Chaotic Hybrid Invasive Weed Optimization for Machinery Optimizing

Lijun Xie
Department of Automotive Engineering, Changsha Vocational & Technical College, Changsha , China
Email: xielijun0209@163.com

Aijia Ouyang
College of Information Science and Engineering, Hunan University, Changsha, China
Email: ouyangaijia@163.com

Libin Liu*
Department of Mathematics and Computer Science, University of Chizhou, Chizhou, China
Email: Liulibin969@163.com

Minghua He
Institute of Higher Education, Jinggangshan University, Ji'an, China
Email: minghuahe@126.com

Xuyu Peng and Xu Zhou
College of Mathematics and Information Engineering, Jiaxing University, Jiaxing , China
Email: {pengxuyu, happypanda2006}@126.com

***Abstract*—We present a chaotic hybrid invasive weed optimization (CHIWO) algorithm with an adaptive penalty function which is introduced to solve the constraints for solving the machinery optimizing problems. The proposed CHIWO algorithm runs consistently well on the studies of 13 Benchmark functions and 3 machinery optimizing problems. Experimental results indicate that CHIWO algorithm is a simple and efficient method which can improve the performance of invasive weed optimization (IWO) in terms of computing accuracy and robustness.**

***Index Terms*—chaos, invasive weed optimization, BFGS method, hybrid algorithm, machinery optimizing**

I. INTRODUCTION

The machinery optimizing belongs to constrained optimization problem. A constrained optimization problem is generally defined as,

Minimize $f(x)$

$$\text{Subject to } g_j(x) \geq 0,\ j=1,2,\cdots,J \quad (1)$$

$$h_k(x) = 0,\ k = 1,2,\cdots,K \quad (2)$$

$$l_i \leq x_i \leq u_i,\ i = 1,2,\cdots,N \quad (3)$$

Where $f(x)$ is the fitness function, $h_k(x)$ is the equality constraint and $x_i \in [l_i,u_i]$, $g_j(x)$ is the jth inequality constraint.

Many traditional mathematical methods such as the Lagrange multiplier methods usually require the derivative information of the fitness function and constraints. Furthermore, the attained solution often tends to be a local optimum only if the search space is convex. Recently, intelligent algorithms (IA) have attracted much attention for a kind of optimization problems on a account of their superior advantages. IAs do not need the fitness function to be derivable or even continuous, and IAs run as global optimization methods due to the well balance between the global search and local search of the whole search space. So far, many approaches have been proposed by incorporating constraint-handling methods into IAs to solve constrained optimization problems.

Invasive weed optimization (IWO) is a swarm intelligence algorithm that mimics the colonizing behavior of weeds [1]. The main feature of a weed is that it grows its population entirely or predominantly in a geographically specified space, which can be substantially large or small. Initially a certain number of weeds are randomly spread over the entire search space. The weeds will grow up at last and perform the steps as follows.

Chaotic map is an efficient method in improving the basic IAs and the chaotic hybrid IWO (CHIWO) is presented [1,2]. The performance of CHIWO is investigated when applied to machinery optimizing problems in this paper. An adaptive penalty function is introduced to solve the constraints in the paper. CHIWO is compared against other approaches proposed in the literature on 13 Benchmark functions and 3 machinery optimizing.

This work is supported by the NSFC (No.61202109).
∗ Corresponding author, Libin Liu, E-mail: liulibin969@163.com.

doi:10.4304/jcp.8.8.2093-2100

The rest of the paper is organized as follows: Section II provides an overview of BFGS method, invasive weed optimization. CHIWO is presented by introducing the chaotic operator and self-adaptive penalty function in Section III. Experimental results of the study are discussed and analyzed in Section IV. Finally, Section V concludes the paper.

II. BFGS Method and Invasive weed optimization

A. BFGS Method

BFGS method belongs to the quasi-Newton methods, and is the most successful one. We can describe the method in this section, the method can minimize $f(x)$ generating a sequence of points X_k in $\Re^n$ such that

$$X_{k+1} = X_K + \alpha_K d_k ; k = 0,1,2,\cdots, \tag{4}$$

where $d_k \in \Re^n$ is a vector which demonstrates the search direction in the iteration of k and α_n is a scalar which gives the definition of the step length in this direction. Thus, in the each iteration of k , we can acquire the search direction of solving a linear system as follows:

$$B_K d_k = -\nabla f_k ; k = 0,1,2,\cdots, \tag{5}$$

where ∇f_k represents the gradient of f in the current point X_k and B_k is a n-order matrix. B_k is the Hessian matrix in the classic Newton method, in quasi-Newton methods B_k is an approximation to the Hessian which is updated at each iteration by means of a low-rank formula [3]. Rank-2 function is used in the BFGS, given by

$$B_k = B_{k-1} - \frac{B_{k-1} s_{k-1} s_{k-1}^T B_{k-1}}{s_{k-1}^T B_{k-1} s_{k-1}} + \frac{y_{k-1} y_{k-1}^T}{y_{k-1}^T s_{k-1}}, k = 1,2,\cdots, \tag{6}$$

where $s_{k-1} = X_k - X_{k-1}$, and $y_{k-1} = \nabla f_k - \nabla f_{k-1}$.

BFGS is not fully derivative-free although BFGS is Hessian-free, when the gradient vector is applied at each step, therefore, to compute the first-order partial derivatives in the function of f is vital for it. One may argue that, in case the gradient ∇f_k is costly or difficult to compute analytically, it can be approximated by finite difference techniques, as done continually in real utilizations. But the same argument holds for the Hessian matrix of f .

B. Invasive Weed Optimization

There are four steps of the invasive weed optimization algorithm as described as follows [4]:

Step 1. Initializing: a number of weeds which represent some trial solutions of the fitness function are initialized equably randomly in the objective space.

Step 2. Reproducing: each weed in the population can produce seeds relying on its own and population's fitness value. The number of seeds of a weed increases linearly from the minimum number of seeds for a plant with the maximum number of seeds for a weed with best fitness (which corresponds to the highest fitness function value for a maximization problem).

Step 3. Distributing: distributing the produced seeds randomly in the objective search space with random numbers with mean equal to zero. The previous step guarantees that the generated seeds will be produced around the parent weed, tending to fall into the trap of local optimum. Nevertheless, the random function's Standard Deviation (SD) is made to decrease with the iterations. Let *sd_min* and *sd_max* denote the minimum and maximum standard deviation respectively. Let *pow* denotes a real number. Hence the standard deviation for a particular iteration can be given as in Eq. (7):

$$sd_{ITER} = \left(\frac{iter_{\max} - iter}{iter_{\max}} \right)^{pow} \left(sd_{\max} - sd_{\min} \right) + sd_{\min} \tag{7}$$

The former step guarantees that the probability of dropping a seed in a distant area decreases nonlinearly with iterations, which leads to grouping fitter weeds and elimination of inappropriate weeds. Consequently, this is a selection operator of invasive weed optimization.

Step 4. Excluding competitively: Once a weed does not produce seeds then it would die out. Therefore, there is a need of some kind of competition between weeds for limiting maximum number of weeds in a population. Initially, the weeds in a population will generate rapidly and all the generated weeds will be included in the population, until the number of weeds in the population reaches a maximum value *pop_max*. Nevertheless, it is expected that by this time the fitter weeds have generated more than undesirable weeds. Since then, only the fittest weeds, among the existing ones and the generated weeds; are taken in the population and the steps 1 – 4 are redone until the maximum generation time has been arrived, i.e. the population size is fixed from thereon to *pop_max*. This method is known as competitive exclusion and is also a selection procedure of invasive weed optimization.

The pseudo-code of the standard invasive weed optimization is described as follows:

Step 1. Produce initially some number of weeds in the objective space.

Step 2. Run from step 3 to step 6 until the termination criterion for iteration.

Step 3. Compute the objective function values of the population then rank them.

Step 4. Let a weed generate some seeds between a predefined maximum and minimum number by users in the light of the following equation:

$$S_i = S_{\min} + (curr\_pop - rank_i) \times (S_{\max} - S_{\min}) \tag{8}$$

In which, $S_{\max}$ and $S_{\min}$ denotes the maximum and minimum number of seeds generated by a weed.

Step 5. Scatter randomly the seeds in the neighborhood of the parent weed. The standard deviation (SD) is computed as the Eq. (7).

Step 6. Outnumbers the maximum number predefined, rank it and abandon all the weeds except the best *np_init* one.

III Chaotic Hybrid Invasive Weed Optimization for Machinery Optimizing

A. Constrained Handling Method

To our knowledge, the penalty function technique has been the most popular constraint-handling method as result of its simple principle and IAs of implementation.

The sum of constrained violations on the solutions is added up to the fitness function so that the original constrained problems are transformed into unconstrained ones. A dynamic penalty function which can adjust the penalty factors' value is introduced to implement the goal that the algorithm can search in the feasible space according to the generations. The self-adaptive dynamic penalty function is as follows [5]:

$$\Phi(x) = f(x) + e^{\alpha(1-\rho_t)} \times \{\sum_{i=1}^{N} (\max[0, g_i(x)]^2 + \sum_{j=1}^{p} |h_j(x)|\} \quad (9)$$

B. Chaotic Hybrid Invasive Weed Optimization

Chaotic operator is incorporated into the invasive weed optimization algorithm in order to overcome the disadvantages of IWO [6-9]. It is remarkable that chaotic search operator has a strong ability to escape from the local optimum [10]; for this reason, the chaotic IWO (CIWO) algorithm has a lesser chance of pre-mature convergence compared to IWO. Moreover, as a result of global search and scanning ability, the execution of the chaotic operator prevents the need for increasing the number of seeds in the objective space. hence, the computing time of the whole algorithm is reduced relatively.

The chaotic map which is used in this paper is introduced in this section. Then, the proposed CHIWO method is described. For more information regarding chaotic maps refer to [11].

1) Chaotic maps

a) Logistic map

Logistic map is one of the famous and simplest chaotic maps, which has been used in some literatures [12] and it is presented by Sir Robert May in 1976.

An implication of this is the possibility that a simple deterministic dynamic system can expose complex chaotic behavior devoid of any stochastic disruptions. The logistic map is given as follows:

$$x_{k+1} = a x_k (1 - x_k) \quad (10)$$

where $x \in [0,1]$ denotes the chaotic variable and $a \in [0,4]$ denotes the control parameter. By managing the control parameter, one can determine whether the system is in the stable state, or in the chaotic one. The chaotic behavior of the sequence is ensured when $a = 4$, provided that the initial value for the chaotic variable (x) is in the range of (0,1) except for points $x = \{0.25, 0.5, 0.75\}$

b) Sinusoidal map

The sinusoidal map or the sinusoid iterator is defined as follows:

$$x_{k+1} = a x_k^2 \sin(\pi x_k) \quad (11)$$

which guarantees chaotic behavior in the span of (0,1). As it can be seen, the performance of the system becomes chaotic when $a = 2.3$.

c) Tent map

The tent map exhibits chaotic dynamics. This mapping generates chaotic sequences in data range (0,1). The following equation defines the tent map:

$$x_{k+1} = \begin{cases} a x_k & x < 0.5 \\ a(1 - x_k) & x \geq 0.5 \end{cases} \quad for \quad \alpha = 2 \quad (12)$$

2) Chaotic hybrid invasive weed optimization

The aim of the optimization algorithm is to minimize the function as follows:

$f(x_1, x_2, \cdots, x_m)$

Subject to

$$x_{\min}^i < x_i < x_{\max}^i \quad i = 1, \cdots, m \quad (13)$$

The steps of the proposed CHIWO algorithm proceed as follows:

Step 1. Initialize the parameters of the maximum and minimum value for each variable exploited in the optimization of fitness function. Chaotically distribute the pioneering seeds in the space employing the logistic maps described in Section 3.1. It is remarkable that the variables should be normalized to the range of (0,1) before employing the logistic map. The normalization procedure is described as follows:

Step 1.1. Transform variable x to $\hat{x}$ confined in the data range (0,1):

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (14)$$

Step 1.2. Execute the chaotic sequence to the transformed variable x generating a new value.

Step 1.3. Transform $\hat{x}$ into the range ($x_{\min}, x_{\max}$):

$$x = x_{\min} + \hat{x}(x_{\max} - x_{\min}) \quad (15)$$

Step 2. Each weed is evaluated and ranked in accordance with its fitness in the population.

Step 3. New seeds are generated in the case of each weed's ranking in the population. The latest produced seeds are distributed randomly in the space with the standard deviation calculated through Eq. (7).

Step 4. The latest produced seeds are chaotically dispersed in the neighborhood of the flowering weed using one of the chaotic maps outlined in the former subsection. If the current chaotically distributed seed has a better estimation than the previous one, maintain the novel one. If not, the chaotic sequence is continued. By making good use of the local search superiorities of chaotic search, the algorithm is ensured to converge much rapider.

Step 5. Rank the seeds once again, and some of them with lower fitness values are eliminated to reach the maximum number of weeds allowed which is preset by the user.

Step 6. Use BFGS method to search accurately.

Step 7. The algorithm continues at step 3 until generation time is arrived or a termination criterion is met.

IV Experimental Results and Analyses

The experiment in the study is divided into two parts in order to evaluate the performance of CHIWO, the first part is computed through 13 Benchmark functions and the second one is evaluated via 3 machinery optimizing problems.

A. Benchmark Functions

13 common Benchmark functions taken from [13] are used in the experiments in order to test the performance

of the algorithm CHIWO. The benchmark functions contain the features which are representative of what could be considered "difficult" constrained optimization problems for an evolutionary algorithm.

ISR is one of current most competitive approaches for constrained optimization, and it can be observed from Table 1 that it has found out the optimum in each run on all functions except g03, g05, g12 and g13. From the results of CHIWO in Table 1, CHIWO is better than ISR in best, mean, worst values and standard deviation on g03, g05, g12 and g13, and converges to the optimum of g13 in all 100 runs. It can be seen our approach CHIWO has distinct superiority over all the functions except g04, g08 and g13 in best, mean, worst values and std.dev. CHIWO can search the optimum 5126.498 among all the intelligent algorithms. The robustness of CHIWO is slightly weaker than IA-PSO but much stronger than other five algorithms on g13. The whole performance of CHIWO is similar to CHIWO, SMES, AI and PSO. We can see that our approach CHIWO has absolute superiority over all the aspects of performance except for robustness on g8. The performance of CHIWO is better than HIWO on all the functions. Therefore, the performance of CHIWO method is efficient and valuable.

TABLE I
COMPARISON PERFORMANCE OF EACH ALGORITHM

Fun	Term	Approaches for constrained optimization						
		ISR[13]	SMES[13]	RDE[13]	IA	IWO	HIWO	CHIWO
g01	Best	**-15.000**	**-15.000**	**-15.000**	**-15.000**	-15.0010	-15.000	**-15.000**
	Mean	**-15.000**	**-15.000**	-14.792	**-15.000**	-13.5678	-14.802	**-15.000**
	Worst	**-15.000**	**-15.000**	-12.743	**-15.000**	-8.1263	-14.115	**-15.000**
	Std.dev	5.8E-14	**0**	NA	**0**	3.4E+01	505E-5	**0**
g02	Best	**0.803619**	0.803601	**0.803619**	**0.803619**	0.803626	0.803605	**0.803619**
	Mean	0.782715	0.785238	0.746236	0.778987	0.782124	0.751014	**0.782414**
	Worst	0.723519	0.751322	0.302179	0.723050	0.710506	0.737342	**0.753502**
	Std.dev	2.2E-02	1.7E-02	NA	2.4E-02	1.87E-02	8.9E-03	**1.7E-05**
g03	Best	1.001	**1.000**	**1.000**	1.0009	1.0003	1.0004	**1.0005**
	Mean	1.001	**1.000**	0.640	1.0004	0.9945	1.0004	**1.0005**
	Worst	1.001	**1.000**	0.029	**1.0000**	0.6852	1.0004	**1.0005**
	Std.dev	8.2E-09	2.1E-04	NA	1.5E-03	4.9E-02	4.1E-08	**9.0E-10**
g04	Best	**-30665.539**	**-30665.539**	-30665.539	-30665.539	-30665.539	-30665.539	**-30665.539**
	Mean	**-30665.539**	**-30665.539**	-30592.154	-30665.539	-30665.539	-30665.539	**-30665.539**
	Worst	**-30665.539**	**-30665.539**	-29986.214	-30665.539	-30665.533	-30665.539	**-30665.539**
	Std.dev	1.1E-11	NA	NA	2.3E-07	5.8E-04	6.3E-10	**6.6E-13**
g05	Best	5126.497	5126.599	5126.497	5126.484	5126.386	5126.497	**5126.498**
	Mean	5126.497	5174.492	5218.729	5128.311	5495.451	5126.497	**5126.498**
	Worst	5126.497	5304.167	5502.410	5130.456	6271.620	5126.497	**5126.498**
	Std.dev	7.2E-13	5.0E+01	NA	6.3E-01	4.3E+02	8.8E-16	**2.6E-22**
g06	Best	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	-6961.829	**-6961.814**	**-6961.814**
	Mean	**-6961.814**	-6961.284	-6367.575	**-6961.814**	-6961.824	**-6961.814**	**-6961.814**
	Worst	**-6961.814**	-6952.482	-2236.950	**-6961.814**	-6961.815	**-6961.814**	**-6961.814**
	Std.dev	1.9E-12	1.9E+00	NA	5.9E-05	3.2E-04	5.2E-15	**1.8E-18**
g07	Best	**24.306**	24.327	**24.306**	**24.306**	24.328	**24.306**	**24.306**
	Mean	**24.306**	24.475	104.599	**24.306**	24.698	**24.306**	**24.306**
	Worst	**24.306**	24.843	1120.541	**24.306**	25.292	**24.306**	**24.306**
	Std.dev	6.3E-05	1.3E-01	NA	4.4E-02	2.3E-01	1.8E-05	**1.9E-06**
g08	Best	**0.095825**	**0.095825**	0.095825	0.095825	0.095824	**0.095825**	**0.095825**
	Mean	**0.095825**	**0.095825**	0.091292	0.095825	0.092424	**0.095825**	**0.095825**
	Worst	**0.095825**	**0.095825**	0.027188	0.095825	0.08876	**0.095825**	**0.095825**
	Std.dev	2.7E-07	**0**	NA	6.4E-10	1.4E-08	**3.6E-14**	3.3E-10
g09	Best	**680.630**	**680.630**	**680.630**	**680.630**	680.631	680.630	**680.630**
	Mean	**680.630**	680.643	692.472	**680.630**	680.645	680.654	**680.630**
	Worst	**680.630**	680.719	839.78	**680.630**	690.543	680.802	**680.630**
	Std.dev	3.2E-13	1.6E-02	NA	9.1E-06	5.2E-03	6.0E-4	**5.3E-20**
g10	Best	**7049.248**	7051.903	**7049.248**	**7049.248**	7088.321	**7049.248**	**7049.248**
	Mean	7049.250	7253.047	8442.661	7049.260	7545.214	7049.252	**7049.248**
	Worst	7049.270	7638.366	15580.370	7049.276	9098.246	7049.286	**7049.249**
	Std.dev	3.2E-03	1.4E+02	NA	4.9E-02	5.8E+02	1.4E-01	**1.5E-05**
g11	Best	**0.75**	**0.75**	**0.75**	**0.75**	0.74	0.75	**0.7499**
	Mean	**0.75**	**0.75**	0.76	**0.75**	0.77	0.76	**0.7499**
	Worst	**0.75**	**0.75**	0.87	**0.75**	0.97	0.77	**0.7499**
	Std.dev	1.1E-16	1.5E-04	NA	7.2E-20	5.4E-02	1.2E-6	**2.2E-23**
g12	Best	1.000	**1.000**	1.000	**1.000**	1.000	1.000	**1.000**
	Mean	1.000	**1.000**	1.000	**1.000**	1.000	1.000	**1.000**
	Worst	1.000	**1.000**	1.000	**1.000**	1.000	1.000	**1.000**
	Std.dev	1.2E-09	**0**	NA	**0**	2.3E-02	5.5E-14	**2.6E-16**
g13	Best	**0.053942**	0.053986	0.053866	0.05321	0.05978	**0.053942**	**0.053942**
	Mean	0.06677	0.166385	0.747227	0.85224	0.82554	**0.053942**	**0.053942**
	Worst	0.438803	0.468294	2.259875	2.01231	2.63645	**0.053942**	**0.053942**
	Std.dev	7.0E-02	1.8E-01	NA	5.0E-02	3.4E-01	2.8E-08	**3.3E-10**

B. Machinery Optimizing Problems

The main goal of this study is to investigate the performance of our proposed CHIWO algorithm for solving the machinery optimizing problems, three well-studied machinery optimizing problems that are chosen from [14] have been solved. And the best results attained by our algorithm CHIWO with adaptive adjusting over 100 independent runs have been compared with the IWO and those reported algorithms in literature.

Example 1. A tension/compression string design problem

This problem is described in Arora (1989), and the aim is to minimize the weight $f(x)$ of a tension/compression spring subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$ and the number of active coils $P(x_3)$.The mathematical formulation of this problem can be described as follows:

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

Subject to $g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0$

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

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0$$

$$0.05 \le x_1 \le 2$$

$$0.25 \le x_2 \le 1.3$$

$$2 \le x_3 \le 15$$

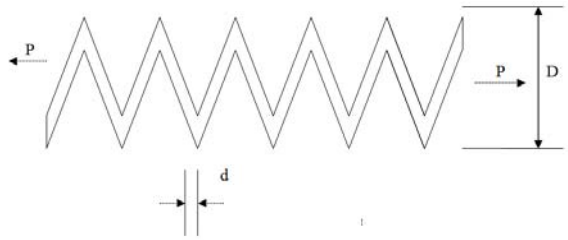

Figure 1. The structure of a tension/compression string

The details of best solutions by different approaches are given in Table II and III on Example 1. The best, worst, mean and standard deviation are shown in Table III. For best value attained by our approach. From Table II and III, we can see that our approach has found out the best solution when compared with CPSO[14], Beleg.[15], Arora[16], Coello[17] and C.M[18]. From Table III, it can be seen that the best, worst, mean and standard deviation of CHIWO is the best among six algorithms. We can conclude that the robustness of CHIWO is the strongest on Example 1 among six methods.

Example 2. A welded beam design problem

The following problem is taken from (Coello,2000), 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 (Pc); end deflection of the beam (δ) and side constraints. There are four design variables as shown in Figure 2, i.e., $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$. The problem can be mathematically formulated as follows:

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

Subject to $g_1(x) = \tau(x) - 13600 \le 0$

$$g_2(x) = \sigma(x) - 30000 \le 0$$

$$g_3(x) = x_1 - x_4 \le 0$$

$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0$$

$$g_5(x) = 0.125 - x_1 \le 0$$

$$g_6(x) = \delta(x) - 0.25 \le 0$$

$$g_7(x) = P - P_c(x) \le 0$$

$$0.1 \le x_1, x_4 \le 2$$

$$0.1 \le x_2, x_3 \le 10$$

Where

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

$$\tau' = \frac{p}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{QR}{J}$$

$$Q = P(L + \frac{x_2}{2})$$

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

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

$$\sigma(x) = \frac{6PL}{x_4x_3^2}$$

$$\delta(x) = \frac{4PL^3}{Ex_3^3x_4}$$

$$P_c(x) = \frac{4.013E\sqrt{x_3^2x_4^6}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000, L = 14, E = 30 \times 10^6, G = 12 \times 10^6$$

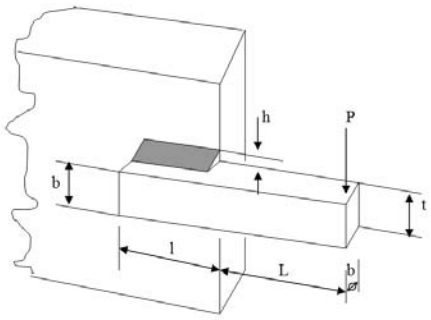

Figure 2. The structure of a welded beam

The details of best solutions by different approaches are given in Table IV on Example 2. The best, worst, mean and standard deviation are shown in Table V. The ability of searching optimum of CHIWO is the best when compared with CPSO[14], C.B.[19], Coello[17] and C.M[18]. The values of mean, worst and standard deviation of CHIWO are the best among the six methods except for the best value of CHIWO is slightly worse than CPSO. From Table V, it can be found that the average searching quality of CHIWO is competitive to that of CPSO and greatly superior to those of other methods. In addition, the standard deviation of the results by CHIWO in 100 independent runs is the smallest.

Example 3. A pressure vessel design problem

A pressure vessel design problem is introduced in this section the fitness is to minimize the total cost, including the cost of material, forming and welding. There are four variables: *Ts* (x_1, thickness of the shell), *Th*, (x_2, thickness of the cylindrical section of the vessel, not including the head). Among the four variables, *Ts* and *Th* integer multiples of 0.0625 in., which are the available thickness of rolled steel plates, and *R* and *L* are continuous variables. The problem can be formulated as follows:

Minimize

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to $g_1(x) = -x_1 + 0.0193x_3 \le 0$

$$g_2(x) = -x_2 + 0.00954x_3 \le 0$$

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$

$$g_4(x) = x_4 - 240 \le 0$$

$$0 \le x_1, x_2 \le 100$$

$$10 \le x_3, x_4 \le 200$$

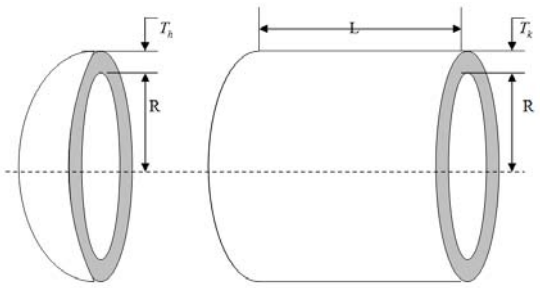

Figure 3. The structure of a pressure vessel

The details of best solutions by different approaches are given in Table VI on Example 3. The best, worst, mean and standard deviation are shown in Table VII. The ability of searching optimum of CHIWO is the best when compared with CPSO[14], Deb [20], KBK[21], S.E.[22], and Coello[17] but slightly worse than C.M[18]. The value of best of CHIWO is the smallest and the standard deviation is also the smallest among the methods. It indicates that CHIWO is a strong robustness algorithm.

TABLE II
COMPARISON OF THE BEST SOLUTION FOR EXAMPLE 1 BY DIFFERENT METHODS

Variables	CHIWO	CPSO[14]	Beleg.[15]	Arora[16]	Coello[17]	C.M[18]
$x_1(d)$	0.051710	0.051728	0.050000	0.053396	0.051480	0.051989
$x_2(D)$	0.357201	0.357644	0.315900	0.399180	0.351661	0.363965
$x_3(P)$	11.250123	11.244543	14.250000	9.185400	11.632201	10.890522
$g_1(x)$	-0.001456	-0.000845	-0.000014	0.000019	-0.002080	-0.000013
$g_2(x)$	-0.004041	-0.000013	-0.003782	-0.000018	-0.000110	-0.000021
$g_3(x)$	**-3.993361**	-4.051300	-3.938302	-4.123832	-4.026318	-4.061338
$g_4(x)$	-0.756236	-0.727090	-0.756067	-0.698283	**-04.026318**	-0.722698
$f(x)$	**0.0126670**	0.012675	0.012833	0.012730	0.012705	0.012681

TABLE III
STATISTICAL RESULTS OF DIFFERENT METHODS FOR EXAMPLE 1

Methods	Best	Mean	Worst	Std.dev.
CHIWO	**0.0126670**	**0.012687**	**0.0127100**	**3.2554e-6**
CPSO[14]	0.012675	0.012730	0.012924	5.198500e-5
Beleg.[15]	0.012833	N/A	N/A	N/A
Arora[16]	0.012730	N/A	N/A	N/A
Coello[17]	0.012705	0.012769	0.012822	3.939000e-5
C.M[18]	0.012681	0.012742	0.012973	5.900000e-5

TABLE IV
COMPARISON OF THE BEST SOLUTION FOR EXAMPLE 2 BY DIFFERENT METHODS

Variable	CHIWO	CPSO[14]	C.B.[19]	C.M.[18]	Coello[17]
$x_1(h)$	0.205020	0.204381	0.205700	0.205986	0.208800
$x_2(l)$	3.482114	3.505107	3.470500	3.471328	3.420500
$x_3(t)$	9.036623	9.033546	9.036600	9.020224	8.997500
$x_4(b)$	0.205801	0.205878	0.205700	0.206480	0.210000
$g_1(x)$	4.47e-02	-12.839796	-5743.826517	-5758.603777	-0.337812
$g_2(x)$	5.5e-01	-1.247467	-4.715097	-255.576901	-353.902604
$g_3(x)$	5.36e-4	-0.001498	0	-0.004400	-0.0012
$g_4(x)$	-3.432012	-3.429347	-3.020289	-2.982866	-3.411865
$g_5(x)$	-0.079126	-0.079381	-0.120500	-0.123900	-0.0838
$g_6(x)$	-0.235537	-0.235536	-0.234208	-0.234160	-0.235649
$g_7(x)$	3.5e-1	-11.681355	-3604.275002	-4465.270928	-363.232384
$f(x)$	1.729021	1.728024	1.724852	1.728226	1.748309

TABLE V
STATISTICAL RESULTS OF DIFFERENT METHODS FOR EXAMPLE 2

Methods	Best	Mean	Worst	Std.dev.
CHIWO	1.729021	**1.738088**	**1.766016**	**0.010786**
CPSO[14]	1.728024	1.748831	1.782143	0.012926
C.B.[19]	1.724852	N/A	N/A	N/A
C.M.[18]	1.728226	N/A	N/A	N/A
Coello [17]	1.748309	1.771973	1.785835	0.011220

TABLE VI
COMPARISON OF THE BEST SOLUTION FOR EXAMPLE 3 BY DIFFERENT METHODS

Variables	CHIWO	CPSO[14]	Deb [20]	KBK[21]	S.E.[22]	Coello[17]	C.M.[18]
$x_1(T_s)$	0.8127	0.8125	1.1250	1.1250	0.9375	0.8125	0.8125
$x_2(T_h)$	0.4374	0.4375	0.6250	0.6250	0.5000	0.4375	0.4375
$x_3(R)$	42.0986	42.0913	47.7000	58.2910	48.3290	40.3239	42.0974
$x_4(L)$	176.6516	176.7465	117.7010	43.6900	112.6790	200.0000	176.6541
$g_1(x)$	-2.6655e-6	-0.0001	-0.2044	0.00002	-0.0048	-0.0343	**-0.00002**
$g_2(x)$	-0.0367	-0.0359	-0.1699	-0.0689	-0.0389	-0.0528	-0.0359
$g_3(x)$	-8.2354	-116.3827	54.2260	-21.2201	-3652.8768	-27.1058	-27.8861
$g_4(x)$	-63.5449	-63.2535	-122.2990	-196.3100	-127.3210	-40.0000	-63.3460
$f(x)$	6061.7024	6061.0777	8129.1036	7198.0428	6410.3811	6288.7445	**6059.9463**

TABLE VII
STATISTICAL RESULTS OF DIFFERENT METHODS FOR EXAMPLE 3

Methods	Best	Mean	Worst	Std.dev.
CHIWO	6061.7024	**6090.1243**	**6285.3042**	15.2545
CPSO[14]	6061.0777	6147.1332	6363.8041	86.4545
Deb [20]	8129.1036	N/A	N/A	N/A
KBK[21]	7198.0428	N/A	N/A	N/A
S.E.[22]	6410.3811	N/A	N/A	N/A
Coello[17]	6288.7455	6293.8432	6308.1497	**7.4133**
C.M.[18]	**6059.9463**	6177.2533	6469.3220	130.9297

V. CONCLUSIONS

In this paper, new CHIWO approach is proposed and applied to solve machinery optimizing problems. The possibilities of exploring the CHIWO efficiency combined with adaptive penalty function are successfully presented, as illustrated by the studies of 13 Benchmark functions and 3 machinery optimizing problems. Our results indicate that CHIWO approach solve such problems efficiently in terms of precision and robustness and, in most cases, the performance of CHIWO is very nice comparing to the IWO and the other algorithms in the literatures.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China under Grant No.61202109, the Education and Science Planning of Hunan Province of China under Grant No.XJK011CJK010 and the Humanities & social sciences research project of Jinggangshan University of China under Grant No. JR1216.

REFERENCES

[1] S. Song, Y. Gan, L. Kong, and J. Cheng, "A Novel PSO Algorithm Based on Local Chaos & Simplex Search Strategy and its Application," *Journal of Software*, vol.6, no.4, pp.604-611, 2011.

[2] W. Liang, L. Zhang, and M. Wang, "The Chaos Differential Evolution Optimization Algorithm and its Application to Support Vector Regression Machine," *Journal of Software*, vol. 6, no.7, pp.1297-1304, 2011.

[3] J. Xie and J. Yang, "A Novel Crossover Operator for Particle Swarm Algorithm," *2010 International Conference on Machine Vision and Human-machine Interface*, Kaifeng, China, pp.161-164, 2010.

[4] D. Kundu, K. Suresh, S. Ghosh, S. Das, B. K. Panigrahi, and S. Das, "Multi-objective optimization with artificial weed colonies," *Information Sciences*, vol.181, no.12, pp.2441-2454, 2011.

[5] A. Ouyang, G. Zhou, and Y. Zhou, "A Self-adaptive immune PSO algorithm for constrained optimization problems," *Proceeding of the 5th International Symposium on Intelligence Computation and Applications*, pp.208-217, 2010.

[6] Z. Wang and P. Du, "Multifractal Analysis and Modeling of Chaotic Channels," *Journal of Software*, vol.7, no.3, pp.718-723, 2012.

[7] Y.-P. Zhang, X. Lin, Q. Wang, and R. O. Sinnott, "A Rapid Cryptography Algorithm Based on Chaos and Public Key," *Journal of Software*, vol.7, no.4, pp.856-860, 2012.

[8] J.-Z. Sun, G.-H. Geng, S.-Y. Wang, and M.-Q. Zhou, "Chaotic Hybrid Bacterial Colony Chemotaxis Algorithm Based on Tent Map," *Journal of Software*, vol.7, no.5, pp.1030-1037, 2012.

[9] H. Li, L. Wang, R. Zhang, and L. Wu, "A High Performance and Secure Palmprint Template Protection Scheme," *Journal of Software*, vol.7, no.8, pp.1827-1834, 2012.

[10] M. Ahmadi and H. Mojallali, "Chaotic invasive weed optimization algorithm with application to parameter estimation of chaotic systems," *Chaos, Solitons & Fractals*, vol.45, no.9–10, pp. 1108-1120, 2012.

[11] H. O. Peitgen, H. Jurgens, and D. Saupe, "*Chaos and fractals: new frontiers of science*," Second ed, New York: Springer, 2004.

[12] Y.-Y. He, J.-Z. Zhou, X.-Q. Xiang, H. Chen, and H. Qin, "Comparison of different chaotic maps in particle swarm optimization algorithm for long-term cascaded hydroelectric system scheduling," *Chaos, Solitons & Fractals*, vol.42, no.5, pp.3169–3176, 2009.

[13] M. Zhang, W. Luo, and X. Wang. "Differential evolution with dynamic stochastic selection for constrained

optimization," *Information Sciences*, vol.178, no.15, 2008, pp.3043–3074.

[14] Q. He and L. Wang, "An efficient co-evolutionary particle swarm optimization for constrained engineering designed problems," *Engineering Applications of Artificial Intelligence*, vol.20, pp.89-99, 2007.

[15] A. D. Belegundu, "*A study of mathematical programming methods for structural optimization*," Departments of Civil and Mechanical Engineering, The University of Iowa, Iowa city, Iowa, 1982.

[16] J. S. Arora, "*Introduction to optimization design*," New York:McGraw-Hill,1989.

[17] C. A. C. Colleo, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol.41, no.2, pp.113-127, 2000.

[18] C. A. C. Colleo and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Advanced Engineering Informatics*, vol.16, pp.193-223, 2002.

[19] C. A. C. Colleo and R. L. Becerra, "Efficient evolutionary optimization through the use of a cultural algorithm," *Engineering Optimization*, vol.36, no.2, pp.219-236, 2004.

[20] K. Deb, "Gene AS: a robust optimal design technique for mechanical component design," In: Dipankar Dasgupta and Zbigniew Michalewicz (eds), *Evolutionary Algorithms in Engineering Applications*, Berlin: Springer-Verlag, pp.497-514, 1997.

[21] B. K. Kannan and S. N. Kramer, "An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its application to mechanical design," *Transactions of the ASME, Journal of Mechanical Design*, vol.116, no.2, pp.318-320, 1994.

[22] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design," *Proceedings of the ASME Design Technology Conference*, Kissimine, FL, USA, pp.95-105, 1988.

Lijun Xie is a Lecturer in the department of automotive engineering, Changsha vocational & technical university, Changsha, Hunan Province, China. She received the B.E. degree in Electronics Engineering from Hunan Normal University, Changsha, China, in 1994. Her research interests include machinery optimizing, intelligence algorithm and nonlinear programming optimization. She has published research articles in international conference and journals of machinery optimizing and intelligence algorithm.

Aijia Ouyang is a Lecturer in the Department of Computer Science at Jiaxing University, Jiaxing, China. He received the M.E. degree in computer science and technology from Guangxi University for Nationalities, Nanning, China, in 2010. He is currently a Ph.D. candidate of Hunan University, Changsha, China. His research interests include parallel computing, cloud computing, supercomputing, mobile computing, and artificial intelligence. He has published research articles in international conference and journals of parallel computing.

Libin liu is a Lecturer in the Department of Mathematics and Computer Science and Technology at Chizhou University, Chizhou, Anhui Province, China. He received the M.E. degree in computer science and technology from Guangxi University for Nationalities, Nanning, China, in 2009. He is currently a Ph.D. candidate in the College of Mathematics and Computer Science and Technology at South China Normal University, Guangzhou, China. His research interests include parallel computing, and Numerical Methods for Differential Equations. He has published research articles in international conference and journals of Differential Equations.

Minghua He received the MS degree in computer-based language instruction from Shanghai International Studies University in 2008. He is currently a researcher of the Institute of Higher Education in Jinggangshan University, Jiangxi Province. His research projects have included computer-assisted language learning and intelligent algorithm-based language teaching. He has a strong interest in application of intelligent algorithms in foreign language instruction and learning. His research interests also include web-based learning, IT-based education reform, and educational technology. Several related papers have been published in domestic journals in recent years.

Xuyu Peng received the B.E. degree in the Department of Computer Science and Technology from Hunan University, Changsha, China, in 2000, and the M.E. degree in the Department of Computer Science and Technology from Hunan University, in 2009, respectively. Her research interests include parallel computing, cloud computing, supercomputing, DNA computing and scheduling algorithms. She has published research articles in international conference and journals of parallel computing and scheduling algorithms. She has published research articles in international conference and journals of cloud computing.

Xu Zhou is a Lecturer in the Department of Computer Science and Technology at Jiaxing University, Jiaxing, Zhejiang Province, China. She received the B.E. degree in the Department of Computer Science and Technology from Hunan University, Changsha, China, in 2006, and the M.E. degree in the Department of Computer Science and Technology from Hunan University, in 2009, respectively. She is currently a Ph.D. candidate in the College of Information Science and Engineering at Hunan University, Changsha, China. Her research interests include parallel computing, cloud computing, DNA computing and database query. She has published research articles in international conference and journals of supercomputing and DNA computing.