IMPROVED CUCKOO SEARCH ALGORITHM FOR NUMERICAL FUNCTION OPTIMIZATION

JIANJUN LIU, MIN ZENG AND YIFAN GE
School of Science, China University of Petroleum
Beijing 102249, China

CHANGZHI WU∗
School of Management
Guangzhou University, Guangzhou, 510006, China
and
Australasian Joint Research Centre for Building Information Modelling
School of Built Environment, Curtin University, 6102, WA, Australia

XIANGYU WANG
Australasian Joint Research Centre for Building Information Modelling
School of Built Environment, Curtin University, 6102, WA, Australia

(Communicated by Changjun Yu)

ABSTRACT. Cuckoo Search (CS) is a recently proposed metaheuristic algorithm to solve optimization problems. For improving its performance both on the efficiency of searching and the speed of convergence, we proposed an improved Cuckoo Search algorithm based on the teaching-learning strategy (TLCS). For a better balance between intensification and diversification, both a dynamic weight factor and an out-of-bound project strategies are also introduced into TLCS. The results of numerical experiment demonstrate that our improved TLCS performs better than the basic CS and other two improved CS methods appearing in literatures.

1. Introduction. For the unconstrained function optimization problem

$$\min f(x), \ x \in \mathbb{R}^n,$$

where $f(x)$ is continuous but may be not differentiable or multimodal. It is difficult to solve this problem by many gradient based methods.

In recent decades, more and more heuristic algorithms inspired by nature are emerging to solve the problem above. Most of those algorithms have been widely applied to solve the problems in various engineering and management areas, such as project scheduling [18, 19] and RFID network planning [26]. Here are some examples as genetic algorithm [6, 13], evolutionary algorithm [14], differential evolution [15, 1], particle swarm optimization [3], ant colony optimization [2], artificial bee colony [7], differential search [9, 10, 21], firefly algorithm [23, 8] and so on. All of these heuristic

2010 Mathematics Subject Classification. Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases. Cuckoo Search, teaching-learning strategy, dynamic weight factor, optimization.

∗ Corresponding author: Changzhi Wu.
This paper is partially supported by the National Natural Science Foundation of China (61473326) and Australian Research Council.
algorithms simulate the behaviors of biotic population. There are two important characteristics in these behaviors, one of them is the selection of the fittest and the other is the adaptation to the environment. Numerically speaking, they can be translated into two crucial characteristics, namely, intensification and diversification, in the population-based algorithm. The intensification strategy enables the algorithm to search around the current best solution and select the best candidate or solution whereas the diversification strategy ensures that the algorithm can efficiently explore the search space. In order to establish a better balance between the intensification and diversification, many hybrid or improved heuristic algorithms are proposed [12, 11].

A recently metaheuristic search algorithm, called Cuckoo Search (CS), has been developed by Yang and Deb [24]. This algorithm is inspired by the reproduction strategy of cuckoos. It requires fewer parameters and has been successfully applied to solve optimization problems. However, there are some drawbacks in CS such as that the algorithm is slowly convergent at the later iterations with low accuracy. Then, several improved or modified Cuckoo Search are developed. Dynamic weight CS algorithm (WCS) is presented in [28] where a dynamic weight local search technique is introduced to enhance the optimization ability and to improve the convergence speed of CS algorithm. A self-adaptive step adjustment Cuckoo search algorithm (ASCS) was presented which speeds up the convergence of CS algorithm and improves the calculation accuracy. Another CS optimization algorithm update the disturbance of particles by Gauss distribution instead of Lévy distribution [27]. These improved or modified strategies are helpful for enhancing the performance of CS. For more recent progress on CS, we can refer to both papers [22, 5] in which the recent advances and applications of CS are reviewed.

For the better performance for CS, this paper introduces three improvements,

1) updating solutions by a teaching-learning strategy with Lévy flight,
2) adjusting dynamically the weight factor,
3) introducing a novel out-of-bound control project into the basic CS.

These improvements can make the CS algorithm to overcome the “premature” and accelerate its convergence to the optimum.

The remainder of this paper is organized as follows: Section 2 describes the principle of the basic Cuckoo search algorithm. In Section 3, our improved CS algorithm is presented in detail. In Section 4, based on the numerical study, we analyze the comparison results on the performance of TLCS, basic CS, WCS and ASCS algorithms. Some conclusions are followed in Section 5.

2. Principle of CS algorithm. Cuckoo Search is a population-based algorithm which simulates the behaviors of cuckoo swarm. Cuckoo Search follows three idealized rules as below [22]:

1) Each cuckoo lays one egg at a time, and dumps its egg in the chosen nest randomly;
2) The best nests with high quality of eggs 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.

Each egg in a nest represents a solution while an egg be laying by a cuckoo represents a new solution. Namely, a cuckoo represents a new solution. The aim is to use the new and potentially better solutions to replace a not-so-good solution in the nests when CS is used to solve the problem (1). The update equation for a nest
is as follows

$$x_{i}^{t+1} = x_{i}^{t} + \alpha \oplus L(\lambda),$$ (2)

where $x_{i}^{t}$ is the solution, i.e., the $i-$th nest in generation $t$, and $x_{i}^{t}=(x_{i1}^{t}, x_{i2}^{t}, \ldots, x_{in}^{t})$.

$\alpha > 0$ is the step size which is related to the scale $n$ of the problem. In most cases, $\alpha = 1$. The product $\oplus$ means entry-wise multiplications. Lévy flight essentially provides a random walk, while this random step is drawn from a Lévy distribution $L(\lambda)$ for a large step.

$$L(\lambda) \sim \mu = t^{-\lambda}, \quad 1 \leq \lambda \leq 3,$$ (3)

which has an infinite variance with an infinite mean. Here, the consecutive steps (or jumps) of a cuckoo essentially come from a random walk process which obeys a power-law step-length distribution with a heavy tail.

In the basic CS algorithm, a Lévy flight strategy is used to generate a search step. However, this step size is unstable [4]. Generally, the larger the flight step is, the more extensive the search is. Nevertheless, a large step might lead to a reduction of the accuracy of solution. On the other hand, the smaller the step is, the slower the convergence rate becomes. Therefore, there exist such drawbacks as low accuracy and slow convergence in CS.

3. TLCS algorithm. In this section, we will improve the CS from three aspects, i.e., teaching-learning strategy, dynamic weight factor and out-of-bound project, to enhance its performance for function optimization.

3.1. Teaching-learning strategy (TL). Inspired by Teaching-Learning based optimization algorithm [16, 17], we introduce a teaching-learning strategy into CS.

The cuckoo swarm is divided into two parts, teacher cuckoos and learner cuckoos. The searching process of cuckoo swarm for optima simulates three modes of learning: teaching phase, learning phase and Lévy flight phase. The “teaching phase” means learning from the teacher, the “learning phase” means learning through the interaction among learners, and the solutions are eliminated out of the current population at “Lévy phase”. Through these three stages, teachers and learners are learning, and a teacher is generated from the learners with the best fitness value. The better the teacher is, the more knowledge the students can obtain.

3.1.1. Teaching phase. At generation $t$, the mean location of all learner cuckoos is denoted as $M^t$, which is represented by

$$M^t = (m_{1}^t, m_{2}^t, \ldots, m_{n}^t).$$ (4)

During the teaching phase, the teacher role is assigned to the best individual $x_{\text{teacher}}$. A teacher will put maximum effort into increasing the knowledge level of the whole swarm, while the learners will gain knowledge according to both the quality of teaching delivered by a teacher and the quality of learners. Considering this fact, the difference (indicted by $Df$) between the result of the teacher and the mean result of the learners is expressed by

$$Df = \gamma(x_{\text{teacher}} - TF \times M^t),$$ (5)

where $x_{\text{teacher}}$ is the level of teacher, $\gamma \in (0, 1)$ is a random number, and the teaching factor $TF$, that decides the value of mean to be changed, is randomly assigned a value of either 1 or 2 [17].
Hence, at the teaching phase the new set of improved learners can be generated by
\[ x_{\text{new}}^t(i) = \omega x_t^i + (Df + \alpha \oplus L(\lambda)), \] (6)
where \( x_{\text{new}}^t(i) \) is the new value of \( i \)-th learner after being updated, \( \omega \) is the weight factor. Generally, for achieving the global optimum it is helpful to encourage the individuals to well dispersed in the search space during the early stages of the search. So we set the weight factor \( \omega \) to adjust dynamically with the current iteration number \( t \) by
\[ \omega = \omega_{\text{max}} - \Delta \omega \cdot \frac{t}{T}, \] (7)
where \( \Delta \omega = \omega_{\text{max}} - \omega_{\text{min}} \). Generally, \( \omega_{\text{max}} = 0.9 \), \( \omega_{\text{min}} = 0.1 \) and they are the maximum and minimum values of weight factor, respectively. \( T \) is the maximum number of allowable iteration. Therefore, this dynamic weight factor might cause the algorithm to exploit the solutions within the interior of a relatively small space in which the potential global optimum locates at the later searching stages, whereas to explore ones in a large space at the earlier stages.

3.1.2. Learning Phase. This phase of the algorithm simulates the learning of the learners through interaction among themselves. A learner will learn new knowledge from the others who are more knowledgable.

The learning phenomenon in this phase is expressed below. After two learners \( i \) and \( j \), \( (i \neq j) \), are randomly selected, the \( i \)-th element \( x_{\text{new}}^t(i) \) in the learning phase is given by
\[ x_{\text{new}}^t(i) = \begin{cases} \omega x_t^i + r_1(x_t^i - x_t^j), & \text{if } f(x_t^i) < f(x_t^j), \\ \omega x_t^i + r_2(x_t^j - x_t^i), & \text{otherwise} \end{cases} \] (8)
where the value of the weight factor \( \omega \) is assigned by (7), \( r_1 \) and \( r_2 \) are two random constants within interval \((0, 1)\).

3.1.3. Lévy flight phase. In this phase, more better solutions will be achieved according to a specified Lévy flight and updated by using the following formula which is similar to (2).
\[ x_{\text{new}}^t(i) = x_t^i + \beta \otimes Q \] (9)
where \( \beta \in \mathbb{R}^n \) is a randomly vector whose elements are bounded in \([lb, ub]\), \( \otimes \) indicates the product of the corresponding elements, and \( Q = (q_1, q_2, \cdots, q_n) \) with
\[ q_i = \begin{cases} 1, & \text{if } r > p_\alpha, \\ 0, & \text{otherwise} \end{cases} (i = 1, 2, \cdots, n), \] (10)
where \( r \in (0, 1) \) is a random number and \( p_\alpha \) is a constant that equals to 0.25 [24].

3.2. Out-of-bound project. During search process, the individual may get into trouble of sticking at the boundary of the search space or out of the boundary. In order to deal with this case we present a project by which the individual out of the boundary will be projected into the search space.

The project is defined as follows:
\[ x_t^i = \begin{cases} 2lb_i - x_t^i, & \text{if } x_t^i < lb_i, \\ 2ub_i - x_t^i, & \text{if } x_t^i > ub_i, \end{cases} \] (11)
Figure 1. Project plot. $x_1, x_2, x_3$ are transformed to $y_1, y_2, y_3$, respectively, by (11) while $x_4$ is first transformed to $y_4$ by (11), then to $y'_4$ by (12).

and

$$x'_i = \begin{cases} \text{lb}_i + r_3(\text{ub}_i - \text{lb}_i), & \text{if } x'_i > \text{ub}_i, \\ \text{lb}_i + r_4(\text{ub}_i - \text{lb}_i), & \text{if } x'_i < \text{lb}_i \end{cases} \quad (12)$$

where $i = 1, 2, \cdots, n$, and $r_3, r_4$ are two random constants within interval $(0, 1)$.

This project devotes to transforming a solution out of the boundary into the search space by (11). If this solution is still not in the search region after transformed by (11), then it is transformed again by (12). This project can ensure that all the solutions are in the search space. We illustrate them by Figure 1 in two dimensional rectangular region. Especially, if there are a large number of new solutions out of the search space but near to boundary, the optimal solution might be near to the boundary. It can keep the transformed solution $y_1$ or $y_3$ still near to the boundary to transform solution $x_1$ or $x_3$ by (11).

Based on the description above, we state the detail procedure of TLCS by Algorithm 1.

4. Experiment, comparison and discussion. In order to demonstrate the performance of our improved algorithm TLCS, 18 benchmark problems will be tested and the numerical comparisons with CS, ASCS and WCS are made. All programs are coded in Matlab R2013a (win64) and executed in a PC computer with Intel(R) Core(TM) i3-380M, double CPU @ 2.53 GHz under Windows 7 operation system.

4.1. Properties of the benchmark functions. All 18 test problems are selected from [12] and listed in Table 1. The objective functions of these problems consist of many different kinds of properties such as unimodal, multimodal, regular, irregular, separable, non-separable and multidimensional. Specifically, there are 4 non-separable and 3 multimodal functions, respectively. Their names, characteristics, ranges, formulas, and optimal function values are also offered in Table 1.

4.2. Parameter settings. All 18 functions of two different dimensions, 5 and 10, are tested. The maximum number of iterations $T = 500$ and the population size is set to 20 for all test functions of different dimensions. Set other three parameters $\omega_{\text{max}} = 0.9$, $\omega_{\text{min}} = 0.1$ and $P_\alpha = 0.25$. 
Algorithm 1 Procedure of TLCS algorithm

Initialization.
Preset the number of population, $N$; the maximum iteration number, $T$; the probability parameter, $p_\alpha$; the teaching factor, $TF$; the range of weight factors $[\omega_{min}, \omega_{max}]$.
Initialize the population by the uniform distribution;
Generate nests that correspond to the solutions in the search space.
repeat
Compute fitness, $f(x^t_i)$, $i = 1, 2, \cdots, N$, of all the nests.
Evaluate nests based on their fitness, select the best.
for $i = 1$; $i < N$; $i++$ do
  //Teaching-learning Phase.
  Update $x'^t_{new(i)}$ by Eqs.(6), (11) and (12).
  if $f(x'^t_{new(i)}) < f(x^t_i)$ then
    $x^t_i = x'^t_{new(i)}$.
  end if
  //Learning-learning Phase.
  Randomly select $i, j (i \neq j)$ and then update $x'^t_{new(i)}$ by Eqs.(8), (11) and (12).
  if $f(x'^t_{new(i)}) < f(x^t_i)$ then
    $x^t_i = x'^t_{new(i)}$.
  end if
  //Lévy flight Phase.
  if a random number $r \in (0, 1) > p_\alpha$ then
    Update $x^t_i$ by Eqs.(9), (11) and (12).
  end if
end for
until Termination criteria are met

4.3. Results and discussion. The best, mean and standard deviation for each test function of 5 and 10 dimensions achieved by different algorithms are listed in Table 2 and Table 3, respectively. To attract attention, the best numerical results achieved by our improved algorithm are marked in bold font in both tables.

As seen from both tables, the performance of TLCS is better than CS, ASCS in the exploitation for most functions. For 5 dimensions benchmarks, TLCS, WCS can find the much better solutions than others for all the functions, and TLCS gets better solutions for $f_5, f_6, f_7, f_{13}, f_{15}, f_{18}$. For 10 dimensions benchmarks, TLCS also gets better solutions for $f_5, f_7, f_{15}, f_{18}$. Although all these four algorithms fail to find the optimal solution for $f_{11}$, the values for $f_{11}$ achieved by TLCS is slightly better than the other three algorithms. It is because that Schwefel function is deceptive in that the global minimum is geometrically distant, over the search space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong point.

The convergence curves of four algorithms for 1-8 benchmarks with 5 dimensions are plotted in Figure 2, respectively. Figure 2(a)-(h) graphically present the comparison on convergence characteristics between four algorithms over some a run for eight functions of 5 dimensions with the maximum number of iterations $T = 500$. 
As illustrated in Figure 2, TLCS always keeps convergent faster than the other algorithms in all cases. Therefore, the teaching-learning strategy does have significant effects on the performance of algorithm. And the dynamic weight factors can keep a faster convergence for our improved algorithm.

| Function Name | C Range | Formula | Opt. |
|---------------|---------|---------|------|
| Sphere S.U.   | [-100,100] | \[f_1(x) = \frac{1}{3} \sum_{i=1}^{n} x_i^3\] | 0 |
| Ackley S.M.   | [-32,32]  | \[f_2(x) = 20 - 20 \cos \left( \frac{1}{2} \sum_{i=1}^{n} x_i^2 \right) - e^{-\frac{1}{4} \sum_{i=1}^{n} \cos \left( 2 \pi x_i \right)} + e\] | 0 |
| Sum-Squares S.U. | [-10,10 ] | \[f_3(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2\] | 0 |
| Rastrigin S.M. | [-5,12,5,12] | \[f_4(x) = 10n + \sum_{i=1}^{n} \left[ x_i^2 - 10 \cos \left( 2 \pi x_i \right) \right]\] | 0 |
| Rosenbrock N.U. | [-10,10]  | \[f_5(x) = \frac{1}{n} \sum_{i=1}^{n-1} \left[ 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right]\] | 0 |
| Griewank N.M. | [-600,600] | \[f_6(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right)\] | 0 |
| Levy N.M.    | [-10,10]  | \[f_7(x) = \sin(\pi x_i) + \sum_{i=1}^{n} \left[ \left( w_i x_i \right)^2 - 10 \cos \left( \pi w_i x_i \right) + 1 \right]\] | 0 |
| Weierstrass N.M. | [-0.5,0.5] | \[f_{10}(x) = \sum_{i,j=1}^{n} \left( \sum_{k=0}^{n} a^k \cos \left( 2 \pi k \right) (x_i - 0.5) \right)\] | 0 |
| Schwefel S.M. | [-500,500] | \[f_1(x) = 418.982887 \cdot n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})\] | 0 |
| Whitley N.M. | [-10,2,10,2] | \[f_{12}(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \sum_{k=0}^{n} a^k \cos \left( 2 \pi k \right) (x_i - 0.5) \right)\] | 0 |
| Trid N. [-n^2, n^2] | \[f_{13}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=1}^{n} x_i x_{i-1}, (n = 5)\] | -30 |
| Styblinski-Tang S.M. | [-5,5] | \[f_{14}(x) = \frac{1}{n} \sum_{i=1}^{n} \left( x_i^4 - 16 x_i^2 + 5 x_i \right), a = 39.16599\] | -nn |
| Alpine N.M. | [-10,10]  | \[f_{15}(x) = \sum_{i=1}^{n} x_i \sin(x_i) + 0.1 \cdot x_i\] | 0 |
| Easom S.U.   | [-100,100] | \[f_{16}(x) = -\cos(\pi x_1) \cos(\pi x_2) e^{-\pi(x_1^2 + x_2^2)} - (x_1^2 + x_2^2)\] | -1 |
| Treccani S.M. | [-3,3]  | \[f_{17}(x) = \left( \frac{x_1}{4} \right)^4 + 4 \left( \frac{x_1}{4} \right)^2 + \left( \frac{x_2}{2} \right)^2\] | 0 |
| Perm(n, β) N.U. | [-n,n]  | \[f_{18}(x) = \sum_{i=1}^{n} \left( \sum_{k=1}^{n} \beta \right) \left( x_i^k - \frac{1}{x_i} \right)^2\] | 0 |

As illustrated in Figure 2, TLCS always keeps convergent faster than the other algorithms in all cases. Therefore, the teaching-learning strategy does have significant effects on the performance of algorithm. And the dynamic weight factors can keep a faster convergence for our improved algorithm.
(Continued Table 2)

| Func | Indicator | CS     | ASCS   | WCS    | TLCS   |
|------|-----------|--------|--------|--------|--------|
|      | Std.      | 4.3639e-04 | 6.6652e-04 | 0.0000e+00 | 0.0000e+00 |
| $f_3$ | Best      | 9.9825e-16 | 5.7419e-19 | 0.0000e+00 | 0.0000e+00 |
|      | Mean      | 3.5974e-14 | 1.8282e-17 | 0.0000e+00 | 0.0000e+00 |
|      | Std.      | 4.4513e-14 | 2.9784e-17 | 0.0000e+00 | 0.0000e+00 |
|      | $f_4$      | Best      | 9.9825e-16 | 5.7419e-19 | 0.0000e+00 | 0.0000e+00 |
|      | Mean      | 3.5974e-14 | 1.8282e-17 | 0.0000e+00 | 0.0000e+00 |
|      | Std.      | 4.4513e-14 | 2.9784e-17 | 0.0000e+00 | 0.0000e+00 |
|      | $f_5$      | Best      | 5.4423e-04 | 3.8454e-03 | 9.0786e-05 | 3.2173e-06 |
|      | Mean      | 4.9548e-02 | 8.8289e-02 | 4.9515e-03 | 9.0026e-05 |
|      | Std.      | 8.2955e-02 | 1.2769e-01 | 4.4433e-03 | 2.4935e-04 |
|      | $f_6$      | Best      | 2.1760e-02 | 6.0844e-02 | 0.0000e+00 | 0.0000e+00 |
|      | Mean      | 4.1580e-01 | 6.0469e-01 | 0.0000e+00 | 0.0000e+00 |
|      | Std.      | 4.1771e-01 | 4.3998e-01 | 0.0000e+00 | 0.0000e+00 |
|      | $f_7$      | Best      | 7.4053e-12 | 4.1549e-11 | 1.4224e-13 | 6.0523e-17 |
|      | Mean      | 4.5672e-10 | 5.7734e-10 | 4.9219e-11 | 1.3148e-15 |
|      | Std.      | 1.2514e-09 | 5.2112e-10 | 4.1786e-11 | 2.0139e-15 |
|      | $f_8$      | Best      | 7.5663e-19 | 1.9396e-20 | 0.0000e+00 | 5.2592e-51 |
|      | Mean      | 3.2565e-17 | 1.2424e-17 | 0.0000e+00 | 1.0280e-44 |
|      | Std.      | 8.2955e-02 | 1.2769e-01 | 4.4433e-03 | 2.4935e-04 |
|      | $f_9$      | Best      | 2.7592e-13 | 1.2952e-16 | 0.0000e+00 | 0.0000e+00 |
|      | Mean      | 2.6490e-12 | 1.6909e-15 | 0.0000e+00 | 0.0000e+00 |
|      | Std.      | 2.9165e-12 | 1.4999e-15 | 0.0000e+00 | 0.0000e+00 |
|      | $f_{10}$   | Best      | 5.5995e-03 | 6.6042e-03 | 0.0000e+00 | 0.0000e+00 |
|      | Mean      | 1.1684e-02 | 1.0426e-02 | 0.0000e+00 | 0.0000e+00 |
|      | Std.      | 6.3211e-03 | 3.5660e-03 | 0.0000e+00 | 0.0000e+00 |
|      | $f_{11}$   | Best      | 2.0752e+03 | 2.0752e+03 | 2.0752e+03 | 2.0616e+03 |
|      | Mean      | 2.0752e+03 | 2.0752e+03 | 2.0752e+03 | 2.0616e+03 |
|      | Std.      | 4.7941e-11 | 1.7782e-10 | 1.5453e-09 | 2.2077e+01 |
|      | $f_{12}$   | Best      | 1.1195e-03 | 1.3950e-03 | 1.1187e-02 | 9.8810e-13 |
|      | Mean      | 7.2247e-01 | 5.6108e-01 | 1.5719e-00 | 5.6092e+00 |
|      | Std.      | 9.9970e-01 | 8.9282e-01 | 1.2832e-00 | 4.2152e+00 |
|      | $f_{13}$   | Best      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | Mean      | 7.8332e+01 | 7.8332e+01 | 7.8332e+01 | 7.8332e+01 |
|      | Std.      | 5.0220e-07 | 8.9447e-07 | 1.2933e-06 | 2.3413e+00 |
|      | $f_{14}$   | Best      | 1.7330e+00 | 3.1330e-03 | 1.1926e+18 | 0.0000e+00 |
|      | Mean      | 8.1827e-03 | 1.4765e-02 | 3.9923e-17 | 0.0000e+00 |
|      | Std.      | 4.3095e-03 | 1.2199e-02 | 0.0000e+00 | 0.0000e+00 |
|      | $f_{15}$   | Best      | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 |
|      | Mean      | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 |
|      | Std.      | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 | 1.0000e+00 |
|      | $f_{16}$   | Best      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | Mean      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | Std.      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | $f_{17}$   | Best      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | Mean      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | Std.      | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 | 3.0000e+00 |
|      | $f_{18}$   | Best      | 5.0220e-07 | 8.9447e-07 | 1.2933e-06 | 2.3413e+00 |
|      | Mean      | 6.0659e-02 | 6.3264e-02 | 1.7993e-05 | 4.1372e-12 |
|      | Std.      | 1.2079e-01 | 1.0323e-01 | 1.8082e-05 | 4.4431e-12 |
Figure 2. Convergence performance of CS, ASCS, WCS and TLCS, on 8 test functions of 5 dimensions. Horizontal and vertical axes represent the number of iterations and the logarithm of the function values, respectively.
Table 3: Statistic results obtained by CS [24], ASCS [25], WCS [28] and TLCS on 10-D functions over 30 independent runs, respectively.

| Fun. | Indicator | CS       | ASCS     | WCS      | TLCS     |
|------|-----------|----------|----------|----------|----------|
| \(f_1\) | Best       | 3.2389e-06 | 1.1207e-08 | 0.0000e+00 | 0.0000e+00 |
|      | Mean       | 8.0057e-06 | 5.3375e-08 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 4.6479e-06 | 2.9549e-08 | 0.0000e+00 | 0.0000e+00 |
| \(f_2\) | Best       | 3.7125e-02 | 8.7783e-02 | 8.8818e-16 | 8.8818e-16 |
|      | Mean       | 6.0272e-01 | 4.0848e-01 | 8.8818e-16 | 8.8818e-16 |
|      | Std.       | 5.8084e-01 | 2.2043e-01 | 0.0000e+00 | 0.0000e+00 |
| \(f_3\) | Best       | 5.2596e-07 | 2.7587e-09 | 0.0000e+00 | 0.0000e+00 |
|      | Mean       | 1.1568e+00 | 1.5915e+00 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 2.4106e+00 | 4.5746e+00 | 0.0000e+00 | 0.0000e+00 |
| \(f_4\) | Best       | 6.0868e+00 | 9.9616e+00 | 0.0000e+00 | 0.0000e+00 |
|      | Mean       | 1.1568e+01 | 1.5915e+01 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 2.4106e+00 | 4.5746e+00 | 0.0000e+00 | 0.0000e+00 |
| \(f_5\) | Best       | 5.5106e-01 | 1.1646e+00 | 2.7452e+00 | 2.2219e+00 |
|      | Mean       | 4.3244e+00 | 3.2423e+00 | 3.6089e+00 | 2.7140e+00 |
|      | Std.       | 1.4901e+00 | 1.5008e+00 | 3.7702e-01 | 3.3872e-01 |
| \(f_6\) | Best       | 5.4516e-02 | 4.9131e-02 | 0.0000e+00 | 0.0000e+00 |
|      | Mean       | 7.6544e-02 | 9.0650e-02 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 1.4901e+00 | 1.5008e+00 | 3.7702e-01 | 3.3872e-01 |
| \(f_7\) | Best       | 1.6338e-04 | 3.3752e-03 | 4.5958e-06 | 9.2416e-10 |
|      | Mean       | 1.0474e-02 | 2.5951e-02 | 1.2202e-04 | 1.7004e-08 |
|      | Std.       | 1.0911e-02 | 1.8264e-02 | 1.1195e-04 | 1.4461e-08 |
| \(f_8\) | Best       | 6.3186e-09 | 1.5939e-09 | 0.0000e+00 | 2.1348e-36 |
|      | Mean       | 4.4004e-08 | 9.4253e-09 | 0.0000e+00 | 7.5924e-26 |
|      | Std.       | 7.6544e-09 | 9.0650e-09 | 0.0000e+00 | 2.3323e-25 |
| \(f_9\) | Best       | 8.7140e-04 | 3.7275e-03 | 4.5958e-06 | 9.2416e-10 |
|      | Mean       | 4.8839e-03 | 8.7991e-05 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 3.2252e-03 | 4.7437e-05 | 0.0000e+00 | 0.0000e+00 |
| \(f_{10}\) | Best      | 4.0273e-01 | 4.6694e-01 | 0.0000e+00 | 0.0000e+00 |
|      | Mean       | 7.8045e-01 | 8.4527e-01 | 0.0000e+00 | 0.0000e+00 |
|      | Std.       | 2.4665e-01 | 2.5778e-01 | 0.0000e+00 | 0.0000e+00 |
| \(f_{11}\) | Best      | 4.1504e+03 | 4.1504e+03 | 4.1504e+03 | 4.0458e+03 |
|      | Mean       | 4.1504e+03 | 4.1504e+03 | 4.1504e+03 | 4.1152e+03 |
|      | Std.       | 1.1924e-04 | 6.4787e-04 | 3.9471e+01 | 3.9471e+01 |
| \(f_{12}\) | Best      | 2.7106e+01 | 3.3064e+01 | 2.4671e+01 | 4.0486e+01 |
|      | Mean       | 4.2321e+01 | 4.4884e+01 | 3.8730e+01 | 4.4565e+01 |
|      | Std.       | 6.7034e+00 | 6.0260e+00 | 4.9097e+00 | 4.9097e+00 |
| \(f_{13}\) | Best      | -1.2467e+02 | -1.2467e+02 | -1.2467e+02 | -1.7996e+02 |
|      | Mean       | -1.2467e+02 | -1.2467e+02 | -1.2467e+02 | -1.6700e+02 |
|      | Std.       | 6.0730e+00 | 5.1284e+00 | 4.6884e-08 | 4.6884e-08 |
| \(f_{14}\) | Best      | -7.8281e+01 | -7.8225e+02 | -7.8051e+02 | -1.7996e+02 |
|      | Mean       | -7.8281e+01 | -7.8225e+02 | -7.8051e+02 | -1.7996e+02 |
|      | Std.       | 3.5470e+00 | 1.4800e+00 | 1.564e+00 | 1.564e+00 |
| \(f_{15}\) | Best      | 2.2441e+01 | 5.7034e+01 | 2.5015e-18 | 0.0000e+00 |
|      | Mean       | 3.8266e+01 | 1.1298e+00 | 4.8905e-19 | 0.0000e+00 |
|      | Std.       | 1.0949e+01 | 4.0051e-01 | 0.0000e+00 | 0.0000e+00 |
| \(f_{16}\) | Best      | -1.0000e+00 | -1.0000e+00 | -1.0000e+00 | -1.0000e+00 |
|      | Mean       | -1.0000e+00 | -1.0000e+00 | -1.0000e+00 | -1.0000e+00 |
|      | Std.       | 9.4746e+13 | 6.1618e-13 | 0.0000e+00 | 0.0000e+00 |
| \(f_{17}\) | Best      | -3.5527e-15 | -3.5527e-15 | 0.0000e+00 | 0.0000e+00 |
Furthermore, we apply statistical significance testing, which is a meaningful way to study the difference between any two stochastic algorithms, to do comparison among four algorithms. A non-parametric test, Wilcoxon rank sum test, is chosen to judge the difference between paired scores. The statistical results based on the best and average function values are presented in Table 4, where rank sum is the sum of ranks based on the absolute value of the difference between two test algorithms. It returns a logical h-value indicating the test decision.

Table 4. Wilcoxon rank sum results on best and average function values obtained by TLCS against CS, ASCS and WCS algorithms for 18 benchmark functions.

| Algorithm  | Dim. | criteria | rank sum | p value | h value |
|------------|------|----------|----------|---------|---------|
| TLCS to CS | 5    | Best     | 421.5000 | 0.051   | 1       |
|            |      | Mean     | 411.0000 | 0.0139  | 1       |
|            |      | Std      | 422.0000 | 0.0048  | 1       |
| TLCS to ASCS | 5 | Best     | 415.0000 | 0.0095  | 1       |
|            |      | Mean     | 406.0000 | 0.0213  | 1       |
|            |      | Std      | 408.5000 | 0.0165  | 1       |
| TLCS to WCS | 5 | Best     | 357.0000 | 0.4363  | 0       |
|            |      | Mean     | 356.5000 | 0.4532  | 0       |
|            |      | Std      | 333.0000 | 1.0000  | 0       |
| TLCS to CS | 10   | Best     | 410.5000 | 0.0143  | 1       |
|            |      | Mean     | 407.0000 | 0.0196  | 1       |
|            |      | Std      | 405.5000 | 0.0213  | 1       |
| TLCS to ASCS | 10 | Best     | 408.5000 | 0.0170  | 1       |
|            |      | Mean     | 406.0000 | 0.0213  | 1       |
|            |      | Std      | 407.5000 | 0.0180  | 1       |
| TLCS to WCS | 10 | Best     | 353.0000 | 0.5183  | 0       |
|            |      | Mean     | 348.0000 | 0.6339  | 0       |
|            |      | Std      | 324.5000 | 0.7810  | 0       |

As a null hypothesis, there is no significant difference between the best/or mean values of two samples. Whereas the alternative hypothesis it is assumed that there is a significant difference in the best and/or mean fitness values of the two samples, at the 5% significance level. We use the function rank sum for the computation of the h-value for these tests, where the number “1” means that the first algorithm is significantly better than the second, and the “0” means that there is no significant difference between both algorithms. From Table 4, TLCS is clearly superior to CS, ASCS on 5D and 10D functions with regard to the best solutions and is equivalent to WCS.

5. Conclusion. In this paper, we introduce three strategies into the Cuckoo Search algorithm and then propose a TLCS algorithm. The TLCS combines the Lévy flight with dynamic weight teaching-learning strategy to establish a better balance between exploitation and exploration in the process of searching optima. In addition, a dynamic weight factor and a out-of-bound project strategy are helpful
to enhance the performance of TLCS. The proposed algorithm has been validated and compared with other improved CS algorithms appearing in recent literatures. Simulations and comparison show that TLCS is superior to others for those multimodal or non-separable objective functions. This is partly due to the fact that three strategies mentioned above are effective in TLCS. This potentially powerful optimization strategy can easily be extended to study constrained optimization or multiobjective optimization.

REFERENCES

[1] S. Das and P. Suganthan, Differential evolution, A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation, 15 (2011), 4–31.
[2] M. Dorigo and C. Blum, Ant colony optimization theory, A survey, Theoretical Computer Science, 344 (2005), 243–278.
[3] R. Eberhart and Y. Shi, Particle swarm optimization, Development, applications and resources, In Proceedings of the 2001 Congress on Evolutionary Computation, 1 (2001), 81–86.
[4] E. Valian, S. Mohanna and S. Tavakoli, Improved cuckoo search algorithm for global optimization, International Journal of Communications and Information Technology, 1 (2011), 31–44.
[5] I. Fister, X. Yang and D. Fister, Cuckoo search, A brief literature review, Studies in Computational Intelligence, 516 (2014), 49–62.
[6] D. Goldberg, Genetic algorithms and machine learning, Machine Learning, 3 (1988), 95–99.
[7] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization, artificial bee colony (ABC) algorithm, Journal of Global Optimization, 39 (2007), 459–471.
[8] J. Liu, S. Dong, L. Zhang, Q. Ma and C. Wu, Estimation of archie parameters by a novel hybrid optimization algorithm, Journal of Petroleum Science and Engineering, 135 (2015), 232–239.
[9] J. Liu, K. Teo, X. Wang and C. Wu, An exact penalty function-based differential search algorithm for constrained global optimization, Soft Computing, 20 (2015), 1305–1313.
[10] J. Liu, C. Wu, G. Wu and X. Wang, A novel differential search algorithm and applications for structure design, Applied Mathematics & Computation, 268 (2015), 246–269.
[11] J. Liu, S. Zhang, C. Wu, J. Liang, X. Wang and K. L. Teo, A hybrid approach to constrained global optimization, Applied Soft Computing, 47 (2016), 281–294.
[12] J. Liu, H. Zhu, Q. Ma, L. Zhang and H. Xu, An artificial bee colony algorithm with guide of global & local optimal and asynchronous scaling factors for numerical optimization, Applied Soft Computing, 37 (2015), 608–618.
[13] Q. Long and C. Wu, A hybrid method combining genetic algorithm and Hooke-Jeeves method for constrained global optimization, Journal of Industrial & Management Optimization, 10 (2014), 1279–1296.
[14] M. Platel, L. Defoin, S. Schliebs and N. Kasabov, Quantum-inspired evolutionary algorithm, a multimodel EDA, IEEE Transactions on Evolutionary Computation, 13 (2009), 1218–1232.
[15] A. Qin, V. Huang and P. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation, 13 (2009), 398–417.
[16] R. Rao, V. Savsani and D. Vakharia, Teaching-learning-based optimization, A novel method for constrained mechanical design optimization problems, Computer-aided Design, 43 (2011), 303–315.
[17] R. Rao, V. Savsani and D. Vakharia, Teaching-learning-based optimization, An optimization method for continuous non-linear large scale problems, Information Sciences, 183 (2012), 1–15.
[18] S. Tao, C. Wu, Z. Sheng and X. Wang, Space-time repetitive project scheduling considering location and congestion, Journal of Computing in Civil Engineering, 32 (2018), Article ID, 04018017.
[19] S. Tao, C. Wu, Z. Sheng and X. Wang, Stochastic project scheduling with hierarchical alternatives, Applied Mathematical Modelling, 58 (2018), 181–202.
[20] M. Tavazoei and M. Haeri, Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms, *Applied Mathematics and Computation*, 187 (2007), 1076–1085.

[21] Z. Wang, D. Gao and J. Liu, Multi-objective sidetracking horizontal well trajectory optimization in cluster wells based on DS algorithm, *Journal of Petroleum Science & Engineering*, 147 (2016), 771–778.

[22] X. Yang and S. Deb, Cuckoo search, Recent advances and applications, *Neural Computing and Applications*, 24 (2014), 169–174.

[23] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2010.

[24] X. S. Yang and S. Deb, Cuckoo search via lévy flights, In *World Congress on Nature & Biologically Inspired Computing*, (2009), 210–214.

[25] Y. Zhang, L. Wang and Q. Wu, Modified adaptive cuckoo search (MACS) algorithm and formal description for global optimisation, *International Journal of Computer Applications in Technology*, 44 (2012), 73–79.

[26] C. Zhao, C. Wu, J. Chai, X. Wang, X. Yang, J. Lee and M. Kim, Decomposition-based multi-objective firefly algorithm for RFID network planning with uncertainty, *Applied Soft Computing*, 55 (2011), 549–564.

[27] H. Zheng and Y. Zhou, A novel cuckoo search optimization algorithm base on Gauss distribution, *Journal of Computational Information Systems*, 8 (2012), 4193–4200.

[28] H. Zheng and Y. Zhou, Self-adaptive step cuckoo search algorithm, *Computer Engineering & Applications*, 49 (2013), 68–71.

Received February 2016; revised January 2018.

E-mail address: LiuJJ@cup.edu.cn
E-mail address: 1085053090@qq.com
E-mail address: 973200690@qq.com
E-mail address: C.Wu@exchange.curtin.edu.au
E-mail address: Xiangyu.Wang@curtin.edu.au