Opposition-based krill herd algorithm with Cauchy mutation and position clamping

Gai-Ge Wang a,b,c,*, Suash Deb d, Amir H. Gandomi e, Amir H. Alavi f

a School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China
b School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China
c School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China
d Department of Computer Science & Engineering, Cambridge Institute of Technology, Cambridge Village, Tatisilwai, Ranchi 835103, Jharkhand, India
e Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China
f Department of Civil and Environmental Engineering, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Article history:
Received 31 August 2015
Received in revised form 1 October 2015
Accepted 12 November 2015
Communicated by: X. Gao

Keywords:
Krill herd
Opposition-based learning
Cauchy mutation
Position clamping
Engineering optimization

ABSTRACT

Krill herd (KH) has been proven to be an efficient algorithm for function optimization. For some complex functions, this algorithm may have problems with convergence or being trapped in local minima. To cope with these issues, this paper presents an improved KH-based algorithm, called Opposition Krill Herd (OKH). The proposed approach utilizes opposition-based learning (OBL), position clamping (PC) and Cauchy mutation (CM) to enhance the performance of basic KH. OBL accelerates the convergence of the method while both PC and heavy-tailed CM help KH escape from local optima. Simulations are implemented on an array of benchmark functions and two engineering optimization problems. The results show that OKH has a good performance on majority of the considered functions and two engineering cases. The influence of each individual strategy (OBL, CM and PC) on KH is verified through 25 benchmarks. The results show that the KH with OBL, CM and PC operators, has the best performance among different variants of OKH.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Inspired by nature, a variety of modern intelligent algorithms have been developed and applied to solve optimization problems. Some of them, like cuckoo search (CS) [1–5], biogeography-based optimization (BBO) [6–13], artificial bee colony (ABC) [14,15], genetic algorithm (GA) [16], genetic programming (GP) [17], study GA (SGA) [18,19], differential evolution (DE) [20–25], ant lion optimizer (ALO) [26], chicken swarm optimization (CSO) [27], wolf search algorithm (WSA) [28], multi-verse optimizer (MVO) [29], earthworm optimization algorithm (EWA) [30], grey wolf optimizer (GWO) [31,32], firefly algorithm (FA) [33–35], dragonfly algorithm (DA) [36], harmony search (HS) [37–41], bird swarm algorithm (BSO) [42], moth–flame optimization (MFO) [43], animal migration optimization (AMO) [44], particle swarm optimization (PSO) [45–48], ant colony optimization (ACO) [49], bat algorithm (BA) [50–54], have solved several complicated challenging problems that are hard to deal with by traditional optimization techniques. Among these algorithms, krill herd (KH) method [55–57] has been studied extensively due to its promising performance for solving most complex problems. KH was first proposed by Gandomi and Alavi by the idealization of communicating and foraging behaviors of krill swarms [55]. KH performed well on various optimization problems [55]. However, in some cases, it might not be capable of escaping from local minima. In order to decrease the influence of this problem for KH, this paper proposes different variants of KH algorithm using opposition-based learning (OBL), position clamping (PC) and Cauchy mutation (CM). The main idea of OBL is to search for a better candidate solution through the simultaneous consideration of a solution and its opposite that is closer to the global optimum. OBL can successfully handle this task by updating the other half krill according to the opposite that is closer to the global optimum. Consequently, a faster convergence can be provided for the KH method. The heavy-tailed CM and PC help the krill not trap into the local optima. Through different experiments, the KH together with OBL, CM and PC operators performs the best among various OKHs. Experimental simulations on 25 benchmark functions and two engineering
optimization problems show that OKH performs well on the majority of benchmark functions and engineering problems.

The remainder of this paper is organized as follows. The next section introduces the main process of the basic KH and OBL theory. Section 3 proposes an improved OKH model by combination of KH, CM operator and PC operator. Then, in Section 4, a series of comparison experiments on various benchmarks and two engineering cases are conducted. The final section provides our concluding remarks and points out our future work orientation.

2. Preliminary

2.1. Krill herd

In computer science, KH [55] is a probabilistic technique for solving computational problems. It is a kind of swarm intelligence algorithms that take advantage of the evolving behaviors of krill individuals. It is based on the idealization of the krill swarms when hunting for food and communicating with each other. The KH method repeats the implementation of the three movements and takes search directions that proceed to the best solution. The behavior of krill is idealized into three actions as:

i. movement influenced by other krill;
ii. foraging action;
iii. physical diffusion.

KH algorithm adopted the following Lagrangian model as Eq. (1).

\[
\frac{dX_i}{dt} = N_i + F_i + D_i
\]  

(1)

where \( N_i \) is the motion induced by other krill; \( F_i \) is the foraging motion, and \( D_i \) is the physical diffusion.

For the first motion, its motion direction, \( \alpha_i \), is primarily decided by the target effect, a local effect, and a repulsive effect. For a krill, it can be given as:

\[
N_i^{\text{new}} = N_i^{\text{max}} \alpha_i + \omega_n N_i^{\text{old}}
\]  

(2)

and \( N_i^{\text{max}} \) is the maximum induced speed, \( \omega_n \) is the inertia weight, \( N_i^{\text{old}} \) is the last motion.

The second motion is mainly decided by the two factors: the food location and its previous experience. For the ith krill, it can be defined as:

\[
F_i = V_i \beta_i + \omega_V F_i^{\text{old}}
\]  

(3)

where

\[
\beta_i = \beta_i^{\text{food}} + \beta_i^{\text{best}}
\]  

(4)

and \( V_i \) is the foraging speed, \( \omega_V \) is the inertia weight for the second motion, \( F_i^{\text{old}} \) is the last motion, \( \beta_i^{\text{food}} \) is the food attractive and \( \beta_i^{\text{best}} \) is the effect of the best fitness of the ith krill so far.

The third motion is essentially a random process. It contains two parts: a maximum diffusion speed and a random directional vector. It can be expressed as follows:

\[
D_i = D_{\text{max}} \delta
\]  

(5)

where \( D_{\text{max}} \) is the maximum diffusion speed, and \( \delta \) is the random vector.

Based on the three above-mentioned movements, the position of a krill from \( t \) to \( t + \Delta t \) is given as:

\[
X_i(t + \Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt}
\]  

(6)

In standard KH, the process of movement influenced by other krill, foraging action and physical diffusion continue for a fixed number of generations or until a termination Condition is satisfied. More detailed description about KH algorithm can be referred as [55].

2.2. Opposition-based learning

OBL [58,59] is a relatively novel technology in optimization field. It has been shown to be a promising approach to improve the performance of many metaheuristic algorithms, such as ACO [60], DE [61–63], population-based incremental learning (PBIL) [64,65], PSO [66] and ABC [67]. Evidently, a simultaneous evaluation of a solution and its opposite can increases the chance of finding an individual close to the global best solution. The definition of the Opposite number (1-dimensional) and opposite point (D-dimensional) are given below.

Opposite number [66]: let \( x \in [a, b] \) be a real number. Its opposite is defined by:

\[
x^* = a + b - x
\]  

(7)

Opposite point [66]: let \( X = (x_1, x_2, \ldots, x_D) \) be a point in a D-dimensional space, where \( x_1, x_2, \ldots, x_D \in R \) and \( x_i \in [a, b] \), \( i \in 1, 2, \ldots, D \). The definition of the opposite point \( X^* = (x_1', x_2', \ldots, x_D') \) is as follows:

\[
x_i' = a_i + b_i - x_i
\]  

(8)

The definition of opposition-based optimization can be given in terms of opposite point as follows.

Opposition-based optimization [66]: let \( X = (x_1, x_2, \ldots, x_D) \) be a candidate solution in search space, and its fitness function is \( f(X) \). As per Eq. (8), \( X^* = (x_1', x_2', \ldots, x_D') \)is the opposite of \( X = (x_1, x_2, \ldots, x_D) \). If \( f(X^*) < f(X) \), then update \( X \) with \( X^* \); otherwise \( X \) is unchanged. Therefore, simultaneously computing and evaluating the current point and its opposite makes the population proceed to the best solution.

3. Improving KH using OBL

This study is aimed at improving the performance of KH by combining it with OBL, CM, and PC operators. The OBL method forces krill to move toward the best solutions, while CM and PC operators are well capable of adding the diversity of the population. These two operators also provide an effective balance between exploration and exploitation.

3.1. The Opposition krill herd method

In the improved OKH method, the OBL idea is combined with the traditional KH in order to improve its search ability. The main steps of the OKH method can be given below.

In the first step, the first half population (\( P \)) is generated by a random distribution. The rest half population (\( OP \)) is initialized as per the first half population (\( P \)) in terms of OBL as shown in Section 2.2.

After initialization, for the first half population (\( P \)), it is subject to update the position of the krill as shown in Section 2.1. Simultaneously, for the rest half population (\( OP \)), it is subjected to update the position of the krill as per the first half population (\( P \)) as shown in Section 2.2.

After the updating of the solutions in the population, the two subpopulations are composed of one population. This operation can make the population size unchanged in all the optimization process. Furthermore, we can sort the population in terms of their fitness and locate the best individual for this generation. The procedure is then iterated.
3.2. The Cauchy mutation operator

Some experiments have proven that the krill in KH will change slightly between around the global best krill before it get final best solution \[57\]. It is possible to improve the search ability of krill by the addition of the neighbors of the global best krill in each generation. This can also help the whole krill individuals proceed to the better positions. The implementation of a CM on the global best krill results in reaching this goal. CM has been successfully combined with many metaheuristic algorithms, such as PSO \[68,69\], DE \[70,71\]. The 1-D Cauchy density function is defined by:

\[
f(x) = \frac{1}{\pi \tau^2} \frac{1}{1 + (x/\tau)^2}, \quad x \in \mathbb{R}
\]

where \(\tau > 0\) is a scale parameter \[68\]. The Cauchy distributed function is

\[
F_c(x) = \frac{1}{\pi} \arctan \left( \frac{x}{\tau} \right)
\]

(10)

This mutation operator is well capable of decreasing the possibility of trapping into a local optimum. The description of the CM operator in OKH is given below:

\[
W(j) = \left( \sum_{i=1}^{NP} x_{ij} \right) / NP
\]

(11)

where \(x_{ij}\) is the \(j^{th}\) position vector of the \(i^{th}\) krill, \(NP\) is the population size, \(W(j)\) is a weight vector.

\[
x'(j) = x(j) + W(j) \cdot C
\]

(12)

where \(C\) is a random number drawn from a Cauchy distribution with \(\tau = 1\).

3.3. The position clamping operator

In order to make good balance between exploration and exploitation, the PC operator is introduced into KH method. The clamping technique has been successfully together with several metaheuristic algorithms (PSO \[72\]).

In basic KH, the values of positions become very large, especially for the krill distant from the best positions. Accordingly, some krill may exceed the boundaries of the search space. In order to get over this problem, positions of krill are clamped to limit the global search. If a krill moves out of the allowed position, it is set to the maximum position \(X_{\text{max}}\). Here, \(X_{\text{max}}\) and \(X_{\text{min}}\) represents the maximum and minimum positions of krill in \(j^{th}\) dimension. The position of krill is updated using Eq. (13).

\[
X_{ij}(t+1) = \begin{cases} X_{ij}(t+1), & \text{if } X_{ij}(t+1) < X_{\text{max}} \quad j \\ X_{\text{max}}, & \text{otherwise} \end{cases}
\]

(13)

In OKH, exploration and exploitation are adjusted by adjusting the value of \(X_{\text{max}}\), and the maximum and minimum positions are initialized by Eqs. (14) and (15).

\[
X_{\text{max}} = \delta_1 (X_{\text{max}} - X_{\text{min}})
\]

(14)

\[
X_{\text{min}} = \delta_2 (X_{\text{min}} - X_{\text{max}})
\]

(15)

where \(\delta_1\) and \(\delta_2\) are constant factors and are respectively taken 0.4 and 0.6 in our OKH solution.

By combination of KH method and OBL, CM and PC operator, the detailed steps of the OKH algorithm are as follows:

Step 1: Initialization

The first half population \((P)\) including \(NP/2\) individuals is generated by a random distribution. The rest \(NP/2\) individuals \((OP)\) is initialized as per the first half population \((P)\) in terms of OBL as shown in Section 2.2. In the present work, the population size \(NP\) is an even number.

Step 2: Evaluation

All the individuals in the population are evaluated according to its position.

Step 3: The KH process

For the \(NP/2\) individuals in \(P\), their positions are updated by the three motions in KH method as described in Section 2.1. The main step of KH process can be described as follows:

\[
\text{for } i=1:NP/2 \quad \text{(all krill in } P) \quad \text{do}
\]

Perform the following motion calculation.

- Motion induced by other individuals
- Foraging motion
- Physical diffusion

Update the krill individual position in the search space.

\[
\text{end for } i
\]

Step 4: The OBL process

Corresponding to \(P\), for the last \(NP/2\) individuals in \(OP\), the position of individual is updated by the rules of OBL as described in Section 2.2. The main step of OBL process can be described as follows:

\[
\text{for } i=NP/2+1:NP \quad \text{(all krill in } OP) \quad \text{do}
\]

Compute \(X_{\text{OP}}(t+1)\) as per \(x_i\) and Eq. (8).

- If \(f(X_{\text{OP}}(t+1)) > f(x_i)\)
- Update \(X_{\text{OP}}(t+1), \text{by using Eq. (8)}\)

\[
\text{end if}
\]

Step 5: Combination

After a population \((P \text{ and } OP)\) of all the individuals are updated in this way, the \(P \text{ and } OP\) are combined into one population.

Step 6: The CM operator

For all the individuals in population \((P \text{ and } OP)\), the CM operator is performed as shown in Section 3.2.

Step 7: The PC operator

For all the individuals in population \((P \text{ and } OP)\), the PC operator is performed as shown in Section 3.3.

Step 8: Finding the best solution

Find out the best solution ever found and evaluate the average performance of the population.

Step 9: Stop or not

If the stopping criterion is satisfied, the OKH algorithm stops, and output the best solution, else return to Step 2.

The description of the proposed OKH process is also shown in Fig. 1.

4. Simulation results

In this section, the OKH method is evaluated from various aspects using a series of experiments on benchmark functions (see Table 1) and two engineering optimization problems. In order to obtain fair results, all the implementations are conducted under the same conditions as discussed in \[73\]. More detailed descriptions of all the benchmarks can be referred as \[6,74\]. Note that, without special clarification, the dimensions of function is set to 20. Population size and maximum generation are set to 50. In order to decrease the influence of the randomness, we have run 200 times for every method on each function and engineering case. The optimal solution for each test problem is bolded in each table. In addition, the scales that are used to do normalizations in the tables are fully different with each other, therefore values from different tables are independent each other. The detailed normalization process can be found in \[73\]. For all the tables, the total
Their performance is tested through 25 benchmarks (see Table 1). In order to investigate the influence of OBL, CM and PC, three search strategies (OBL, CM and PC) have been combined with the basic KH method. In order to represent “Opposition-based learning”, “Cauchy mutation operator”, and “Position clamping operator”, respectively. “1” represents KH with this strategy, while “0” means the method including KH instead of the responding strategy. We take OKH5 as an example. OKH is the combination of KH, OBL and PC operator while not CM operator. Note that, OKH0 is essentially a basic KH method, there we use KH directly in KH process. From Tables 3, OKH0 is KH II indeed. The results are recorded in Tables 3 and 4.

### Table 1: Benchmark functions.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>No.</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01</td>
<td>Ackley</td>
<td>F14</td>
<td>Quartic with noise</td>
</tr>
<tr>
<td>F02</td>
<td>Alpine</td>
<td>F15</td>
<td>Rastrigin</td>
</tr>
<tr>
<td>F03</td>
<td>Brown</td>
<td>F16</td>
<td>Rosenbrock</td>
</tr>
<tr>
<td>F04</td>
<td>Dixon &amp; Price</td>
<td>F17</td>
<td>Schwefel 2.26</td>
</tr>
<tr>
<td>F05</td>
<td>Fletcher-Powell</td>
<td>F18</td>
<td>Schwefel 2.2</td>
</tr>
<tr>
<td>F06</td>
<td>Griewank</td>
<td>F19</td>
<td>Schwefel 2.22</td>
</tr>
<tr>
<td>F07</td>
<td>Holzma 2 function</td>
<td>F20</td>
<td>Schwefel 2.21</td>
</tr>
<tr>
<td>F08</td>
<td>Levy</td>
<td>F21</td>
<td>Sphere</td>
</tr>
<tr>
<td>F09</td>
<td>Pathological function</td>
<td>F22</td>
<td>Step</td>
</tr>
<tr>
<td>F10</td>
<td>Penalty #1</td>
<td>F23</td>
<td>Sum function</td>
</tr>
<tr>
<td>F11</td>
<td>Penalty #2</td>
<td>F24</td>
<td>Zakharov</td>
</tr>
<tr>
<td>F12</td>
<td>Perm #1</td>
<td>F25</td>
<td>Wavyl</td>
</tr>
<tr>
<td>F13</td>
<td>Powel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. Flowchart of the OKH algorithm.

### Table 2: Various OKHs with three search strategies.

<table>
<thead>
<tr>
<th></th>
<th>OKH1</th>
<th>OKH2</th>
<th>OKH3</th>
<th>OKH4</th>
<th>OKH5</th>
<th>OKH6</th>
<th>OKH7</th>
</tr>
</thead>
<tbody>
<tr>
<td>KH</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OBL</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PC</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 3: Mean function values obtained by various OKHs.

<table>
<thead>
<tr>
<th></th>
<th>KH</th>
<th>OKH1</th>
<th>OKH2</th>
<th>OKH3</th>
<th>OKH4</th>
<th>OKH5</th>
<th>OKH6</th>
<th>OKH7</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01</td>
<td>289.64</td>
<td>1.88</td>
<td>150.43</td>
<td>1.24</td>
<td>1.92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F02</td>
<td>67.34</td>
<td>1.32</td>
<td>4.7E3</td>
<td>1.81</td>
<td>2.36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F03</td>
<td>59.44</td>
<td>5.93</td>
<td>11.36</td>
<td>8.6E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F04</td>
<td>2.18</td>
<td>1.53</td>
<td>153.86</td>
<td>10.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F05</td>
<td>19.17</td>
<td>9.83</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F06</td>
<td>1.02</td>
<td>2.16</td>
<td>11.84</td>
<td>8.6E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F07</td>
<td>6.0E11</td>
<td>3.93</td>
<td>11.36</td>
<td>8.6E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F08</td>
<td>1.04</td>
<td>2.80</td>
<td>13.83</td>
<td>21.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F09</td>
<td>1.00</td>
<td>3.14</td>
<td>14.00</td>
<td>28.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>2.5E3</td>
<td>1.53</td>
<td>153.86</td>
<td>10.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F11</td>
<td>9.8E5</td>
<td>6.0E9</td>
<td>104.61</td>
<td>21.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>1.1E18</td>
<td>8.6E7</td>
<td>104.61</td>
<td>21.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13</td>
<td>8.7E5</td>
<td>7.76</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F14</td>
<td>5.0E11</td>
<td>3.93</td>
<td>11.36</td>
<td>8.6E4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F15</td>
<td>7.15</td>
<td>1.00</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F16</td>
<td>4.9E5</td>
<td>6.0E9</td>
<td>104.61</td>
<td>21.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17</td>
<td>1.3E6</td>
<td>7.76</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F18</td>
<td>3.82</td>
<td>1.02</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F19</td>
<td>3.82</td>
<td>1.02</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F20</td>
<td>2.5E3</td>
<td>1.53</td>
<td>153.86</td>
<td>10.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F21</td>
<td>2.16</td>
<td>2.16</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F22</td>
<td>2.16</td>
<td>2.16</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F23</td>
<td>7.15</td>
<td>7.15</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F24</td>
<td>7.15</td>
<td>7.15</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F25</td>
<td>7.15</td>
<td>7.15</td>
<td>11.84</td>
<td>17.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Table 2, “OBL”, “CM”, and “PC” represent “Opposition-based learning”, “Cauchy mutation operator”, and “Position clamping operator”, respectively. “1” represents KH with this strategy, while “0” means the method including KH instead of the responding strategy. We take OKH5 as an example. OKH is the combination of KH, OBL and PC operator while not CM operator. Note that, OKH0 is essentially a basic KH method, there we use KH directly in Table 2. In addition, Gandomi and Alavi have concluded that, the KH II (KH with crossover operator) has the best performance among four types of KHS [55]. In the present work, the KH II is therefore selected as standard KH algorithm. That is to say, here OKHO is KH II indeed. The results are recorded in Tables 3-5.

From Tables 3-5, OKH7 is well capable of searching for the best function values for most cases. This indicates the combination of KH, OBL, CM and PC operators leads to the krill move toward the best solutions. For other OKHs, OKH6 (KH+OBL+CM) is only worse than OKH7. OKH4 (KH+OBL) and OKH5 (KH+OBL+PC) have the similar performance between each other. Further, considering OKH1, OKH2 and OKH4, i.e., KH together with only one strategy, OKH4 is the best, OKH2 is the second best, and OKH1 is the third best. It has proven that, OBL has played the greatest role among three different strategies. In general cases, the ranks from the best to the worst are approximately as follows: OKH7 > OKH6 > OKH5 > OKH4 > OKH2 > OKH3 > OKH1 > OKH0.

Please cite this article as: G.-G. Wang, et al., Opposition-based krill herd algorithm with Cauchy mutation and position clamping, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.11.018
Comprehensively considering, OKH7 is selected as the best method to compare with other methods in next experiments.

### 4.2. Comparisons of OKH7 with other methods

In order to explore the benefits of OKH, its performance was compared with seven optimization methods on 25 optimization problems, which are ABC [14], ACO [49], BBO [6], DE [20], GA [18], KH [55], and SGA [18]. The same parameters for KH and OKH7 are set as shown in Section 3 and above mentions. For the parameters used in the other methods, their settings are the same as [6,75]. The obtained function values are recorded in Tables 6–9.

From Table 6, we see that, on average, OKH7 has the best performance on five out of 25 benchmarks. Usually, BBO and SGA have the second best performance on five functions out of 25 functions. For best solutions, Table 7 shows that OKH7 is well capable of finding the optimal solutions on seventeen out of 25 benchmarks. SGA and BBO can search for the best solutions on five and two out of 25 benchmarks. For the worst function values shown in Table 8, the function values obtained by OKH7 are better than BBO and SGA.
on 14 out of 25 benchmarks. Generally, other methods perform slightly differently with each other. Furthermore, standard deviation (Std) of the eight methods (see Table 9) indicates that, OKH7 has the smallest Std on sixteen out of 25 benchmarks. In other words, for most benchmarks, OKH7 is able to find the optimal solutions within smaller range than other methods. And it is therefore more practical and feasible than others. From the Tables 6–9, for most high-dimensional benchmarks, OKH7 is the best method at searching for the optimal function values.

Furthermore, to intuitively clearly show the superiority of the OKH7, evolution curves of BBO, OKH7 and SGA on some most typical benchmarks are also provided in this section (see Figs. 2–4).

From Fig. 2, for F01 function, it can be seen that, OKH7 is far better than BBO and SGA, while BBO is little better than SGA. For F04, SGA has obtained the better final value than BBO though it is worse than OKH7. For F06, though BBO, OKH7 and SGA converge to the similar value finally, the function value of OKH7 is smaller than BBO and SGA all the time. For F07, OKH7 has the better function value than BBO and SGA that have similar trend and final value.

From Fig. 3, for F09 and F12, they have the similar convergent trend. OKH7 is far better than BBO and SGA that has the similar final values. For F10 and F11, they have similar convergent trend, too. OKH7 is far better than BBO and SGA.

From Fig. 2, it is clear that OKH7 is better than BBO and SGA for F18. For F19, though BBO has a relative good initial value, the three methods converge to the similar values finally. For F20 and F22, OKH7 is better than BBO and SGA that converge to the similar values. Especially, OKH7 converge sharply about generation 20. It is indicated that the OBL, CM operator and PC operator used in OKH7 have a good influence on the performance of OKH7.
From the above Tables 6–9 and Figs. 2–4, we can conclude that our proposed OKH7 algorithm is well capable of finding the best function minimum. In general, SGA and BBO are only inferior to OKH7 among eight methods.

4.3. Comparisons using t-test

As per experimental results of 200 trials on each function, Table 10 presents the t-values on every function of the two-tailed test with the 5% level of significance between the OKH7 and other optimization methods. In the table, the value of t with 398 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. Boldface means that OKH7 is better than the compared algorithm. For the last three rows, the “Better”, “Equal”, and “Worse” represent that OKH7 is better than, equal to, and worse than certain comparative method on certain benchmark. Here, we take the OKH7 and the BBO for instance. OKH7 performed better than BBO on thirteen functions, does as good as BBO on six functions, and is worse than the latter on six functions. In conclusion, this indicates that the OKH7 generally performs better than BBO on the solution accuracy. Furthermore, for KH, the number of “Better”, “Equal”, and “Worse” are 17, 3, and 5, respectively. These three numbers indicate that OKH7 has the better performance than KH on most functions. Though the OKH7 is outperformed on some functions, Table 10 still reveals that it performs better than the other seven methods for most functions.

4.4. Analyze the number of fitness evaluation

Here, in order to fully investigate the advantage of the OKH7 method, the number of fitness evaluations is also studied. We look at the number of fitness evaluations on 25 functions for each method to certain fixed function values. In our experiments, the fixed function values are set to $opt + 2$. $opt$ is the optimum for every function. The maximum of fitness evaluations is set to 50,000. That is to say, if a method is not able to find the values $opt + 2$ within the maximum of fitness evaluations (50,000), it will stop, too, and return the 50,000. All the results are recorded in Table 11. Table 11 shows that, for the 16 functions (F01–F02, F04–F07, F12, F15–F16, and F18–F24), OKH7 can find the satisfactory results with the least fitness evaluations. SGA, DE and BBO have worse performance than OKH7, and they can converge to the fixed values with the least fitness evaluations on the six functions (F03, F08, F10–F11, F13, and F17), two functions (F09, F25) and one function (F14), respectively. In conclusion, OKH7 can find the satisfactory solution by using the least fitness evaluations on most function.

4.5. Engineering optimization problems

Except the standard functions discussed in the section above, two engineering optimization problems are also used to validate the OKH7 method.
4.5.1. Test-Sheet Composition (TSC) problem

Examination has a great role in motivating teaching and student learning. Automatic TSC using computer is to find a combination of questions to satisfy constraints from item bank. In essence, TSC model is a multi-constraint multi-objective optimization problem.

In general, the error between eight constraints and evaluation requirements can be considered as objective function $f$. The mathematical model for the TSC problem is as follows:

$$
\begin{align*}
\min f &= \sum_{j=1}^{3} (d_{ij}^d + d_{ij}^d) \times \omega_j + \sum_{i=1}^{h} (d_{ij}^w + d_{ij}^w) \times \omega_i + \sum_{j=1}^{l} (d_{ij}^d + d_{ij}^d) \\
&\times \omega_d + \sum_{i=1}^{p} (d_{ij}^d + d_{ij}^d) \times \omega_t \\
+ \sum_{i=1}^{s} (d_{ij}^d + d_{ij}^d) \times \omega_h + \sum_{j=1}^{q} (d_{ij}^d + d_{ij}^d) \times \omega_j
\end{align*}
$$

where, $d_{ij}^d$, $d_{ij}^d$, $d_{ij}^d$, and $d_{ij}^d$ are the positive deviation between the test-sheet property and constraints; $d_{ij}^d$, $d_{ij}^d$, $d_{ij}^d$, and $d_{ij}^d$ are the negative deviation between the test-sheet property and constraints. The coefficient of the positive deviation and negative deviation is $0$, i.e., $d^d \times d^d = 0$; $\omega_j (j = 1, 2, \ldots, 8)$ is weight for TSC problem, whose sum is $1$.

In practice, the weight $\omega_j$ has an important impact on the TSC problem. In the present work, Analytic hierarchy process (AHP) [76] is used to address the weights. We get the goal weights for test-sheet property and constraints; $\omega_h, \omega_t, \omega_d, \omega_p, \omega_s, \omega_q$ are the weight for TSC model that are $\{0.3423, 0.1412, 0.0792, 0.0319, 0.2244, 0.0417\}$ according to the AHP [77].

In OKHs, the standard continuous encoding of OKH is not proper solve TSC problem directly. In order to apply OKH to TSC problem, preprocessing and encoding should be implemented firstly. More details about AHP, preprocessing and encoding can be referred as [77]. More description of the mathematical model for TSC problem can be referred as [77].

When using OKH to solve TSC problem, the status code for each krill individual represents a candidate solution. Krill $i$ in population is evaluated by the objective function $f(X_i)$ in TSC model. Here, the eight algorithms are used to find the best combination of questions. The constraints of the TSC problem are set as shown in [77].

From Table 12, we see that OKH7 is the most effective method on the best, mean and worst values. Moreover, OKH7 has the least Std among eight methods. In sum, from Table 12, OKH7 is well capable of finding the most satisfactory solution for TSC problem. Since TSC problem is an essentially a discrete problem, therefore we can say, OKH7 is suitable for solving discrete optimization problem, too.

4.5.2. Sensor selection problem

The sensor selection problem [6] for aircraft engine health estimation can be considered as a test problem to validate the OKH method.

The Modular Aero Propulsion System Simulation (MAPSS) [6] is used as the engine simulation in sensor selection problem. Based on the analyses in [6], the objective function of the health estimation problem can be expressed as

$$
J = \sum_{i=4}^{13} \sqrt{\sum_{j=1}^{C_2} f_i(x_j) + \alpha C_0}
$$

where, $\alpha$ and $C_0$ are used for normalization. $C_0$ is the covariance, and $C_0$ is the cost of setting the aircraft engine with all 11 sensors. $\alpha$ is a scale factor that can balance the importance of financial cost and estimation accuracy.

We see that the selection of sensors in order to generate the minimum for $J$ is essentially an optimization problem. In fact, optimization methods can be used to solve the sensor selection problem. A population member contains a vector of integers, and its element is corresponding to a sensor number. Here suppose that a total of 20 sensors can be used form unique 11 sensors, and each sensor is only selected less than or equal to four times. The number of possible sensor sets is equal to the coefficient of $x^{10}$ as

$$
q(x) = (1 + x + x^2 + x^3 + x^4)^{11}
$$

For $x^{10}$ in Eq. (18), its coefficient is $3,755,070$. In order to minimize $J$ in Eq. (17), the total number of sensor sets must be searched. $\Sigma_f$ for a sensor set should be solved to compute $J$ for a single sensor set. In order to solve for $\Sigma_f$, if a discrete algebraic Riccati equation (DARE) [6] is used, this brute-force search would consume $21 h$ [6].

Instead of a brute-force 21-h search, eight optimization methods are used to search for a sub-optimal sensor set. The results on the sensor selection problem are recorded in Table 13. We see that OKH7 performs the best in terms of both average performance and best performance. For worst function values, SGA performs the best among different methods, and OKH7 is only inferior to SGA. Furthermore, the Std of OKH7 is much smaller than other methods except SGA. To sum up, OKH7 has a relatively high ability of finding the satisfactory sensor set.

5. Conclusion

This paper presented various OKH methods for solving the continuous and discrete optimization problems. In OKH, first of all, a half population of candidate solutions is randomly initialized. And then, the rest half population is initialized as per the first half population based on the OBL theory. After initialization, new solutions are created by applying the KH and OBL process. By simultaneous consideration of the krill in the KH process and OBL process, OKH can provide a higher chance of finding solutions which are closer to the global optimum. The two subpopulations are combined into one population. In order to add the diversity of the krill population and balance the exploration and exploitation, the CM and PC operators are introduced into the OKH method.

After the implementation of the CM and PC operators, the fitness

<table>
<thead>
<tr>
<th>Table 12</th>
<th>Optimization results for the sensor selection problem.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Mean Worst Std</td>
</tr>
<tr>
<td>ABC</td>
<td>3.89  4.54  4.87  0.36</td>
</tr>
<tr>
<td>ACO</td>
<td>6.02  6.62  6.77  0.31</td>
</tr>
<tr>
<td>BBO</td>
<td>3.40  4.69  4.85  0.34</td>
</tr>
<tr>
<td>DE</td>
<td>5.44  6.13  6.15  0.28</td>
</tr>
<tr>
<td>GA</td>
<td>4.12  5.06  4.84  0.37</td>
</tr>
<tr>
<td>KH</td>
<td>3.96  5.52  6.12  0.52</td>
</tr>
<tr>
<td>OKH7</td>
<td>3.35  4.47  4.79  0.27</td>
</tr>
<tr>
<td>SGA</td>
<td>3.73  4.87  4.86  0.47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 13</th>
<th>Optimization results for the Sensor Selection problem.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Mean Worst Std</td>
</tr>
<tr>
<td>ABC</td>
<td>8.02  8.10  8.14  0.021</td>
</tr>
<tr>
<td>ACO</td>
<td>8.10  8.24  8.15  0.040</td>
</tr>
<tr>
<td>BBO</td>
<td>7.20  8.02  8.22  0.015</td>
</tr>
<tr>
<td>DE</td>
<td>7.70  8.09  8.11  0.021</td>
</tr>
<tr>
<td>GA</td>
<td>8.04  8.06  8.06  0.038</td>
</tr>
<tr>
<td>KH</td>
<td>8.03  8.08  8.11  0.014</td>
</tr>
<tr>
<td>OKH7</td>
<td>7.16  8.00  8.06  0.010</td>
</tr>
<tr>
<td>SGA</td>
<td>8.02  8.02  8.02  0</td>
</tr>
</tbody>
</table>
of the resulting solutions are evaluated and find the best solution for this generation. Several experiments are carried out on an array of well-known benchmark problems and two engineering optimization problems and the related results are compared with seven other methods. The experimental results verify that the proposed OKH model is a more efficient and practical method in solving optimization problems.

In addition, in order to verify the influence of the OBL, CM and PC strategies, KH is combined with different combinations of them. The results show that KH combined with the OBL, CM and PC operators, i.e., OKH7, has the best performance among different variants of OKH.

In future, our research highlights would be focused on the following points. Firstly, the influence of the OKH parameters on convergence and performance would be carefully analyzed and investigated. Secondly, from Table 3–5, we claimed that OKH7 is the best performer compared with other seven OKHs. But it is obvious that the computational complexity of OKH7 is the most high. Because it has more strategies than other OKHs, which can be seen from Table 2. Thus, OKH7 will spend more CPU time than other OKHs, in order to find optimal solutions. It should be noted that CPU time is a significant factor to the implementation of most optimization methods. The time used by OKH should be studied. Based on this time study, how to make the OKH consume less time is still worthy of further study. Thirdly, through OKH, we can see, an improved method significantly outperforms the original methods. In our future studies, the KH will be combined with other search strategies. Fourthly, we would use more benchmarks to test our method, such as constrained optimization and complex engineering optimization problems [78,79]. Finally, future analysis of our OKH using dynamic system and Markov chain can give us theoretical insight into their advantages and disadvantages.

Acknowledgments

This work was supported by Jiangsu Province Science Foundation for Youths (No. BK20150239) and National Natural Science Foundation of China (No. 61503165).

References


Please cite this article as: G.-G. Wang, et al., Opposition-based krill herd algorithm with Cauchy mutation and position clamping, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2015.11.018

Amir H. Gandomi is the pioneer of Krill Herd Algorithm. He was selected as an elite in 2008 by Iranian National Institute of Elites. He used to be a lecturer in Tafresh University and served as a researcher in National Elites Foundation. He is currently a researcher in the Department of Civil Engineering at the University of Akron, OH. Amir Hossein Gandomi has published over 70 journal papers and several discussion papers, conference papers, and book chapters. He has two patents and has published three books in Elsevier. His research interests are Metaheuristics modeling and optimization.

Amir H. Alavi is a senior researcher at the Department of Civil & Environmental Engineering, Michigan State University (MSU), MI, USA. His area of expertise includes energy harvesting, sensor technology, big data analysis, artificial intelligence, statistical and probabilistic methods, and metaheuristic modeling and optimization. He has published three books and over one hundred research papers in book chapters, indexed journals, and conference proceedings, along with two patents. He is on the editorial board of several journals. He is the pioneer of Krill Herd (KH) optimization algorithm.