J. Cent. South Univ. (2014) 21: 3197−3204
DOI: 10.1007/s11771-014-2291-y

Springer

A hybrid cuckoo search algorithm with feasibility-based rule for constrained structural optimization

LONG Wen(龙文)[1], ZHANG Wen-zhuan(张文专)[1], HUANG Ya-fei(黄亚飞)[2], CHEN Yi-xiong(陈义雄)[2]

1. Guizhou Key Laboratory of Economics System Simulation, Guizhou University of Finance & Economics, Guiyang 550004, China;
2. School of Information Science and Engineering, Central South University, Changsha 410083, China

Abstract: Constrained optimization problems are very important as they are encountered in many science and engineering applications. As a novel evolutionary computation technique, cuckoo search (CS) algorithm has attracted much attention and wide applications, owing to its easy implementation and quick convergence. A hybrid cuckoo pattern search algorithm (HCPS) with feasibility-based rule is proposed for solving constrained numerical and engineering design optimization problems. This algorithm can combine the stochastic exploration of the cuckoo search algorithm and the exploitation capability of the pattern search method. Simulation and comparisons based on several well-known benchmark test functions and structural design optimization problems demonstrate the effectiveness, efficiency and robustness of the proposed HCPS algorithm.

Key words: constrained optimization problem; cuckoo search algorithm; pattern search; feasibility-based rule; engineering optimization

1 Introduction

Constrained optimization problems are always inevitable in many engineering disciplines, such as tension and/or compression spring design optimization problem [1], and welded beam design problem [2], and so on. The general constrained optimization problem with equality, inequality, lower bound, and upper bound constraints is defined as

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_j(\mathbf{x}) \le 0,\ j=1,2,\cdots,p \\ & h_j(\mathbf{x}) = 0,\ j=p+1,\cdots,m \\ & l_i \le x_i \le u_i,\ i=1,2,\cdots,n \end{aligned} \tag{1}$$

where $\boldsymbol{x}=(x_1, x_2, \cdots, x_n)$ is a dimensional vector of n decision variables, $f(\boldsymbol{x})$ is an objective function, $g_j(\boldsymbol{x}) \le 0$ and $h_j(\boldsymbol{x})=0$ are known as inequality and equality constraints, respectively. p is the number of inequality constraints and $m-p$ is the number of equality constraints, l_i and u_i are the lower bound and the upper bound of x_i, respectively.

Evolutionary algorithms have many advantages over conventional nonlinear programming techniques: easy implementation, reliable and robust performance, the gradients of the const function and constraint functions are not required, and the change of being trapped by a local minimum is lower. Due to those advantages, evolutionary algorithms have been successfully applied to solve constrained optimization problems in the past decade [3−6]. Recently, the cuckoo search (CS) algorithm, which is a population based stochastic global search method, has been proposed by YANG and DEB [7]. This algorithm is a novel evolutionary one, which is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. CS algorithm has shown good performance both on benchmark unconstrained functions [8] and on real-world problems, including data fusion in wireless sensor networks [9], embedded system design optimization [10], manufacturing scheduling [11], and structural optimization [12].

Although the CS algorithm is good at exploring the search space and locating the region of global minimum, it is slow at exploiting the solutions [13]. Pattern search method [14] is good at improving the accuracy of that approximation. Recently, evolutionary algorithms combined with other evolutionary operators or local search schemes have achieved good performance on variety of optimization problems, which have wide

Foundation item: Projects([2013]2082, [2009]2061) supported by the Science Technology Foundation of Guizhou Province, China; Project([2013]140) supported by the Excellent Science Technology Innovation Talents in Universities of Guizhou Province, China; Project(2008040) supported by the Natural Science Research in Education Department of Guizhou Province, China
Received date: 2013−05−06; **Accepted date:** 2013−10−28
Corresponding author: LONG Wen, Associate Professor, PhD; Tel: +86−13639086822; E-mail: lw084601012@gmail.com

applications in a variety of fields. Thus, a novel hybrid cuckoo search algorithm based on a local pattern search method that relies on a feasibility-based rule for constraint-handling is proposed to solve constrained optimization problems in this work. The proposed method would inherit both the advantages of the stochastic exploration ability of the cuckoo search algorithm and the exploitation ability of the pattern search algorithm. Simulation results and comparisons are demonstrated.

2 Proposed hybrid method

2.1 Cuckoo search algorithm

For simplicity in describing the cuckoo search algorithm, the following three idealized rules are used [7]:

1) Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;

2) The best nests with high quality of eggs (solutions) will carry over to the next generations;

3) The number of available host nests is fixed, and a host can discover an alien egg with a probability $P_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.

For simplicity, this last assumption can be approximated by a fraction P_a of the n nests being replaced by new nests (with new random solutions at new locations). For a maximization problem, the quality or fitness of a solution can simply be proportional to the objective function. Other forms of fitness can be defined in a similar way to the fitness function in GA. Based on these three rules, the basic steps of the cuckoo search algorithm can be summarized as the pseudo code as below.

When generating new solution $x^{(t+1)}$, e.g., cuckoo i, a Lévy flight is performed as

$$x_i^{(t+1)} = x_i^t + \alpha \oplus \text{Lévy}(\lambda) \tag{2}$$

where α>0 is the step size which should be related to the scales of the problem of interest. In most cases, we can see α=O(1). The product $\oplus$ means entry-wise multiplications. Lévy flight essentially provides a random walk, while their random steps are drawn from a Lévy distribution for large steps which has an infinite variance with an infinite mean as

$$\text{Lévy} \sim u = t^{-\lambda}, (1 < \lambda < 3) \tag{3}$$

Here, the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail. Pseudo code of the cuckoo search algorithm is presented below.

Algorithm 1 Cuckoo search algorithm

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}}$

Generation t=1;

Initial a population of n host nests x_i(i=1, 2, …, n);

While (t<Maximum Generation) or (stop criterion)

  Get a cuckoo (say i) randomly by Lévy flights;

  Evaluate fitness for cuckoo F;

  Choose a nest among n (e.g., j) randomly;

  If (F_i>F_j) then

  Replace j by the new solution;

  End if

  Abandon a fraction (P_a) of worse nests and build new ones;

  Keep the best solutions (or nests with quality solutions)

  Rank the solutions and find the current best;

  Update the generation number t=t+1;

End while

2.2 Pattern search method

Pattern search (PS) method [14] is adopted in this work due to its smaller memory requirement and effective local search capability. The basic PS algorithm is a simple direct search method that does not require derivative or second derivative information. PS is traditionally employed when the gradient of the function is not reliable when performing the search. The basic PS algorithm moves along the coordinate axes or other user defined positive spanning set to improve an existing solution in a greedy way. The step size is reduced, typically in half, when an improvement is not found in any direction. The pattern search component is illustrated in Fig. 1.

2.3 Constraint-handling method

Obviously, it is important to choose an appropriate constraint-handling technique for solving constrained optimization problems. The feasibility-based rule method is proposed in this work to handle constraints because it does not need to tune additional penalty parameters. According to feasibility-based rules, the constraint violation of each individual should be calculated to judge whether the individual is in feasible region. Equation (4) gives the mathematic formula for constraint violation calculation in our proposed algorithm.

$$f_{\text{viol}}(\boldsymbol{x}) = \sum_{j=1}^{p} \max(0, g_j(\boldsymbol{x})) + \sum_{j=1}^{m-p} \max(0, \text{abs}(h_j(\boldsymbol{x}))) \tag{4}$$

Motivated by Ref. [15], a feasibility-based rule is employed to handle constraints, which is described as follows:

1) Any feasible solution is preferred to any infeasible solution;

2) Between two feasible solutions, the one having

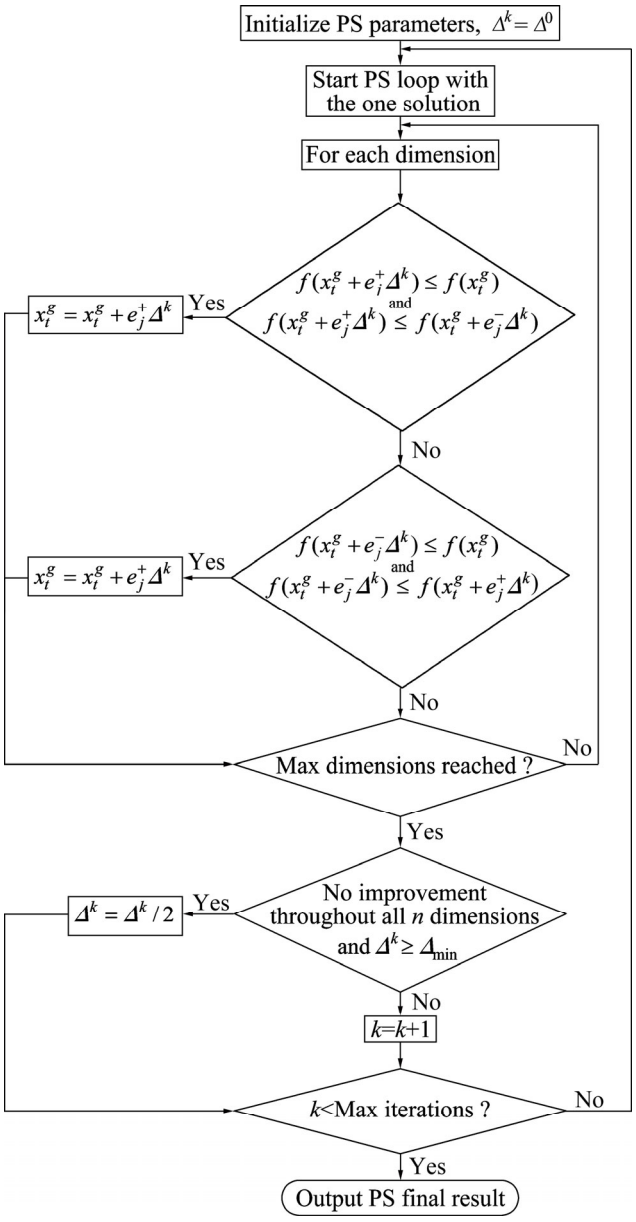

Fig. 1 Procedure of pattern search component

better objective function value is preferred;

3) Between two infeasible solutions, the one having smaller constraint violation is preferred.

Based on the above criteria, objective function and constraint violation information are considered, separately. Consequently, penalty factors are not used at all.

2.4 Hybrid CPS algorithm based on feasibility rule

While cuckoo search algorithms are good at quickly finding a reasonable solution, they may be slow to converge to the local optimal solution from a nearby solution and they may be stuck in a local optimum [13]. To overcome this problem, a local search improvement phase can be added to the CS algorithm. Pattern search is adopted in this work due to its smaller memory requirement and effective local search capability. After each iteration of the cuckoo search phase, the current best solution is improved independently with a local pattern search of neighboring solutions. The PS is adapted from the algorithm described in subsection 2.2. The pseudo code of the hybrid cuckoo pattern search (HCPS) algorithm with feasibility-based rule is presented below.

Algorithm 2 Hybrid cuckoo pattern search (HCPS) algorithm

Parameter setting;
Generation t=1;
Initial a population of n host nests x_i(i=1, 2, …, n);
While (t<Maximum Generation) or (stop criterion)
  Get a cuckoo (say i) randomly by Lévy flights;
  Evaluate fitness for cuckoo F;
  Choose a nest among n (say j) randomly;
  If (F_i>F_j) then
  Replace j by the new solution;
  End if
  Abandon a fraction (P_a) of worse nests and build new ones;
  Keep the best solutions (or nests with quality solutions), Rank the solutions and find the current best;
  Improve the chromosomes with pattern search algorithm;
  Select the offspring by feasibility-based rule;
  Update the generation number t=t+1;
End while

3 Simulations and comparisons

In this section, numerical simulation are carried out to investigate the performances of the proposed HCPS algorithm, where several constrained benchmark functions [16] and three well-studied structural design optimization problems [1] are used for testing. The parameters used by HCPS algorithm are the following: the population size n=50 nests, α=1, P_a=0.25, The maximum generation number is 3000, The maximum number of generations k_{max} for pattern search is set on 20 generations. All parameters of the HCPS algorithm are kept constant for all problems.

3.1 Benchmark test functions

Choose three well-known benchmark test functions to evaluate the performance of the proposed HCPS algorithm. The first test function is a maximization problem with twenty design variables and two inequality constraints. In this work, the maximization problems are transformed into minimization using $-f(\boldsymbol{x})$, and this problem can be stated as

$$\begin{cases} \text{F1}: \text{Max } f_1(\boldsymbol{x}) = \left| \dfrac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \\ \text{s.t.}: g_1(\boldsymbol{x}) = 0.75 - \prod_{i=1}^{n} x_i \le 0 \\ \quad g_2(\boldsymbol{x}) = \sum_{i=1}^{n} x_i - 0.75n \le 0 \end{cases} \tag{5}$$

and n=20 and $0 \le x_i \le 10$ (i=1, 2, ⋯, n).

The second problem is a minimization problem with two variables and two inequality constraints, and this problem can be stated as

$$\begin{cases} \text{F2}: \text{Min } f_2(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \\ \text{s.t.}: g_1(\boldsymbol{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0 \\ \quad g_2(\boldsymbol{x}) = -(x_1 - 6)^2 - (x_2 - 5)^2 - 82.81 \le 0 \end{cases} \tag{6}$$

and $13 \le x_1 \le 100$, and $0 \le x_2 \le 100$.

The third test function is a minimization problem with seven variables and four inequality constraints, and this problem can be stated as

$$\begin{cases} \text{F3: Min } f_3(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + \\ \quad 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\ \text{s.t.}: g_1(\boldsymbol{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0 \\ \quad g_2(\boldsymbol{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0 \\ \quad g_3(\boldsymbol{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0 \\ \quad g_4(\boldsymbol{x}) = 4x_1^2 + x_2^2 - x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 \end{cases} \tag{7}$$

and $-10 \le x_i \le 10 (i = 1, \cdots, 7)$.

Table 1 contains as summary of the execution results of the same three benchmark functions, which are compared with those of MOEA [3], CPSO [4], MABC [5], SR [16], HGPS [17], SAFF [18], and SMES [19]. For the sake of comparison, the table also gives the reference optimal values. As listed in Table 1, the proposed HCPS is able to find the global optima consistently on test functions over 30 runs except for F1. With respect to test function F1, yet the optimal solutions are not consistently found, the best results achieved are very close to the global optimal solutions. The distribution of the resulting solutions for test function F1 is shown in Fig. 2. Note that the standard deviations over 30 runs for all the functions are relatively small, which reflects that HCPS is capable of performing a robust and stable search. In addition, the convergence result for each problem is shown in Fig. 3 so that the efficiency of HCPS can be demonstrated more explicitly. As can be seen, HCPS achieves the convergence at a very early stage for three problems.

From Table 1, compared with SR, HGPS, and SAFF, HCPS finds better "best", "mean", and "worst" results for three test functions. With respect to MABC and SMES, HCPS provides better "mean", and "worst" results for function F1 and better results for functions F2 and F3. The better "best" results obtained by MABC and SMES for function F1. Compared with MOEA, HCPS finds similar results for functions F2 and F3. CPSO is one of the most competitive algorithms known to date. With respect to CPSO, HCPS provides similar results for three functions.

The above discussion validates that HCPS is an effective and efficient method for constrained optimization, and that it is capable of providing competitive results.

3.2 Structural design optimization problems

Structural design optimization problems are complex,

Table 1 Experimental results of HCPS and other methods for three benchmark functions

Function	Optimal		MOEA[3]	MABC[5]	SR[16]	HGPS[17]	SAFF[18]	SMES[19]	CPSO[4]	HCPS
F1	−0.803619	Best	−0.803619	−0.803611	−0.803515	−0.611330	−0.80297	−0.803601	−0.803619	−0.803519
		Mean	−0.803220	−0.795430	−0.781975	−0.556323	−0.79010	−0.785238	−0.801653	−0.801041
		Worst	−0.792608	−0.770319	−0.726288	−0.526660	−0.76043	−0.751322	−0.784076	−0.792458
		Std.	2.0×10^{-3}	9.47×10^{-3}	2.0×10^{-2}	2.50×10^{-2}	1.2×10^{-2}	1.7×10^{-2}	4.7×10^{-3}	3.4×10^{-3}
F2	−6961.814	Best	−6961.814	−6961.814	−6961.814	−6961.814	−6961.800	−6961.814	−6961.814	−6961.814
		Mean	−6961.814	−6961.814	−6875.940	−6961.814	−6961.800	−6961.284	−6961.814	−6961.814
		Worst	−6961.814	−6961.813	−6350.262	−6961.809	−6961.800	−6961.482	−6961.814	−6961.814
		Std.	1.8×10^{-12}	4.0×10^{-4}	1.6×10^{2}	1.3×10^{-3}	0.000	1.900	3.8×10^{-11}	5.1×10^{-10}
F3	680.630	Best	680.630	680.631	680.630	680.6301	680.64	680.632	680.630	680.630
		Mean	680.630	680.636	680.656	680.6301	680.72	680.643	680.630	680.630
		Worst	680.630	680.641	680.763	680.6301	680.87	680.719	680.630	680.630
		Std.	4.7×10^{-13}	2.6×10^{-3}	3.4×10^{-2}	0.000	5.9×10^{-2}	1.6×10^{-2}	5.9×10^{-17}	2.8×10^{-12}

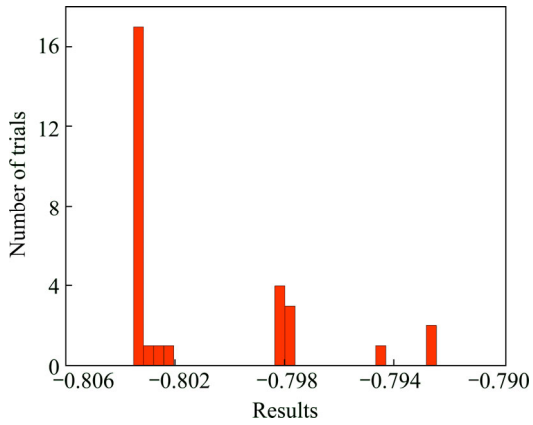

Fig. 2 Distribution of resulting solutions for test function F1

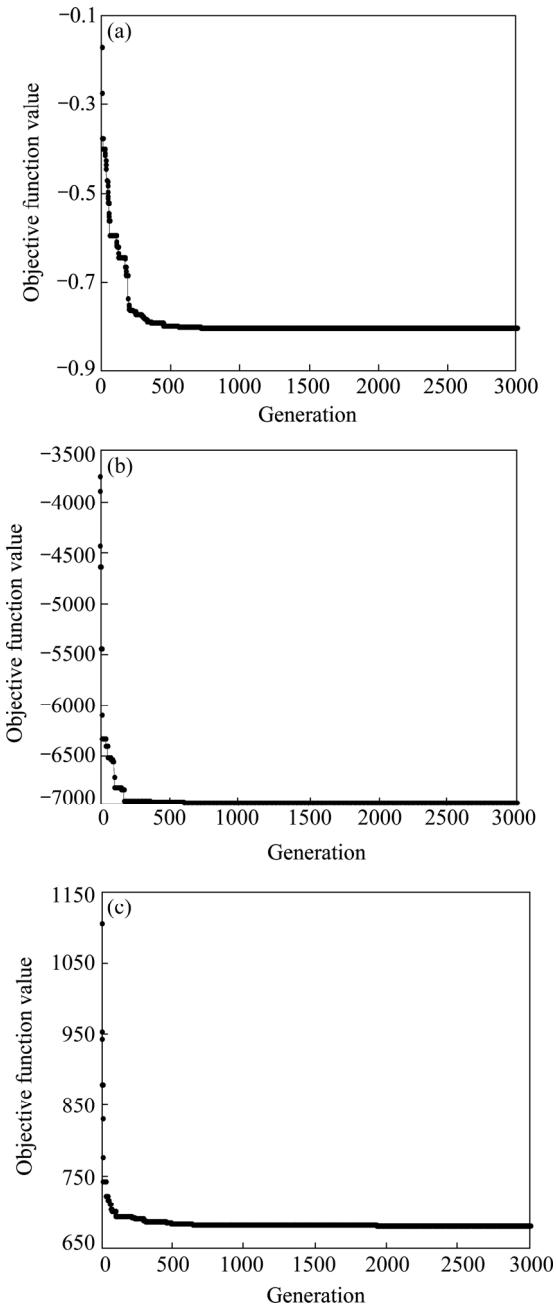

Fig. 3 Convergence results for three test functions: (a) F1; (b) F2; (c) F3

sometimes even the optimal solutions of interest do not exist. In order to see how HCPS algorithm performs, three standard structural engineering test problems are solved.

3.2.1 Three-bar truss structural design

This case considers a three-bar planar truss structure shown in Fig. 4. The volume of a statically loaded three-bar truss is to be minimized subjected to stress (σ) constraints on each of the truss members. The objective is to evaluate the optimal cross sectional areas (x_1, x_2).

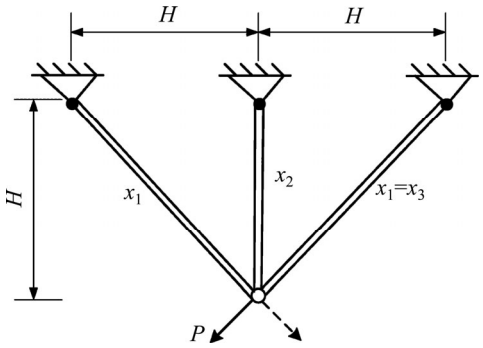

Fig. 4 Three-bar planar truss structural design problem

The mathematical formulation is given as follows:

$$\begin{cases} \text{Min}: f(\boldsymbol{x}) = (2\sqrt{2x_1} + x_2) \times l \\ \text{s.t.}: g_1(\boldsymbol{x}) = \dfrac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\ g_2(\boldsymbol{x}) = \dfrac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\ g_3(\boldsymbol{x}) = \dfrac{1}{x_1 + \sqrt{2}x_2} P - \sigma \le 0 \end{cases} \tag{8}$$

where $0 \ge x_1 \ge 1$ and $0 \ge x_2 \ge 1$, l=100 cm, P=2 kN/cm^2, and σ=2 kN/cm^2.

This design problem is a nonlinear fractional programming problem. The comparison of obtained statistical results for the HCPS with previous studies including water cycle algorithm (abbreviated by WCA) [1], cuckoo search (denoted as CS) [12] and swarm with intelligent information sharing (denoted as SIIS) [20] are given in Table 2. As shown in Table 2, it is evident that the searching quality of HCPS is higher than those of other methods. Table 3 presents the best solutions obtained by HCPS and those reported by CS, WCA, and

Table 2 Comparison of statistical results obtained from various algorithms for three-bar truss structural design problem

Method	Best	Mean	Worst	Std.
SIIS[20]	264.3	N/A	N/A	N/A
WCA[1]	263.895843	263.895903	263.896201	8.71×10^{-5}
CS[12]	263.97156	264.0669	N/A	9.00×10^{-5}
HCPS	263.895843	263.895912	263.896052	5.27×10^{-6}

Table 3 Comparison of best solution obtained from previous algorithms for three-bar truss structural design problem

Parameter	SCA[20]	WCA[1]	CS[12]	HCPS
x_1	0.795	0.788651	0.78867	0.78867515
x_2	0.395	0.408316	0.40902	0.40824826
$g_1(x)$	−0.00169	0	−0.00029	-7.55×10^{-14}
$g_2(x)$	−0.26124	−1.464024	−0.26853	−1.46410165
$g_3(x)$	−0.74045	−0.535975	−0.73176	−0.53589835
$f(x)$	264.3	263.895843	263.9716	263.895843

SCA. From Table 3, it is clear that the best solution obtained by HCPS is better than those by SCA and CS. Compared with WCA, HCPS provides similar results.

3.2.2 Tension/compression spring structural design

The tension/compression spring structural design problem is described in Ref. [1] for which the objective is to minimize the weight ($f(x)$) of a tension/compression string (as shown in Fig. 5) subjected to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. There are three design variables: the wire diameter $d(x_1)$, the mean coil diameter $d(x_2)$, and the number of active coils $p(x_3)$.

The tension/compression spring structural design problem is stated as follows:

$$\begin{cases} \text{Min}: f(\boldsymbol{x}) = (x_3+2)x_2x_1^2 \\ \text{s.t.}: g_1(\boldsymbol{x}) = 1-\dfrac{x_2^3x_3}{71785x_1^4} \le 0 \\ \quad g_2(\boldsymbol{x}) = \dfrac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\dfrac{1}{5108x_1^2}-1 \le 0 \\ \quad g_3(\boldsymbol{x}) = 1-\dfrac{140.45x_1}{x_2^2x_3} \le 0 \\ \quad g_4(\boldsymbol{x}) = \dfrac{x_1+x_2}{1.5}-1 \le 0 \end{cases} \tag{9}$$

where $0.25\le x_1\le 1.3$, $0.05\le x_2\le 2.0$, and $2\le x_3\le 1.5$.

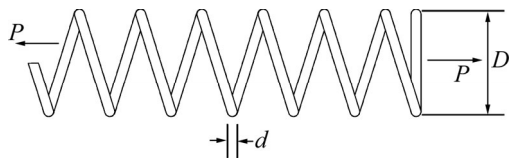

Fig. 5 Tension/compression spring structural design problem

The optimization engines previously applied to tension/compression spring structural design problem include WCA [1], DEDS [6], and CS [12]. The comparisons for the best solutions given by different algorithms are presented in Table 4. Their comparison statistical results are listed in Table 5.

It can be observed from Table 4 that the objective values by four algorithms. From Table 5, with respect to

Table 4 Comparison of best solution obtained from previous algorithms for spring structural design problem

Parameter	WCA[1]	DEDS[6]	CS[12]	HCPS
x_1	0.051680	0.051689	0.051690	0.0516891
x_2	0.356522	0.356717	0.356750	0.3567178
x_3	11.300410	11.288965	11.287126	11.288955
$g_1(x)$	-1.65×10^{-13}	1.45×10^{-9}	-3.56×10^{-5}	3.46×10^{-6}
$g_2(x)$	-7.9×10^{-14}	-1.19×10^{-9}	2.18×10^{-5}	-1.98×10^{-6}
$g_3(x)$	−4.053399	−4.053785	−4.053787	−4.0537925
$g_4(x)$	−0.727864	−0.727728	−0.727707	−0.7277287
$f(x)$	0.012665	0.012665	0.012665	0.012665

Table 5 Comparison of statistical results obtained from various algorithms for spring structural design problem

Method	Best	Mean	Worst	Std.
WCA[1]	0.012665	0.012764	0.012952	8.06×10^{-5}
DEDS[6]	0.012665	0.012669	0.012738	1.30×10^{-5}
CS[12]	0.012665	0.012729	0.013056	1.21×10^{-4}
HCPS	0.012665	0.012667	0.012701	1.60×10^{-6}

WCA, DEDS, and CS, HCPS finds similar "best" results and better "mean", "worst", and "Std." results for tension/compression spring structural design problem.

3.2.3 Speed reducer structural design

In this problem (see in Fig. 6), the weight of speed reducer is to be minimized subjected 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 to x_7 represent the face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), and the diameter of first (d_1) and second shafts (d_2), respectively.

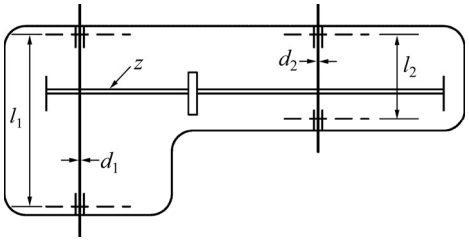

Fig. 6 Speed reducer structural design problem

The mathematical formulation can be summarized as follows:

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

$$\text{s.t.}: g_1(\boldsymbol{x}) = \frac{27}{x_1x_2^2x_3}-1 \le 0$$

$$g_2(\boldsymbol{x})=\frac{397.5}{x_1x_2^2x_2^2}-1\le 0$$

$$g_3(\boldsymbol{x})=\frac{1.93x_4^3}{x_2x_6^4x_3}-1\le 0$$

$$g_4(\boldsymbol{x})=\frac{1.93x_5^3}{x_2x_7^5x_3}-1\le 0$$

$$g_5(\boldsymbol{x})=\frac{[(745x_4/x_2x_3)^2+16.9\times10^6]^{1/2}}{110x_6^3}-1\le 0$$

$$g_6(\boldsymbol{x})=\frac{[(745x_5/x_2x_3)^2+157.5\times10^6]^{1/2}}{85x_7^3}-1\le 0$$

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

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

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

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

$$g_{11}(\boldsymbol{x})=\frac{1.1x_6+1.7}{x_5}-1\le 0 \tag{10}$$

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

This problem has been solved previously using cuckoo search (denoted as CS) [12], swarm with an intelligence information sharing (abbreviated by SIIS) [20], simple evolutionary algorithm (denoted as SEA) [21], and socio-behavioral simulation model (Abbreviated by SBSM) [22]. Table 6 lists the statistical results that have been determined by the above mentioned approaches as well as the proposed HCPS in this work. Table 7 shows the comparisons of the best solutions obtained by the proposed HCPS and other compared methods.

Table 6 Comparison of statistical results obtained from various algorithms for speed reducer structural design problem

Method	Best	Mean	Worst	Std.
CS[12]	3000.9810	3007.1997	3009.0000	4.9634
SIIS[20]	2732.9006	2758.8878	2758.3071	N/A
SEA[21]	3025.005	3088.7778	3078.5918	N/A
SBSM[22]	3008.08	3012.1200	3028.2800	N/A
HCPS	2994.47107	2994.62318	2994.76112	6.86×10^{-2}

From Tables 6 and 7, HCPS finds better "best", "mean", and "worst" results than CS, SEA, and SBSM. Even the worst solution found by HCPS is better than the best solutions by the CS, SEA, and SBSM. Although the best objective value derived by SIIS is better than those of HCPS, the reported value is not feasible. This is because the fifth and sixth constraints ($g_5(x)$, $g_6(x)$) are

Table 7 Comparison of best solution obtained from various studies for speed reducer structural design problem

Parameter	CS[12]	SIIS[20]	SEA[21]	SBSM[22]	HCPS
$b(x_1)$	3.5015	3.514185	3.506163	3.506122	3.5000
$m(x_2)$	0.7000	0.700005	0.700831	0.700006	0.7000
$z(x_3)$	17.0000	17.000000	17.000000	17.000000	17.00000000
$l_1(x_4)$	7.6050	7.497343	7.460181	7.549126	7.3000
$l_2(x_5)$	7.8181	7.8346	7.962143	7.85933	7.715320
$d_1(x_6)$	3.3520	2.9018	3.3629	3.365576	3.350215
$d_2(x_7)$	5.2875	5.0022	5.3090	5.289773	5.286654
$g_1(x)$	−0.0743	−0.0777	−0.0777	−0.0755	−0.07391528
$g_2(x)$	−0.1983	−0.2012	−0.2013	−0.1994	−0.19799853
$g_3(x)$	−0.4349	−0.0360	−0.4741	−0.4562	−0.49917245
$g_4(x)$	−0.9008	−0.8754	−0.8971	−0.8994	−0.90464387
$g_5(x)$	−0.0011	0.5395	−0.0110	−0.0132	-2.990×10^{-7}
$g_6(x)$	−0.0004	0.1805	−0.0125	−0.0017	-2.639×10^{-7}
$g_7(x)$	−0.7025	−0.7025	−0.7022	−0.7025	−0.70250000
$g_8(x)$	−0.0004	−0.0040	−0.0006	−0.0017	0
$g_9(x)$	−0.5832	−0.5816	−0.5831	−0.5826	−0.58333333
$g_{10}(x)$	−0.0890	−0.1660	−0.0691	−0.0796	−0.05132568
$g_{11}(x)$	−0.0130	−0.0552	−0.0279	−0.0179	-0.778×10^{-8}
$f(x)$	3000.9810	2732.9006	3025.005	3008.08	2994.47107

significantly violated in the results of SIIS.

Based on the aforementioned simulation and comparison results validate that the proposed HCPS has the substantial capability in handling various constrained structural design optimization problems and its solution quality is quite stable. So, it can be concluded that the proposed HCPS is a good alternative for constrained structural optimization.

4 Conclusions

1) A hybridization scheme for constrained structural optimization is presented which combines the feasibility-based rule for handling constraints with a cuckoo search algorithm as global optimizer and a pattern search method as local optimizer.

2) The proposed algorithm has demonstrated better performance than the other approaches in literature on solving three constrained optimization benchmark functions and three constrained structural design optimization problems.

3) In the future, we will apply HCPS to various problems found in the real world. Meanwhile, we are interested in extending our method so that it can deal with multi-objective optimization problems.

References

[1] ESKANDAR H, SADOLLAH A, BAHREININEJAD A, HAMDI M. Water cycle algorithm: A novel metaheuristic optimization method for solving constrained engineering optimization problems [J]. Computers and Structures, 2012, 110/111: 151−166.

[2] LONG Wen, LIANG Xi-ming, HUANG Ya-fei, CHEN Yi-xiong. A hybrid differential evolution augmented Lagrangian method for constrained numerical and engineering optimization [J]. Computer-Aided Design, 2013, 45(12): 1562−1574.

[3] CAI Zi-xing, WANG Yong. A multi-objective optimization-based evolutionary algorithm for constrained optimization [J]. IEEE Trans-actions on Evolutionary Computation, 2006, 10(6): 658−675.

[4] DANESHYARI M, YEN G G. Constrained multiple-swarm particle swarm optimization within a cultural framework [J]. IEEE Trans-actions on Systems, Man and Cybernetics, 2012, 42(2): 475−490.

[5] KARABOGA D, AKAY B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems [J]. Applied soft Computing, 2011, 11(3): 3021−3031.

[6] ZHANG Min, LUO Wen-jian, Wang Xu-fa. Differential evolution with dynamic stochastic selection for constrained optimization [J]. Information Sciences, 2008, 178(15): 3043−3074.

[7] YANG Xin-she, DEB S. Cuckoo search via Levy flights [C]// The World Congress on Nature and Biologically Inspired Computing, India: IEEE Publications, 2009: 210−214.

[8] TUBA M, SUBOTIC M, STANAREVIC N. Modified cuckoo search algorithm for unconstrained optimization problems [C]// The 5th European Conference on European Computing Conference. Wisconsin: WSEAS Publications, 2011: 263−268.

[9] DHIVYA M, SUNDARAMBAL M, ANAND L N. Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach[J]. International Journal of Communications, Network and System Sciences, 2011, 4(4): 249−255.

[10] KUMAR A, CHAKARVERTY S. Design optimization using genetic algorithm and cuckoo search [C]// The International Conference on Electro/Information Technology. Mankato: IEEE Press, 2011: 1−5.

[11] BURNWAL S, DEB S. Scheduling optimization of flexible manufa-cturing system using cuckoo search-based approach [J]. International Journal of Advanced Manufacturing Technology, 2013, 64(5/8): 951−959.

[12] GANDOMI A H, YANG X S, ALAVI A H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problem [J]. Engineering with Computers, 2013, 29(1): 17−35.

[13] LI Xiang-tao, YIN Ming-hao. Parameter estimation for chaotic systems using the cuckoo search algorithm with an orthogonal learning method [J]. Chinese Physics B, 2012, 21(5): 050507.

[14] ZHU Wei-hang. Massively parallel differential evolution-pattern search optimization with graphics hardware acceleration: An investigation on bound constrained optimization problems [J]. Journal of Global Optimization, 2011, 50(3): 417−437.

[15] DEB K. An efficient constraint handling method for genetic algorithms [J]. Computer Methods in Applied Mechanics and Engineering, 2000, 186(2/4): 311−338.

[16] RUNARSSON T P, YAO X. Stochastic ranking for constrained evolutionary optimization [J]. IEEE Transactions on Evolutionary Computation, 2000, 4(3): 284−294.

[17] COSTA L, SANTO I, FERNANDES E. A hybrid genetic pattern search augmented Lagrangian method for constrained global optimization [J]. Applied Mathematics and Computation, 2012, 218(18): 9415−9426.

[18] FARMANI R, WRIGHT J A. Self-adaptive fitness formulation for constrained optimization [J]. IEEE Transactions on Evolutionary Computation, 2003, 7(5): 445−455.

[19] MEZURA-MONTES E, COELLO C A C. A simple multimembered evolution strategy to solve constrained optimization problems [J]. IEEE Transactions on Evolutionary Computation, 2005, 9(1): 1−17.

[20] RAY T K, SAINI P. Engineering design optimization using a swarm with an intelligent information sharing among individuals [J]. Engineering Optimization, 2001, 33(6): 735−748.

[21] MONTES E M, COELLO C A C, RICARDO L. Engineering optimization using a simple evolutionary algorithm [C]// The 15th International Conference on Tools with artificial Intelligence, CA, USA, 2003: 149−156.

[22] AKHTAR S, TAI K, RAY T. A socio-behavioural simulation model for engineering design optimization [J]. Engineering Optimization, 2002, 34(4): 341−354.

(Edited by DENG Lü-xiang)