Chaotic Krill Herd algorithm

Gai-Ge Wang a, Lihong Guo b,⇑, Amir H. Gandomi c,1, Guo-Sheng Hao a, Heqi Wang b

a School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, Jiangsu 221116, China
b Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China
c Department of Civil Engineering, The University of Akron, Akron, OH 44325, USA

Article info
Article history:
Received 20 August 2012
Received in revised form 26 September 2013
Accepted 10 February 2014
Available online 3 March 2014

Keywords:
Global optimization problem
Krill Herd
Chaotic maps
Multimodal function

Abstract

Recently, Gandomi and Alavi proposed a meta-heuristic optimization algorithm, called Krill Herd (KH). This paper introduces the chaos theory into the KH optimization process with the aim of accelerating its global convergence speed. Various chaotic maps are considered in the proposed chaotic KH (CKH) method to adjust the three main movements of the krill in the optimization process. Several test problems are utilized to evaluate the performance of CKH. The results show that the performance of CKH, with an appropriate chaotic map, is better than or comparable with the KH and other robust optimization approaches.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

The process of optimization is essentially the choice of a vector within a search space. The selected vector can maximize/minimize an objective function to provide the best solution. Generally, modern intelligent approaches are used to deal with these types of optimization problems. Such optimization approaches can be categorized into two groups in view of their natures: (1) deterministic, and (2) random intelligent approaches. The deterministic approaches using gradient have a strict step. They produce the identical solution if its initial starting values are the same with each other when solving the same problem. Contrary to the deterministic approaches, gradient-free stochastic algorithms are based on random walks. Therefore, the optimization process cannot be repeated under any conditions. However, in most cases, both of them are capable of finding the same final optimal solutions [44]. Recently, nature-inspired meta-heuristic algorithms show a powerful and efficient performance for dealing with high-dimension nonlinear optimization problems [12,53].

To some extent, all meta-heuristic approaches manage to make a trade-off between intensification (local search) and randomization (global search) [50]. These robust nature-inspired meta-heuristic approaches are utilized to tackle NP-hard problems such as task-resource assignment. They can fully exploit the useful information of the whole population to find optimal solutions. Up to now, significant research has been done on evolution theory. The process of evolution is idealized as a kind of gradient-free method, called genetic algorithms (GAs) [6,18,55]. Since then various nature-inspired meta-heuristic approaches have been proposed such as evolutionary strategy (ES) [1,2], particle swarm optimization (PSO) [27,28,40,56], ant colony optimization (ACO) [4], differential evolution (DE) [10,19,38], firefly algorithm (FA) [7], biogeography-based optimization (BBO) [29,37], cuckoo search (CS) [9,49], probability-based incremental learning (PBIL) [36], big bang–big
crunch algorithm [5,23,25,26], harmony search (HS) [15,17,47], charged system search (CSS) [24], animal migration optimization (AMO) [31], teaching–learning-based optimization (TLBO) [3,35], bat algorithm (BA) [11,52], and the KH method [8]. In fact, KH is a new kind of swarm intelligence optimization approach inspired by the herding behavior of krill [8]. In KH, the objective function for the krill movement is decided by the distances of each krill from density of the krill swarm and food. The position for each krill is made up of three parts: (i) movement induced by other individuals (ii) foraging motion, and (iii) physical diffusion. The crucial advantage of KH algorithm is its simplicity, making KH implement easily and lending itself to parallel computation [44].

In general, KH lends itself strongly to exploitation. However, it cannot always implement global search well. Thus, in some cases, KH fails to find global optimal solution. The search strategy used in basic KH is mainly based on random walks. Thus, it cannot always deal with the problem successfully [45]. Different strategies have been added to the basic KH method [43,45] with the aim of improving its performance [45].

With the development of the nonlinear dynamics, chaos concept has been widely considered in various applications [34]. In this context, one of the most famous applications is the introduction of chaos theory into the optimization methods [48]. Up to now, the chaos theory has been successfully combined with several meta-heuristic optimization methods [13]. Some major efforts in this area includes hybridizing chaotic sequences with memetic differential evolution algorithm [22], FA [13], gravitational search algorithm [20], imperialist competitive algorithm [41], charged system search [33,42], PSO [14,46], and GAs [21].

In the present study, chaotic KH-based (CKH) methods are introduced for the purpose of accelerating the convergence of KH. Various one-dimensional chaotic maps are employed in place of the parameters used in KH. The performance of the proposed approach is tested on fourteen benchmark problems. Experimental results indicate that CKH performance is superior to KH, ACO, BA, CS, DE, ES, GA, PBIL, and PSO. This is mainly because deterministic chaotic signals are used to replace linearly declined values.

The organization of this paper is as follows. Firstly, a brief overview of the basic KH algorithm and 12 chaotic maps are given in Section 2. The detailed presentation of the proposed CKH approach is provided in Section 3. Subsequently, the tuning of the inertia weights and selecting the optimal CKH are described in Section 4. In addition, the performance of the CKH approach is verified using fourteen benchmarks. Finally, a summary of the present work is represented in Section 5.

2. Overview of the KH and chaotic maps

2.1. KH method

KH [8] is a new type of meta-heuristic method for solving optimization problems. This method is inspired by the herding of krill swarms when searching for food in nature. For each krill, its position in search space is influenced by three components described below [45]:

i. movement induced by other krill;
ii. foraging action;
iii. random diffusion.

For simplicity, the above three motions in KH can be idealized to the following Lagrangian model [8] as shown in Eq. (1).

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

where \(N_i\), \(F_i\), and \(D_i\) are, respectively, corresponding to the above three motions for the \(i\)th krill [8]. The krill number is represented by \(N_i\), and \(t\) is considered as generation.

2.1.1. Motion induced by other krill

The direction of the first motion, \(\alpha_i\), is approximately calculated according to the following three factors: target effect, local effect, and repulsive effect. For krill \(i\), this movement can be modeled below:

\[
N_{i,\text{new}} = N_{i,\text{max}} \alpha_i + \omega_n N_{i,\text{old}}
\]

(2)

where

\[
\alpha_i = \alpha_i^{\text{local}} + \alpha_i^{\text{target}}
\]

(3)

and \(N_{i,\text{max}}\) is the maximum speed, \(\omega_n\) is the inertia weight in \([0,1]\), \(N_{i,\text{old}}\) is the previous motion, \(\alpha_i^{\text{local}}\) and \(\alpha_i^{\text{target}}\) are the local effect and the target effect, respectively [45]. According to the literature [8], we set \(N_{i,\text{max}}\) to 0.01 (m s\(^{-1}\)) in our study.

2.1.2. Foraging motion

The second motion is determined by the two main factors: the food location and the previous experience with respect to the food position. For the \(i\)th krill, the expression of this motion can be provided below [45]:
\[ F_i = V_f \beta_i + \omega_f F_i^{\text{old}} \]  

where

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

and \( V_f \) is the foraging speed, \( \omega_f \) is the inertia weight between 0 and 1, \( F_i^{\text{old}} \) is the previous foraging motion, \( \beta_i^{\text{food}} \) is the food attraction and \( \beta_i^{\text{best}} \) is the effect of the best fitness. In our study, we set \( V_f \) to 0.02 [8].

### 2.1.3. Physical diffusion

The third motion is essentially a random process. The model of this motion can be expressed according to two factors: a maximum diffusion speed and a random vector. This model can be expressed below [8]:

\[ D_i = D^{\text{max}} \delta \]

where \( D^{\text{max}} \) is the diffusion speed, and \( \delta \) is the random vector in \([-1,1]\) [45].

### 2.1.4. Main procedure of the KH method

The second and third motion involve two global and two local schemes. These schemes can work simultaneously which makes KH a robust and efficient method [45]. The position of krill \( i \) from \( t \) to \( t + \Delta t \) can be formulated by Eq. (7) [45].

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

Note that \( \Delta t \) is a key constant and should be fine-tuned in terms of the specific problem. The reason is that \( \Delta t \) can be regarded as a scale factor of the speed vector. Moreover, in KH, the inertia weights \((\omega_m, \omega_f)\) are set to 0.9 at the start of the KH to highlight exploration. They are later linearly decreased to 0.1 in order to stimulate exploitation [8].

Furthermore, in order to enhance the performance of the KH, genetic reproduction mechanisms have been introduced into the KH algorithm [8].

In general, the KH method can be described as shown in Fig. 1, and its responding flowchart is illustrated in Fig. 2. Further information about the above movements and KH method can be found in [8].

### 2.2. Chaotic maps

There is a special kind of random-based optimization algorithms, namely chaotic optimization algorithm (COA) [13]. COA employs chaotic variables rather than random variables [13]. The chaos has the property of the non-repetition and ergodicity. Therefore, it can perform downright searches at higher speeds compared to the stochastic searches that mainly rely on probabilities [13]. In the present study, 1-D, non-invertible maps are used to produce chaotic sets. Twelve distinguished 1-D

---

**Krill herd algorithm**

**Begin**

**Step 1: Initialization.** Initialize the generation counter \( G \) the population \( P \) of \( NP \) krill randomly, \( V_f \), \( D^{\text{max}} \), and \( N^{\text{max}} \).

**Step 2: Fitness calculation.** Calculate fitness for each krill according to its initial position.

**Step 3: While** \( G < \text{MaxGeneration} \) **do**

- Sort the population according to their fitness.
- For \( i = 1:NP \) (all krill) **do**
  - Perform the following motion calculation.
    - Motion induced by other individuals
    - Foraging motion
    - Physical diffusion
  - Implement the genetic operators.
  - Update the krill position in the search space.
  - Calculate fitness for each krill according to its new position.
- **end for** \( i \)
- \( G = G + 1 \).

**Step 4: end while**

**End.**

---

**Fig. 1.** Krill Herd algorithm.
maps [13] are described in Table 1. In this table, $k$ means the index of the chaotic sequence, and $x_k$ represents the $k$th number in the chaotic sequence.

3. The proposed CKH approach

As it is presented in Eqs. (2) and (4), the main parameters of KH are the inertia weights ($\omega_n, \omega_f$) that represent the variations of the global optimal attraction. The values of the weights are vitally important in deciding the convergence speed and how to update the position of the krill in KH.

Though the basic KH is very powerful, the solutions are still deviating from each other at the end of the search. In KH, the inertia weights ($\omega_n, \omega_f$) are set to 0.9 at the early search stage to highlight exploration. Finally, they are linearly decreased to 0.1 to stimulate exploitation [8].

In the standard KH, there is no need to keep linearly decrements. In fact, a chaotic varying inertia weights ($\omega_n, \omega_f$) may be more favorable for the search, which may also make the method converge to the best solution with a fast speed as will be demonstrated in the following section. After normalization, the range of a chaotic map is always between 0 and 1. So, in this paper, we use chaotic maps to adjust inertia weights ($\omega_n, \omega_f$), and such chaos-improved KH can be named as the chaotic KH.

The basic framework of the CKH method and its responding flowchart are shown in Figs. 3 and 4, respectively. In Figs. 3 and 4, $1 \leq i \leq NP$ represents the krill number, and $NP$ is the total number of the krill in the population, i.e. population size.

The tuning of inertia weights ($\omega_n, \omega_f$) using chaotic maps also improve the ability of KH to avoid local optima when solving a multimodal function. Comparing with other optimization methods, this could be a merit for this method.
Moreover, the second significant improvement is the addition of elitism scheme to the CKH. As with other meta-heuristic methods, a kind of elitism is typically introduced in order to hold the optimal solutions. This puts a stop to the best solutions from being corrupted by motion calculation operators. Here, an elitism strategy is utilized to memorize the property of the best krill in CKH. Consequently, even if motion calculation operation ruins its corresponding krill, it has been held and can be recovered if required.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Chebyshev map</td>
<td>$x(k+1) = \cos(k \cos^{-1}(x(k)))$</td>
</tr>
<tr>
<td>M2</td>
<td>Circle map</td>
<td>$x(k+1) = x(k) + b - (a/2\pi) \sin(2\pi k \mod 1)$</td>
</tr>
<tr>
<td>M3</td>
<td>Gaussian map</td>
<td>$x_k = \begin{cases} 0 &amp; x_k = 0 \ 1/x_0 \mod 1 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>M4</td>
<td>Intermittency map</td>
<td>$x_{k+1} = \begin{cases} \varepsilon + x_k + C_1 x_k &amp; 0 &lt; x_k &lt; P \ 0 &amp; P &lt; x_k &lt; 1 \end{cases}$</td>
</tr>
<tr>
<td>M5</td>
<td>Iterative map</td>
<td>$x_{k+1} = \sin \left( \frac{x_k}{2} \right)$, $a \in (0, 1)$</td>
</tr>
<tr>
<td>M6</td>
<td>Liebovitch map</td>
<td>$x_{k+1} = \begin{cases} ax_k &amp; 0 &lt; x_k \leq P \ \frac{1}{x_k} &amp; P &lt; x_k &lt; P_1 \ P_1 &lt; x_k &lt; 1 \end{cases}$</td>
</tr>
<tr>
<td>M7</td>
<td>Logistic map</td>
<td>$x_{k+1} = a(1 - x_k)$</td>
</tr>
<tr>
<td>M8</td>
<td>Piecewise map</td>
<td>$x_{k+1} = \begin{cases} \frac{x_k}{3x_k} &amp; 0 \leq x_k &lt; P \ \frac{1}{x_k} &amp; P \leq x_k &lt; P_1 \ 1 - P &lt; x_k &lt; 1 \end{cases}$</td>
</tr>
<tr>
<td>M9</td>
<td>Sine map</td>
<td>$x_{k+1} = \frac{1}{2} \sin(\pi x_k)$, $0 &lt; \alpha \leq 4$</td>
</tr>
<tr>
<td>M10</td>
<td>Singer map</td>
<td>$x_{k+1} = \mu(7.86 x_k - 23.31 x_k^3 + 28.75 x_k^4 - 13.302875 x_k^5)$</td>
</tr>
<tr>
<td>M11</td>
<td>Sinusoidal map</td>
<td>$x_{k+1} = ax_k^2 \sin(\pi x_k)$</td>
</tr>
<tr>
<td>M12</td>
<td>Tent map</td>
<td>$x_{k+1} = \begin{cases} \frac{10}{(1 - x_k)} &amp; x_k &lt; 0.7 \ \frac{1}{x_k} &amp; x_k \geq 0.7 \end{cases}$</td>
</tr>
</tbody>
</table>

*With $a = 0.5$ and $b = 0.2$, it generates a chaotic sequence in $(0, 1)$.

**CKH algorithm**

**Begin**

**Step 1: Initialization.** Initialize the generation counter $G$, the population $P$ of NP krill randomly, $V_k$, $D_{max}$, and $N_{max}$ and KEEP; set initial value of the chaotic map $c_0$ randomly, and inertia weights $\omega = (\omega_x, \omega_y) = c_0$.

**Step 2: Fitness calculation.** Calculate fitness for each krill according to its initial position.

**Step 3: While** $G < \text{MaxGeneration}$ **do**

Sort the population according to their fitness.

Memorize the KEEP best krill.

Update inertia weights using chaotic maps ($\omega = c_{t+1}$).

for $i = 1$: NP(all krill) do

Perform the following motion calculation.

Motion induced by the presence of other individuals

Foraging motion

Physical diffusion

Update the krill position in the search space.

Calculate fitness for each krill according to its new position.

end for $i$

Replace the KEEP worst krill with the KEEP best krill.

$G = G+1$.

**Step 4: end while**

**End.**

Fig. 3. CKH algorithm.
4. Experimental results

In this section, various experiments on optimization benchmark problems are implemented to verify the performance of the proposed meta-heuristic CKH method.

In order to get an unbiased comparison of CPU times, all the experiments are performed using the same PC with the detailed settings as shown in Table 2.
Fourteen different benchmark functions are used to evaluate our proposed meta-heuristic CKH method. The expressions and properties of these benchmarks can be presented in Tables 3 and 4, respectively [44]. More information with reference to all the benchmarks can be found in [54].

For all the algorithms used in this work, population size $N_P$, an elitism parameter $K_{keep}$, and maximum generation $Maxgen$ are assigned to 50, 2 and 50, respectively. The optimal results obtained by each algorithm for each benchmark are explained in the following sections. It should be pointed out that different scales are used to normalize the results in the tables. Therefore, values cannot be comparative between them. The dimension of each function used in our work is 20 (i.e., $d = 20$).

### 4.1. The performance of CKH with different chaotic maps

CKH is capable of significantly improving the solution quality by using the chaotic maps. Here, the work of adjusting the inertia weights $\omega = (\omega_0, \omega_f)$ are implemented. And the value of $\omega$ is substituted with various chaotic maps presented in Section 2.2. We normalized all the chaotic maps in $[0.1, 0.9]$ so as to achieve this.

100 implementations of the CKH algorithm are carried out on fourteen benchmarks to obtain typical statistical results. The results of the simulations are recorded in Tables 5 and 6. Table 5 represents the average values obtained by the CKH method, averaged over 100 simulations. Table 6 also illustrates the best values obtained by CKH method from 100 simulations. We must point out, the marks M1, M2, …, M12 in the Tables 5 and 6 are short for the twelve responding chaotic maps in the Section 2.2.

### Table 2
The detailed settings.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Detailed settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Pentium IV processor</td>
</tr>
<tr>
<td>Frequency</td>
<td>2.0 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>512 MB</td>
</tr>
<tr>
<td>Hard drive</td>
<td>160 GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>Detailed settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Windows XP3</td>
</tr>
<tr>
<td>Language</td>
<td>MATLAB R2012a (7.14)</td>
</tr>
</tbody>
</table>

### Table 3
Benchmark functions.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01</td>
<td>Ackley</td>
<td>$f(x) = 20 + e - 20 \cdot e^{-0.2 \sqrt{\sum_{i=1}^{n} x_i^2} - e^{\sum_{i=1}^{n} \sin(2\pi x_i)}}$</td>
</tr>
<tr>
<td>F02</td>
<td>Fletcher–Powell</td>
<td>$f(x) = \sum_{i=1}^{n} (A_i - B_i)^2 \cdot A_i = \sum_{i=1}^{n} (a_0 \sin x_i + b_0 \cos x_i) \quad B_i = \sum_{j=1}^{n} (a_j \sin x_j + b_j \cos x_j)$</td>
</tr>
<tr>
<td>F03</td>
<td>Griewank</td>
<td>$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{1000} - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1$</td>
</tr>
<tr>
<td>F04</td>
<td>Penalty #1</td>
<td>$f(x) = \frac{1}{50} \left( 10 \sin^2(\pi x_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot (1 + 10 \sin^2(\pi y_{i+1})) + (y_n - 1)^2 \right) + \sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = 1 + 0.25(x_i + 1)$</td>
</tr>
<tr>
<td>F05</td>
<td>Penalty #2</td>
<td>$f(x) = 0.1 \left( \sin^2(3.14 x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \cdot (1 + \sin^2(3.14 y_{i+1})) + (x_n - 1)^2 + \sin^2(2 \pi x_n) \right) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$</td>
</tr>
<tr>
<td>F06</td>
<td>Quartic with noise</td>
<td>$f(x) = \sum_{i=1}^{n} \left( i \cdot x_i^4 + U(0, 1) \right)$</td>
</tr>
<tr>
<td>F07</td>
<td>Rastrigin</td>
<td>$f(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \pi x_i))$</td>
</tr>
<tr>
<td>F08</td>
<td>Rosenbrock</td>
<td>$f(x) = \sum_{i=1}^{n-1} \left[ 100(x_i - x_{i+1}^2)^2 + (x_i - 1)^2 \right]$</td>
</tr>
<tr>
<td>F09</td>
<td>Schwefel 2.26</td>
<td>$f(x) = 418.9829 \times D - \sum_{i=1}^{n} x_i \sin(x_i^1/2)$</td>
</tr>
<tr>
<td>F10</td>
<td>Schwefel 1.2</td>
<td>$f(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2$</td>
</tr>
<tr>
<td>F11</td>
<td>Schwefel 2.22</td>
<td>$f(x) = \sum_{i=1}^{n}</td>
</tr>
<tr>
<td>F12</td>
<td>Schwefel 2.21</td>
<td>$f(x) = \max(</td>
</tr>
<tr>
<td>F13</td>
<td>Sphere</td>
<td>$f(x) = \sum_{i=1}^{n} x_i^2$</td>
</tr>
<tr>
<td>F14</td>
<td>Step</td>
<td>$f(x) = 6 \cdot n + \sum_{i=1}^{n}</td>
</tr>
</tbody>
</table>

1 In benchmark function F02, the matrix elements $a_{i,n}, b_{i,n} \in (-100, 100)$, $x_{m,n} \in (-\pi, \pi)$ are draw from uniform distribution.

2 In benchmark functions F04 and F05, the definition of the function $u(x_i, a, k, m)$ is as follows:

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$
From Tables 5 and 6, it can be seen that the basic KH combined with the M9 (Sine map) or M10 (Singer map) perform better than others. For M9 (Sine map) and M10 (Singer map), the M10 (Singer map) significantly outperforms the M9 (Sine map) when multiple runs are made. Accordingly, we choose it as the proper map to form the best chaotic KH (CKH) that will be studied in more detail in Section 4.2.

### 4.2. CKH vs. other optimization algorithms

The performance of CKH was compared on benchmark problems with nine other optimization algorithms [45]. The methods included in the comparative study are ACO [4], BA [51,52], CS [49], DE [30,38], ES [1,2], GA [18], KH [8], PBIL [36], and PSO.
[16, 27]. In DE, the DE/rand/1/bin scheme is used. It is worth mentioning that Gandomi and Alavi [8] have demonstrated that the KH II shows the best performance compared to the existing algorithms. Thus, in this work, KH II is considered as the basic KH method. The parameter settings in all experiments for each algorithm are shown in Table 7 [44].

Tables 8 and 9 record the average and the best results from 100 runs, respectively. Table 8 shows that, on average, CKH performs better than other methods on eleven of the fourteen benchmarks (F01–F08, and F12–F14) when searching for function minimum. CS, DE and GA are the second most effective, performing best on one of the fourteen benchmarks (F10, F11 and F09, respectively). From Table 9, it can be seen that, CKH performs better than other methods on eight of the fourteen benchmarks (F01, F03, F06–F08, and F12–F14). GA and ACO are the second and third most effective, respectively, performing the best on the benchmarks F02, F09, F10 and F04–F05 when multiple runs are made. CS and DE are the fourth most effec-

<table>
<thead>
<tr>
<th>Table 7</th>
<th>Parameters settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Parameters</td>
</tr>
<tr>
<td>ACO</td>
<td>Initial pheromone value $\tau_0 = 1E–6$, pheromone update constant $Q = 20$, exploration constant $q_o = 1$, global pheromone decay rate $\rho_s = 0.9$, local pheromone decay rate $\rho_l = 0.5$, pheromone sensitivity $s = 1$, and visibility sensitivity $b = 5$</td>
</tr>
<tr>
<td>BA</td>
<td>Loudness $A = 0.95$, pulse rate $r = 0.5$, and scaling factor $\sigma = 0.1$</td>
</tr>
<tr>
<td>CKH</td>
<td>The foraging speed $V_f = 0.02$, the maximum diffusion speed $D_{max} = 0.005$, the maximum induced speed $N_{max} = 0.01$</td>
</tr>
<tr>
<td>CS</td>
<td>A discovery rate $p_d = 0.25$; for DE, a weighting factor $F = 0.95$ and a crossover constant $CR = 0.4$</td>
</tr>
<tr>
<td>DE</td>
<td>A weighting factor $F = 0.95$ and a crossover constant $CR = 0.4$</td>
</tr>
<tr>
<td>ES</td>
<td>The number of offspring $k = 10$ produced in each generation, and standard deviation $r = 1$ for changing solutions</td>
</tr>
<tr>
<td>GA</td>
<td>Roulette wheel selection, single point crossover with a crossover probability of 1, and a mutation probability of 0.01</td>
</tr>
<tr>
<td>KH</td>
<td>The foraging speed $V_f = 0.02$, the maximum diffusion speed $D_{max} = 0.005$, the maximum induced speed $N_{max} = 0.01$</td>
</tr>
<tr>
<td>PBIL</td>
<td>A learning rate of 0.05, 1 good population member and 0 bad population members to use to update the probability vector each generation, an elitism parameter of 1, and a 0 probability vector mutation rate</td>
</tr>
<tr>
<td>PSO</td>
<td>An inertial constant = 0.3, a cognitive constant = 1, and a social constant for swarm interaction = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 8</th>
<th>Mean normalized values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>BA</td>
</tr>
<tr>
<td>F01</td>
<td>3.51</td>
</tr>
<tr>
<td>F02</td>
<td>2.65</td>
</tr>
<tr>
<td>F03</td>
<td>2.25</td>
</tr>
<tr>
<td>F04</td>
<td>3.8E3</td>
</tr>
<tr>
<td>F05</td>
<td>1.1E3</td>
</tr>
<tr>
<td>F06</td>
<td>8.68</td>
</tr>
<tr>
<td>F07</td>
<td>1.78</td>
</tr>
<tr>
<td>F08</td>
<td>14.74</td>
</tr>
<tr>
<td>F09</td>
<td>1.21</td>
</tr>
<tr>
<td>F10</td>
<td>1.57</td>
</tr>
<tr>
<td>F11</td>
<td>2.36</td>
</tr>
<tr>
<td>F12</td>
<td>4.85</td>
</tr>
<tr>
<td>F13</td>
<td>21.70</td>
</tr>
<tr>
<td>F14</td>
<td>3.54</td>
</tr>
<tr>
<td>Time</td>
<td>3.18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 9</th>
<th>Best normalized values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>BA</td>
</tr>
<tr>
<td>F01</td>
<td>4.32</td>
</tr>
<tr>
<td>F02</td>
<td>10.52</td>
</tr>
<tr>
<td>F03</td>
<td>2.52</td>
</tr>
<tr>
<td>F04</td>
<td>1.00</td>
</tr>
<tr>
<td>F05</td>
<td>1.00</td>
</tr>
<tr>
<td>F06</td>
<td>6.7E7</td>
</tr>
<tr>
<td>F07</td>
<td>2.0E7</td>
</tr>
<tr>
<td>F08</td>
<td>13.56</td>
</tr>
<tr>
<td>F09</td>
<td>2.68</td>
</tr>
<tr>
<td>F10</td>
<td>1.15</td>
</tr>
<tr>
<td>F11</td>
<td>2.18</td>
</tr>
<tr>
<td>F12</td>
<td>7.94</td>
</tr>
<tr>
<td>F13</td>
<td>52.12</td>
</tr>
<tr>
<td>F14</td>
<td>2.83</td>
</tr>
</tbody>
</table>
They have the best performance on the benchmarks F10 and F11, respectively. It can be seen that replacing inertia weights $\omega$ with chaotic maps can greatly enhance the performance of the KH.

Moreover, the calculated time demands of the ten optimization algorithms were similar. The average running times for these algorithms are collected and presented in Table 8. BA was the most efficient algorithm, while CKH was the tenth fastest of the ten methods.

Furthermore, optimization process of each algorithm is given in Figs. 5–18. The values shown in these figures are the average function optimum achieved from 100 runs. Here, all the values are not normalized, and they are true function values. In addition, it should be noted that the global optima of the benchmarks (F04, F05, and F11) are illustrated in the form of the semi-logarithmic convergence plots. Also, KH II is short for KH in Figs. 5–18.

Fig. 5 shows the values obtained by the ten methods on the F01 function, which is a multimodal function with a narrow global minimum basin ($F01_{\text{min}} = 0$) and many minor local optima. From Fig. 5, CKH performance is superior to the nine methods in optimization process, while KH II performs the second best in this benchmark. Moreover, all the methods have the almost same initial values, while CKH overtakes the other 9 methods. All methods clearly outperform the PBIL algorithm.

Fig. 6 shows the optimization values for F02 function. For this case, the figure clearly shows that CKH, DE and GA, KH II performances differ from each other. From Table 8 and Fig. 6, CKH has the best performance for this multimodal function. ACO, BA, CS, ES, PBIL, and PSO fail to find the best solution in this benchmark function.

![Fig. 5. Performance comparison on the F01 Ackley function.](image)

![Fig. 6. Performance comparison on the F02 Fletcher–Powell function.](image)
Fig. 7. Performance comparison on the F03 Griewank function.

Fig. 8. Performance comparison on the F04 Penalty #1 function.

Fig. 9. Performance comparison on the F05 Penalty #2 function.
Fig. 10. Performance comparison on the F06 Quartic (with noise) function.

Fig. 11. Performance comparison on the F07 Rastrigin function.

Fig. 12. Performance comparison on the F08 Rosenbrock function.
Fig. 7 shows the optimization values for F03 function. F03 has a strange property, as it is much easier to solve for higher dimensions than lower dimensions [32]. From Fig. 7, it can be seen that CKH has the best performance for this benchmark. For other algorithms, ACO, BA, GA, and PSO have the fastest convergence rate initially towards the global optimum. However, they are outperformed by CKH after some generations. Further, ACO and DE work very well because they rank 2 and 3, respectively, among ten algorithms.

Fig. 8 illustrates the values for F04 Penalty #1 function. From Fig. 8, apparently, CKH overtakes all other approaches. For this case, PSO and GA show a faster convergence rate initially, however they seem to be trapped into sub-optimal values as the procedure proceeds, especially PSO. Although slower, DE performs the second best in the optimization process.

Fig. 9 illustrates the values achieved on F05 function. For this benchmark, very similar to F04 in Fig. 8, CKH significantly overtakes all other approaches in the optimization process.

Fig. 10 illustrates the values achieved for the ten methods when using F06. With reference to this benchmark, very similar to F02 Fletcher–Powell function as shown in Fig. 6, the figure reveals that ACO, CKH, DE and GA perform slightly different from each other. However, Table 8 and Fig. 10 show that CKH provides better results for this problem. BA, CS, ES, KH II, PBIL, and PSO do not succeed in this benchmark function. At last, DE and GA converge to the value that is very close to CKH’s.

Fig. 11 illustrates the functions values for the F07 Rastrigin function that is a difficult multimodal function with a unique global minimum of $F07_{min} = 0$ and several local optima. When attempting to solve F07, methods may be converged into a local value. Therefore, a method that can keep a larger diversity is well capable of producing better values. At first glance...
Fig. 15. Performance comparison on the F11 Schwefel 2.22 function.

Fig. 16. Performance comparison on the F12 Schwefel 2.21 function.

Fig. 17. Performance comparison on the F13 Sphere function.
it is clear that CKH greatly overtakes all other approaches. Moreover, PSO has the fastest convergence rate initially towards global value. However, it is outperformed by CKH after 16 generations. Further, DE and GA compares well compared with other algorithms.

Fig. 12 shows the functions values for F08 function, which can also be regarded as a multimodal function. For this case, analogous to the F07 in Fig. 11, clearly, CKH greatly overtakes all other approaches. In addition, PSO has the fastest convergence rate initially towards the global solution. However, it is outperformed by CKH after 22 generations. Further, the performance of DE and GA is very good for this case.

Fig. 13 reveals the equivalent values for the F09 Schwefel 2.26 function. From Fig. 13, GA performs better than other nine methods involving CKH in the optimization process, while ACO performs the second best in this multimodal benchmark. Moreover, KH II is superior to CKH.

Fig. 14 displays the values for F10 function. From Fig. 14, for this case, analogous to the F02 Fletcher–Powell function as shown in Fig. 6, it can be concluded that: (i) the performance of CKH and CS is similar, (ii) KH II is only inferior to CKH and CS, and performs the third best in this benchmark function. However, referring to Table 8 and Fig. 14, CS slightly outperforms CKH and the other approaches. PSO has the fastest convergence rate initially at finding the global minimum. However, it is overtaken by CKH and CS after 26 iterations.

Fig. 15 reveals the function values for F11 Schwefel 2.22 function. From Fig. 15, DE has the stable convergence rate towards the global solution and overtakes all other approaches in this unimodal benchmark function. CKH is only inferior to DE, and performs the second best in this unimodal function.

Fig. 16 reveals the function values for F12 Schwefel 2.21 function. At first glance, it is obvious that CKH has the fastest convergence rate towards the global solution in the whole optimization process. CKH reaches the optimal solution significantly earlier than other algorithms. KH II is only inferior to CKH, and performs the second best in this unimodal function.

Fig. 17 shows the function values for F13 Sphere function, which is also known as de Jong’s function, and has a single global value $F_{13, \text{min}} = 0$, therefore, it is easy to solve. From Fig. 17, CKH has the fastest convergence rate towards the global solution and overtakes all other methods. Fig. 17 shows that GA performs better than DE at the beginning of the search, but DE eventually finds the value of CKH, while GA cannot. KH II fails to find the global value within the maximum number of iterations.

Fig. 18 displays the function values for F14 Step function. CKH shows the fastest convergence rate towards the global solution and overtakes all other methods. Referring to Fig. 18, ACO shows a little faster convergence than DE. Both ACO and DE are only inferior to CKH and reach a minimum that is very close to CKH’s.

Considering the results shown in Figs. 5–18, it can be concluded that the CKH’s performance is superior to or at least quite competitive with the other nine algorithms. Further, benchmarks F02, F04–F08, and F10 illustrate that PSO performs much better than the other methods initially, while later it is overtaken by other approaches.

### 4.3. Comparisons using t-test

Based on the final search results of 100 independent trials on every function, Table 10 presents the $t$ values on every function of the two-tailed test with the 5% level of significance between the CKH and other optimization methods. In the table, the value of $t$ with 198 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test. **Boldface** indicates that CKH performs
significantly better than the compared algorithms. For instance comparing the CKH and DE, the former significantly outperformed the latter on twelve functions (F01, F03–F08, and F10–F14), does as good as the latter on two functions (F02 and F09). This indicates that the CKH generally performs better than DE on the solution accuracy. Though the CKH performed slightly weaker on some functions, Table 10 also reveals that it outperforms the other nine methods for most functions.

4.4. The study of the number of fitness evaluations

In order to fully investigate the advantage of the CKH method, the number of fitness evaluations is further studied. We look at the number of fitness evaluations for every function for each method to the fixed function values. In the experiments, the fixed function values are set to $opt + 1$. $opt$ indicates the optimum for every function that can be found in fifth column of Table 4. The maximum of fitness evaluations is set to 50,000. That is to say, if a method cannot converge to the values $opt + 1$ before the maximum number of fitness evaluations is reached (50,000), it will stop, too, and return only up to 50,000 fitness evaluations. All the results are recorded in Table 11. Table 11 shows that, for the six functions F04–F05, F10, F12–F14, CKH can find the satisfactory results with the least fitness evaluations. KH is somewhat inferior to the CKH method in terms of number of fitness evaluations, and it can converge to the fixed values with the least fitness evaluations for the five functions F02–F03 and F07–F09. Not only do DE and GA also perform well, but also they use the least evaluations when solving the test problems F01, F11 and F06.

5. Discussion

From the experiments conducted in Sections 4.1–4.4 on the benchmarks, it is demonstrated that CKH performs better than or comparable to the basic KH and other optimization approaches. Moreover, the implementation of CKH is simple and easy, and no effect is made to fine-tune the parameters. The work carried out in this paper demonstrates the CKH to be robust over all kinds of benchmark functions.

There are many ways to verify the performance of the meta-heuristic optimization algorithms. Benchmark evaluation is a simple way that is widely used. However, it is not a perfect approach. Firstly, the parameters used in these methods cannot significantly better than the compared algorithms. For instance comparing the CKH and DE, the former significantly outperformed the latter on twelve functions (F01, F03–F08, and F10–F14), does as good as the latter on two functions (F02 and F09). This indicates that the CKH generally performs better than DE on the solution accuracy. Though the CKH performed slightly weaker on some functions, Table 10 also reveals that it outperforms the other nine methods for most functions.

4.4. The study of the number of fitness evaluations

In order to fully investigate the advantage of the CKH method, the number of fitness evaluations is further studied. We look at the number of fitness evaluations for every function for each method to the fixed function values. In the experiments, the fixed function values are set to $opt + 1$. $opt$ indicates the optimum for every function that can be found in fifth column of Table 4. The maximum of fitness evaluations is set to 50,000. That is to say, if a method cannot converge to the values $opt + 1$ before the maximum number of fitness evaluations is reached (50,000), it will stop, too, and return only up to 50,000 fitness evaluations. All the results are recorded in Table 11. Table 11 shows that, for the six functions F04–F05, F10, F12–F14, CKH can find the satisfactory results with the least fitness evaluations. KH is somewhat inferior to the CKH method in terms of number of fitness evaluations, and it can converge to the fixed values with the least fitness evaluations for the five functions F02–F03 and F07–F09. Not only do DE and GA also perform well, but also they use the least evaluations when solving the test problems F01, F11 and F06.

5. Discussion

From the experiments conducted in Sections 4.1–4.4 on the benchmarks, it is demonstrated that CKH performs better than or comparable to the basic KH and other optimization approaches. Moreover, the implementation of CKH is simple and easy, and no effect is made to fine-tune the parameters. The work carried out in this paper demonstrates the CKH to be robust over all kinds of benchmark functions.

There are many ways to verify the performance of the meta-heuristic optimization algorithms. Benchmark evaluation is a simple way that is widely used. However, it is not a perfect approach. Firstly, the parameters used in these methods cannot
be adjusted carefully. It is well-known that for these optimization methods, they always have some randomness. Therefore, different tuning parameters might make a big difference in their performance. Secondly, most practical application problems are not necessarily related with benchmark functions. Finally, benchmark evaluation might arrive at diverse conclusions if we change population size or maximum number of generations [44].

When a meta-heuristic method is used to solve a certain problem, the running time is a key factor. If a method wastes a long time searching for the best or second best solution, it must be impractical [45]. CKH does not seem to demand an impractical amount of computational effort. In comparison with other algorithms considered in this study, CKH was the tenth fastest. In CKH, calculating the chaotic sequence takes a relatively long time. How to make the CKH’s converge faster still deserves further scrutiny.

6. Conclusions

The chaos theory and KH method are hybridized in order to design a novel improved meta-heuristic chaotic KH method (CKH) for solving optimization problems. Various chaotic maps are used to regulate the inertia weights, $\omega$, of KH. The Singer map is selected as its $\omega$ through comparing various chaotic KH variants to form the best CKH. The simulations show that the usage of deterministic chaotic signals instead of linearly decreasing values is an important modification of the KH method. The experimental results show that the tuned KH significantly enhances the reliability of the global optimality and the quality of the solutions. In addition, the results reveal that CKH considerably enhances the KH’s performances on multi-modal and unimodal problems. Moreover, CKH is simple and easy to implement.

Another advantage of CKH is that there are fewer parameters to adjust compared with other population-based optimization methods. The performance of a method that relies on parameters is mainly determined by those parameters. In such cases, it is hard to select the optimal parameters for an optimization method [14]. The present study proposes a new technique to deal with this difficult issue by using chaotic maps in place of these parameters. In CKH, the chaos theory is used to adjust the key parameter, $\omega$. However, the introduction of chaos theory leads to another problem, which makes the approach take a long time to select the best chaotic map in place of the parameter, $\omega$.

A limitation of this study is that only fourteen benchmarks are used to verify the performance of CKH. Thus, in order to evaluate the proposed method more in-depth, more comprehensive problems should be tested such as the high-dimensional ($d \geq 20$) CEC 2005 test suite [39]. Furthermore, CKH may be compared with other state-of-the-art optimization methods. In the future, the CKH method can be extended to solve constrained optimization problems such as the constrained optimization CEC 2005 test suite [39]. Herein, two mechanisms are used to improve the KH method. An interesting topic for future research would be investigating the real impact of elitism and the use of chaos in order to analyze their contributions to the algorithm performance. For more verification, the CKH method might be applied to solve practical engineering problems such as benchmark structural optimization problems. We will also perform mathematical analysis of the proposed method using dynamic system such as Markov chain to prove and explain its convergence.

References