Combining Penalty Function with Modified Chicken Swarm Optimization for Constrained Optimization

Y.L. Chen[1,2,3], P.L. He[1,2] & Y.H. Zhang[1,2]

[1]Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu, Sichuan, China

[2]Key Laboratory of Optical Engineering, Chinese Academy of Sciences, Chengdu, Sichuan, China

[3]University of Chinese Academy of Sciences, Beijing, China

Keywords: Chicken Swarm Optimization; Bio-inspired algorithm; Nonlinear constraints; Penalty function; Optimization applications.

Abstract. In many mechanical designs, such as airborne electro-optical platform, optical lenses, mechanical containers, speed reducer, and so on, lightweight design has always been our goal. Under various constraints, obtaining the minimum of some parameter is the optimization problem we often encounter in the engineering works. Chicken Swarm Optimization (CSO), a new bio-inspired algorithm, is namely applied to deal with these kinds of problems. This paper firstly describes the origin and the basic model of the CSO and shows the result of applying the CSO to the algorithm test functions and a fair statistical comparison of the CSO with Bat Algorithm (BA) and modified Bat Algorithm based on Differential Evolution (DEBA) on the same test functions. Then, the CSO algorithm is modified. After that, the modified CSO is used to do the test on the previous test functions in order to be compared with the basic CSO, BA and DEBA. Finally, the modified CSO is combined with a dynamic penalty function to solve nonlinear constrained optimization problems and compared with other algorithms. From the results of all the tests, we can see that the CSO outperforms many other algorithms or their modified ones in terms of both optimization accuracy and stability. However, the modified CSO gets better performances than the CSO. As well, the modified CSO combined with penalty function is better than the CSO and many other optimization algorithms for constrained optimization problems.

Introduction

CSO algorithm (Meng, X.B. et al. 2014) was proposed on the Sixth International Conference on Swarm Intelligence in 2014. Through the observation of the individual and the entire flock of chickens, the researchers found that chicken has mature cognitive ability, communication skill and learning ability, and in the flock there exists an almost strict hierarchical order, similar to our teams. Flock activity exhibits complex and efficient swarm intelligence, which can be associated with the objective problem to be optimized. For example, in the optimization design for inner frame of airborne electro-optical platform, in order to improve the dynamic performance of airborne electro-optical platform and reduce the adverse effect of vibration environment on image quality, there needs to minimize the combined compliance index with the constraint that the fundamental frequency is below a certain value (Wang, P. et al. 2014).

Through ideally modeling the flock structure, identity of chicken, chickens' relationships and foraging law, researchers got this new bio-inspired algorithm.

Mimicking flock activity pattern and foraging law, the CSO gathers the whole wisdom of the flock. The CSO, owning the characteristics of simplicity and good scalability, is a naturally adaptive multiple swarm algorithm. Like many swarm intelligence algorithms, the CSO is a kind of stochastic optimization algorithm, using an iterative approach to solve the objective problem. Thus it just needs low mathematical analyticity and doesn't require that the target function is derivable. It can handle not only continuous problems but also discrete problems. Besides, its parameter configuration is simple.

Mathematical Model

The mathematical model of the CSO can be understood in the following way: firstly, confirm the flock structure, namely the number of the roosters, the hens, the chicks and the mother hens; secondly, set fixed identities for all of the chickens; thirdly, establish the mathematical model by chickens' identities and their foraging laws; and fourthly, set a certain interval to update the relationship of chickens regularly.

Flock Structure

Suppose there are *N* chickens in total in the flock. The proportion of roosters and hens is *rp* and *hp*, respectively. In hens, the proportion of mother hens is *mp*. Assume *RN*, *HN*, *CN* and *MN* indicate the number of the roosters, the hens, the chicks and the mother hens, respectively. Then the relationship among them can be expressed as follows.

$$RN = N * rp \tag{1}$$

$$HN = N * hp \tag{2}$$

$$CN = N - RN - HN \tag{3}$$

$$MN = HN * mp \tag{4}$$

The sizes of *RN*, *HN*, *CN* and *MN* directly decide the flock structure, which is one of the main factors that affect the performance of the CSO. From the optimization goal and the natural law, *HN* should be greater than *RN*, because hens can bring more benefits, such as eggs. Obviously, *HN* should also be greater than *MN*, because not all hens feed chicks. The model also sets the number of adult chickens to be greater than the number of chicks, *CN* (Meng, X.B. et al. 2014).

Indentity of Chicken

Assume that all the chickens are depicted by their own two properties, "fitness" and "position". So the model is built as follows.

$$\begin{cases} f_i(t) \\ x_{i,j}(t) \end{cases}, i \in [1, N],\ j \in [1, D] \tag{5}$$

Where *t* represents time step. *N* is the total number of chickens. *D* represents the dimension of the space in which the chickens can search for food. *f* denotes the value of "fitness", which corresponds to the value of the objective function. *x* represents the "position", which corresponds to the decision variables in the optimization problem to be solved. In this model, the aim of optimization is obtaining the minimum. Therefore, the smaller the value of *f*, the better the chicken's fitness (Meng, X.B. et al. 2014).

Foraging Law

According to the foraging law and identities of chickens, the authors of the literature (Meng, X.B. et al. 2014) proposed the core part of the CSO, that is, the movement of the chickens.

Rooster Model

The rooster model is built by the truth that the roosters with better fitness values can search for food in a wider range of place than those with worse fitness values (Meng, X.B. et al. 2014). This is formulated below.

$$x_{i,j}(t+1) = x_{i,j}(t) * (1 + N(0, \sigma^2)) \tag{6}$$

$$\sigma = \begin{cases} 1 & , f_i \le f_k \\ \exp\left(\dfrac{(f_k - f_i)}{|f_i| + \varepsilon}\right), & f_i > f_k \end{cases},\ k \in [1, N], k \ne i \tag{7}$$

Where $N(0, \sigma^2)$ is a normal distribution with mean 0 and standard deviation σ. ε is the smallest constant in the computer, which is used to avoid zero-division-error. *k* is a rooster's index, randomly selected from the rooster groups. The *k*th rooster is different from the *i*th rooster. *f* is the fitness value corresponding to *x* (Meng, X.B. et al. 2014).

From the established model, we can analyze the activity range of the current rooster. If $f_i \leq f_k$, then σ=1. Thus, the current *i*th rooster has better fitness, and it will have a wider range of activity next time. If $f_i > f_k$, then $0<\sigma<1$. Thus, the current *i*th rooster has worse fitness, and it will have a narrower range of activity next time.

Hen Model

Hen groups are more complex and diverse than rooster groups and chick groups. Hens can follow their group-mate rooster to search for food or do it by themselves. Hen would randomly steal the good food found by others. The more dominant hens would have advantage in searching for food. Besides, a part of hens will raise their own chicks (Meng, X.B. et al. 2014). These situations can be represented by the following formula.

$$\begin{aligned} x_{i,j}(t+1) = x_{i,j}(t) + c_1 * U_1(0,1) * (x_{r1,j}(t) - x_{i,j}(t)) \\ + c_2 * U_2(0,1) * (x_{r2,j}(t) - x_{i,j}(t)) \end{aligned} \quad (8)$$

$$c_1 = \exp\left(\frac{(f_i - f_{r1})}{|f_i| + \varepsilon}\right) \quad (9)$$

$$c_2 = \exp(f_{r2} - f_i) \quad (10)$$

Where $U_1(0,1)$ and $U_2(0,1)$ are uniform distributions over [0, 1]. $r1 \in [1, N]$, is the index of the head rooster in the *i*th hen's group. $r2 \in [1, N]$, is the index of a chicken different from the rooster *r*1, which is selected randomly from a specific group. In the specific group, every chicken's fitness value is better than the *i*th hen's.

Thus, $c_2<1<c_1$ only if $f_i > f_{r1}$ and $f_i > f_{r2}$. If c_1=0, then the *i*th hen wouldn't move around her head rooster, and it will search for food by herself. The smaller the *i*th hen's fitness value, the nearer c_1 approximates to 1 and the smaller the gap of positions between the *i*th hen and her head rooster. The larger the difference between f_i and f_{r2}, the larger the gap of positions between the *i*th hen and the chicken *r*2 and the smaller c_2. Therefore, the *i*th hen couldn't easily steal the food found by the chicken *r*2. If c_2=0, then the *i*th hen would search for food on her own territory. In addition, there exist competitions in the group, so the equation forms between c_1 and c_2 are different (Meng, X.B. et al. 2014).

Chick Model

The chicks move around their mother to search for food (Meng, X.B. et al. 2014). Use relatively simple way to simulate their activities below.

$$x_{i,j}(t+1) = x_{i,j}(t) + FM * (x_{m,j}(t) - x_{i,j}(t)) \quad (11)$$

Where $x_{m,j}(t)$ indicates the position of the *i*th chick's mother ($m \in [1, N]$). *FM* is a parameter ($FM \in (0, 2)$). For different chicks, the values of *FM* will be selected randomly. In practice, *FM*, belonging to the interval [0.4, 1], generally performed well.

Relationship Update

In addition to the model described above, there are two key parameters in the CSO, that is, the total number of iterations *M* and the interval of relationship update *G*. For *M* and *G*, they should be set to select suitable values based on the particular problem. If *G* is too large, it is not conducive to convergence to the global optimum quickly for the CSO; if *G* is too small, the algorithm may fall into local optimum. Typically, $G \in [2, 20]$ would be a good choice (Meng, X.B. et al. 2014).

Tests of CSO

The CSO can directly deal with most unconstrained optimization problems similar to Benchmark functions. Moreover, when dealing with such problems, the algorithm exhibits excellent performance. This section selects a part of the standard test functions in the literature (Xiao, H.H. & Duan, Y.M. 2014) to test the CSO, and then compares the results with the data in the literature (Xiao, H.H. & Duan, Y.M. 2014). Test functions are shown in Table 1.

In order to reflect the consistency of test, we ran the program of the CSO 20 times for each test function independently, and ensured that the parameters were same for each test function. The parameter settings of the CSO are shown in Table 2. List the best value, the worst value, the mean and the standard deviation of the results obtained by the CSO, and compare them with the data listed in the literature (Xiao, H.H. & Duan, Y.M. 2014). The experimental results, which are shown in Table 3, demonstrates that the performance of the CSO are much better than BA (Yang, X.S. 2010) and DEBA. All the original data created by the program of the CSO in the table are rounded and the first 9 digits after the decimal point are preserved below.

Modified Chicken Swarm Optimization (m-CSO)

In the flock, the number of roosters and chicks are smaller than that of hens, and their structures are relatively simple. The number of hens is the biggest, and the hens' structure is the most complex in the flock. Therefore, the hen model would directly affect the performance of the CSO. Through many times experiments, we analyzed the whole model of the CSO, modified the latter part of the hen model following the chick model, and kept the rest remained. The equation (8) was changed to the form below, that equation (12).

$$\begin{aligned} x_{i,j}(t+1) = x_{i,j}(t) + c_1 * U_1(0,1) * (x_{r1,j}(t) - x_{i,j}(t)) \\ + c_2 * U_2(0,1) * (x_{i,j}(t) - x_{r2,j}(t)) \end{aligned} \quad (12)$$

Table 1. Test functions

ID	Name	Expressions and Conditions	Optimum
F1	Schaffer F6	$f_1(x) = \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5, -100 \le x_i \le 100, n = 2$	-1
F2	Griewank	$f_2(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}), -600 \le x_i \le 600, n = 10$	0
F3	Rosenbrock	$f_3(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], -30 \le x_i \le 30, n = 10$	0

Table 2. Parameter settings

M	*N*	*G*	*rp*	*hp*	*mp*	*FM*
1000	100	10	0.2	0.6	0.1	[0.5, 0.9]

Table 3. Statistical comparison of CSO with BA and DEBA on F1-F3

Function	Algorithm	Best	Worst	Mean	Std.
F1	BA	-0.921810818	-0.511191588	-0.640175003	1.240439745e-01
	DEBA	-1	-0.999438907	-0.999925903	1.321207680e-04
	CSO	-1	-1	-1	0
F2	BA	48.356822946	4.884648927e+02	1.934004011e+02	1.142338058e+02
	DEBA	3.773594144e-07	3.907074747e-05	1.314841126e-05	1.105930329e-05
	CSO	0	0	0	0
F3	BA	4.138213137e+03	5.382751350e+07	1.013266796e+07	1.372576961e+07
	DEBA	22.860694188	88.599479455	55.249891509	17.805778520
	CSO	6.339087813	7.226184561	6.942939149	0.322589855

Table 4. Optimization results of m-CSO on the function F3

Function	Algorithm	Best	Worst	Mean	Std.	M
F3	m-CSO	1.677240079	2.802966172	2.240676430	0.332482466	1000
		0.081880091	0.180520462	0.127903339	0.030576332	2000
		0.005862121	0.027200427	0.015156016	0.004572982	3000
		0.001201132	0.004775559	0.002830799	8.357847107e-04	4000

The modified CSO algorithm, hereinafter referred to as the m-CSO algorithm, was tested independently on the third test functions F3 with the same parameters and the same running times. The results given by the program of the m-CSO are shown in Table 4 below. The parameter settings are same as above shown in Table 2. Obviously, with the same parameters, the m-CSO is superior to the CSO considering optimization accuracy and stability. Based on this, if only increasing the number of iterations, higher accuracy and stability can be achieved.

Applying Penalty Function to m-CSO for Constrained Optimization

In this paper, a penalty function was combined with the m-CSO, hereinafter referred to the pm-CSO algorithm, to solve the nonlinear constrained optimization problem containing the inequality constraints and the equality constraints.

Constrained Optimization Problem

The general form of constrained optimization problem is depicted as follows.

$$\min\ f(x)$$
$$s.t.\ \begin{cases} g_i(x) \le 0, & i = 1,\dots,k \\ h_i(x) = 0, & i = k+1,\dots,m \end{cases},\quad x \in R^n \tag{13}$$

Where, $x=(x_1, x_2, \dots, x_n)$ represents the decision variables. $g_i(x)$ and $h_i(x)$, respectively, represent the inequality constraints and the equality constraints. Generally, the equality constraints $h_i(x)=0$ can be transformed to the inequality constraints $-\sigma \le h_i(x) \le \sigma$. σ is the tolerance limit of the equality constraints (Mi, Y.Q. & Gao, Y.L. 2015). In the constrained optimization problem, $max[f(x)] \leftrightarrow min[-f(x)]$, and $g_i(x) \ge 0 \leftrightarrow (-g_i(x) \le 0)$, so the formula (13) can represent all the constrained optimization problems.

Penalty Function

In the course of solving constrained optimization problems, how to deal with the constraints is the key. The quality of processing the constraints is directly related to the quality of the final result. In this paper, a classical method to deal with the constraints was adopted, that is, the penalty function method. Although there were many different forms of penalty functions, the core of the penalty function method for constrained optimization problems had never been changed. That is, by weighting on the inequality constraints and the equality constraints, then combining them with the original objective function, a new objective function is created to replace the original one. Using such a simple method, the original constrained optimization problems are transformed to the unconstrained ones (Mi, Y.Q. & Gao, Y.L. 2015).

In this paper, a dynamic penalty function method was selected to handle constrained optimization problems (Yang, J.M. et al. 1997) as follows.

$$F(x,t) = f(x) + d(t) * p(x) \tag{14}$$

$$d(t) = \begin{cases} t * \sqrt{t} \\ 5 * t \\ \sqrt{t} \\ 0.0025 * t \\ \dots \end{cases} \tag{15}$$

$$p(x)=\sum_{i=1}^{n}\alpha_i * cond_i(x)^{\beta_i} \quad (16)$$

$$cond_i(x)=\begin{cases}\max(0,g_i(x)), & i=1,...,k\\ \max(0,|h_i(x)|), & i=k+1,...,m\end{cases} \quad (17)$$

$$\alpha_i=\begin{cases}1, & cond_i(x)<1\\ 2, & cond_i(x)\geq 1\end{cases} \quad (18)$$

$$\beta_i=\begin{cases}10, & cond_i(x)<0.01\\ 20, & 0.01\leq cond_i(x)\leq 0.1\\ 100, & 0.1<cond_i(x)\leq 1\\ 300, & cond_i(x)>1\end{cases} \quad (19)$$

Where $F(x, t)$ is the new objective function, a function of the decision variables x and the time step t. $f(x)$ is the original objective function. $d(t)$ is the penalty factor. $p(x)$ is a penalty term. $cond_i(x)$ is the degree function to violate the ith constraint. α_i and β_i are different levels of punishment function and punishment intensity. About $d(t)$, the selection depends on the specific circumstance.

Thus, the original constrained optimization problems can be transformed to equivalent unconstrained ones, that is, $min[f(x)]\leftrightarrow min[F(x, t)]$. The following is to validate the performance of the pm-CSO using two practical engineering problems.

Experiments and Analyses

Experiment 1: QQR-T1-4 Problem

The mathematical model of QQR-T1-4 problem (Hock, W. & Schittkowski, K. 1981) is expressed as follows.

$$\min\ f(x)=(x_1-2)^2+(x_2-1)^2,\quad x=(x_1,x_2) \quad (20a)$$

$$s.t.\begin{cases}g_1(x)=\dfrac{x_1^2}{4}+x_2^2-1\leq 0\\ h_2(x)=x_1-2x_2+1=0\end{cases},0\leq x_1,x_2\leq 10 \quad (20b)$$

Table 5. Optimization results of QQR-T1-4 by pm-CSO

Number	f	x_1	x_2	g_1	h_2
1	1.393464981	0.822875656	0.911437828	-1.33095757e-11	-2.39808e-14
2	1.393464989	0.822875652	0.911437826	-4.731791958e-09	-3.89470678e-11
3	1.393466416	0.822875074	0.911437452	-9.235509753e-07	1.694994745e-07
4	1.393464994	0.822875650	0.911437825	-7.219941978e-09	2.87960766e-11
5	1.393466749	0.822874932	0.911437466	-9.573716975e-07	-9.1955332e-12
6	1.393564702	0.822834897	0.911416575	-5.550970447e-05	1.747053610e-06
7	1.393465451	0.822875463	0.911437733	-2.518178630e-07	-3.219815081e-09
8	1.393477181	0.822870688	0.911434971	-7.250921217e-06	7.458435856e-07
9	1.393464988	0.822875652	0.911437826	-4.120968455e-09	1.50921498e-11
10	1.393501268	0.822860841	0.911429871	-2.059876895e-05	1.097870112e-06

This is a typical nonlinear optimization problem containing equality constraints and inequality constraints. Until now, there have been many scholars who had studied it. Literatures (Homaifar, A. et al. 1994, Fogel, D.B. 1995, Myung, H. et al. 1995, Yang, J.M. et al. 1997) showed us this issue. Literature (Yang, J.M. et al. 1997) listed the result of this issue, that is, f =1.3934651, which corresponds to the value of the decision variables (x_1, x_2) = (0.8228756, 0.9114378). It is the best currently known result.

Using the pm-CSO to solve this problem, and running the program 10 times independently, we got the results shown in Table 5. The parameters were same as ones in Table 2 except M. M is 2000 for this issue. And, the penalty factor $d(t)$ is "0.0025*t". As can be seen from the table, all the function values belong to the range [1.393464980713918, 1.393564702486817]; all the decision variables and inequality constraints are to meet the requirements; only equality constraints fluctuate around 0. In all results, we found that the 1st, 2nd, 4th, 9th result are relatively closer to the known optimum, and the precision of the first result is the highest. If the accuracy of equality constraints can be appropriately set to a little wider, we can say that the first result is better than the best currently known results.

Experiment 2: Speed Reducer Design

The mathematical model of speed reducer design problem (Reynolds, R. & Ali, M. 2008) is expressed as follows.

$$\begin{aligned}\min\ f(x) = {} & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 \\ & - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + \\ & 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2), \\ & x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)\end{aligned} \tag{21a}$$

$$s.t.1\begin{cases} g_1(x) = \dfrac{27}{x_1x_2^2x_3}, \qquad g_2(x) = \dfrac{397.5}{x_1x_2^2x_3^2}, \\ g_3(x) = \dfrac{1.93x_4^3}{x_2x_6^4x_3}, \qquad g_4(x) = \dfrac{1.93x_5^3}{x_2x_7^4x_3}, \\ g_5(x) = \dfrac{1}{110x_6^3}\sqrt{(\dfrac{745x_4}{x_2x_3})^2 + 16.9*10^6}, \\ g_6(x) = \dfrac{1}{85x_7^3}\sqrt{(\dfrac{745x_5}{x_2x_3})^2 + 157.5*10^6}, \\ g_7(x) = \dfrac{x_2x_3}{40}, g_8(x) = \dfrac{5x_2}{x_1}, g_9(x) = \dfrac{x_1}{12x_2}, \\ g_{10}(x) = \dfrac{1.5x_6 + 1.9}{x_4}, g_{11}(x) = \dfrac{1.1x_7 + 1.9}{x_5} \end{cases} \tag{21b}$$

$$s.t.2\begin{cases} 2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, \\ 7.3 \le x_4 \le 8.3, 7.8 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, \\ 5 \le x_7 \le 5.5, \\ g_i(x) \le 1 (i = 1, 2, ..., 11), \\ g = (g_1, g_2, g_3, ..., g_{10}, g_{11}) \end{cases} \tag{21c}$$

Use the pm-CSO to optimize this problem, run the program independently 20 times, and compare the result of the pm-CSO with ones of other optimization algorithms. The results are shown in Table 6. The parameter settings are same as above shown in Table 2. For this issue, the penalty factor $d(t)$ was "$t*t^{0.5}$".

Table 6. Optimization results of the speed reducer design

Algorithm		Best	Worst	Mean	Std.
Robert, et al (Reynolds, R. & Ali, M. 2008)	SFI(lBest)	3000.737	3044.332	3015.026	9.95
	SFI(square)	2996.974	3007.301	3000.278	2.804
	SFI(gBest)	2998.991	3034.973	3011.062	9.105
Mezura, et al (Mezura, M.E. & Hernandez, O.B. 2009)		2999.264	N/A	3014.759	11.0
Akay, et al (Akay, B. & Karaboga, D. 2012)		2997.05841	N/A	2997.05841	0
Gandomi, et al (Gandomi, A.H., Yang, X.S. & Alavi, A.H. 2013)		3000.981	3009	3007.2	4.963
CSO (Meng, X.B. et al. 2014)		2996.605	3007.258	2997.764	0.165
pm-CSO		2996.348164995	2996.348505298	2996.348205146	8.912827902e-05

According to the table above, the optimal value obtained by the pm-CSO is 2996.348164995. The decision variables corresponding to the optimal value are

x = (3.500000000004351, 0.700000000000000, 17.000000000005702, 7.300000000000000, 7.800000000107652, 3.350214666099439, 5.286683229790418).

The corresponding constraints are

g = (0.926084719600665, 0.802001472856516, 0.500827751895624, 0.098528302386299, 0.999999999997317, 0.999999999981577, 0.297500000000100, 0.999999999998757, 0.416666666667185, 0.948674246458789, 0.989147634956792).

From the results, the pm-CSO is better than another seven kinds of optimization algorithms, including the CSO. Compared with the original CSO, the pm-CSO has greatly improved the performance, whether the optimization accuracy or stability.

Conclusions and Discussions

As a new bio-inspired algorithm, the CSO inherits many advantages of other algorithms. In dealing with most unconstrained optimization problems, the CSO demonstrated its excellent performance. It can be far better than many of the current optimization algorithms in terms of the optimization precision and stability. However, the basic CSO could not perform satisfactorily while dealing with some specific unconstrained optimization problems and constrained ones. By modifying the original hen model of the CSO, we got the m-CSO, which performed better in the specific problems the CSO couldn’t deal with well. The test results in section 3 illustrated this point well.

In addition, we selected a dynamic penalty function to combine with the m-CSO for complex constrained optimization problems. Like this, we got the pm-CSO. With penalty function, the constrained optimization problems could be transformed to unconstrained ones. Then the pm-CSO can quickly optimize these unconstrained optimization problems. In this paper, the modification of the CSO and the combination with the penalty function make the CSO obtain more advantages in dealing with the optimization problem. The pm-CSO will be largely beneficial to solving engineering optimization problems such as the lightweight design of airborne electro-optical platform.

Of course, there are still many aspects about the CSO we can study. The initial parameters directly decide the flock structure and the update interval of flock relationship. And their changes will affect the overall performance of the CSO. Moreover, the dynamic penalty function selected in this paper could not deal with all the constrained optimization problems. For different constrained optimization problems, the pm-CSO needs different penalty factors and initial values of the objective function to prevent the algorithm to diverge in the iterative process.

References

[1] Akay, B. & Karaboga, D. 2012. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*. 23(4): 1001–1014.

[2] Fogel, D.B. 1995. A comparison of Evolutionary Programming and Genetic Algorithm on selected Constrained Optimization Problems. *Simulation*. 64(6): 397-404.

[3] Gandomi, A.H., Yang, X.S. & Alavi, A.H. 2013. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers*. 29(1): 17–35.

[4] Hock, W. & Schittkowski, K. 1981. *Test Examples for Nonlinear Programming Codes*. 187.

[5] Homaifar, A., Qi, C.X. & Lai, S.H. 1994. Constrained Optimization via Genetic Algorithms. *Simulation*. 2(4): 242-254.

[6] Meng, X.B., Liu, Y. & Gao, X.Z. et al. 2014. A new bio-inspired algorithm: chicken swarm optimization. *International Conference on Swarm Intelligence*. 8794: 86-94.

[7] Mezura, M.E. & Hernandez, O.B. 2009. Modified bacterial foraging optimization for engineering design. *Artificial Neural Networks in Engineering Conference*. 19: 357–364.

[8] Mi, Y.Q. & Gao, Y.L. 2015. The improved particle swarm optimization algorithm for solving constrained optimization Problems. *Journal of Jiangxi Normal University(Natural Science)*. 39(1): 59-63.

[9] Myung, H., Kim, J.H. & Fogel, D.B. 1995. *Preliminary Investigations into a Two-Stage Method of Evolutionary Optimization on Constrained Problems*: 449-463.

[10] Reynolds, R. & Ali, M. 2008. Embedding a social fabric component into cultural algorithms toolkit for an enhanced knowledge-driven engineering optimization. *International Journal of Intelligent Computing and Cybernetics*. 1(4): 563–597.

[11] Wang, P., Zhang, G.Y. & Liu, J.Y., et al. 2014. Topology Optimization Design for Inner Frame of Airborne Electro-optical Platform. *Journal of Mechanical Engineering.* 50(13): 135-141.

[12] Xiao, H.H. & Duan, Y.M. 2014. Research and application of improved bat algorithm based on DE algorithm. *Computer Simulation*. 31(1): 272-277.

[13] Yang, J.M., Chen, Y.P. & Horng, J.T., et al. 1997. Applying Family Competition to Evolution Strategies for Constrained Optimization. *Evolutionary Programming*. 1213: 201-211.

[14] Yang, X.S. 2010. A new metaheuristic bat-inspired algorithm. GONZALEZ J R et al. *Nature Inspired Cooperative Strategies for Optimization* (*NICSO* 2010): 65-74.