Applied Mathematics & Information Sciences
An International Journal

A Hybrid Glowworm Swarm Optimization Algorithm for Constrained Engineering Design Problems

Yongquan Zhou[1,2], *Guo Zhou*[3] , *Junli Zhang*[1]

[1] College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning, Guangxi, 530006, People's Republic of China.
[2] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, Guangxi, 530006, People's Republic of China.
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100081, People's Republic of China.

Received: 28 Jan. 2012; Revised 5 Aug. 2012; Accepted 7 Oct. 2012
Published online: 1 Jan. 2013

Abstract: In this paper, a novel hybrid glowworm swarm optimization (HGSO) algorithm is proposed. Firstly, the presented algorithm embeds predatory behavior of artificial fish swarm algorithm (AFSA) into glowworm swarm optimization (GSO) algorithm and combines the improved GSO with differential evolution (DE) on the basis of a two-population co-evolution mechanism. Secondly, under the guidance of the feasibility rules, the swarm converges towards the feasible region quickly. In addition, to overcome premature convergence, the local search strategy based on simulated annealing (SA) is used and makes the search near the true optimum solution gradually. Finally, the HGSO algorithm is for solving constrained engineering design problems. The results show that HGSO algorithm has faster convergence speed, higher computational precision, and is more effective for solving constrained engineering design problems.

Keywords: Glowworm swarm optimization, differential evolution, feasibility rules, simulated annealing, hybrid optimization algorithm, engineering design problems.

1. Introduction

Generally speaking, in science calculation and engineering application field, many problems can be transformed into optimization problems, optimization can be divided into no constraint and constraints, and among them a constrained optimization problem can be described as follows:

$$\begin{aligned} & \max f(x) \\ s.t.\ & g_i(x) \le 0, i = 1, 2, \cdots, n. \\ & h_j(x) = 0, j = 1, 2 \cdots, p. \\ & a_k \le x_k \le b_k, k = 1, 2, \cdots, d. \end{aligned} \quad (1)$$

where $S = \{x \mid x \in R^d, a_k \le x_k \le b_k, k = 1, 2, \cdots, d\}$ denotes the search space, $x = [x_1, x_2, \cdots, x_d]^T$ denotes the decision solution vector, d is dimension of the decision variable, $F = \{x \mid x \in S, g_i(x) \le 0, h_j(x) = 0, i = 1, 2, \cdots, n, j = 1, 2, \cdots, p\}$ denotes the feasible region, obviously, $x \in F \subseteq S$. In fact, an equality constraint $h_j(x) = 0$ can be replaced by a couple of inequality constraint $h_j \le \delta$ and $h_j \ge -\delta$ (δ is a small tolerant amount). If we find $min\ f(x)$, we will transform it into solving $max\ g(x)$ by $g(x) = -f(x)$. Constrained optimization problem is that the objective function is made find the optimal solution in the feasible region.

For solving constrained optimization design problem, most of traditional algorithms are based on the concept of gradient, they request that the objective function and constraint conditions should be differentiable, and the obtained solution is mostly local optimal solution. Penalty function methods are simple, convenient and don't strictly require problem itself, but how to determine the suitable penalty factors is more difficult. In addition, in view of the deficiencies of the low accuracy and the poor stability for solving constrained optimization problems, this paper a hybrid glowworm swarm optimization (HGSO) algorithm is proposed. The proposed HGSO introduces preda-

* Corresponding author: e-mail: yongquanzhou@126.com

tory behavior of artificial fish swarm algorithm (AFSA) into glowworm swarm optimization (GSO) algorithm and combines the improved GSO with differential evolution (DE). In the evolutionary process, HGSO uses the constraint processing technology based on feasibility rules to update the optimal location of the population, which makes the population rapidly convergence to feasible regions and find better feasible solution. Besides, to avoid premature, HGSO adopts the local search strategy based on simulated annealing (SA) to optimize the local optimal value.

The rest of the paper is organized as follows: In Section 2, AFSA, GSO and DE are simply described. In Section 3, the HGSO hybrid strategy is proposed and explained in detail. Simulation and comparisons based on engineering design problems of HGSO are presented in Section 4 and in the end some conclusions in Section 5.

2. Basic Algorithms

2.1. Artificial Fish Swarm Algorithm (AFSA)

AFSA is a random search algorithm based on simulating fish swarm behaviors [1]. Assume that X_i is the position of artificial fish (AF) $i, y = f(X)$ is the fitness value at position $X, d_{ij} = \|X_i - X_j\|$ represents the distance between the AF i and j, $Visual$ and δ represent the visual distance and crowd factor of the AF respectively, nf is the number of its fellows within the visual,$step$is the step of the AF moving, $S = \{X_j \mid \|X_i - X_j\| < Visual\}$ is the set of AF i exploring area at the present position. The typical behaviors of the AF are expressed as follows:

(1) **AF-Prey:** Suppose that X_i is the AF state at present $X_j(X_j \in S)$ is the state of AF attempt within the visual, $trynumber$ is the maximum number of AF attempts. The behavior of prey can be expressed as follows:

$$prey(X_i) = \begin{cases} X_i + step\dfrac{X_j - X_i}{\|X_j - X_i\|} & if\ y_j > y_i \\ X_i + (2rand - 1) \cdot step & else \end{cases} \tag{2}$$

where *rand* is random function.

(2) **AF-Swarm:** Suppose that X_i is the AF state at present, and $X_c = \sum_{X_i \in S} X_i/nf$ is the center position of the AF within the visual. The behavior of swarm can be described as follows:

$$swam(X_i) = \begin{cases} X_i + step\dfrac{X_c - X_i}{\|X_c - X_i\|} & if\ \dfrac{y_c}{nf} > \delta y_i \\ prey(X_i) & else \end{cases} \tag{3}$$

(3) **AF-Follow:** Suppose that X_i is the AF state at present, and $y_{max} = max\{f(X_j) \mid X_j \in S\}$. The behavior of follow can be expressed in the following equation:

$$follow(X_i) = \begin{cases} X_i + step\dfrac{X_{max} - X_i}{\|X_{max} - X_i\|} & if\ \dfrac{y_{max}}{nf} > \delta y_i \\ prey(X_i) & else \end{cases} \tag{4}$$

2.2. Glowworm Swarm Optimization (GSO)

Glowworm swarm optimization (GSO) proposed by Krishnanand K. N and Ghose D. in 2005[2] [3]. Each iteration of GSO is consists of a luciferin-update phase followed by a movement phase based on a transition rule.

Luciferin-update phase: The luciferin update depends on the function value at the glowworm position. During the luciferin-update phase, each glowworm adds, to its previous luciferin level, a luciferin quantity proportional to the fitness of its current location in the objective function domain. Also, a fraction of the luciferin value is subtracted to simulate the decay in luciferin with time. The luciferin update rule is given by:

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma f(x_i(t+1)) \tag{5}$$

where $l_i(t)$ represents the luciferin level associated with glowworm i at time t, ρ is the luciferin decay constant $(0 \leq \rho \leq 1)$, γ is the luciferin enhancement constant, and $f(x_i(t))$ represents the value of the objective function at agent $i's$ location at time t.

Movement phase: During the movement phase, each glowworm decides, using a probabilistic mechanism, to move toward a neighbor that has a luciferin value higher than its own. That is, glowworms are attracted to neighbors that glow brighter. The set of neighbors of glowworm i at time t is calculated as follows:

$$N_i(t) = \{j : \|x_j(t) - x_i(t)\| < r_d^i(t);\ l_i(t) < l_j(t)\} \tag{6}$$

where the $\|\mathbf{x}\|$ is the Euclidean norm of $\mathbf{x}$, and r_d^i represents the variable neighborhood range associated with glowworm i at time t, which is bounded above by a circular sensor range $r_s(0 < r_d^i(t) < r_s)$. For each glowworm i, the probability of moving toward a neighbor $j \in N_i(t)$ is given by:

$$P_{ij} = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \tag{7}$$

Let glowworm i select a glowworm $j \in N_i(t)$ with $P_{ij}(t)$ given in (7). Then, the discrete-time model of the glowworm movements can be stated as:

$$x_i(t+1) = x_i(t) + s\Big(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\Big) \tag{8}$$

where $x_i(t) \in R^d$ is the location of glowworm i, at time t, in the d-dimensional real space R^d, and $s(> 0)$ is the step size.

Neighborhood range update rule: We associate each agent i with a neighborhood whose radial range $r_d^i(t)$ is dynamic in nature. Let r_0 be the initial neighborhood range of each glowworm (that is, $r_d^i(0) = r_0, \forall i$). To adaptively update the neighborhood range of each glowworm, the rule as follows:

$$r_d^i(t+1) = min\{r_s, max\{0, r_d^i(t) + \beta(n_i - |N_i(t)|)\}\} \tag{9}$$

where β is a constant parameter and n_t is a parameter used to control the number of neighbors.

2.3. Differential Evolution (DE)

DE is proposed by Storn and K. Price[4] for solving Chebyshev polynomial, it is a heuristic random search algorithm based on population differences, generates new individual through mutation and crossover, and retains excellent individuals according to survival of the fittest. Suppose that N_p denotes population size, $x_i(k) = [x_i^1(k), x_i^2(k), \cdots, x_i^d(k)]$ denotes the position of the i-th individual at the k-th iteration. The procedure of DE is summarized as follows:

Step 1 *Initialization*. Randomly initialize the positions $x_i(k)(i = 1, 2, \cdots, N_p)$ of N_p individuals in the search space, and let the number of iterations $k = 0$.

Step 2 *Mutation*. According to the following equation in (10), DE achieves individual variation through the difference strategy.

$$v_i(k+1) = x_{r_1}(k) + F \times (x_{r_2}(k) - x_{r_3}(k)) \tag{10}$$

where F denotes scaling factor, $r_1, r_2, r_3 \in \{1, 2, \cdots, N_p\}$ are three random numbers and $i \neq r_1 \neq r_2 \neq r_3$. If $\nu_i^j(k+1) < a_j$, then let $\nu_i^j(k+1) = a_j$; If $\nu_i^j(k+1) > b_j$, then let $\nu_i^j(k+1) = b_j, j = 1, \cdots, d$.

Step 3 *Crossover*. According to the following equation (11), implement crossover operation for the population $\{x_i(k)\}$ at the k-th iteration and variable intermediate $v_i(k+1)$:

$$u_i^j(k+1) = \begin{cases} v_i^j(k+1) & if\ randb(j) \leq CR\ or\ j = randr(i), \\ x_i^j(k) & otherwise \end{cases} \tag{11}$$

where $randb(j) \in [0, 1]$ denotes the j-th value generated by the same random generator, $CR \in [0, 1]$ denotes mutation rate, $randr(i) \in [1, 2, \cdots, d]$ is a random selection index, which ensures that $u_i(k+1)$ can get at least a parameter from $v_i(k+1)$.

Step 4 *Selection*. According to the following equation (12), DE selects the individuals into the next generation population by greed strategy:

$$x_i(k+1) = \begin{cases} u_i(k+1)\ if\ f(u_i(k+1)) \geq f(x_i(k)), \\ x_i^j(k)\ otherwise \end{cases} \tag{12}$$

Step 5 If the maximum number of iterations is met, then calculate the fitness value $f(x_i(k+1))(i = 1, \cdots, N_p)$ of each individual, then stop and output the optimal position and the optimal value of the population; otherwise, let $k = k + 1$, return **Step 2**.

3. Hybrid Glowworm Swarm Optimization Strategies (HGSO)

3.1. Improved GSO (IGSO) Based on Predatory Behavior of AFSA

In the basic GSO algorithm, each glowworm only in accordance with luciferin values of glowworms in its neighbor set, selects the glowworm by a certain probability and moves towards it. However, if the search space of a problem is very large or irregular, the neighbor sets of some glowworms may be empty, which leads these glowworms to keep still in iterative process. To avoid this case and ensure that each glowworm keeps moving, we will introduce predatory behavior of AFSA into GSO and propose an improved GSO (IGSO) algorithm. The idea of IGSO is as follows: the glowworms whose neighbor sets are empty are carried out predatory behavior in their dynamic decision domains. Assume that N represents population size, $x_i(t) = [x_i^{(1)}(t), x_i^{(2)}(t), \cdots, x_i^{(d)}(t)]$ denotes the position of the i-th glowworm at the t-th iteration. The procedure of IGSO can be described as follows:

Step 1 let $l_i(0) = l_0$, $r_d^i(t) = r_0$, $t = 0$, here, t denotes the number of GSO iterations. Randomly initialize the position $x_i(t)(i = 1, 2, \cdots, N)$ of each glowworm in the search space. Calculate the fitness value $f(x_i)$ of each glowworm. Initialize the current optimal position x^* and the current optimal value f_x^* according to the fitness values.

Step 2 Update the luciferin value $l_i(t)$ of each glowworm according to (5).

Step 3 Calculate $N_i(t)$ and $P_{ij}(t)$ for each glowworm according to (6) and (7).

Step 4 For each glowworm, if $N_i(t)$ is not empty, then according to $P_{ij}(t)$ and roulette method, select the j-th glowworm in $N_i(t)$ and move toward it, calculate $x_i(t+1)$ according to (8), Or else, implement predatory behavior in $r_d^i(t)$ and get $x_i(t+1)$. If $x_i^j(t+1) < a_j$, then $x_i^j(t+1) = a_j$; If $x_i^j(t+1) > b_j$, then $x_i^j(t+1) = b_j$, where $j = 1, 2, \cdots, d$.

Step 5 Calculate the current fitness value $f(x_i(t))$ of each glowworm, if the optimal position and optimal value of the current population are better than x^* and f_x^*, then update x^* and f_x^*, or else, don't update.

Step 6 If the maximum number of iterations is met, then stop and output x^* and f_x^*; or else, calculate $r_d^i(t+1)$ according to (9) and let $t = t + 1$, return **Step 2**.

3.2. The Feasibility Rules

The penalty function methods are a kind of constraint processing technology that is the most commonly used, it achieves balances between the objective function and constraints by adjusting the penalty factors, but how to select the suitable penalty factors is one difficulty of using penalty function methods. However, the constraint processing technology based on the feasibility rules will separate constraint conditions and objective function; it brings no additional parameters and is implemented easily. The rules are described as follows[5]: Assume that x_i and x_j denote the positions of the i-th individual and j-th individual respectively, if any one happens in the following cases, we will rule x_i is better than x_j: (1) x_j is infeasible, but x_i is feasible; (2) Both x_i and x_j are feasible, but $f(x_i) > f(x_j)$; (3) Both x_i and x_j are infeasible, but $viol(x_i) < viol(x_j)$. In the first and the third cases, the search tends to the feasible region rather than infeasible region, and in the second case, the search tends to the feasible region with better solution. In this paper, the constraint violation value of an infeasible solution is calculated as follows:

$$viol(x) = \sum_{i=1}^{N} max[g_i(x), 0], \tag{13}$$

where it is supposed that all equality constraints have already been transformed into inequality constraints.

3.3. The Local Search Based on SA

Simulated annealing (SA) [5] [6] is a stochastic searching algorithm with jumping property, which can make the search avoid falling into the local optimum. In the search process, SA accepts a better solution with probability 1, but also accepts a worse solution with a certain probability. Such a probabilistic jumping property can be controlled by adjusting the temperature, that is to say, the probability decreases as the temperature decreases. When the temperature tends to zero the probability will also approach to zero. It has been theoretically proved that under certain conditions SA is globally convergent in probability 1. Assume that p_g^t denotes the optimal position of the population at the t-th generation, to avoid premature convergence, we adopt the local search strategy on the basis of the feasibility rules and SA for p_g^t. Its process is described as follows:

Step 1 Let $m = 1, p_g^{'} = p_g^t$.

Step 2 Generate a new solution according to (14):

$$x^{'} = p_g^{'} + \eta \times (X_{max} - X_{min}) \times N(0,1) \tag{14}$$

where, η denotes the step size of the search, $N(0,1)$ denotes a random number normally distributed with mean 0 and variance 1, X_{min} and X_{max} denote the upper and lower bounds of the solutions defined by the problem.

Step 3 According to the following criteria computation p_a:

3.1. If $x^{'}$ is feasible and $p_g^{'}$ is infeasible, let $p_a = 1$.

3.2. If $x^{'}$ is infeasible and $p_g^{'}$ is feasible, let $p_a = 0$.

3.3. If both $x^{'}$ and $p_g^{'}$ are feasible, let $p_a = min\{1, exp[(f(x^{'}) - f(p_g^{'}))/T(k)]\}$.

3.4. If both $X^{'}$ and $p_g^{'}$ are infeasible, let $p_a = min\{1, exp[(viol(p_g^{'}) - viol(x^{'}))/T(k)]\}$.

where $T(k)$ denotes the temperature at the k-th generation.

Step 4 If $p_a \geq U(0,1)$, then $p_g^{'} = x^{'}$, where $U(0,1)$ represents a random number uniformly distributed in the range of $[0,1]$.

Step 5 Let $m = m+1$. If $m > 1$, stop and output $p_g^{'}$ as the new optimal position of the population, where L is a user-defined maximum number of iterations; else go to **Step 2**.

3.4. IGSO-DE Strategy

In this section, a new algorithm called IGSO-DE based on IGSO and DE is introduced. IGSO-DE is a two-group co-evolution algorithm, whose principle is described as follows: in the search space, the entire population is divided equally into two groups randomly, one group evolves according to IGSO, and the other group evolves according to DE. After the end of each generation, based on an optimal information sharing mechanism, that is, if the current optimal solution of IGSO is better than that of DE, then the current optimal solution of DE is updated by that of IGSO, which guides DE group towards the direction of the optimal solution, or else, the current optimal solution of IGSO is updated by that of DE, which guides IGSO group towards the direction of the optimal solution. As a consequence, two groups can obtain information not only from their own group but also from another group in evolution, which can make two groups achieve co-evolution and avoid premature.

3.5. HGSO Algorithm

The proposed HGSO takes IGSO-DE as the basic framework, updates the optimum position of the population using updating strategy on the basis of the feasibility rules in the search process, and adopts the local search strategy based on SA for the optimal position of each generation. In addition, in this paper, the initial temperature is determined by the following empirical formula:

$$T(0) = -\frac{f_{max} - f_{min}}{ln(0,1)} \tag{15}$$

where f_{max} and f_{min} are the maximum and minimum objective values of the solutions in the initial swarm respectively. Besides, the exponential annealing, i.e. $T(k+1) = \lambda T(k)$, is used, where the annealing rate satisfies $0 < \lambda < 1$. The procedure of HGSO can be described as follows:

Step 1 Let $k = 0$, here, k denotes the mark of HGSO iteration. Randomly initialize the positions of N individuals in the search space, and calculate initial temperature $T(k)$ according to (15), initialize the optimal position X^* and the optimal value $f(X^*)$ of the population according to the feasibility rules.

Step 2 The entire population is divided equally into two swarms at random: the glowworm $swam_1$ and the differential evolution $swam_2$.

Step 3 According to Section 3.1, implement IGSO for the glowworm $swam_1$, then according to the feasibility rules, determine the current optimal position $X^k_{GSO-best}$ of the glowworm $swam_1$, apply the local search strategy based on SA to $X^k_{GSO-best}$ and get the new position $X^k_{GSO-newbest}$.

Step 4 According to Section 2.3, implement DE for the differential evolution $swam_2$, and according to the feasibility rules, determine the current optimal position $X^k_{DE-best}$ of the differential evolution $swam_2$, apply the local search strategy based on SA to $X^k_{DE-best}$ and get the new position $X^k_{DE-newbest}$.

Step 5 According to the feasibility rules, if $X^k_{GSO-newbest}$ is better than $X^k_{DE-newbest}$, and then $X^k_{GSO-newbest}$ updates $X^k_{DE-newbest}$, otherwise,

$X^k_{DE-newbest}$ updates $X^k_{GSO-newbest}$. In addition, according to the feasibility rules, update the optimal position X^* and the optimal value $f(X^*)$ of the entire population.

Step 6 If the maximum number of iterations is met, then stop and output the optimal position X^* and the optimal value $f(X^*)$ of the entire population; or else, let $T(K+1) = \lambda T(k)$, $k = k+1$, go back to **Step 3**.

4. Simulation Experiments

4.1. Engineering Optimization Problems

To verify the reliability and validity of HGSO, the following five typical engineering constrained design problems are used to test the performance of HGSO.

4.2. Experimental Environment

The HGSO are coded in MATLAB R2009a and implemented on 2.00GHz CPU machine with 1.92GB RAM under Windows XP platform. The parameters of the HGSO are set as follows: population size $N = 250$, the luciferin decay constant $\rho = 0.4$, the luciferin enhancement constant $\gamma = 0.6$, the rate of change of the neighbourhood range $\beta = 0.08$, the neighbourhood threshold $n_t = 5$, step-size of the movement $s = 0.03$, the initial luciferin value $l_0 = 5$, the maximum number of attempts of glowworms in predatory behavior $trynumber = 15$, scaling factor $F = 0.4$, mutation probability $CR = 0.9$, annealing rate $\lambda = 09415$, step size of the local search $\eta = 0.00315$, the number of iterations of the local search at each generation $L = 20$.

Example 1. A welded beam design problem

This problem is taken from [12], in which a welded beam is designed for minimum cost ($f(x)$) subject to constraints on shear stress (τ); bending stress in the beam (θ); buckling load on the bar (P_c); end deflection of the beam (δ); and side constraints. There are four design variables as shown in Figure 1, i.e. $h(x_1), l(x_2)$,$t(x_3)$ and $b(x_3)$ The problem can be mathematically formulated as follows:

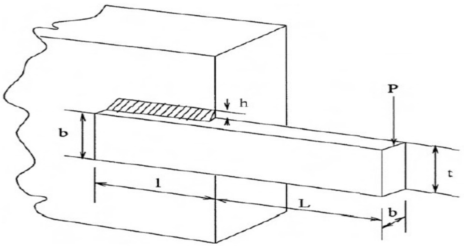

Figure 1: The welded beam design problem

$$
\begin{aligned}
& \min f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \\
s.t.\ & g_1(x) = \tau(x) - 13600 \le 0 \\
& g_2(x) = \sigma(x) - 30000 \le 0 \\
& g_3(x) = x_1 - x_4 \le 0 \\
& g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) \\
& \qquad - 5.0 \le 0 \\
& g_5(x) = 0.125 - x_1 \le 0 \\
& g_6(x) = \delta(x) - 0.25 \le 0 \\
& g_7(x) = 6000 - P_c(x) \le 0 \\
& 0.1 \le x_1, x_4 \le 2; 0.1 \le x_2, x_3 \le 10
\end{aligned}
$$

where

$$
\begin{aligned}
& \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''x_2/(2R) + (\tau'')^2}, \\
& \tau^{'} = 6000/(\sqrt{2}x_1x_2), \\
& \tau^{''} = MR/J, \\
& M = 6000(14 + x_2/2), \\
& R = \sqrt{[x_2^2 + (x_1 + x_3)^2]/4}, \\
& J = 2\{\sqrt{2}x_1x_2[x_2^2/12 + (x_1 + x_3)^2/4]\}, \\
& \sigma(x) = 504000/(x_4x_3^2), \delta(x) = 2.1952/(x_3^3x_4), \\
& P_c(x) = 4.013 \cdot E \cdot (1 - 0.0282346x_3) \cdot x_3x_4^3/(6L^2), \\
& L = 14, E = 30 \times 10^6.
\end{aligned}
$$

The circular sensor range r_s and the initial dynamic decision domain r_0 of the glowworms are all set to 5 for this problem, HGSO-1 represents the maximum number of generations is set to $T_{max} = 300$, which is in accordance with those of HPSO [5]and DSS-MDE [11], HGSO-2 denotes the maximum number of generations is set to $T_{max} = 400$, the other parameters are set to the same as those of Section 4.2. Simulation results and comparisons are shown in Table 1 and Table 2.

Table 1 Comparison of the best solution for Example 1 by different methods

Methods	$x(h)_1$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$f(x)$
Deb[13]	0.2455	6.1960	8.2730	0.2422	2.385937
Ray, Liew[14]	0.2444382760	6.2379672340	8.2885761430	0.2445661820	2.3854347
FSA[7]	0.24435257	6.2157922	8.2939046	0.24435258	2.381065
DSS-MDE[11]	0.2443689758	6.2175197152	8.2914713905	0.2443689758	2.38095658
Coello[15]	0.2088	3.4205	8.9975	0.21	1.748309
Coello, Montes[16]	0.2060	3.4713	9.0202	0.2065	1.728226
CPSO[8]	0.204381	3.505107	9.033546	0.205878	1.728024
Coello, Becerra[17]	0.205700	3.470500	9.036600	0.205700	1.724852
HPSO[5]	0.205730	3.470489	9.036624	0.205730	1.724852
CPSOSA[18]	0.20572961513-4341	3.47048900451-6055	9.03662445443-3026	0.205729619072-119	1.724852
PSO-DE[10]	0.205729640	3.470488666	9.036623910	0.205729640	1.724852309
HGSO-1	**0.205729619-261986**	**3.47048893-3856535**	**9.036624361-111119**	**0.20572961-9262203**	**1.724852234-810284**
HGSO-2	**0.205729619-262158**	**3.470488933-849532**	**9.036624361-111766**	**0.205729619-262158**	**1.724852234-809372**

Table 2 Statistical results of different methods for Example 1

Methods	Best	Mean	Worst	Std. Dev.
Deb[13]	2.385937	N/A	N/A	N/A
Ray, Liew[14]	2.3854347	3.0025883	6.3996785	0.959078
FSA[7]	2.381065	2.404166	2.488967	N/A
DSS-MDE[11]	2.38095658	2.38095658	2.38095658	3.19E-10
Coello[15]	1.748309	1.771973	1.785835	0.011220
Coello, Montes[16]	1.728226	1.792654	1.993408	N/A
CPSO[8]	1.728024	1.748831	1.782143	0.012926
Coello, Becerra[17]	1.724852	1.971809	3.179709	0.443131
HPSO[5]	1.724852	1.749040	1.814295	0.040049
CPSOSA[18]	1.724852	1.724853	1.724861	1.705146210-873380e-005
PSO-DE[10]	1.724852309	1.724852309	1.724852309	**6.7e-016**
HGSO-1	**1.72485223481-0284**	**1.72485223481-5348**	**1.72485223482-5736**	**3.599302351912-389e-012**
HGSO-2	**1.72485223480-9372**	**1.72485223480-9373**	**1.72485223480-9374**	**8.853024971371-397e-016**

From Table 1, it is observed that the best feasible solution obtained by HGSO is competitive to the results obtained in [5], [17] and [18], but better than the results obtained by other methods. From Table 2, it can be seen that the average searching quality of HGSO is greatly superior to those of the other methods. In addition, even the worst solution obtained by HGSO is better than the best solutions reported in [7], [8], [10], [11], [13], [14], [15] and [16], in 30 independent runs, the standard deviation of the results by HGSO-1 is very small and its precision reaches 10^{-12}, which is slightly worst than that of PSO-DE [10] but better than those of the other methods, however, the maximum number of generations by PSO-DE [10] is unknown. In addition, the standard deviation of the results by HGSO-2 is already competitive to that of PSO-DE [10], its precision reaches 10^{-16}, which shows that the robustness of HGSO is the best to solve this problem.

Example 2. A tension/compression string design problem

This problem is described in [19], and the aim is to minimize the weight ($f(x)$) of a tension/compression spring (as shown in Figure 2) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter $D(x_2)$, the wire diameter $d(x_1)$ and the number of active coils $P(x_3)$.

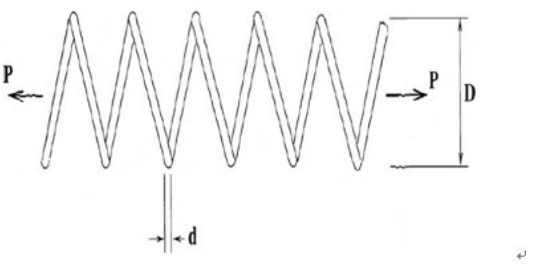

Figure 2: The tension/compression string design problem

The problem can be mathematically formulated as follows:

$$\begin{aligned} \min f(x) &= (x_3+2)x_2x_1^2 \\ s.t. g_1(x) &= 1 - x_2^3x_3/(71785x_1^4) \le 0; \\ g_2(x) &= (4x_2^2 - x_1x_2)/[12566(x_2x_1^3 - x_1^4)] \\ &\quad +1/(5108x_1^2 - 1 \le 0 \\ g_3(x) &= 1 - 140.45x_1/(x_2^2x_3) \le 0; \\ g_4(x) &= (x_1+x_2)/1.5 - 1 \le 0; \\ &0.05 \le x_1 \le 2; 0.25 \le x_2 \le 1.3; 2 \le x_3 \le 15. \end{aligned}$$

The circular sensor range r_s and the initial dynamic decision domain r_0 of the glowworms are all set to 7 for this problem, the maximum number of generations is set to $T_{max} = 300$, which is consistent with that of HPSO[5], the other parameters are set to the same as those of Section 4.2. Simulation results and comparisons are listed in Table 3 and Table 4.

Table 3 Comparison of the best solution for Example 2 by different methods

Methods	$x_1(d)$	$x_2(D)$	$x_3(P)$	$f(x)$
Belegundu[20]	0.050000	0.315900	14.250000	0.0128334
Arora[19]	0.053396	0.399180	9.185400	0.0127303
Coello[15]	0.051480	0.351661	11.632201	0.0127048
Coello,Montes[16]	0.051989	0.363965	10.890522	0.0126810
Coello,Becerra[17]	0.050000	0.317395	14.031795	0.0127210
CPSO[8]	0.051728	0.357644	11.244543	0.0126747
Ray, Liew[14]	0.0521602170	0.368158695	10.6484422590	0.01266924934
HPSO[5]	0.051706	0.357126	11.265083	**0.0126652**
FSA[7]	0.0517425034-0926	0.35800478-345599	11.213907362-78739	0.012665258
DSS-MDE[11]	0.0516890614	0.3567177469	11.2889653382	0.012665233
CPSOSA[18]	0.0516537117-70636	0.3558679160-29741	11.338963731-041684	0.01266525
PSO-DE[10]	0.0516888101	0.3567117001	11.289319935	0.012665233
HGSO	**0.051689060-896103**	**0.356717735-308878**	**11.288966014-881048**	**0.012665232-788319**

Table 4 Statistical results of different methods for Example 2

Methods	Best	Mean	Worst	Std. Dev.
Belegundu[20]	0.0128334	N/A	N/A	N/A
Arora[19]	0.0127303	N/A	N/A	N/A
Coello[15]	0.0127048	0.0127690	0.012822	3.9390e-005
Coello,Montes[16]	0.0126810	0.0127420	0.012973	5.9000e-005
Coello,Becerra[17]	0.0127210	0.0135681	0.015116	8.4152e-004
CPSO[8]	0.0126747	0.0127300	0.012924	5.1985e-004
Ray,Liew[14]	0.01266924934	0.012922669	0.016717272	5.92e-004
HPSO[5]	**0.0126652**	0.0127072	0.0127191	1.5824e-005
FSA[7]	0.012665258	0.012665299	0.012665338	N/A
DSS-MDE[11]	0.012665233	0.012669366	0.012738262	1.25e-005
CPSOSA[18]	0.01266525	0.012668	0.012671	3.685693653588-679e-006
PSO-DE[10]	0.012665233	0.012665233	0.012665233	4.9e-012
HGSO	**0.0126652327-88319**	**0.0126652327-88321**	**0.0126652327-88342**	**4.350272215639-758e-015**

From Table 3, it can be found that the best feasible solution obtained by HGSO is competitive to that of HPSO [5], but better than the results obtained by the other methods. From Table 4, it can be seen that compared with the results reported by the other methods, the average searching quality of HGSO is the best. Besides, even the worst solution obtained by HGSO is better than the best solutions obtained by the other methods except HPSO [5]. In 30 independent runs, the standard deviation of the results by HGSO is also the smallest, its precision reaches 10^{-15}, which shows that the robustness of HGSO is the best for solving this problem.

Example 3. A pressure vessel design problem

This problem is taken from [21], in which the objective is to minimize the total cost ($f(x)$), including the cost of the material, forming and welding. A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 3. There are four design variables: 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, not including the head). Among the four variables, T_s and T_h are integer multiples of 0.0625 in, which are the available thicknesses of rolled steel plates, and R and L are continuous variables.

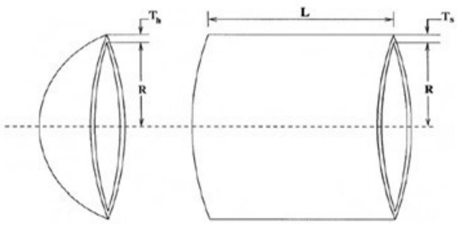

Figure 3 Center and end section of pressure vessel design problem

The problem can be mathematically formulated as follows:

$$
\begin{aligned}
& \min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\
& +3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
s.t.\quad & g_1(x) = -x_1 + 0.0193x_3 \leq 0; \\
& g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\
& g_3(x) = -\pi x_3^2x_4 - 4\pi x_3^3/3 + 1296000 \leq 0; \\
& g_4(x) = x_4 - 240 \leq 0 \\
& 0 \leq x_1, x_2 \leq 100; \\
& 10 \leq x_3, x_4 \leq 200
\end{aligned}
$$

The circular sensor range r_s and the initial dynamic decision domain r_0 of the glowworms are all set to 100 for this problem, the maximum number of generations is set to $T_{max} = 300$, which is consistent with that of HPSO[5], the other parameters are set to the same as those of Section 4.2. Simulation results and comparisons are given in Table 5 and Table 6.

Table 5 Comparison of the best solution for Example 3 by different methods

Methods	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	$f(x)$
Deb,Gene[22]	1.1250	0.6250	47.7000	117.7010	8129.8000
Kannan,Kramer[21]	1.1250	0.6250	58.2910	43.6900	7198.0428
Sandgren[23]	0.9315	0.5000	48.3290	112.6790	6410.3811
Coello[15]	0.8125	0.4375	40.3239	200.0000	6288.7445
Coello, Montes[16]	0.8125	0.4375	42.0974	176.6540	6059.9463
CPSO[8]	0.8125	0.4375	42.0913	176.7465	6061.0777
HPSO[5]	0.8125	0.4375	42.0984	176.6366	6059.7143
CPSOSA[18]	0.812500000-000000	0.437500000-000000	42.098445593-492706	176.6365958-720004	6059.7143
PSO-DE[10]	0.8125	0.4375	42.098445596	176.636595842	6059.714335
HGSO	**0.812500000-000000**	**0.437500000-000000**	**42.0984455-958549**	**176.6365958-424395**	**6059.7143-350484**

Table 6 Statistical results of different methods for Example 3

Methods	Best	Mean	Worst	Std. Dev.
Deb,Gene[22]	8129.8000	N/A	N/A	N/A
Kannan,Kramer[21]	7198.0428	N/A	N/A	N/A
Sandgren[23]	6410.3811	N/A	N/A	N/A
Coello[15]	6288.7445	6293.8432	6308.1497	7.4133
Coello, Montes[16]	6059.9463	6177.2533	6469.3220	130.9297
CPSO[8]	6061.0777	6147.1332	6363.8041	86.4545
HPSO[5]	6059.7143	6099.9323	6288.6770	86.2022
CPSOSA[18]	6059.7143	6059.7143	6059.7146	2.2641547578664-23e-006
PSO-DE[10]	6059.714335	6059.714335	6059.714335	1.0e-010
HGSO	**6059.71433504-84**	**6059.7143350484**	**6059.7143350484**	**9.2504274601307-37e-013**

From Table 5, it can be found that the best feasible solution obtained by HGSO is competitive to those of HPSO [5], PSO-DE [10] and CPSOSA [18], but better than the results obtained by the other methods. From Table 6, it can be seen that the mean solution and the worst solution of HGSO are competitive to those of PSO-DE [10], but better than those of the other methods. What is more, the standard deviation of the results by HGSO is also the smallest, its precision reaches 10^{-13}, which shows that the robustness of HGSO is the best for solving this problem.

Example 4. A speed reducer design problem

This problem is described in [24]. In this constrained optimization problem, the weight of speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shafts. The variables $x_1 \sim x_7$ represent the face width, length of the first shaft between bearings, lengths of the second shaft between bearings, and the diameter of first and second shafts respectively. This is an example of a mixed integer programming problem. The third variable x_3 (number of teeth) is of integer value while all left variables are continuous.

The problem can be mathematically formulated as follows:

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

subject to

$$
\begin{aligned}
g_1(x) &= \frac{27}{x_1x_2^2x_3} - 1 \leq 0 \\
g_2(x) &= \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0 \\
g_3(x) &= \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \leq 0 \\
g_4(x) &= \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leq 0 \\
g_5(x) &= \frac{[(745x_4/x_2x_3)^2 + 16.9 \times 10^6]^{1/2}}{110.0x_6^3} - 1 \leq 0
\end{aligned}
$$

$$g_6(x) = \frac{[(745x_5/x_2x_3)^2 + 157.5 \times 10^6]^{1/2}}{85.0x_7^3} - 1 \leq 0$$
$$g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0$$
$$g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0$$
$$g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0$$
$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$
$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$

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

The circular sensor range r_s and the initial dynamic decision domain r_0 of the glowworms are all set to 6 for this problem, the maximum number of generations is set to $T_{max} = 300$, which is in accordance with that of DSS-MDE[11], the other parameters are set to the same as those of Section 4.2. Simulation results and comparisons are shown in Table 7 and Table 8.

Table 7 Comparison of the best solution for Example 4 by different methods

Design variables	Akhtar,Tai, Ray[25]	Montes, Coello, Reyes[26]	PSO-DE[10]	Ray,Liew[14]	DSS-MDE[11]	**HGSO**
x_1	3.506122	3.500010	3.50000-00	3.50000-681	3.5000000-000	**3.500000000-000053**
x_2	0.700006	0.700000	0.7000000	0.70000-001	0.700000-0000	**0.700000000-000000**
x_3	17	17	17.000000	17	17	**17.000000000-000000**
x_4	7.549126	7.300156	7.30000000-0013	7.32760-205	7.3000000-000	**7.300000000-000000**
x_5	7.859330	7.800027	7.80000000-0005	7.71532-175	7.7153199-115	**7.715319911-478490**
x_6	3.365576	3.350221	3.35021466-6097	3.35026-702	3.35021-46661	**3.350214666-096505**
x_7	5.289773	5.286685	5.28668322-9758	5.28665-450	5.28665-44650	**5.286654464-980233**
$f(x)$	3008.08	2996.356-689	2996.348-165	2994.744-241	2994.471-066	**2994.471066-146868**

Table 8 Statistical results of different methods for Example 4

Methods	Best	Mean	Worst	Std. Dev.
Akhtar, Tai,Ray[25]	3008.08	3012.12	3028.28	N/A
Montes,Coello, Reyes[26]	2996.356689	2996.367220	N/A	8.2e-003
PSO-DE[10]	2996.348165	2996.348165	2996.348166	1.0e-007
Ray, Liew[14]	2994.744241	3001.758264	3009.964736	4.00914232
DSS-MDE[11]	2994.471066	2994.471066	2994.471066	3.58e-012
HGSO	**2994.47106614-6868**	**2994.47106614-7077**	**2994.47106614-7409**	**1.4410164949197-02e-010**

From Table 7, it can be found that the best feasible solution obtained by HGSO is competitive to that of DSS-MDE[11], but better than the results obtained by the other methods. From Table 8, it can be seen that the mean solution and the worst solution of HGSO are almost the same as the best solution of HGSO, even the worst solution of HGSO is better than the best solutions reported in [10], [14], [25] and [26]. Furthermore, the standard deviation of the results by HGSO is also very small, and its precision is 10^{-10}. It is a little worst than that of DSS-MDE [11], but markedly superior to those of the other methods.

Example 5. A three-bar truss design problem

This problem is taken from [27], which deals with the design of a three-bar truss structure where the volume is to minimize subject to stress constraints.

The problem can be mathematically formulated as follows:

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

subject to

$$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0$$
$$g_2(x) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0$$
$$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0,$$
$$0 \leq x_1, x_2 \leq 1$$

where $l = 100cm$, $P = 2KN/cm^2$, $\sigma = 2Kn/cm^2$.

The circular sensor range r_s and the initial dynamic decision domain r_0 of the glowworms are all set to 0.5 for this problem, the maximum number of generations is set to $T_{max} = 300$, which is consistent with that of DSS-MDE[11], the other parameters are set to the same as those of Section 4.2. Simulation results and comparisons are shown in Table 9 and Table 10.

Table 9 Comparison of the best solution for Example 5 by different methods

Methods	x_1	x_2	$f(x)$
Hernendez[27]	0.788	0.408	263.9
Ray, Saini[28]	0.795	0.395	264.3
Ray, Liew[14]	0.7886210370	0.4084013340	263.8958466
DSS-MDE[11]	0.7886751359	0.4082482868	263.8958434
PSO-DE[10]	0.788675134746	0.408248290037	263.89584338
HGSO	**0.788675130017075**	**0.408248303411660**	**263.8958433764684**

Table 10 Statistical results of different methods for Example 5

Methods	Best	Mean	Worst	Std. Dev.
Hernendez[27]	263.9	N/A	N/A	N/A
Ray, Saini[28]	264.3	N/A	N/A	N/A
Ray, Liew[14]	263.8958466	263.9033	263.96975	1.26e-002
DSS-MDE[11]	263.8958434	263.8958436	263.8958498	9.72e-007
PSO-DE[10]	263.89584338	263.89584338	263.89584338	4.5e-010
HGSO	**263.8958433764684**	**263.8958433764684**	**263.8958433764684**	**0**

From Table 9, it is observed that the best feasible solution obtained by HGSO is the best. From Table 10, it can be seen that the mean solution and the worst solution of HGSO are the same as the best solution of HGSO in 30 independent runs, they are better than the best solutions reported in [10], [11], [14], [27] and [28]. In addition, the standard deviation of the results by HGSO is also the smallest and its precision is 0, which illustrates that the robustness of HGSO is the best to solve this problem.

5. Conclusions

In this paper, a new algorithm named HGSO is proposed. Firstly, predatory behavior of AFSA is introduced to GSO and the IGSO algorithm is got, then, based on an optimum information sharing mechanism, IGSO is integrated with DE, besides, the constraint processing technology based on the feasibility rules is used to update the optimum position of the population. To escape from the local optimum, the local search strategy based on SA is applied to the best solution of the population of each generation. Finally, HGSO is tested on five benchmark functions and five engineering design problems. The experimental results show that the HGSO outperforms algorithms in the literature in terms of efficiency, precision, reliability and robustness, so, the HGSO is very effective for solving constrained design problems.

Acknowledgements

The authors are very grateful to the referees for their valuable comments and suggestions. These works are supported by National Science Foundation of China (61165015) and the funded by open research fund program of key lab of intelligent perception and image understanding of ministry of education of China under Grant: IPIU012011001.

References

[1] Xiaolei Li, Zhijiang Shao, Jixin Qian. *An Optimizing Method Based on Autonomous Animats: Fish-swarm Algorithm* . Systems Engineering and Theory and Practice, 22(11): 32-38 (2002).

[2] Krishnanand, K. N. and Ghose, D. *Detection of Multiple Source Locations Using a Glowworm Metaphor with Applications to Collective Robotics*, in Proceedings of IEEE Swarm Intelligence Symposium, pp. 84-91 (2005a).

[3] Krishnanand, K. N., Ghose, D. *Glowworm Swarm Optimisation: A New Method for Optimising Multi-modal Functions*. Int. J. Computational Intelligence Studies, 1(1): 93-119 (2009).

[4] Price K. V. *Differential Evolution: A Fast and Simple Numerical Optimizer. Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society*. Piscataway, NJ, USA: IEEE, 524-527 (1996).

[5] He Q, Wang L. *A Hybrid Particle Swarm Optimization with a Feasibility-Based Rule for Constrained Optimization*. Applied Mathematics and Computation, 186: 1407-1422 (2007).

[6] Kirkpatric S., Gelatt C. D. and Vecchi M. P. *Optimization by simulated annealing*. Science, 1983, 220: 671-680.

[7] A.R. Hedar, M. Fukushima, *Derivative-free filter simulated annealing method for constrained continuous global optimization*, Journal of Global Optimization. 2006,35 (4):521-549.

[8] He Q., Wang L. *An Effective Co-evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems*. Engineering Applications of Artificial Intelligence, 20: 89-99 (2007).

[9] W. Zhu, M. M. Ali, *Solving Nonlinearly Constrained Global Optimization Problem via An Auxiliary Function Method*, Journal of Computational and Applied Mathematics, 12-17 (2009).

[10] Hui Liu, Zixing Cai, Yong Wang. *Hybridizing Particle Swarm Optimization with Differential Evolution for Constrained Numerical and Engineering Optimization*. Applied Soft Computing, 10:629-640 (2010).

[11] Min Zhang, Wenjian Luo, Xufa Wang. *Differential Evolution with Dynamic Stochastic Selection for Constrained Optimization*. Information Sciences, 178, 3043-3074 (2008).

[12] Rao S. S. *Engineering Optimization (Third ed)*. New York: Wiley, 1996.

[13] Deb K. *Optimal Design of A Welded Beam via Genetic Algorithms*. AIAA Journal, 29: 2013-2015 (1991).

[14] T. Ray, K. M. Liew. *Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior*, IEEE Trans. Evol. Comput. 7 (4): 386-396 (2003).

[15] Coello C. A. C. *Use of A Self-adaptive Penalty Approach for Engineering Optimization Problems*. Computers in Industry, 41: 113-127 (2000).

[16] Coello C. A. C., Montes E. M. *Constraint Handling in Genetic Algorithms Through the Use of Dominance-Based Tournament Selection*. Advanced Engineering Informatics, ,16: 193-203 (2002).

[17] Coello C. A. C., Becerra R. L., *Efficient Evolutionary Optimization Through the Use of A Cultural Algorithm*. Engineering Optimization, 36: 219-236 (2004).

[18] Yongquan Zhou, Shengyu Pei. *A Hybrid Co-evolutionary Particle Swarm Optimization Algorithm for Solving Constrained Engineering Design Problems*. Journal of Computers, 5(6): 965-972 (2010).

[19] Arora J. S. *Introduction to Optimum Design*. New York: McGraw-Hill, (1989).

[20] Belegundu A. D., *A Study of Mathematical Programming Methods for Structural Optimization*. Dept. of Civil and Environmental Engineering, Univ. of Iowa, Iowa City, Iowa, (1982).

[21] Kannan B. K., Kramer S. N.*An augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design*. Trans. of the ASME, Journal of Mechanical Design, 116: 318-320 (1994).

[22] Deb K., Gene A. S., *A Robust Optimal Design Technique for Mechanical Component Design*. In: Dasgupta D and Michalewicz Z (Eds.), Evolutionary Algorithms in Engineering Applications, Berlin: Springer-Verlag, 497-514 (1997).

[23] Sandgren E., *Nonlinear Integer And Discrete Programming in Mechanical Design*. In: Proc. of the ASME Design Technology Conference. 95-105 (1988).

[24] J. K. Kuang, S. S. Rao, L. Chen, *Taguchi-aided Search Method for Design Optimization of Engineering Systems*, Eng. Optim. 30:1-23 (1998).

[25] S. Akhtar, K. Tai, T. Ray, *A Socio-behavioural Simulation Model for Engineering Design Optimization*, Eng. Optim. 34 (4): 341-354 (2002).

[26] E.M. Montes, C. A. C. Coello, J. V. Reyes, *Increasing Successful Offspring And Diversity in Differential Evolution for Engineering Design*, in: Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture (ACDM 2006), pp. 131-139 (2006).

[27] S. Hernendez, *Multiobjective structural optimization*, in: S. Kodiyalam, M. Saxena (Eds.), Geometry And Optimization Techniques for Structural Design, Elsevier, Amsterdam, The Netherlands, pp. 341-362 (1994).

[28] T. Ray, P. Saini, *Engineering Design Optimization Using a Swarm with An Intelligent Information Sharing Among Individuals*, Eng. Optim. 33 (3):735-748 (2001).

Yongquan Zhou Ph. D., professor. His current research interests include neural networks, intelligent computing.
Guo Zhou Ph. D. Student. His current research interests include virtual reality, human – computer interaction system.
Junli Zhang Master's, Her current research interests include the intelligent computing, optimization algorithm.