the others are minimization problems. In this study, the maximization problems are transformed into minimization using $f(x)$. In addition, only test problems g03, g05, g11, and g13 contain equality constraints.

4.1.2. Comparison with other constraint handling methods

We compare our proposed S-type dynamic penalty method with two constraint handling methods, superiority of feasible solution (SF) [39,40] and $\varepsilon-$constraints (EC) [41], to show their performances on the 24 problems in CEC 2006 set whose characters are listed in Table 1.

To handle the constraints, SF adopts the following fitness function:

$$fitness(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible;} \\ f_{\text{worst}}(x) + v(x), & \text{otherwise.} \end{cases} \tag{9}$$

where f_{worst} is the objective value of the worst feasible solution and $v(x)$ is the overall constraint violation. $v(x)$ is calculated through the equation:

$$v(x) = \sum_{i=1}^{m} \frac{v_i(x)}{v_{\text{imax}}} \tag{10}$$

where v_{imax} is the current maximum violation of constraint i obtained so far. If there is no feasible solution in the current population, f_{worst} is zero.

The basic idea of ε-constraint handling technique is similar to SF and the difference is the relaxation of the constraint violation during the early stages of the search process using an ε parameter. The ε is updated until the generation counter k reaches the control generations T_c. After T_c generations, the ε is set to be zero and EC is the same as SF. The ε is updated by the following equation:

$$\varepsilon(k) = \begin{cases} \varepsilon(0)\left(1 - \dfrac{k}{T_c}\right)^{cp}, & 0 \le k \le T_c \\ 0, & k \ge T_c. \end{cases} \tag{11}$$

where $\varepsilon(0) = v(x_\theta)$ with x_θ is the top θth individual and $\theta = (0.05SN)$; parameter cp is recommended to be [2, 10] and T_c is in $[0.1G_{\max}, 0.8G_{\max}]$.

In order to observe the effect of three constraint handling techniques, we make sure that all the components such as initial population, two random parameters p_1, p_2 in DS, are same, respectively. The parameters are set to: $T_c = 0.2G_{\max}$, $c_p = 5$ in CE. All three methods perform within 150,000 function evaluations.

We investigate their performances through the results presented in Table 2 where SF-DS (EC-DS) is the DS algorithms where the constraints are handled through the SF (EC) technique. Both parameters $\theta_1 = 6$ and $\theta_2 = 8$.

From this table, we can clearly observe that for the problems g04, g08, g12 and g24, the successful rate for all three methods are 100%. For the problems g10, g13, g17, g19, g20 and g21, all the three methods cannot secure optimal solutions. For all the other problems except g03, D-DS achieves the best performance in terms of successful rate SR. For g03, SF-DS achieves the best performance.

Another kind of penalty methods in the literature is to increase the penalty factor if the current solution is not feasible. However, there is lacking of a rule to guide the update of the penalty factor. In implementation process, this factor is always set as $p_j(gen) = 10 * p_j(gen - 1)$. Let this kind of DS algorithm be referred to as infinite penalty DS (IP-DS). After several iterations, this number becomes huge and the problem becomes ill-conditioned. Here we pick up the problems g05, g11 and g18 as examples to solve it by our S-type penalty method and that one by $p_j(gen) = 10 * p_j(gen - 1)$. The following table shows the corresponding results. From Table 3, we can clearly observe that although IP-DS can find optimal solutions for the problems g11 and g18, its successful rate is much lower than that obtained by D-DS. Furthermore, the mean clearly shows that the problem becomes ill-conditioned when the penalty factor becomes large.

4.2. Comparison with other population-based algorithms on test problems in CEC2006 benchmark set

Several well-known and widely-used evolution algorithms are applied to solve the 24 problems in CEC 2006 or part of them, where the parameters and the methods for handling constraints in each algorithm are described below.

GA uses Deb's rules as constraint handling method [42]. All equality constraints have been converted into inequality constraints, $|h_j| \le \epsilon$ with ϵ varying dynamically. For GA, population size is 200, the maximum number of generations is 1200, the crossover rate is 0.8, the mutation rate is 0.6 and the number of objective function evaluations is 240,000.

CVI-PSO introduces the total constraints violation as an objective function to minimize [23]. Interval arithmetic is used to normalize the total violations. The resulting objective problem is solved using a simple lexicographic method. The swarm size is 50 and the generation number is 500. Hence, PSO performs 25,000 objective function evaluations. CVI-PSO used a w decreasing with iteration t in a way to balance global and local searches and to achieve better convergence.

In DE[43], the parameter which affects the differential variation between two solutions is set to 0.5 in experiments. Value of crossover rate, which controls the change of the diversity of the population, is chosen to be 0.9 as recommended. Population size is 90, maximum generation number is 2666 and Deb's rules is adopted.

In ABC algorithm[3], the colony size is 40 and the maximum iteration number is 6000. Thus, the total objective function evaluations number is 240,000. The value of limit is equal to $0.5 \times sn \times D$, where D is the dimension of the problem and sn is the number of solutions in the population, and the scout production period is also set as $0.5 \times sn \times D$.

Especially, population sizes are set differently with different test problems and maximum number of objective function evaluations is 500,000 in modified APM-ES [17]. Equality constraints are transformed into inequalities of the form $|g_j(x)| - \epsilon \le 0$, and $\epsilon = 10^{-4}$.

Experiments are performed 25 runs independently. The results obtained by those existing methods are taken from the references cited. Their swarm sizes, evaluations and maximum generations are listed in Table 4. We apply our proposed D-DS algorithm to solve these 24 problems for the comparison study with $\theta_1 = 6$ and $\theta_2 = 8$ in Eq. (8). The numerical results obtained by our method as well as many other swarm-based methods is listed in Table 5.

Table 5 shows the numerical results by D-DS as well as some other swarm-based optimization methods. The results for APM-ES are taken from Table 1 given in [17]. For the test problems from g01 to g24, we apply D-DS to solve them and compare the best results with GA in [42], CVI-PSO in [23], DE variants in [43], ABC in [3] and APM-ES in [17]. Among these six methods, our proposed algorithm D-DS performs much better than other algorithms and it can successfully achieves 16 optimal results more than 12, 14 ones which achieved by recent algorithms, APM-ES and DE variants, respectively. Then, it follows DE variants in [43]. GA in [42] is the worst one.

D-DS can successfully secure optimal solutions for 16 problems among 24 problems. We notice that all six algorithms cannot solve g20 and g22. From the literature, we know that there is no feasible solution for g20, while for g22 it is rarely to find a feasible solution [37]. Therefore, the solutions obtained by D-DS are close to the true optimal solutions for all the other problems except g13, g20 and g22. However, the local convergence for D-DS is very slow. For g13, D-DS fails to identify an approximate solution. Thus, the failure of D-DS to identify a global optimal solution may be caused by either stuck in local optimal or slow local convergence.

From Table 5, we also notice that D-DS, APM-ES and DE variants achieve more optimal results for test problems from g14 to g24.

In addition, statistical significance testing is a meaningful way to study the difference between any two stochastic algorithms. We have chosen a non-parametric test, Wilcoxon signed rank test that has been applied in literatures [44,45], to judge the difference between paired scores. The statistical results based on the best fitness values are presented in Table 6, where $R = R_-$

Table 2
Statistical results obtained by DS algorithm with three constraint handling techniques for 24 test functions over 25 independent runs using 150,000 objective function evaluations. Std. – standard deviation; SR. – successful rate; FR. – feasible rate.

Prob.	Optimal value	Method	Best	Mean	Worst	Std.	SR.(%)	FR.(%)
g01	−15.000	D-DS	−15.0000	−14.9999	−12.0630	0.0004	98	100
		SF-DS	−15.0000	−14.7883	−9.7708	0.6660	10	100
		EC-DS	−15.0000	−15.0000	−15.0000	0.0000	26	100
g02	−0.803619	D-DS	−0.8036	−0.7944	−0.7704	0.0083	62	100
		SF-DS	−0.8277	−0.4763	−0.2783	0.6745	5	100
		EC-DS	−0.7925	−0.7836	−0.7644	0.0230	6	100
g03	−1.000	D-DS	−1.0000	−0.9988	−0.8745	0.5895	60	100
		SF-DS	−1.0005	−0.8034	−0.6531	0.7635	73	100
		EC-DS	−0.7190	−0.3448	−0.1376	0.8745	0	100
g04	−30665.539	D-DS	−30665.539	−30665.2313	−30665.1899	0.0252	100	100
		SF-DS	−30665.5508	−30665.5508	−30665.5508	0.0000	100	100
		EC-DS	−30665.5508	−30665.5508	−30665.5508	0.0000	100	100
g05	5126.498	D-DS	5126.4981	5309.6169	5751.7177	194.7202	68	100
		SF-DS	5126.4969	5158.3317	5329.3866	173.8937	56	100
		EC-DS	5127.6752	5257.1609	5400.7901	0.0000	0	100
g06	−6961.814	D-DS	−6961.814	−6959.1871	−6943.1037	6.0243	96	100
		SF-DS	−6961.8141	−6961.8101	−6961.8371	0.4500	34	25
		EC-DS	−6961.8140	−6961.8071	−6961.8071	0.0320	56	34
g07	24.306	D-DS	24.3065	24.4592	25.0048	0.2352	76	100
		SF-DS	24.3484	24.6694	25.8783	0.0000	14	100
		EC-DS	24.3526	24.4261	24.4963	0.0000	8	100
g08	−0.095825	D-DS	−0.095825	−0.0958	−0.0958	0.0000	100	100
		SF-DS	−0.0958	−0.0958	−0.0958	0.0000	100	100
		EC-DS	−0.0958	−0.0958	−0.0958	0.0000	100	100
g09	680.63	D-DS	680.630	680.6303	680.6326	0.0005	100	100
		SF-DS	680.6300	680.6705	680.6326	0.0015	100	100
		EC-DS	680.6307	680.7758	681.1061	0.3652	30	100
g10	7049.25	D-DS	7056.2024	7087.2024	7234.3331	80.0434	0	100
		SF-DS	7054.1548	7118.6628	7123.6326	34.0982	0	100
		EC-DS	7049.9551	7093.3699	7701.0655	12.0430	0	100
g11	0.7499	D-DS	0.7499	0.7837	1.0000	0.0702	58	100
		SF-DS	0.7499	0.7561	1.0500	0.7572	58	100
		EC-DS	0.7516	0.8622	0.9587	0.2060	0	100
g12	−1.000	D-DS	−1.000	−1.0000	−1.0000	0.0000	100	100
		SF-DS	−1.000	−1.0000	−1.0000	0.0000	100	100
		EC-DS	−1.000	−1.0000	−1.0000	0.0000	100	100
g13	0.5395	D-DS	0.6485	7.3028	200.2849	45.1173	0	87
		SF-DS	0.6965	8.9956	55.7326	24.3650	0	42
		EC-DS	0.6850	48.9476	511.7326	102.0546	0	35
g14	−47.7649	D-DS	−47.7649	−47.1031	−46.2942	0.2893	50	100
		SF-DS	−46.1251	−43.9162	−42.2627	0.5463	0	100
		EC-DS	−45.5565	−45.0119	−44.5309	0.8750	0	25
g15	961.7150	D-DS	961.7150	963.4312	969.3198	2.2317	52	100
		SF-DS	962.5683	965.0846	967.5207	3.0520	22	100
		EC-DS	961.7520	961.9402	962.2946	0.9365	10	100
g16	−1.9052	D-DS	−1.9052	−1.8961	−1.8119	0.0238	36	100
		SF-DS	−1.9052	−1.8938	−1.8533	0.0438	40	100
		EC-DS	−1.9052	−1.9052	−1.9051	0.0005	16	100
g17	8853.5397	D-DS	8853.8303	9035.2630	9305.3250	112.7391	0	100
		SF-DS	8853.8426	9014.6844	9196.8202	23.3106	0	50
		EC-DS	8945.8154	9080.0846	9251.9289	10.1740	0	82
g18	−0.8660	D-DS	−0.8660	−0.8506	−0.6732	0.0532	92	100
		SF-DS	−0.8660	−0.8525	−0.6680	0.1326	20	100
		EC-DS	−0.8660	−0.8471	−0.7904	0.0927	56	100
g19	32.6555	D-DS	32.8047	35.8122	46.1658	2.8073	0	100
		SF-DS	37.4245	49.5936	76.9940	5.2300	0	100
		EC-DS	33.6618	39.4379	45.5592	2.0540	0	100
g20	0.2050	D-DS	0.1962	3.3456	4.2527	5.7404	0	65
		SF-DS	0.7131	0.8589	1.5347	1.5923	0	15
		EC-DS	0.1970	3.7204	4.2345	4.7738	0	25
g21	193.7245	D-DS	193.7814	193.7816	193.7817	0.9680	0	100
		SF-DS	193.7780	267.1908	1001.6579	0.6578	0	100
		EC-DS	193.7320	240.3608	334.8531	63.5037	0	100
g22	0.2050	D-DS	–	–	–	–	–	–
		SF-DS	–	–	–	–	–	–
		EC-DS	–	–	–	–	–	–
g23	−400.0551	D-DS	−400.0551	−385.1774	−350.9571	34.8560	66	100
		SF-DS	−84.8438	−66.8793	−108.6493	98.5740	12	70
		EC-DS	−242.0679	−130.3189	−20.0670	54.3809	10	79
g24	−5.5080	D-DS	−5.5080	−5.5080	−5.5080	0.0000	100	100
		SF-DS	−5.5080	−5.5080	−5.5080	0.0000	100	100
		EC-DS	−5.5080	−5.5080	−5.5080	0.0000	100	100

Table 3
Statistical results on D-DS and IP-DS.

Prob.	Optimal value	Method	Best	Mean	Worst	Std.	SR.(%)	FR.(%)
g05	5126.498	IP-DS	5126.5035	3978805.9158	19288508.8500	6889294.7261	0	10
		D-DS	5126.4959	5126.5014	5126.5478	0.0163	72	100
g011	0.7499	IP-DS	0.7500	0.7531	0.7706	0.0067	12	18
		D-DS	0.7499	0.7500	0.7500	0.0000	78	100
g018	−0.8660	IP-DS	−0.8660	1433005.3342	2935012.3834	1301722.0942	15	20
		D-DS	−0.8660	−0.8107	−0.6733	0.0864	91	100

Table 4
The size of swarm, evaluation number of objective function and maximum number of generation in the GA, PSO, DE variants, ABC, APM-ES and D-DS algorithms for 24 test problems. The numbers of size and generation adopted in APM-ES vary with the test problems in literature.

Algorithm	GA	CVI-PSO	DE variants	ABC	APM-ES	D-DS
Size	200	50	90	40		30
Evaluation	240,000	25,000	240,000	240,000	500,000	150,000
Generation	1200	500	2666	6000		5000

Table 5
The best results obtained by the GA [42], CVI-PSO [23], DE [43], ABC [3] and APM-ES [17] algorithms for 24 test problems over 25 independent runs. "−" means no feasible solution were found by the algorithm. "NA" means not be applied to solve this problem by algorithm.

Prob.	Optimal	GA variants	CVI-PSO	DE variants	ABC	APM-ES	D-DS
g01	−15.000	−14.440	−14.9999	**−15.000**	**−15.000**	−14.9999	**−15.000**
g02	−0.803619	−0.796231	−0.800097	−0.704009	−0.803598	−0.4938	**−0.8036**
g03	−1.000	−0.990	**−1.000000**	−0.461	**−1.000**	**−1.000**	**−1.0000**
g04	−30665.539	−30626.053	−30665.8217	**−30665.539**	**−30665.539**	**−30665.5386**	**−30665.539**
g05	5126.498	–	5127.2776	5126.497	5126.484	5126.4967	5126.4946
g06	−6961.814	−6952.472	−6961.8138	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**
g07	24.306	31.097	24.473826	**24.306**	24.330	**24.306**	**24.306**
g08	−0.095825	**−0.095825**	−0.105459	**−0.095825**	**−0.095825**	−0.095821	**−0.095825**
g09	680.63	685.994	680.635400	**680.630**	680.634	**680.630**	**680.630**
g10	7049.25	9079.770	7049.2765	**7049.248**	7053.904	**7049.2480**	7056.2024
g11	0.7499	0.75	0.750000	0.75	0.750	**0.7499**	**0.749951**
g12	−1.000	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**
g13	0.053950	0.134057	0.055555	0.0597	0.760	0.0564	0.6485
g14	−47.7649	−45.038	−47.453011	**−47.764888**	NA	−47.7648	**−47.764888**
g15	961.7150	–	961.715707	**961.715022**	NA	**961.7150**	**961.7150**
g16	−1.9052	−1.899	−1.905154	−1.905155	NA	−1.9050	**−1.905158**
g17	8853.5397	–	**8853.5398**	8853.541289	NA	8854.4803	8853.8303
g18	−0.8660	−0.852457	−0.864631	**−0.866025**	NA	**−0.8660**	**−0.866034**
g19	32.6555	102.080	32.827027	**32.655593**	NA	**32.6556**	32.804792
[1]g20	0.2050	–	0.214689	–	NA	–	0.1962410
g21	193.7245	–	193.786925	193.724510	NA	**193.7245**	193.7320
g22	236.4310	–	NA	–	NA	–	–
g23	−400.0551	–	−400.000000	**−400.055093**	NA	−400.0550	**−400.0551**
g24	−5.5080	−5.508	**−5.508013**	**−5.508013**	NA	−5.5079	**−5.508043**

[1] Infeasible solution.

or R_+ is the sum of ranks based on the absolute value of the difference between two test algorithms. The sign of the difference between both independent samples is used to classify cases into one of two samples: differences above zero (positive rank R_+), or below zero (negative rank R_-).

As a null hypothesis, there is no significant difference between the best/or mean values of two samples. Whereas the alternative hypothesis it is assumed that there is a significant difference in the best and/or mean fitness values of the two samples, at the 10% significance level. We use the well-known statistical software packages SPSS 19 for the computation of the p-value for these tests. Based on the test results/rankings, we assign one of three signs ($+$, $-$, and $\approx$) for the comparison of any two algorithms column), where the sign "+" and "−" mean that the first algorithm is significantly better and worse than the second, respectively. And the " $\approx$ " sign means that there is no significant difference between the two algorithms. From Table 6, D-DS is clearly superior to GA and ABC with regard to the best solutions.

Table 6
Wilcoxon sign rank results on the best function values obtained by D-DS against GA, PSO, DE variants, ABC and APM-ES algorithms for CEC2006 set.

Algorithm	Better	Equal	Worse	R_-	R_+	p Value	Decision
D-DS to GA	20	2	1	99	6	0.004	+
D-DS to CVI-PSO	16	2	5	111	99	0.823	≈
D-DS to DE variants	10	8	5	44	61	0.594	≈
D-DS to ABC	5	6	2	22	6	0.017	+
D-DS to APM-ES	12	6	5	54	37	0.552	≈

Table 7
Details of the 18 test problems in CEC2010 benchmark set (NonSep.: Non-Separable; Sep.: Separable).

Prob.	Objective type	Number of constraints		Feasibility (ρ%)	
		E	I	10D	30D
C01	NonSep.	0	2, (NonSep.)	0.997689	1.000000
C02	Sep.	1, (Sep.)	2, (Sep.)	0.000000	0.000000
C03	NonSep.	1, (NonSep.)	0	0.000000	0.000000
C04	Sep.	4, (2 NonSep., 2 Sep.)	0	0.000000	0.000000
C05	Sep.	2, (Sep.)	0	0.000000	0.000000
C06	Sep.	2, (Rotated)	0	0.000000	0.000000
C07	NonSep.	0	1, (Sep.)	0.505123	0.503725
C08	NonSep.	0	1, (Rotated)	0.379512	0.375278
C09	NonSep.	1, (Sep.)	0	0.000000	0.000000
C10	NonSep.	1, (Rotated)	0	0.000000	0.000000
C11	Rotated	1, (NonSep.)	0	0.000000	0.000000
C12	Sep.	1, (NonSep.)	1, (Sep.)	0.000000	0.000000
C13	Sep.	0	3, (2 Sep., 1 NonSep.)	0.000000	0.000000
C14	NonSep.	0	3, (Sep.)	0.003112	0.006123
C15	NonSep.	0	3, (Rotated)	0.003210	0.006023
C16	NonSep.	2, (Sep.)	2, (1 Sep., 1 NonSep.)	0.000000	0.000000
C17	NonSep.	1, (Sep.)	2, (NonSep.)	0.000000	0.000000
C18	NonSep.	1, (Sep.)	1, (Sep.)	0.000010	0.000000

4.3. Performance on test problems in CEC2010 benchmark set

4.3.1. Test problems in CEC2010 benchmark set

In order to further test the performance of our algorithm, we will apply D-DS to CEC2010 benchmark set. It has been proved that CEC2010 set is much complex than CEC2006 set [38]. Details of the 18 test problems in CEC2010 benchmark set are listed in Table 7. The value of ρ is still the ratio of the feasible region to the given box constrained area. I is the number of inequality constraints, E is the number of equality constraints. 10D and 30D represent the dimension of problem.

4.3.2. Numerical performance of D-DS on CEC2010 benchmark set

In the numerical experiment on the CEC2010 benchmark set we set the population number $N = 20$, the maximum number of generations $gen = 10,000$ for 10D problems. Thus, the maximum number of objective function evaluations is $Max_{\text{fes}} = 200{,}000$, which equals to Max_{fes} adopted in APM-ES [18], εDEag (the CEC2010 competition winning algorithm) [46], CCMA(A constraint consensus memetic algorithm) [47] and BGRA (bacterial gene recombination algorithm)[48]. For 30D problems of CEC2010 set, the population number $N = 60$, the maximum number of generations $gen = 20,000$. The same as the setting in experiment on CEC2006 benchmark set, we assume that a solution is feasible if violation of any equality constraint is less than or equals to 10^{-4}. All the tests are independently run over 25 trials.

The numerical results on 10D problems of CEC2010 obtained by D-DS are presented in Table 8. The best solutions obtained among the five algorithms are marked in bold font. From Table 8, we can observe that among the 18 test problems, D-DS achieves 14 best results, APM-ES achieves 1 best result, εDEag achieves 10 best results, CCMA achieves 14 best results and EAwAC achieves 14 best results. In addition, for the test problems C12 and C13, the results obtained by D-DS are much better than all the others.

Table 9 presents the numerical results obtained by D-DS and other three algorithms on 30D problems of CEC2010 set. The best solutions obtained among the four algorithms are marked in bold font. From Table 9, we can observe that among the 18 test problems, D-DS achieves 11 best results, εDEag achieves 1 best results, CCMA achieves 2 best results and EAwAC achieves 6 best results.

According to the non-parametric statistical test with the 10% significance level, as shown in Table 10, for the 10D test problems, D-DS is better than APM-ES in terms of the best values and there is no significant difference to εDEag, CCMA and EAwAc. For the 30D test problems, D-DS is better than εDEag and CCMA with regard to the best results, while there is no significant difference with regard to the average results.

Table 8
Comparison of performances of D-DS, APM-ES [18], εDEag [46], CCMA [47] and EAwAC [56] for 18 test problems of 10D over 25 independent runs using 200,000 objective function evaluations. Best means the best known result for CEC2010 problems. "–" indicates that the data are not listed in literature.

Prob.	Alg.	Best	Mean	Worst	Std.
C01	D-DS	**−7.473104e-01**	−7.4713e-01	−7.4604e-01	3.8754e-04
	APM-ES	−7.4731e-01	−7.2926e-01	−6.7825e-01	–
	εDEag	**−7.473104e-01**	−7.470402e-01	−7.405572e-01	1.323339e-03
	CCMA	**−7.473104e-01**	−7.467701e-01	–	1.869859e-03
	EAwAC	**−0.7473104**	−0.7440821	−0.7292159	4.804888e-03
C02	D-DS	**−2.277710e-00**	−2.2648e-00	−2.2423e-00	9.4201e-03
	APM-ES	−2.2777e+00	−2.2700e+00	−2.2612e+00	–
	εDEag	−2.277702e+00	−2.258870e+00	−2.174499e+00	2.389779e-02
	CCMA	**−2.277710e+00**	−2.277707e+00	–	2.517499e-06
	EAwAC	−2.277709e+00	−2.255688e+00	−2.225884e+00	1.728680e-02
C03	D-DS	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
	APM-ES	6.3847e-17	1.3845e-16	1.0118e-16	–
	εDEag	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C04	D-DS	**−1.0000e-05**	−2.2450e-03	−4.3334e-06	1.0563e-03
	APM-ES	−9.9974e-06	−9.2646e-06	−1.7039e-06	–
	εDEag	−9.992345e-06	−9.918452e-06	−9.282295e-06	1.546730e-07
	CCMA	**−1.000000e-05**	−1.000000e-05	–	−1.000000e-05
	EAwAC	**−1.0000e-05**	−1.0000e-05	−1.0000e-05	0.0000e+00
C05	D-DS	−4.8090e+02	−2.0935e+02	−1.0556e+02	4.1581e-01
	APM-ES	−4.8361e+02	−4.8361e+02	−4.8361e+02	–
	εDEag	−4.836106e+02	−4.836106e+02	−4.836106e+02	3.890350e-13
	CCMA	−4.836106e+02	−4.836106e+02	–	7.046557e-09
	EAwAC	**−483.610625**	−483.610625	−483.610625	5.884488e-06
C06	D-DS	−5.7866e+02	−5.6368e+02	−3.7382e+02	5.6724e-01
	APM-ES	−5.7866e+02	−5.7866e+02	−5.7866e+02	–
	εDEag	−5.786581e+02	−5.786528e+02	−5.786448e+02	3.627169e-03
	CCMA	**−5.786624e+02**	−5.786602e+02	–	2.546542e-03
	EAwAC	−5.78662e+02	−5.78269e+02	−5.76425e+02	6.253216e-01
C07	D-DS	**0.0000e+00**	1.9032e-22	1.9024e-21	6.0155e-22
	APM-ES	5.0579e-17	4.3422e-16	1.6036e-15	–
	εDEag	**0.000e+00**	0.000e+00	0.000e+00	0.000e+00
	CCMA	**−0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.000e+00**	2.375369e-27	2.681343e-26	5.412546e-27
C08	D-DS	**0.000e+00**	1.0601e-15	3.9897e-08	1.5312e-09
	APM-ES	1.8208e-16	1.0281e+02	7.7107e+02	–
	εDEag	**0.000e+00**	6.727528e+00	1.537535e+01	5.560648e+00
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.000e+00**	3.837529e+00	1.094154e-01	5.229141e+00
C09	D-DS	**0.0000e+00**	0.9421e-11	1.4618e-12	3.4685e-05
	APM-ES	4.565e-18	2.1880e-17	7.9361e-17	–
	εDEag	**0.000e+00**	0.000e+00	0.000e+00	0.000e+00
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C10	D-DS	**0.0000e+00**	2.2246e-11	2.3527e-02	5.1273e-05
	APM-ES	4.5359e-18	3.4579e-17	1.8404e-16	–
	εDEag	**0.000e+00**	0.000e+00	0.000e+00	0.000e+00
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C11	D-DS	−1.5227e-03	−1.520231e-03	−1.710080e-04	1.542761e-04
	APM-ES	−1.5227e-03	−6.4461e-04	−8.734e-04	–
	εDEag	**−1.522713e-03**	−1.522713e-03	−1.522713e-03	6.341035e-11
	CCMA	−1.52271e-03	−1.52271e-03	–	8.912164e-12
	EAwAC	**1.522713e-03**	1.522713e-03	1.522713e-03	0.0000e+00
C12	D-DS	**−5.7018e+02**	−3.7076e+02	−2.4201e-01	1.2146e+02
	APM-ES	−1.2667e+01	−1.8616e+00	−1.9925e-01	–
	εDEag	−5.700899e+02	−3.367349e+02	−0.1989129	1.782166e+02
	CCMA	−3.054888e+02	−6.118898e+01	–	1.244952e+02
	EAwAC	−1.992458e-01	−1.990785e-01	−1.950644e-01	8.362746e-04
C13	D-DS	**−6.84294e+01**	−6.8417e+01	−6.7660e+01	2.7470e-03
	APM-ES	−6.5578e+01	−6.2129e+01	−5.7140e+01	–
	εDEag	−6.842937e+01	−6.842936e+01	−6.842936e+01	1.02596e-06
	CCMA	−6.842937e+01	−6.692766e+01	–	1.837361e+00
	EAwAC	−68.42937	−6.126890e-01	−5.722131e-01	2.490391e+00

(*continued on next page*)

Table 8 (*continued*)

Prob.	Alg.	Best	Mean	Worst	Std.
C14	D-DS	**0.0000e+00**	8.4168e-08	2.7719e-02	1.0642e-02
	APM-ES	3.0316e-17	9.4623e-17	1.6937e-16	–
	εDEag	**0.000e+00**	0.000e+00	0.000e+00	0.000e+00
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C15	D-DS	**0.0000e+00**	3.3007e-18	6.7360e-06	1.3493e-11
	APM-ES	3.6732e+00	7.6642e+03	5.0119e+04	–
	εDEag	**0.000e+00**	1.798978e-01	4.497445	8.813156e-01
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C16	D-DS	**0.0000e+00**	7.7381e-13	3.6872e-05	1.2662e-10
	APM-ES	**0.0000e+00**	0.0000e+00	0.0000e+00	–
	εDEag	**0.000e+00**	3.702054e-01	1.018265	3.710479e-01
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C17	D-DS	7.8860e-22	2.0009e-15	6.7929e-12	2.4019e-12
	APM-ES	8.7999e-18	3.8026e-17	1.0929e-16	–
	εDEag	1.463180e-17	1.249561e-01	7.301765e-01	1.937197e-01
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C18	D-DS	**0.0000e+00**	2.0911e-20	2.0911e-19	6.6127e-20
	APM-ES	8.4022e-20	9.8756e-19	5.9669e-18	–
	εDEag	3.731439e-20	9.678765e-19	9.227027e-18	1.811234e-18
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	3.614679e-03	5.590398e-02	1.151939e-02

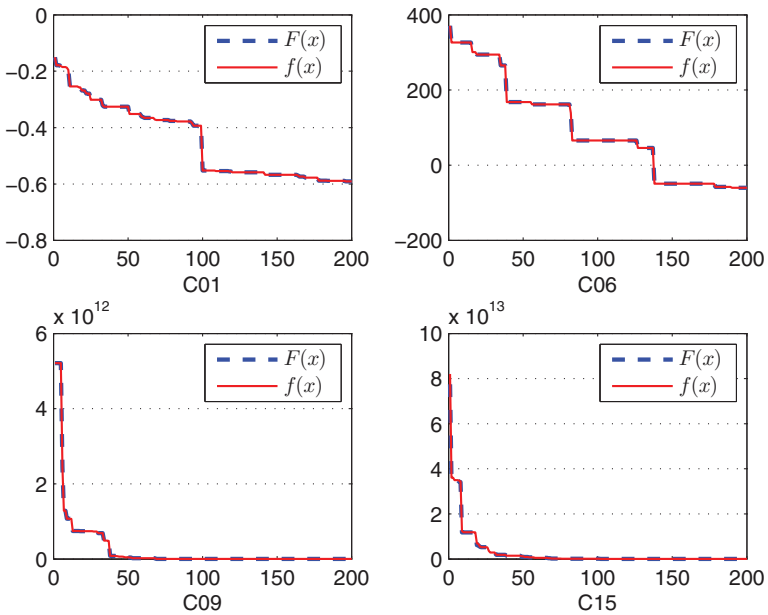

Fig. 3. A search trial of D-DS for four test problems selected from CEC2010 benchmark set.

Fig. 3 present the convergence plots for four problems, C01, C06, C09 and C15, which are selected from CEC2010 set. Also we can find that the objective function $f(x)$ is quickly close to the augmented Lagrangian function $F(x)$ along with the increasing of iteration. It indicates that the dynamic S-type soft-threshold penalty method in D-DS is effective to ensure the search process performing in feasible region.

4.4. Numerical performance of S-type threshold dynamic penalty

In this part, we will compare our method with static penalty function method and the dynamic penalty method in [42]. If replacing the S-type dynamic penalty with a constant penalty in Algorithm 2, we refer such a method as S-DS. Taking the test problems g07 and g19 from CEC2006 set as examples, we apply both D-DS and S-DS to solve them. For S-DS, the static penalty parameter is taken as 10^{10} in order to ensure the feasibility of the obtained solution. The changes of $\log f(x)$ and $\log H(\beta, x)$ in

Table 9
Comparison of performances of D-DS, εDEag [46], CCMA [47] and EAwAC [56] for 18 test problems of 30D over 25 independent runs using 600,000 objective function evaluations.

Prob.	Alg.	Best	Mean	Worst	Std.
C01	D-DS	**−8.218847e-01**	−8.115832e-01	−7.916960e-01	7.027006e-03
	εDEag	−8.218255e-01	−8.208687e-01	−8.195466e-01	7.103893e-04
	CCMA	−8.218842e-01	−8.141462e-01	–	6.999374e-03
	EAwAC	−8.072452e-01	−7.889774e-01	−7.607646e-01	1.220152e-02
C02	D-DS	**−2.280916e+00**	−2.271313e+00	−2.250444e+00	7.142902e-03
	εDEag	−2.169248e+00	−2.151424e+00	−2.117096e+00	1.197582e-02
	CCMA	−2.277471e+00	−2.270847e+00	–	5.973133e-03
	EAwAC	−2.280567e+00	−2.195402e+00	−2.109924e+00	5.466699e-02
C03	D-DS	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
	εDEag	2.867347e+01	2.883785e+01	3.278014e+01	8.047159e-01
	CCMA	3.240477e-24	8.442296e-02	–	4.218478e-01
	EAwAC	**0.0000e+00**	2.150488e-15	3.554911e-14	7.894484e-15
C04	D-DS	**−5.2688e-06**	−3.9381e-04	−2.0000e-06	2.0937e-04
	εDEag	4.698111e-03	8.162973e-03	1.777889e-02	3.067785e-03
	CCMA	−3.321029e-06	−3.235763e-06	–	8.985472e-08
	EAwAC	−1.835577e-06	3.435516e-05	1.572209e-04	4.714218e-05
C05	D-DS	−2.572078e+02	−2.282596e+02	−1.026403e+02	3.497743e+01
	εDEag	−4.531307e+02	−4.495460e+02	−4.421590e+02	2.899105
	CCMA	**−4.836106e+02**	−4.836093e+02	–	2.447876e-03
	EAwAC	−4.835202e+02	−4.816334e+02	−4.767044e+02	2.017331e+00
C06	D-DS	**−5.306385e+02**	−5.306378e+02	−5.306374e+02	1.220010e-04
	εDEag	−5.285750e+02	−5.279068e+02	−5.264539e+02	4.748378e-01
	CCMA	−5.306377e+02	−5.303425e+02	–	9.714491e-01
	EAwAC	−5.302936e+02	−5.281453e+02	−5.257907e+02	1.413775e+00
C07	D-DS	3.208756e-26	2.716274e-23	2.909083e-22	7.625977e-23
	εDEag	1.147112e-15	2.603632e-15	5.481915e-15	1.233430e-15
	CCMA	7.431642e-26	2.832806e-21	–	1.306951e-20
	EAwAC	**0.0000e+00**	2.262586e-17	5.187586e-16	1.035254e-16
C08	D-DS	**1.939944e-27**	4.116162e+01	1.235943e+02	5.540764e+01
	εDEag	2.518693e-14	7.831464e-14	2.578112e-13	4.855177e-14
	CCMA	3.210560e-25	4.595020e+00	–	1.709443e+01
	EAwAC	2.127521e-27	1.076735e+02	3.302350e+02	1.369843e+02
C09	D-DS	3.697681e-25	2.199693e+01	1.671157e+02	4.751215e+01
	εDEag	2.770665e-16	1.072140e+01	1.052759e+02	2.821923e+01
	CCMA	5.615222e-25	3.196975e-06	–	1.577724e-05
	EAwAC	**0.0000e+00**	1.184249e-21	2.960558e-20	5.921111e-21
C10	D-DS	7.718375e-25	1.966411e-12	6.505399e-02	2.165925e+02
	εDEag	3.252002e+01	3.326175e+01	3.463243e+01	4.545577e-01
	CCMA	3.493753e-26	9.758560e-01	–	2.925281e+00
	EAwAC	**0.0000e+00**	2.951691e-25	5.616014e-24	1.136121e-24
C11	D-DS	**−1.783980e-03**	2.312681e+04	2.274368e+05	6.569011e+04
	εDEag	−3.268462e-04	−2.863882e-04	−2.236338e-04	2.707605e-05
	CCMA	−3.923425e-04	−3.923363e-04	–	9.433135e-09
	EAwAC	−3.923440e-04	−3.923440e-04	−3.923440e-04	1.503347e-14
C12	D-DS	**−2.4953e-01**	−4.4660e+02	−5.5676e-01	1.9182e+02
	εDEag	−1.991453e-01	3.562330e+02	5.461723e+02	2.889253e+02
	CCMA	−1.992635e-01	−1.992633e-01	–	2.446028e-07
	EAwAC	−1.992635e-01	−1.992635e-01	−1.992635e-01	6.843205e-10
C13	D-DS	**−6.843070e+01**	−6.797596e+01	−6.670606e+01	5.471462e-01
	εDEag	−6.642473e+01	−6.535310e+01	−6.429690e+01	5.733005e-01
	CCMA	−6.842931e+01	−6.202288e+01	–	3.895266e+00
	EAwAC	−6.155306e+01	−5.731579e+01	−5.489979e+01	1.744858e+00
C14	D-DS	9.915193e-26	6.312528e-24	2.532991e-23	1.071981e-23
	εDEag	5.015863e-14	3.089407e-13	2.923513e-12	5.608409e-13
	CCMA	7.770359e-26	8.655464e-20	–	2.581885e-19
	EAwAC	**0.0000e+00**	3.869646e-21	1.006104e-19	1.973133e-20
C15	D-DS	2.571726e-25	4.4065e+01	3.1311e+02	8.5266e+01
	εDEag	2.160345e+01	2.160376e+01	2.160403e+01	1.104834e-04
	CCMA	5.279373e-26	9.508252e-16	–	4.735074e-15
	EAwAC	**0.0000e+00**	9.301206e-27	3.216107e-26	1.015868e-26
C16	D-DS	**0.0000e+00**	6.7754e-04	2.6632e-03	8.3846e-04
	εDEag	**0.0000e+00**	2.168404e-21	5.421011e-20	1.062297e-20
	CCMA	**0.000000e+00**	0.000000e+00	–	0.000000e+00
	EAwAC	**0.0000e+00**	0.0000e+00	0.0000e+00	0.0000e+00
C17	D-DS	4.912460e-30	2.108026e-01	3.513379e-01	1.924356e-01
	εDEag	2.165719e-01	6.326487e+00	1.889064e+01	4.986691e+00
	CCMA	2.226623e-10	1.328045e-01	–	1.728278e-01
	EAwAC	**1.030373e-32**	3.029488e-32	5.960753e-32	1.507599e-32
C18	D-DS	**3.666971e-30**	1.760165e-24	2.519038e-23	6.021888e-24
	εDEag	1.226054e+00	8.754569e+01	7.375363e+02	1.664753e+02
	CCMA	2.214269e-07	5.406446e-01	–	1.793574e+00
	EAwAC	5.348396e-07	5.030256e-01	4.247972e+00	9.789280e-01

Table 10
Wilcoxon sign rank results on best function values obtained by D-DS against APM-ES, εDEag, EAwAC and BGRA algorithms for CEC2010 set.

Algorithm	Dim.	Criteria	Better	Equal	Worse	R_-	R_+	p Value	Decision
D-DS to εDEag	10D	Best	7	9	2	34	21	0.508	$\approx$
		Mean	9	1	8	92	61	0.463	$\approx$
D-DS to APM-ES		Best	14	3	1	97	23	0.036	+
		Mean	10	0	8	89	82	0.879	$\approx$
D-DS to CCMA		Best	3	10	5	18	27	0.594	$\approx$
		Mean	3	1	14	53	100	0.266	$\approx$
D-DS to EAwAC		Best	5	10	3	32	13	0.260	$\approx$
		Mean	9	0	9	110	43	0.113	$\approx$
D-DS to εDEag	30D	Best	16	1	1	131	22	0.010	+
		Mean	11	0	7	103	68	0.446	$\approx$
D-DS to CCMA		Best	13	1	4	114	39	0.076	+
		Mean	10	0	8	79	92	0.777	$\approx$
D-DS to EAwAC		Best	9	2	7	85	51	0.379	$\approx$
		Mean	11	0	7	98	73	0.586	$\approx$

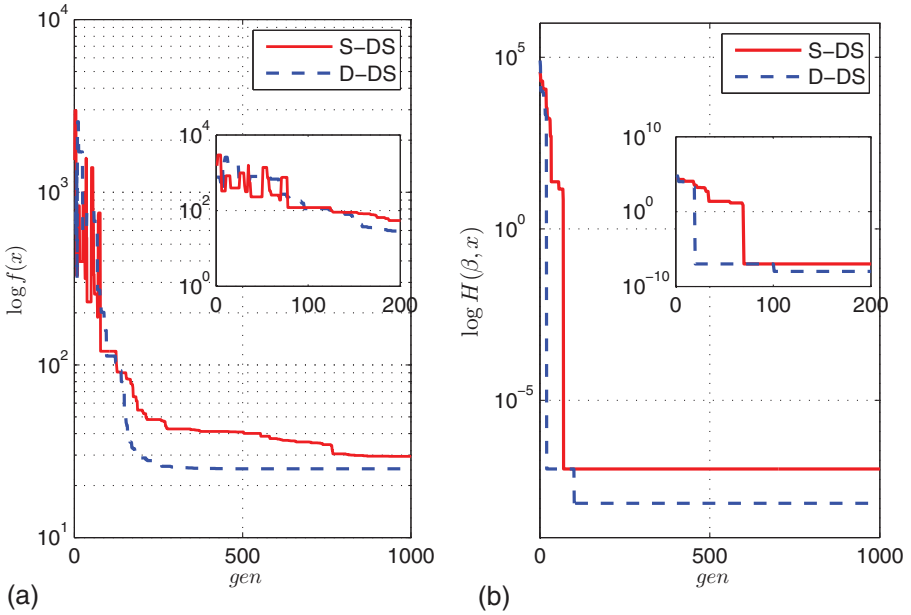

Fig. 4. Computational details of problem g07 by D-DS and S-DS: (a) $\log f(x)$ versus *gen*; (b) $\log H(\beta, x)$ versus *gen*.

terms of *gen* are depicted in Figs. 4 and 5, respectively. Here $\beta = 1$ and logarithm values are adopted for the clear presentation. Meanwhile, $\log H(\beta, x)$ is taken as the threshold value 10^{-7} if $\log H(\beta, x) \leq 10^{-7}$ in Figs. 4(b) and 5(b). From Figs. 4(b) and 5(b), we can observe that at the initial stage, D-DS has bigger constraint violations than S-DS. However, the decrease of the constraint violations by D-DS is much faster than that by S-DS. Only after a few generations (about 20 for g07 and about 30 for g19), the constraint violations decrease to under the threshold, i.e., $\log H(\beta, x) \leq 10^{-7}$. Furthermore, the decrease of the objective function $f(x)$ by D-DS is also faster than that by S-DS. Thus, the dynamic penalty method is much more efficient than static penalty method.

We also compare our dynamic penalty method with the dynamic penalty method in [28] in which the dynamic penalty factor is generated through a polynomial function rather than our exponential formula in (8). For the test problem g07, the numerical results obtained D-DS as well as the penalty method in [28] integrated DS algorithm is depicted in Fig. 6 in which $\log H(\beta, x)$ is taken as the threshold value 10^{-7} if $\log H(\beta, x) \leq 10^{-7}$. Fig. 6 clearly shows that our proposed dynamic penalty method can capture an optimal solution faster than that by the dynamic method in [28] while keeping smaller violations. The superiority of proposed dynamic method may own to the S-type exponential relationship between the dynamic penalty parameter $p(gen)$ and *gen*. However, the relationship between the dynamic penalty parameter $p(gen)$ and *gen* is polynomial in [28].

5. Applications D-DS to structure design optimization problems

In this section, four structure design problems are used to test the performance of D-DS on the solution of the complex real-world problems. During D-DS implementation process, the maximum iteration number is set as 5000 and the population size is set as 40. Thus, the evaluation number of the objective function is 20000. We assume that a solution is feasible if violation of

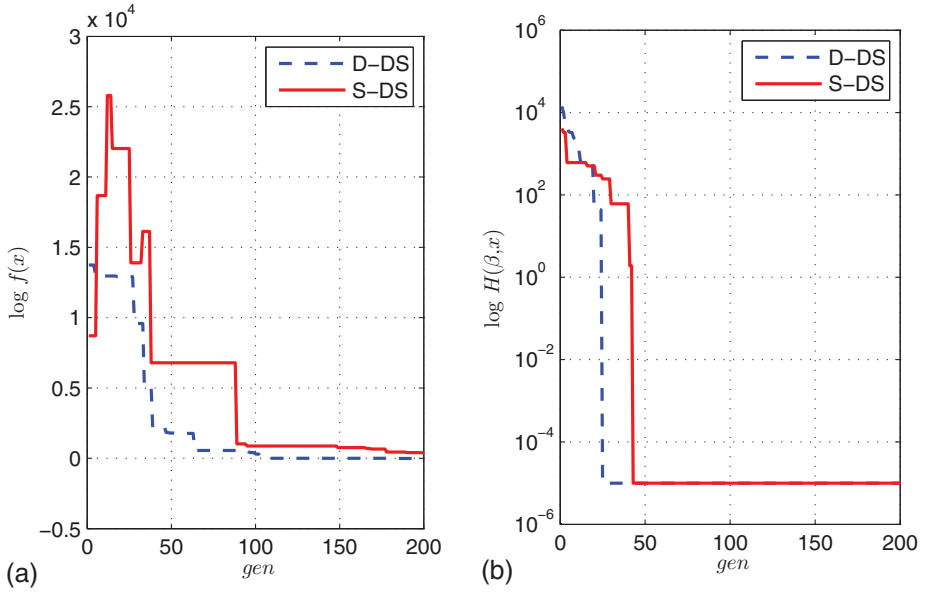

Fig. 5. Computational details of problem g19 by D-DS and S-DS: (a) $\log f(x)$ versus *gen*; (b) $\log H(\beta, x)$ versus *gen*.

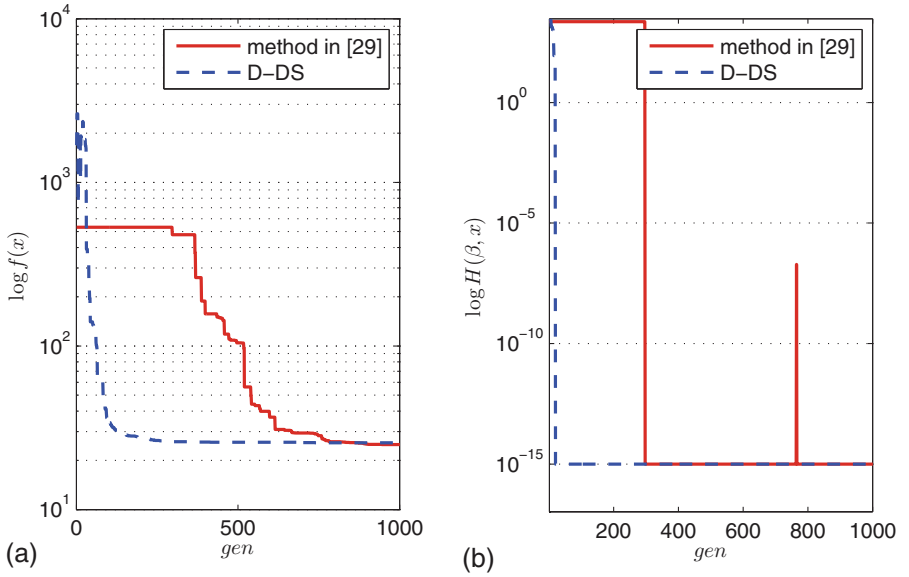

Fig. 6. Comparison of the performances of dynamic penalty results between D-DS and other method on problem g07:(a) $\log f(x)$ versus *gen*; (b) $\log H(\beta, x)$ versus *gen*.

any constraint is less than or equals to 10^{-5}. To observe the statistics of D-DS, we also do 25 independent trials for each design problem.

5.1. Tension/compression spring design optimization problem

For this problem, the aim is to minimize the weight of a tension/compression spring (see Fig. 7) subject to constraints on minimum deflection, shear stress, surge frequency and limits on outside diameter [49].

The design variables are the mean coil diameter x_1, the wire diameter x_2 and the number of active coils x_3. The problem is expressed as follows:

$$\min\ f(x) = (x_3 + 2) \cdot x_2 \cdot x_1^2$$

$$\text{s.t. } g_1(x) = 1 - \frac{x_2^3 x_3}{(71785 x_1^4} \leq 0$$

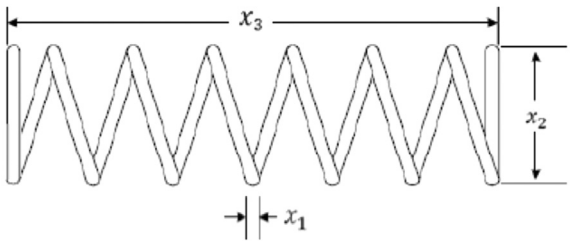

Fig. 7. Schematic of a tension/compression string.

Table 11
Statistics of the results obtained by our method as well as some other swarm based algorithms on the spring design problem.

	G-QPSO [22]	ABC [50]	SSO-C [9]	D-DS
Best	0.012665	0.01266523390012	0.01266523278831	0.012664908519200
Mean	0.013524	0.01285071830167	0.01276488817830	0.012669592082267
Worst	0.017759	0.01321040556169	0.01286791658161	0.012700649624332
Std.	0.001268	0.00011845110272	0.00009287497805	0.000010940514965
Fun. eval	20000	20000	25000	20000

Table 12
The solutions for the spring design problem obtained by our method as well as some other swarm-based algorithms.

	G-QPSO [22]	ABC [50]	SSO-C [9]	D-DS
x_1	0.051515	0.05168906749	0.0516890616572	0.051650228409694
x_2	0.352529	0.35671789406	0.3567177536211	0.355788669929791
x_3	11.538862	11.2889567081	11.2889649412891	11.343367592535893
g_1	−4.8341e-5	−1.65e-13	−2.000000e-15	9.999359540e-06
g_2	−3.5774e-5	−7.90e-14	−2.000000e-015	9.998254664e-06
g_3	−4.0455	−4.053399	−4.0537856576339	−4.052051349917477
g_4	−0.73064	−0.727864	−1.0915931847215	−1.092561101660515
f	0.012665	0.012665	0.0126652328	**0.012664908519200**

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \le 1$$
$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$
$$g_4(x) = x_1 + x_2 - 1.5 \le 0$$

The variables take their vales in their respective ranges: $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$, and $2.0 \le x_3 \le 15.0$.

In this problem, θ_1 and θ_2 are taken as $\theta_1 = 8$ and $\theta_2 = 12$ in Eq. (8). The statistics of 50 independent trials computed by D-DS as well as some other metaheuristic methods are listed in Table 11. The best solutions obtained by different methods are listed in Table 12. From Table 12, we can see that all methods have captured global solution approximately. From Table 11, we can observe that under the same number of function evaluations, our method is better than G-QPSO in [22] and ABC in [50] in terms of the average performance. However, it is worse than MAL-DE in [11]. In fact, it is not strange that MAL-DE in [11] achieves the best performance among them because it is involved with the most function evaluations. The objective function value $f(x)$ and constraint violation $H(\beta, x)$ at each generation obtained by D-DS is depicted in Fig. 8. Fig. 8 shows that only after a few generations, our method not only captures a feasible solution, but also obtains an approximate optimal solution successfully. However, the decrease of the objective function value $f(x)$ becomes flat at the latter stage.

5.2. Golinskis speed reducer

Golinskis speed reducer optimization problem is one of the 10 standard examples which are used by NASA for the multidisciplinary design optimization researchers to test the efficiency and effectiveness of their newly developed optimization methods [51,52]. A speed reducer is part of the gear box of a mechanical system, and it is also used for many other types of applications. This problem aims to design a simple gearbox that may be used in a light airplane between the engine and propeller so as to allow each to rotate at its most efficient speed.

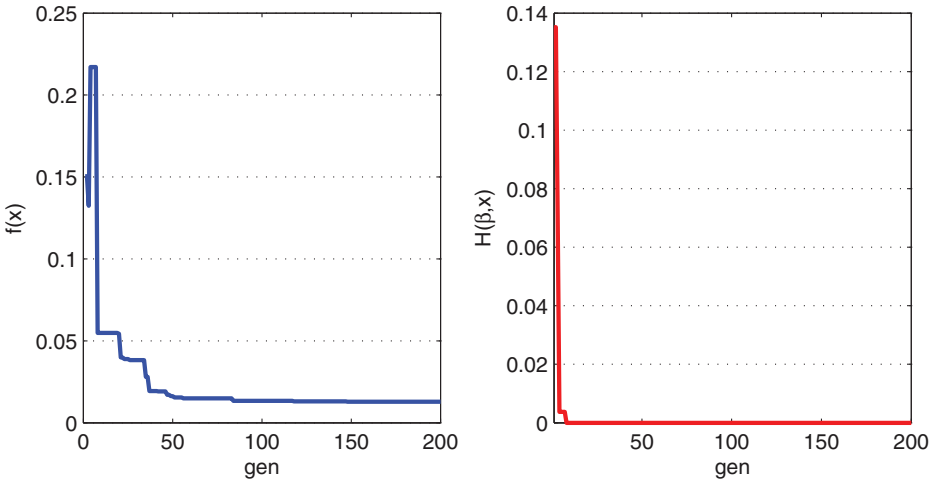

Fig. 8. Convergence curves and violation for the tension/compression spring design optimization problem.

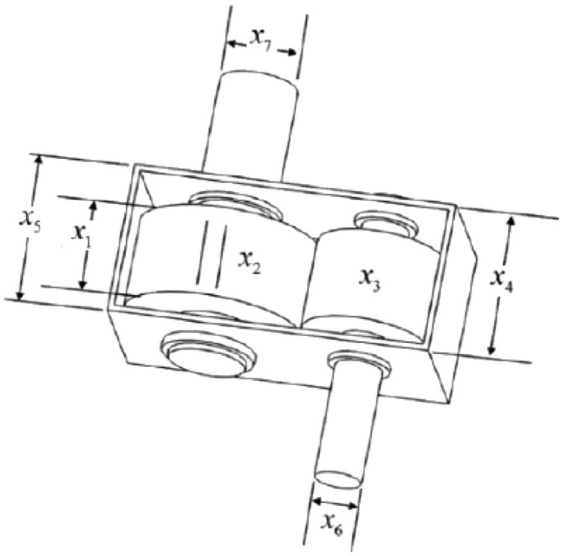

Fig. 9. Schematic of the speed reducer.

The gearbox is shown in Fig. 9 and its seven design variables are labeled below. The speed reducer is modeled so that its weight is minimized subject to constraints on bending stress of the gear teeth, traverse deflections of the shafts and stresses in the shaft.

There are seven variables which are used to describe the problem. They are: x_1, face width. x_2, module of teeth. x_3, number of teeth in the pinion. x_4, length of the first shaft between bearings. x_5, length of the second shaft between bearings. x_6, diameter of the first shaft. x_7, diameter of the second shaft.

$$\begin{aligned}
\min\ f(x) &= 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
&\quad - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\
&\quad + 0.7854(x_4x_6^2 + x_5x_7^2) \\
\text{s.t. } g_1(x) &= \frac{27}{x_1x_2^2x_3} \le 1 \\
g_2(x) &= \frac{397.5}{x_1x_2^2x_3^2} \le 1
\end{aligned}$$

Table 13
Statistical results of different approaches for the speed reducer design problem.

	SCA [57]	PSO-DE [12]	MAL-DE [11]	D-DS
Best	2994.744241	2996.348167	2994.471066	2994.444113
Mean	2994.474392	2996.348174	2994.471066	2994.444113
Worst	3009.964736	2996.348204	2994.471066	2994.444113
Std.	4.00000e+00	6.40000e-06	0.00000e+00	4.793457e-13
Fun. eval	54456	42100	120000	20000

Table 14
The solutions for the speed reducer design problem obtained by D-DS and some other swarm-based algorithms.

	SCA [57]	PSO-DE [12]	MAL-DE [11]	D-DS
x_1	3.5000681	3.500000	3.50000000	3.499965000349998
x_2	0.70000001	0.700000	0.70000000	0.700000000000000
x_3	17	17	17	17
x_4	7.32760205	7.300000	7.30000000	7.300000000000004
x_5	7.71532175	7.800000	7.71531991	7.715223339086296
x_6	3.35026702	3.350214	3.35021467	3.350203498788676
x_7	5.28665450	5.2866832	5.28665446	5.286636810290617
f	2994.744241	2996.348167	2994.471066	**2994.444113**

$$g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} \le 1$$
$$g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} \le 1$$
$$g_5(x) = \frac{\sqrt{(\frac{745x_4}{x_2x_3})^2 + 1.69 \times 10^7}}{110.0x_6^3} \le 1$$
$$g_6(x) = \frac{\sqrt{(\frac{745x_5}{x_2x_3})^2 + 1.575 \times 10^8}}{85.0x_7^3} \le 1$$
$$g_7(x) = \frac{x_2x_3}{40} \le 1$$
$$g_8(x) = \frac{5x_2}{x_1} \le 1$$
$$g_9(x) = \frac{x_1}{12x_2} \le 1$$
$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} \le 1$$
$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} \le 1$$

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

During the solution process, θ_1 and θ_2 are taken as $\theta_1 = 5$ and $\theta_2 = 8$ in Eq. (8). Numerical results obtained by D-DS as well as some other swarm-based optimization methods are listed in Tables 13 and 14. The convergence curve and constraint violations of D-DS are depicted in Fig. 10. Table 13 shows that the best solution is obtained by D-DS and MAL-DE in [11].

5.3. Pressure vessel design problem

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Fig. 5. This design problem is firstly proposed in [53]. The objective is to minimize the total cost of pressure vessel considering the cost of material, forming, and welding.

Through design four variables the pressure vessel design problem can be described as follow. R, inner radius. L, length of the cylindrical selection of the vessel. T_s, thickness of the shell. T_h, thickness of the head.

$$\min\ f(x) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R$$
$$\text{s.t. } g_1(x) = -T_s + 0.0193R \le 0$$
$$g_2(x) = -T_h + 0.00954R \le 0$$

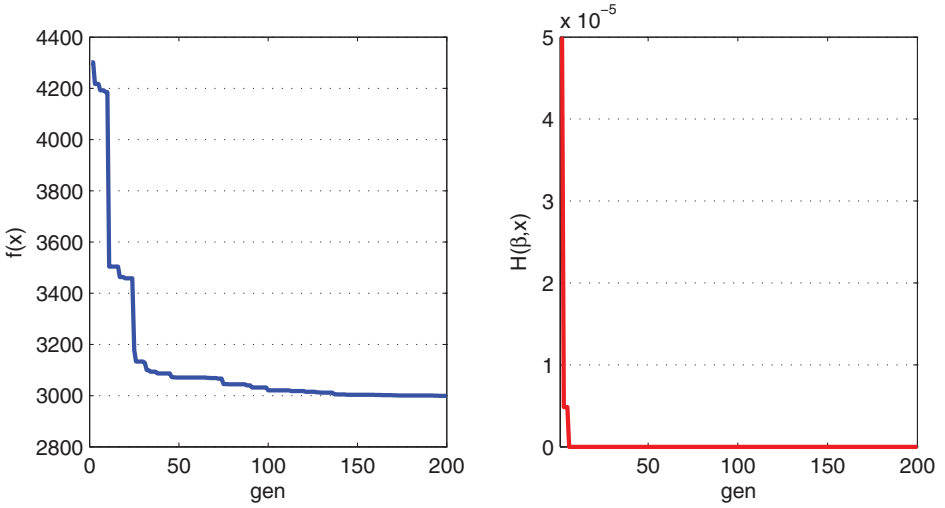

Fig. 10. Convergence curves and violation for the speed reducer design problem.

Table 15
Statistical results of different approaches for the pressure vessel design problem.

	FGA [54]	G-PSO [22]	MAL-DE [11]	D-DS
Best	6059.9643	6059.7208	6059.7143	6059.714335
Mean	6177.2533	6147.1332	6059.7143	6121.420389
Worst	6469.3220	6363.8041	6059.7143	6410.024261
Std.	130.9297	86.4500	1.35288e-12	23.81119795
Fun. eval	25000	8000	120000	20000

Table 16
Statistical results of different approaches for the pressure vessel design problem.

	FGA [54]	G-PSO [22]	MAL-DE [11]	D-DS
T_s	0.8125	0.8125	0.8125	0.81250000000
T_h	0.4375	0.4375	0.4375	0.43750000000
R	42.0974	42.0984	42.098445	42.0984455958549
L	176.6540	176.6372	176.636595	176.6365958424394
g_1	−2.01e-03	−8.7999e-07	−1.14e-08	−0.000000e-12
g_2	−3.58e-02	−3.5881e-02	−0.0358808	−0.0358808290155
g_3	−24.7593	−0.2179	−3.11e-05	−0.000000e-12
g_4	−63.3460	−63.3628	−63.363405	−63.36340405
f	6059.9463	6059.7208	6059.7143	**6059.7143350484**

$$g_3(x) = -\pi R^2 L - \frac{4}{3}\pi R^3 + 1296000 \leq 0$$
$$g_4(x) = L - 240 \leq 0$$

where T_s and T_h are discrete values which are integer multiples of 0.0625, and that $0.0625 \times 1 \leq T_s, T_h \leq 0.0625 \times 99$, $10 \leq R \leq 200$ and $10 \leq L \leq 200$.

For this problem, the parameters θ_1 and θ_2 are taking as $\theta_1 = 8$ and $\theta_2 = 12$. Then, we apply D-DS to solve it. The numerical results obtained by our method as well as those obtained by FGA in [54], G-PSO in [22] and MAL-DE in [11] are presented in Table 15. The corresponding solutions are listed in Table 16. The iteration details on the objective function $f(x)$ and the constraint violation $H(\beta, x)$ are depicted in Fig. 12. Table 15 shows that among the four methods, G-PSO in [22] achieves the best solution. Then, it follows D-DS and MAL-DE in [11]. The worst algorithm is FGA in [54]. However, G-PSO in [22] has larger mean and standard variations than that of D-DS and MAL-DE in [11]. D-DS achieves the much better solution than other algorithms. But MAL-DE in [11] has smaller mean and standard variations. However, the number of function evaluations by D-DS is only 1/6 that of MAL-DE in [11] (Fig. 11).

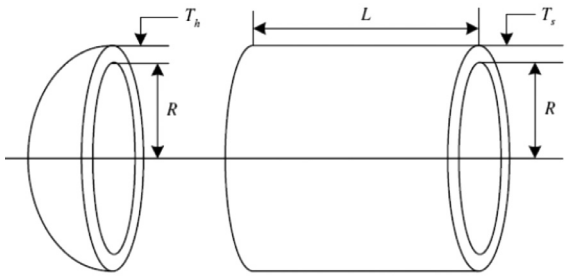

Fig. 11. Schematic of the pressure vessel.

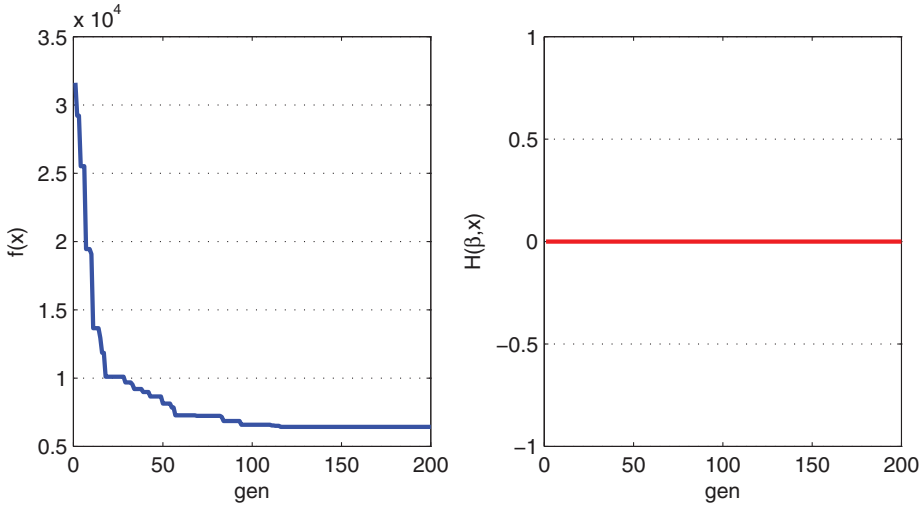

Fig. 12. Convergence curves and violation for the pressure vessel design problem.

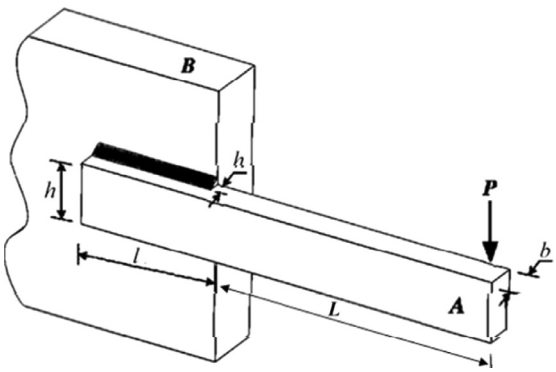

Fig. 13. A welded beam and its design features.

5.4. Welded beam design optimization problem

The welded beam structure, shown in Fig. 13, is a practical design problem. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress (τ), bending stress (σ), buckling load (p_c), end deflection (δ), and side constraint. There are four design variables, the thickness of the weld h, length of the weld l, width of the beam a, and the beam thickness b. The mathematical formulation for this problem is described as follows:

$$\begin{aligned} \min\ & f(x) = 1.10471h^2l + 0.04811ab(14.0 + l) \\ \text{s.t. } & g_1(x) = \tau - \tau_{max} \le 0 \\ & g_2(x) = \sigma - \sigma_{max} \le 0 \\ & g_3(x) = h - b \le 0 \end{aligned}$$

Table 17
Statistical results of different approaches for the welded beam design problem.

	CDE [24]	CPSO [55]	SSO-C [9]	ABC [50]	D-DS
Best	1.733461	1.84640843	1.72485230	1.6952638	1.69524716
Mean	1.768158	2.01114601	1.74646161	1.6953084	1.69524770
Worst	1.824105	2.23738868	1.79933176	1.6953706	1.69525173
Std.	0.022194	0.10851264	0.02572985	2.8362e-05	1.385962e-06
Fun. eval	204,800	200,000	25,000	20,000	20,000

Table 18
The solutions for the welded beam design problem obtained by our method as well as some other swarm-based algorithms.

	CDE [24]	CPSO [55]	SSO-C [9]	ABC [50]	D-DS
x_1	0.203137	0.202369	0.20572963	0.20572450	0.20572963
x_2	3.542998	3.544214	3.47048866	3.25325369	3.25312004
x_3	9.033498	9.048210	9.03662391	9.03664438	9.03662391
x_4	0.206179	0.205723	0.20572963	0.20572999	0.20572963
g_1	−842.628458	−811.41816628	−771.20213105	−0.17975428	−1.818989e-12
g_2	−44.6635338	−75.814077792	−0.0000000000	−0.18697948	−0.00000000
g_3	−0.00304200	−0.0033540000	−0.0000000000	−0.00000549	−2.775557e-17
g_4	−3.42372619	−3.4245721222	−3.4329837853	−3.45240767	−3.45242553
g_5	−0.23555638	−0.2355953327	−0.2355403225	−0.23554044	−0.23554032
g_6	−3802826803	−4.4728583202	−0.0000000000	−0.03957707	−3.637978e-12
f	1.73346218204	1.72802400000	1.72485230859	1.695263880	**1.695247165**

$$g_4(x) = 1.10471h^2 + 0.04811ab(14.0 + l) - 5 \le 0$$
$$g_5(x) = \delta - \delta_{max} \le 0$$
$$g_6(x) = P - P_c \le 0$$

where

$$M = P(L + l/2)$$
$$R = \sqrt{\frac{1}{4(l^2 + (h+a)^2)}}$$
$$J = 2\sqrt{2}hl\left(\frac{1}{4}(l^2 + (h+a)^2)\right)$$
$$\tau' = \frac{P}{\sqrt{2}hl}$$
$$\tau'' = \frac{MR}{J}$$
$$\tau = \sqrt{(\tau')^2 + tau'tau''\frac{l}{R} + (\tau'')^2}$$
$$P_c = \frac{4.013Eab^3}{6L^2}\left(1 - \frac{a}{2L}\sqrt{\frac{E}{4G}}\right)$$
$$\sigma = \frac{6PL}{ba^2}$$
$$\delta = \frac{4PL^3}{Ea^3b}$$

$P = 6000$ lb, $L = 14$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{max} = 13600$ psi, $\sigma_{max} = 30000$ psi, $\delta_{max} = 0.25$ in. The boundary conditions are $0.125 \le h \le 2$, $0.1 \le l, a \le 10$, and $0.1 \le b \le 2$.

For this problem, we compare D-DS with some swarm-based algorithms, including CDE, CPSO, SSO-C and ABC. The parameters θ_1 and θ_2 are taken as $\theta_1 = 5$ and $\theta_2 = 8$ in Eq. (8). The results for 25 runs are reported in Tables 17 and 18. The convergence curve and constraint violation of the best solution obtained by D-DS is depicted in Fig. 14. According to Table 17, we can clearly observe that D-DS achieves better performance in terms of all statics aspects. However, the number of function evaluations in D-DS is only 2000, which is smaller than that of CDE in [24], CPSO in [55] and SSO-C in [9].

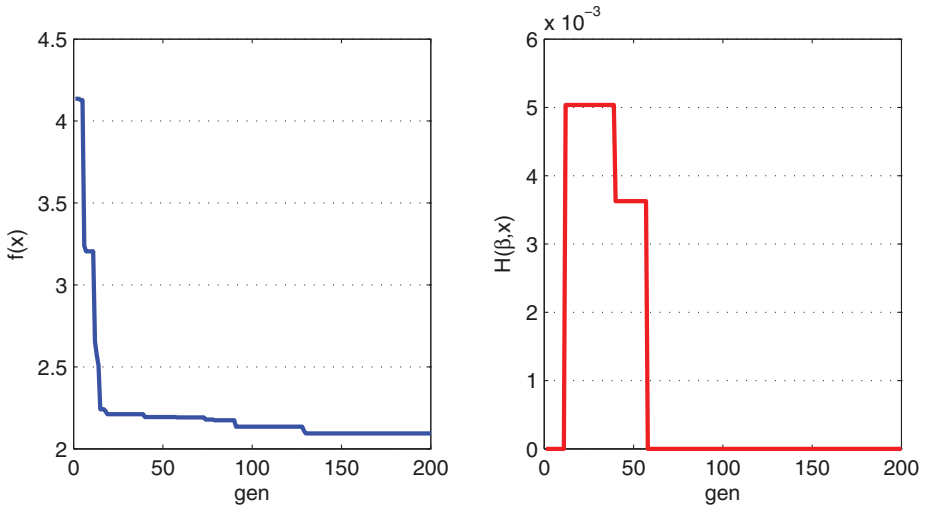

Fig. 14. Convergence curves and violation for the welded beam design problem.

6. Conclusion

A novel D-DS approach is proposed to solve constrained optimization problems, in particular for engineering design optimization problems. To solve the constrained optimization problems, the constraints are appended to the objective function through penalty method. Different from the most used static penalty methods, a dynamic penalty method is adopted in which the dynamic penalty parameter is generated through a S-type function. Then, DS algorithm is applied to solve the penalized unconstrained optimization problems.

In contrast to other population-based algorithms, D-DS proposed in the paper has more potential for the large-scaled constrained optimization problems involving multi-modal functions. This is because D-DS algorithm has no inclination to move correctly towards the best possible solution of the problem. To show the efficiency of the proposed algorithm D-DS, two benchmark sets, CEC2006 and CEC2010, as well as four structure design optimization problems are solved by D-DS and some other swarm-based algorithms. For these optimization problems, D-DS have successfully captured their optimal solutions for most of test problems except several problems such as g13 and C05, C11, et al. The numerical experiments show that D-DS can even achieve better performance through fewer function evaluations. Thus, it is more suitable for large-scaled constrained optimization problems than other similar swarm-based algorithms. However, we also observe that D-DS is sometimes suffered from slow convergence at later stage. Thus, the exploitation strategy in D-DS is still required to improve in future. Additionally, it is also required to compare our proposed penalty technique to handle constraints with those existing in further study.

Acknowledgment

The authors gratefully acknowledge the financial support from the National Natural Science Foundation of China (Grant nos. 11371371, 61473326), the Natural Science Foundation of Chongqing (cstc2013jjB00001 and cstc2013jcyjA00029) and the Foundation of China University of Petroleum (Grant no. KYJJ2012-06-03).

References

[1] S.S. Rao, S. Rao, Engineering Optimization: Theory and Practice, John Wiley & Sons, 2009.
[2] D. Karaboga, B. Basturk, Artificial Bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: Foundations of Fuzzy Logic and Soft Computing, Springer, 2007, pp. 789–798.
[3] D. Karaboga, B. Akay, A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, Appl. Soft Comput. 11 (3) (2011) 3021–3031.
[4] A.M. Deshpande, G.M. Phatnani, A.J. Kulkami, Constraint handling in Firefly algorithm, in: Proceedings of the IEEE International Conference on Cybernetics (CYBCONF), 2013, IEEE, 2013, pp. 186–190.
[5] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Mixed variable structural optimization using Firefly algorithm, Comput. Struct. 89 (23) (2011) 2325–2336.
[6] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method for constrained optimization problems, Intell. Technol. Theory Appl: New Trends Intell. Technol. 76 (2002) 214–220.
[7] G.T. Pulido, C.A.C. Coello, A constraint-handling mechanism for particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, 2004. CEC2004, 2, IEEE, 2004, pp. 1396–1403.
[8] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2) (2007) 1407–1422.
[9] E. Cuevas, M. Cienfuegos, A new algorithm inspired in the behavior of the Social-Spider for constrained optimization, Expert Syst. Appl. Int. J. 41 (2) (2014) 412–425.
[10] M. Ali, W. Zhu, A penalty function-based differential evolution algorithm for constrained global optimization, Comput. Optim. Appl. 54 (3) (2013) 707–739.
[11] W. Long, X. Liang, Y. Huang, Y. Chen, A hybrid differential evolution augmented Lagrangian method for constrained numerical and engineering optimization, Comput. Aided Des. 45 (12) (2013) 1562–1574.

[12] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Appl. Soft Comput. 10 (2) (2010) 629–640.
[13] S.M. Elsayed, R.A. Sarker, D.L. Essam, A self-adaptive combined strategies algorithm for constrained optimization using differential evolution, Appl. Math. Comput. 241 (0) (2014) 267–282.
[14] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Appl. Math. Comput. 219 (15) (2013) 8121–8144.
[15] P. Civicioglu, Artificial cooperative search algorithm for numerical optimization problems, Inf. Sci. 229 (2013) 58–76.
[16] Q. Long, C. Wu, T. Huang, X. Wang, A genetic algorithm for unconstrained multi-objective optimization, Swarm Evolut. Comput. 22 (2015) 1–14.
[17] V.V. De Melo, G. Iacca, A modified covariance matrix adaptation evolution strategy with adaptive penalty function and restart for constrained optimization, Expert Syst. Appl. 41 (16) (2014) 7077–7094.
[18] A.O. Kusakci, M. Can, An adaptive penalty based covariance matrix adaptation–evolution strategy, Comput. Oper. Res. 40 (10) (2013) 2398–2417.
[19] N. Hansen, Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, ACM, 2009, pp. 2389–2396.
[20] Q. Long, C. Wu, A hybrid method combining Genetic Algorithm and Hook-Jeeves method for constrained global optimization, J. Ind. Manag. Optim. 10 (4) (2014) 1279–1296.
[21] L.d. S. Coelho, V.C. Mariani, Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, Expert Syst. Appl. 34 (3) (2008) 1905–1913.
[22] L.d. S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2) (2010) 1676–1683.
[23] I. Mazhoud, K. Hadj-Hamou, J. Bigeon, P. Joyeux, Particle swarm optimization for solving engineering problems: a new constraint-handling mechanism, Eng. Appl. Artif. Intell. 26 (4) (2013) 1263–1273.
[24] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (1) (2007) 340–356.
[25] T.W. Liao, Two hybrid differential evolution algorithms for engineering design optimization, Appl. Soft Comput. 10 (4) (2010) 1188–1199.
[26] M.Z. Ali, N.H. Awad, A novel class of niche hybrid cultural algorithms for continuous engineering optimization, Inf. Sci. 267 (2014) 158–190.
[27] P. Civicioglu, Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm, Comput. Geosci. 46 (2012) 229–247.
[28] O. Yeniay, Penalty function methods for constrained optimization with genetic algorithms, Math. Comput. Appl. 10 (1) (2005) 45–56.
[29] M. Thakur, S.S. Meghwani, H. Jalota, A modified real coded genetic algorithm for constrained optimization, Appl. Math. Comput. 235 (0) (2014) 292–317.
[30] D. Goswami, S. Chakraborty, Differential search algorithm-based parametric optimization of electrochemical micromachining processes, Int. J. Ind. Eng. Comput. 5 (1) (2014) 41–54.
[31] J. Liu, K. Teo, X. Wang, C. Wu, An exact penalty function-based differential search algorithm for constrained global optimization, Soft Comput. (2015) 1–9, doi:10.1007/s00500-015-1588-6.
[32] P. Civicioglu, Circular antenna array design by using evolutionary search algorithms, Prog. Electromagn. Res. B 54 (2013) 265–284.
[33] V. Waghole, R. Tiwari, Optimization of needle roller bearing design using novel hybrid methods, Mech. Mach. Theory 72 (2014) 71–85.
[34] T. Kurban, P. Civicioglu, R. Kurban, E. Besdok, Comparison of evolutionary and swarm based computational techniques for multilevel color image thresholding, Appl. Soft Comput. 23 (2014) 128–143.
[35] X.-S. Yang, Swarm-based metaheuristic algorithms and no-free-lunch theorems, in: Dr. R. Parpinelli (Ed.), Theory and New Applications of Swarm Intelligence, InTech, 2012. Available from: http://www.intechopen.com/books/theory-and-new-applications-of-swarm-intelligence/swarmbased-metaheuristic-algorithms-and-no-free-lunch-theorems.
[36] X.-S. Yang, X. He, Firefly algorithm: recent advances and applications, Int. J. Swarm Intell. 1 (1) (2013) 36–50.
[37] J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C.C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, J. Appl. Mech. 41 (2006).
[38] R. Mallipeddi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization, Nanyang Technological University, Singapore, 2010.
[39] D. Powell, M.M. Skolnick, Using genetic algorithms in engineering design optimization with non-linear constraints, in: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., 1993, pp. 424–431.
[40] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Meth. Appl. Mech. Eng. 186 (2) (2000) 311–338.
[41] T. Takahama, S. Sakai, N. Iwane, Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm, in: AI 2005: Advances in Artificial Intelligence, Springer, 2005, pp. 389–400.
[42] E. Mezura Montes, C.A. Coello Coello, A simple multimembered evolution strategy to solve constrained optimization problems, Evolut. Comput. IEEE Trans. 9 (1) (2005) 1–17.
[43] E. Mezura-Montes, M.E. Miranda-Varela, R. del Carmen Gómez-Ramón, Differential evolution in constrained numerical optimization: an empirical study, Inf. Sci. 180 (22) (2010) 4223–4262.
[44] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[45] S.M. Elsayed, R.A. Sarker, D.L. Essam, On an evolutionary approach for constrained optimization problem solving, Appl. Soft Comput. 12 (10) (2012) 3208–3227.
[46] T. Takahama, S. Sakai, Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2010, IEEE, 2010, pp. 1–9.
[47] N.M. Hamza, R.A. Sarker, D.L. Essam, K. Deb, S.M. Elsayed, A constraint consensus memetic algorithm for solving constrained optimization problems, Eng. Optim. 46 (11) (2014) 1447–1464.
[48] T.-J. Hsieh, A bacterial gene recombination algorithm for solving constrained optimization problems, Appl. Math. Comput. 231 (2014) 187–204.
[49] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2) (2000) 113–127.
[50] H. Garg, Solving structural engineering design optimization problems using an artificial bee colony algorithm, J. Ind. Manag. Optimi. 10 (3) (2014) 777–794.
[51] T. Ray, Golinski's speed reducer problem revisited, AIAA J. 41 (3) (2003) 556–558.
[52] S. Tosserams, L.P. Etman, J. Rooda, An augmented Lagrangian decomposition method for quasi-separable problems in MDO, Struct. Multidiscip. Optim. 34 (3) (2007) 211–227.
[53] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (2) (1990) 223–229.
[54] C.A. Coello Coello, E. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inform. 16 (3) (2002) 193–203.
[55] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (1) (2007) 89–99.
[56] S.M. Elsayed, R.A. Sarker, D.L. Essam, Adaptive configuration of evolutionary algorithms for constrained optimization, Appl. Math. Comput. 222 (2013) 680–711.
[57] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, Evolut. Comput. IEEE Trans. 7 (4) (2003) 386–396.