Krill herd algorithm with chaotic time interval and elitism scheme

Shuxia Li and Yuzhe Tian

1. Introduction

The process of optimization is searching for a vector in a function that produces an optimal solution. All of feasible values are available solutions and the extreme value is optimal solution. In general, modern intelligent algorithms are applied to solve optimization problems. A general classification way for these intelligent algorithms is considering the nature of the algorithms, and optimization algorithms can be divided into two main categories: deterministic algorithms, and stochastic algorithms. Deterministic algorithms using gradient such as hill-climbing have a rigorous move, and will generate the same set of solutions if the iterations start with the same initial starting point. On the other hand, stochastic algorithms without using gradient often generate different solutions even with the same initial value (Wang & Tan, 2019). However, generally speaking, the final values, though slightly different, will converge to the same optimal solutions within a given accuracy. The emergence of metaheuristic optimization method as a blessing from the global optimization theorem has opened up a new facet to fulfill the optimization of a function. Recently, nature-inspired metaheuristic algorithms perform powerfully and efficiently in solving modern nonlinear numerical global optimization problems. To some extent, all metaheuristic algorithms strive for making balance between diversification/exploration/randomization (global search) and intensification/exploitation (local search) (Yang, 2010a).

Inspired by nature, these strong metaheuristic algorithms are applied to solve NP-hard problems such as scheduling (Abdel-Basset, Manogaran, El-Shahat, & Mirjalili, 2018; Hu, Yin, & Li, 2011), feature selection (Zhang, Gong, Hu, & Zhang, 2015; Zhang, Song, & Gong, 2017) and detection (Jian, Lam, & Dong, 2014a), path planning (Wang et al., 2012a; Wang, Chu, & Mirjalili, 2016a), floor-planning (Zou, Wang, Pan, & Qi, 2016; Zou, Wang, Sangaiha, & Kong, 2017), cyber-physical social system (Cui, Sun, Wang, Xue, & Chen, 2017; Wang, Cai, Cui, Min, & Chen, 2017), texture discrimination (Jian, Lam, & Dong, 2014b), factor evaluation (Wang, Guo, Duan, Liu, & Wang, 2012b), saliency detection (Jian, Lam, Dong, & Shen,
S. LI AND Y. TIAN

2015), classification (Wang, Lu, Dong, & Zhao, 2016b),
gesture segmentation (Liu, Gong, Meng, Chen, & Wang,
2017a), economic load dispatch (Parouha & Das, 2018;
Rizk-Allah, El-Sehiemy, & Wang, 2018), IIR system iden-
tification (Zou, Deb, & Wang, 2018), shape design (Rizk-
Allah, El-Sehiemy, Deb, & Wang, 2017), big data and large-
scale optimization (Yi et al., 2018a; Yi, Deng, Alavi, &
Wang, 2018b), multi-objective optimization (Sun et al.,
2019), unit commitment (Srikanth et al., 2018), many-
objective optimization (Gong, Sun, & Ji, 2013; Liu, Gong,
Sun, & Jin, 2017b; Liu, Gong, Sun, & Zhang, 2017c), vehi-
cle routing (Chen, Chen, Wang, Gao, & Sangaih, 2018),
knapsack problem (Abdel-Basset & Zhou, 2018; Feng &
Wang, 2018; Feng, Wang, & Wang, 2018a), and fault diag-
nosis (Yi, Wang, & Wang, 2016), path planning (Wang,
Guo, Duan, Liu, & Wang, 2012c; Wang, Guo, Duan, Liu, &
Wang, 2012d), image processing (Zhang & Wang, 2012),
and test-sheet composition (Duan, Zhao, Wang, & Feng,
2012). All the problems that search for extreme value
could be solved by optimization algorithms. These kinds of
metaheuristic algorithms carry out on a population of
solutions and always find optimal or near-optimal solu-
tions. During the 1950s and 1960s, computer scientists
studied the possibility of conceptualizing evolution as
an optimization method and this generated a subset of
gradient-free methods, namely genetic algorithms (GAs)
(Goldberg, 1998; Wang, Gandomi, & Alavi, 2014a). Since
then many other nature-inspired metaheuristic algo-
rithms have emerged, such as bat algorithm (BA) (Wang &
Guo, 2013; Yang & Gandomi, 2012), biogeography-based
optimization (BBO) (Simón, 2008; Wang, Guo, Duan, Liu,
& Wang, 2012e), ant colony optimization (ACO) (Dorigo
& Stutzle, 2004), cuckoo search (CS) (Li, Li, Tian, & Zou,
2019; Yang & Deb, 2010), differential evolution (DE)
(Gandomi, Yang, Talatahari, & Deb, 2012; Storn & Price,
1997), earthworm optimization algorithm (EWA) (Wang, Deb,
& Coelho, 2018a), elephant herding optimization (EHO)
(Wang, Deb, & Coelho, 2015a; Wang, Deb, Gao, & Coelho,
2016c), moth search (MS) algorithm (Wang, 2018), evolu-
tionary strategy (ES) (Beyer, 2001), firefly algorithm (FA)
(Yang, 2010b), artificial bee colony (ABC) (Karaboga &
Basturk, 2007; Liu, Sun, Wang, & Wu, 2018; Wang & Yi,
2018), harmony search (HS) (Geem, Kim, & Loganathan,
2001), monarch butterfly optimization (MBO) (Wang, Deb,
& Cui, 2015b; Wang, Deb, Zhao, & Cui, 2018c), fireworks
algorithm (FWA) (Tan, 2015), and brain storm optimization
(PSO) (Shi, 2011; Shi, Xue, & Wu, 2013), particle swarm
optimization (PSO) (Kennedy & Eberhart, 1995; Sun, Jiao,
Deng, & Wang, 2017), genetic programming (Gandomi &
Alavi, 2011), cuckoo search (CS) (Gandomi, Yang, & Alavi,
2013a; Wang et al., 2012a; Wang, Deb, Gandomi, Zhang,
& Alavi, 2016d; Wang, Gandomi, Zhao, & Chu, 2016e) and
more recently, the krill herding (KH) algorithm (Gandomi &
Alavi, 2012; Wang, Gandomi, & Alavi, 2014b; Wang, Gan-
domi, Alavi, & Deb, 2016f) that is based on imitating the
herding behaviour of krill individuals in nature.

Firstly presented by Gandomi and Alavi in 2012,
inspired by the herding behaviour of krill individuals, krill
herd (KH) is a novel swarm intelligence approach for opti-
mizing possibly non-differentiable and nonlinear func-
tions in continuous space (Gandomi & Alavi, 2012). In KH
algorithm, the objective function for the krill movement
is decided by the minimum distances of each krill indi-
vidual from food and from the highest density of the
herd. The time-dependent position of the krill individ-
uals consists of three main components: (i) movement
led by other individuals (ii) foraging motion, and (iii) ran-
dom physical diffusion. One of the remarkable advantage
of the KH algorithm is that the KH needs not derivative
information because it uses a stochastic search rather
than a gradient search. Furthermore, comparing with
other population-based metaheuristic algorithms, this
new method requires few control variables, in essence
only a single parameter $\Delta t$ (time interval) to regulate,
which makes KH easy to implement, more robust, and is
very appropriate for parallel computation.

KH is an efficient and powerful algorithm in exploita-
tion but at times it may trap into some local optima so
that it cannot implement global search well. For KH, the
search depends completely on random walks, so there is
no guarantee for a fast convergence. In order to improve
KH in optimization problems, a method has been pro-
posed (Wang et al., 2014c), which introduces mutation
scheme into KH to add the diversity of population. Also,
recently, some other updating strategies have been pro-
posed to further improve the performance of the stan-
ard KH algorithm (Wang, Gandomi, Alavi, & Deb, 2016g).

On the other hand, recent significant advances in the-
ories and applications of nonlinear dynamics, especially
of chaos, have drawn more attention in many fields. One
of these fields is the applications of chaos in function
optimization methods (Yang, Li, & Cheng, 2007).

Antecedently, chaotic sequences have been intro-
duced into some heuristic optimization methods such as
firefly algorithm (FA) (Gandomi, Yang, Talatahari, & Alavi,
2013b), monarch butterfly optimization (MBO) (Feng,
Yang, Wu, Lu, & Zhao, 2018b), harmony search (HS)
(Alatas, 2010), imperialist competitive algorithm (ICA)
(Talatahari, Farahmand Azar, Sheikholeslami, & Gandomi,
2012) and particle swarm optimization (PSO) (Gandomi,
Yun, Yang, & Talatahari, 2013c; Wang, Gandomi, & Alavi,
2013a).

Firstly presented here, chaotic KH-based methods are
proposed in this paper, with the aim of accelerating con-
vergence speed, thus making the approach more feasible
for a wider range of practical applications without losing
the attractive characteristics of the standard KH. In these algorithms, we use thirteen different 1-D chaotic maps to replace the parameter of the KH. Therefore different methods that use chaotic maps as quite efficient alternatives to pseudorandom sequences have been proposed. The proposed method is evaluated on twenty-four standard benchmark functions that have ever been applied to verify optimization algorithms in continuous optimization problems. Experimental results show that the CKH performs more efficiently and accurately than standard KH, ACO, BA, CS, DE, ES, GA, HS, PBIL, and PSO. The results reveal the improvement of the novel methods because of the application of deterministic chaotic signals in place of constant time interval.

The structure of this paper is organized as follows: Section 2 gives a brief description of the standard KH algorithm and 13 chaotic maps. Our proposed method is presented in detail in Section 3. Subsequently, the tuning of the time interval and finding the optimal chaotic KH are discussed in Section 4. In addition, comparing with ACO, BA, CS, DE, ES, GA, HS, PBIL, and PSO, our method is also evaluated through twenty-four benchmark functions. Finally, Section 5 involves the conclusion and proposals for future work.

2. Preliminary

At first, in this section we will provide a background on the krill herd algorithm and 13 chaotic maps in brief.

2.1. Krill herd algorithm

Krill herd (KH) (Gandomi & Alavi, 2012) is a novel metaheuristic optimization method for solving optimization problems, which is based on the imitation of the herding of the krill swarms in response to specific biological and environmental processes. The time-dependent position of an individual krill in 2-D surface is decided by three main actions described as follows:

(1) movement influenced by other krill individuals;
(2) foraging action;
(3) random physical diffusion.

Kh algorithm adopted the following Lagrangian model in a d-dimensional decision space as shown in Equation (1).

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

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

In movement induced by other krill individuals, the direction of motion induced, \(\alpha_i\), is estimated by the target swarm density (target effect), a local swarm density (local effect), and a repulsive swarm density (repulsive effect). For a krill individual, this movement can be given as

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

and \(N_i^{\text{max}}\) is the maximum induced speed, \(\omega_n\) is the inertia weight of the motion induced in [0, 1], \(N_i^{\text{old}}\) is the last motion induced.

The foraging motion is determined by the two main factors. One factor is the food location and the other one is the previous experience about the food location. For the \(i\)th krill individual, this motion can be defined as follows:

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

where

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

and \(V_i\) is the foraging speed, \(\omega_f\) is the inertia weight of the foraging motion between 0 and 1, \(F_i^{\text{old}}\) is the last foraging motion.

The physical diffusion of the krill individuals can be treated as a random process. This motion can be described according to a maximum diffusion speed and a random directional vector. It can be expressed as follows:

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

where \(D_i^{\text{max}}\) is the maximum diffusion speed, and \(\delta\) is the random directional vector and its arrays are random values in \([-1, 1]\).

Based on the three above-mentioned movements, using different effective parameters of the motion during the time, the position vector of a krill individual during the interval \(t\) to \(t + \Delta t\) is formulated by the following equation:

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

Note that \(\Delta t\) is one of the most important constants and should be carefully regulated in terms of the given practical optimization problem. This is because this parameter can be considered to be a scale factor of the speed vector. \(\Delta t\) completely relies on the search space and it seems to be simply obtained from the following expression:

\[
\Delta t = C_i \sum_{j=1}^{NV} (UB_j - LB_j)
\]

where \(NV\) is the total number of variables, and \(UB_j\) and \(LB_j\) are upper and lower bounds of the \(j\)th variables (\(j = 1,\)
2, . . . , NV), respectively. Thus, the absolute of their subtraction shows the search space. Very clearly, low values of \( \Delta t \) make the krill individuals perform the search in the space carefully. More details about the three main motions and KH algorithm can be found in (Bolaji, Al-Betar, Awadallah, Khader, & Abualigah, 2016; Gandomi & Alavi, 2012; Wang, Gandomi, Alavi, & Gong, 2019).

### 2.2. Chaotic maps

In random-based optimization methods, the methods using chaotic variables instead of random variables are called chaotic optimization algorithm (COA). In these algorithms, because chaos has the feature of the non-repetition and ergodicity, it can perform overall searches at higher speeds than stochastic searches that rely on probabilities. To fulfill this matter, herein 1-D non-invertible maps are used to produce chaotic sets. In our present study, the following 13 well-known 1-D chaotic maps are applied to form chaotic KH represented in Table 1. More details about the 13 chaotic maps can be found in (Gandomi et al., 2013b).

### Table 1. Thirteen chaotic maps.

| ID | Name                  | ID | Name                | ID | Name            |
|----|-----------------------|----|---------------------|----|-----------------|
| M1 | Chebyshev map         | M6 | Liebowski map       | M10| Singer map      |
| M2 | Circle map            | M7 | Logistic map        | M11| Sinusoidal map  |
| M3 | Gauss/Mouse map       | M8 | Piecewise map       | M12| Tent map        |
| M4 | Intermittency map     | M9 | Sine map            | M13| Sawa tooth map  |
| M5 | Iterative map         |    |                     |    |                 |

The standard KH is very efficient and powerful, but the solutions have slight changes as the optima are approaching. In KH, the time interval \( \Delta t \) is unaltered constant evaluated by Equation (7).

In the standard KH, there is no need to keep constant time interval value. As a matter of fact, a bigger time interval emphasizes exploration at the beginning of the search, while a smaller time interval encourages exploitation and makes the krill individuals to search the space carefully at the end of the search. Therefore, a chaotic varying time interval \( \Delta t \) may be advantageous, which may also result in the speedup of convergence of the method as we will see in the next section. As all chaotic maps are normalized, the variations of a chaotic map are always in the range \([0, 1]\). Therefore, chaotic maps can be applied to tune the parameter of the time interval \( \Delta t \), and such chaos-based KH is called the chaotic KH. The pseudo code of chaotic KH algorithm can be simply summarized as shown in Figure 1.

The tuning of time interval \( \Delta t \) using chaotic maps also enhance the ability of the KH to evade trapping into local optima in a multimodal landscape, thus making the method more feasible for a wider range of applications but without losing the attractive characteristics of the original method. Comparing with other optimization methods, this could be an advantage for this algorithm as we can see in the simulations below.

In addition, another important improvement is the addition of elitism scheme into the CKH. As with other population-based optimization methods, we typically combine some sort of elitism so as to keep the best solutions in the population. In the major cycle of the CKH, to begin with, the \( KEEP \) best solutions are stored in a variable \( KEEPKRILL \). In general, eventually the \( KEEP \) worst solutions are replaced by the \( KEEP \) best solutions. This elitism scheme can guarantee that the whole population cannot be declined to the population with worse fitness than the former. This prevents the best solutions from being ruined by motion calculation operator. Note that we use an elitism strategy to save the property of the krill that has the best fitness in the CKH process, so even if motion calculation operation corrupts its corresponding krill, we have stored it and can recover to its former good status if needed.

### 3. Our approach: CKH

As it is presented in Equation (6), the main parameter of the KH is the time interval \( \Delta t \) that can be considered to be a scale factor of the speed vector and features the variations of the global best attraction, and its value is of vital importance in determining the speed of the convergence and how the KH works.

The standard KH is very efficient and powerful, but the solutions have slight changes as the optima are approaching. In KH, the time interval \( \Delta t \) is unaltered constant evaluated by Equation (7).

### 4. Simulation experiments

In this section, we evaluate the performance of our proposed method CKH to global numerical optimization through a series of experiments conducted in benchmark functions.

To allow an unbiased comparison of running times, all the experiments were implemented on a PC with a Pentium IV processor running at 2.0 GHz, 512 MB of RAM and a hard drive of 160 Gbytes. Our implementation was compiled using MATLAB R2012a (7.14) running under Windows XP3. No commercial KH tool was used in the following simulation experiments.

Well-defined problem sets are good for testing the performance of optimization methods proposed in our work. Based on mathematical functions, benchmark functions can be applied as objective functions to carry out such tests. In our study, twenty-four different benchmark functions are applied to verify our proposed metaheuristic method CKH. The benchmark functions described in Table 2 are standard testing functions. More details of all
**Krill herd algorithm with chaotic time interval and elitism scheme**

Begin

**Step 1: Initialization.** Set the generation counter \( t = 1 \); initialize the population \( P \) of \( NP \) krill individuals randomly and each krill corresponding to a potential solution to the given problem; set the foraging speed \( V_f \), the maximum diffusion speed \( D_{\text{max}} \), and the maximum induced speed \( N_{\text{max}} \); set initial value of the chaotic map \( c_0 \) randomly; time interval \( \Delta t = c_0 \) and elitism parameter \( \text{KEEP} \): how many of the best krill to keep from one generation to the next.

**Step 2: Fitness evaluation.** Evaluate each krill individual according to its position.

**Step 3: While** the termination criteria is not satisfied or \( t < \text{MaxGeneration} \) do

Sort the population/krill from best to worst.

Store the KEEP best krill as \( \text{KEEPKRILL} \).

for \( i = 1 \) to \( NP(\text{all krill}) \) do

Perform the following motion calculation.

Motion induced by the presence of other individuals
Foraging motion
Physical diffusion
Update time interval \( \Delta t \) using chaotic maps (\( \Delta t = c_{i+1} \)).
Update the krill individual position in the search space by Eq. (6).
Evaluate each krill individual according to its new position \( x_{i+1} \).
end for

Replace the KEEP worst krill with the KEEP best krill stored in \( \text{KEEPKRILL} \).
Store the population/krill from best to worst and find the current best.

\( t = t + 1 \);

**Step 4: end while**

**Step 5: end while**

End.

---

**Figure 1.** Pseudo code of krill herd algorithm with chaotic time interval and elitism scheme.

**Table 2.** Benchmark functions.

| ID | Name       | ID | Name       | ID | Name       | ID | Name       |
|----|------------|----|------------|----|------------|----|------------|
| F01| Ackley     | F07| Rastrigin  | F13| Sphere     | F19| Hartman    |
| F02| Fletcher-Powell | F08| Rosenbrock | F14| Step       | F20| Kowalik    |
| F03| Griewank   | F09| Schwefel 2.26 | F15| Branin     | F21| Shekel 1   |
| F04| Penalty #1 | F10| Schwefel 1.2 | F16| Shekel’s Foxholes | F22| Shekel 2   |
| F05| Penalty #2 | F11| Schwefel 2.22 | F17| Goldstein and Price | F23| Shekel 3   |
| F06| Quartic    | F12| Schwefel 2.21 | F18| Camel Back–6 Hump | F24| Hartman 2  |

the benchmark function can be found in (Ali, Khompattapon, & Zabinsky, 2005; Yao, Liu, & Lin, 1999).

**4.1. The performance of CKH with thirteen different chaotic maps**

Different chaotic KH variants were benchmarked using fourteen well-known numerical examples. In our present study, we use the benchmark functions F01–F14 to select the optimal chaotic map. These benchmark functions are provided in Table 2.

It is feasible to improve the solution quality by using the chaotic maps. In this subsection, tuning the time interval \( \Delta t \) are carried out. Here, the value of \( \Delta t \) is replaced with different chaotic maps represented in Subsection 2.2. In order to fulfill this, all the maps are normalized in [0, 1].

We set population size \( NP = 50 \), an elitism parameter \( \text{Keep} = 2 \) and maximum generation \( \text{Maxgen} = 50 \) for CKH algorithm. We ran 100 Monte Carlo simulations of CKH method on fourteen selected benchmarks to get typical performances. Table 3 illustrates the results of the simulations. The first half part in Table 3 shows the average minima found by CKH algorithm, averaged over 100 Monte Carlo runs. The second half part in Table 3 shows the absolute best minima found by CKH algorithm over 100 Monte Carlo runs. The best value obtained for each benchmark is marked in bold. Note that the normalizations for average and best values in the tables are based on different scales, so values cannot be compared
between the two parts. Each of the functions in this experiment has 20 independent variables (i.e. $d = 20$).

From Table 3, it can be seen that the M9 (Sine map) is significantly outperforms and more effective than other chaotic maps when multiple runs are made. Therefore, it is selected as the best map to be used for chaotic KH (CKH) which will be compared and discussed in more detail in the next subsection.

### 4.2. General performance of CKH

In order to prove the superiority of CKH, in this subsection we compared its performance on global numeric optimization problem with nine other population-based optimization methods (except KH), which are ACO, BA, CS, DE, ES, GA, HS, PBIL, and PSO. ACO (ant colony optimization) (Dorigo & Stutzle, 2004) is an efficient swarm intelligence algorithm for solving optimization problems which is based on the pheromone deposition of ants. BA (bat algorithm) (Yang & Gandomi, 2012) is a new powerful metaheuristic optimization method inspired by the echolocation behaviour of bats with varying pulse rates of emission and loudness. CS (cuckoo search) (Gandomi et al., 2013a) is a metaheuristic optimization algorithm inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. DE (differential evolution) (Storn & Price, 1997) is a simple but robust optimization method that uses the difference between two solutions to probabilistically adapt a third solution. An ES (evolutionary strategy) (Beyer, 1994) is an evolutionary algorithm (Hao, Lim, Ong, Huang, & Wang, 2019) that generally distributes equal importance to mutation and recombination, and that allows more than two parents to reproduce an offspring. A GA (genetic algorithm) (Goldberg, 1998) is a search heuristic that mimics the process of natural evolution. HS (harmony search) (Geem et al., 2001; Wang et al., 2013b) is a new metaheuristic approach inspired by behaviour of musicians’ improvisation process. PBIL (probability-based incremental learning) (Baluja, 1994) is a type of genetic algorithm where the genotype of an entire population (probability vector) is evolved instead of individual members. PSO (particle swarm optimization) (Kennedy & Eberhart, 1995) is also a swarm intelligence algorithm which is based on the swarm behaviour of bird, and fish schooling in nature. In addition, we must point out that, in (Gandomi & Alavi, 2012), Gandomi and Alavi have proved that, comparing all the algorithms, the KH II (KH with crossover operator) performed the which confirms the robustness.

---

**Table 3.** Best normalized optimization results in fourteen benchmark functions with twelve different chaotic maps.

| M1  | M2  | M3  | M4  | M5  | M6  | M7  | M8  | M9  | M10 | M11 | M12 | M13 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| F0  | 1.06| 1.12| 1.02| 3.28| 1.04| 3.89| 1.08| 1.04| 1.00| 1.75| 1.00| 1.05| 1.73 |
| F02 | 1.12| 1.09| 1.07| 2.53| 1.00| 4.40| 1.04| 1.02| 1.00| 1.78| 1.08| 1.00| 1.17 |
| F03 | 1.08| 1.64| 1.30| 20.10| 1.09| 35.65| 1.09| 1.12| 1.24| 16.07| 1.27| 1.00| 5.28 |
| F04 | 1.00| 4.65| 2.23| 1.723| 1.13| 6.263| 1.26| 1.45| 5.79| 2.663| 1.98| 1.50| 56.89 |
| F05 | 1.07| 3.60| 1.75| 341.49| 1.52| 978.49| 1.48| 1.00| 3.09| 375.41| 1.62| 1.48| 23.65 |
| F06 | 1.03| 2.28| 1.45| 59.92| 1.00| 183.83| 1.10| 1.23| 1.31| 73.31| 1.47| 1.16| 7.15 |
| F07 | 1.06| 1.11| 1.08| 2.27| 1.09| 3.12| 1.03| 1.07| 1.00| 1.85| 1.04| 1.11| 1.29 |
| F08 | 1.21| 1.65| 1.50| 12.61| 1.21| 30.52| 1.33| 1.20| 1.00| 11.51| 1.39| 1.34| 3.15 |
| F09 | 1.11| 1.04| 1.05| 1.58| 1.07| 1.89| 1.09| 1.05| 1.10| 1.27| 1.00| 1.03| 1.07 |
| F10 | 1.06| 1.19| 1.17| 2.21| 1.15| 4.10| 1.08| 1.15| 1.00| 2.12| 1.12| 1.08| 1.33 |
| F11 | 1.01| 1.08| 1.08| 1.90| 1.04| 3.06| 1.00| 1.05| 1.05| 1.44| 1.04| 1.50| 1.73 |
| F12 | 1.05| 1.17| 1.06| 4.35| 1.07| 6.41| 1.00| 1.02| 1.02| 2.43| 1.08| 1.02| 1.99 |
| F13 | 1.14| 2.05| 1.52| 25.57| 1.23| 48.96| 1.23| 1.13| 1.00| 16.04| 1.47| 1.19| 6.45 |
| F14 | 1.13| 1.94| 1.42| 24.51| 1.02| 49.71| 1.10| 1.09| 1.30| 12.39| 1.41| 1.00| 6.17 |
| Total| 1.02| 1.02| 1.02| 1.03| 1.02| 1.07| 1.00| 1.01| 1.00| 1.01| 1.01| 1.01| 1.01 |

Notes: The values shown are the minimum objective function values found by CKH algorithm. The first half part in Table 3 shows the average minima found by CKH algorithm, averaged over 100 Monte Carlo runs. The second half part in Table 3 shows the absolute best minima found by CKH algorithm over 100 Monte Carlo runs.

*The values are normalized so that the minimum in each row is 1.00.*
Carlo runs. In other words, Table 4 shows the average per-
best minima found by each algorithm over 100 Monte Carlo
simulations of each method on each benchmark.

Looking carefully at Table 4, for high-dimensional bench-
marks F15-F18 (d = 2), we set loudness A = 0.95, pulse
rate r = 0.5, and scaling factor ϵ = 0.1; for CS, a discovery
rate p_c = 0.25. For HS, we set harmony memory accepting
rate = 0.75, and pitch adjusting rate = 0.7.

For each algorithm, we set population size NP = 50, an
elitism parameter Keep = 2 and maximum gener-
ation Maxgen = 50 for benchmark functions F01-F14
(d = 20); we set NP = 15, Keep = 2 and Maxgen = 20 for
benchmark functions F15-F18 (d = 2); we set NP = 20,
Keep = 2 and Maxgen = 30 for benchmark functions
F19-F23 (d = 4); we set NP = 40, Keep = 2 and Max-
gen = 30 for benchmark F24 (d = 6). We ran 100 Monte
Carlo simulations of each method on each benchmark
function to get representative performances. Tables 4 and
5 illustrate the results of the simulations. Table 4 shows
the average minima found by each method, averaged
over 100 Monte Carlo runs. Table 5 shows the absolute
best minima found by each algorithm over 100 Monte Carlo
runs. In other words, Table 4 shows the average per-
f each test problem is marked in bold. Note that the normal-
izations in Tables 4 and 5 are based on different scales,
so values cannot be compared between the two tables.
In our present study, benchmarks F01-F14, F15-F18, F19-
F23 and F24 have 20, 2, 4, and 6 independent variables
(i.e. d = 20, 2, 4 and 6), respectively.

From Table 4, we see that, on average, CKH is the most
effective at finding objective function minimum on sev-
ten of the twenty-four benchmarks (F01-F08, F12-F16,
and F21-F23). DE is the second most effective, perform-
ing best on eight of the twenty-four benchmarks (F11,
F11-F20 and F24) when multiple runs are made. CS and
GA are the third most effective, performing best on four
of the twenty-four benchmarks (F10, F15, F16, F18 and
F09, F16-F18, respectively) when multiple runs are made.
Looking carefully at Table 4, for high-dimensional bench-
marks F01-F14 (d = 20), CKH performs significantly better
than other algorithms by and large. However, for bench-
marks F15-F24 (d = 2, 4, and 6), there is little difference
between CKH and other algorithms, especially for bench-
marks F15-F18 (d = 2). This phenomenon demonstrates
the superiority of CKH in solving the high-dimensional
function optimization problems with certain appropriate
chaotic map.

From Table 5 shows that CKH performs the best on
twenty of the twenty-four benchmarks (F01, F03, F05-
F08, and F10-F24). DE and KH are the second and third
most effective, performing the best on the benchmarks

| ID | ACO | BA | CHK | CS | DE | ES | GA | HS | KH | PBIL | PSO |
|----|-----|----|-----|----|----|----|----|----|----|------|-----|
| F01| 3.41 | 4.28 | 1.00 | 3.77 | 2.71 | 4.17 | 3.74 | 4.28 | 1.23 | 4.34 | 3.60 |
| F02| 2.66 | 3.90 | 1.00 | 1.96 | 1.07 | 2.89 | 1.10 | 2.58 | 2.84 | 2.42 | 2.22 |
| F03| 2.38 | 4.19 | 1.00 | 15.72 | 4.08 | 19.64 | 7.54 | 38.24 | 6.40 | 43.09 | 14.97 |
| F04| 4.683 | 5.13 | 1.00 | 285.94 | 18.07 | 21.63 | 36.14 | 3.533 | 287.57 | 4.883 | 354.06 |
| F05| 717.57 | 912.75 | 1.00 | 984.3 | 9.61 | 429.93 | 22.56 | 680.56 | 77.60 | 907.56 | 112.72 |
| F06| 9.30 | 161.94 | 1.00 | 22.85 | 3.71 | 126.20 | 10.54 | 121.71 | 28.66 | 156.23 | 29.93 |
| F07| 1.94 | 2.84 | 1.00 | 2.21 | 1.68 | 2.64 | 1.82 | 2.49 | 2.17 | 2.69 | 1.99 |
| F08| 16.86 | 18.89 | 1.00 | 4.95 | 2.57 | 21.98 | 4.26 | 15.49 | 5.94 | 18.65 | 5.39 |
| F09| 1.20 | 4.14 | 2.20 | 3.03 | 2.29 | 2.79 | 1.00 | 3.44 | 4.76 | 3.64 | 3.48 |
| F10| 1.59 | 4.11 | 1.14 | 1.00 | 2.25 | 2.52 | 1.71 | 2.35 | 4.79 | 2.51 | 1.72 |
| F11| 2.34 | 3.72 | 1.29 | 2.34 | 1.00 | 3.54 | 1.69 | 2.85 | 5.210 | 2.87 | 2.02 |
| F12| 4.22 | 7.03 | 1.00 | 4.90 | 5.28 | 6.50 | 5.52 | 6.72 | 1.52 | 6.92 | 5.47 |
| F13| 20.18 | 41.34 | 1.00 | 15.82 | 3.74 | 44.28 | 14.08 | 37.83 | 6.77 | 44.14 | 15.34 |
| F14| 3.67 | 45.30 | 1.00 | 16.35 | 4.08 | 28.06 | 8.36 | 42.03 | 6.79 | 46.35 | 16.20 |
| F15| 1.02 | 2.29 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.21 | 1.00 | 1.00 |
| F16| 1.00 | 12.81 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| F17| 1.00 | 4.24 | 1.73 | 1.01 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| F18| 1.00 | 1.35 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| F19| 3.07 | 4.10 | 1.54 | 1.19 | 1.00 | 1.85 | 3.02 | 2.80 | 7.76 | 3.05 | 1.71 |
| F20| 1.39 | 20.94 | 4.07 | 1.29 | 1.00 | 1.97 | 2.58 | 3.65 | 1.93 | 2.88 | 2.37 |
| F21| 1.34 | 1.47 | 1.00 | 1.29 | 1.10 | 1.31 | 1.30 | 1.35 | 1.25 | 1.40 | 1.34 |
| F22| 1.46 | 1.60 | 1.00 | 1.38 | 1.20 | 1.45 | 1.39 | 1.48 | 1.33 | 1.53 | 1.47 |
| F23| 1.45 | 1.61 | 1.00 | 1.40 | 1.24 | 1.45 | 1.39 | 1.48 | 1.33 | 1.53 | 1.47 |
| F24| 1.62 | 1.88 | 1.02 | 1.18 | 1.00 | 1.87 | 1.13 | 1.47 | 1.61 | 1.50 | 1.68 |
| Time| 10 | 2 | 5 | 6 | 4 | 7 | 11 | 8 | 3 | 1 | 9 |
| Total| 3 | 0 | 17 | 4 | 8 | 1 | 4 | 1 | 0 | 1 | 1 |

*The values are normalized so that the minimum in each row is 1.00. These are not the absolute minima found by each algorithm, but the average minima found by each algorithm.

Table 4. Mean normalized optimization results in twenty-four benchmark functions. The values shown are the minimum objective function values found by each algorithm, averaged over 100 Monte Carlo simulations.
In Table 4, for benchmarks F01-F14 (d = 20), CKH performs significantly better than other algorithms. However, for benchmarks F15-F24 (d = 2, 4, and 6), there is slight difference between CKH and other algorithms. Furthermore, for benchmarks F15-F18 (d = 2), all the eleven optimization algorithms perform almost identically with each other. Benchmark functions F15-24 are too simple to clarify the performance difference among eleven methods. As it can be seen from Tables 4 and 5, replacing time interval $\Delta t$ with chaotic maps can definitely improve the performance of the KH.

Moreover, the computational requirements of the eleven optimization algorithms were homologous. We congregated the average computational time of the optimization algorithms as used to the 24 benchmarks discussed in this section. The results are shown in the last row in Table 4. For this case, we can see that, clearly PBIL was the most efficient optimization method. CKH was the fifth fastest of the eleven methods. However, note that in the great majority of practical applications, it is the fitness function evaluation that is by far the most expensive part of a population-based optimization method.

Further, limited by the length of the paper, some convergence graphs of ACO, BA, CKH, CS, DE, ES, GA, HS, KH II, PBIL, and PSO are shown in Figures 2–7 which represent the process of optimization. Figures 2–7 illustrate the convergence plot for high-dimensional benchmarks F01, F07, F12, F13 (d = 20) and low-dimensional benchmarks F21, F22 (d = 4), respectively. The values shown in Figures 2–7 are the mean objective function optimum achieved from 100 Monte Carlo simulations, which are the true objective function values, not normalized. In addition, we use KH short for KH II in the legend of Figures 2–7.

Figure 2 shows the results obtained for the eleven methods when the F01 Ackley function is applied. Ackley function is a multimodal function with a narrow global minimum basin (F01min = 0) and many minor local optima. From Figure 2, clearly, we can draw the conclusion that, CKH is significantly superior to the other algorithms during the process of optimization, while KH II performs the second best in this multimodal benchmark function. Here, all the algorithms show the almost same starting point, while CKH has a stably faster convergence rate than other algorithms. All though slower, DE performs the third best at finding the global minimum. All methods clearly outperform the PBIL algorithm.

Figure 3 shows the optimization results for the F07 Rastrigin function, which is a complex multimodal problem with a unique global minimum of F07min = 0 and a
Figure 2. Comparison of the performance of the different methods for the F01 Ackley function.

Figure 3. Comparison of the performance of the different methods for the F07 Rastrigin function.

Figure 4. Comparison of the performance of the different methods for the F12 Schwefel 2.21 function.

Figure 5. Comparison of the performance of the different methods for the F13 Sphere function.

Figure 6. Comparison of the performance of the different methods for the F21 Shekel 1 function.

Figure 7. Comparison of the performance of the different methods for the F22 Shekel 2 function.
great number of local optima. When trying to solve F07, methods may easily trap into a local optimum. Therefore, a method capable of maintaining a larger diversity is likely to produce better solutions. At first glance, it is clear that, CKH significantly outperforms all other approaches. For other algorithms, PSO has the fastest convergence rate initially at finding the global minimum. However, it seems to be attracted to sub-minima as the distance from the global minimum increases slightly and it is outperformed by CKH after 15 generations. Further, DE and GA works very well because the DE and GA have ranks of 2, 3, respectively, among eleven algorithms.

Figure 4 shows the results for F12 Schwefel 2.21 function. At first glance, it is obvious that CKH has the fastest convergence rate finding the global minimum in the whole optimization progress. CKH reaches the optimal solution significantly superiorly to other algorithms. KH II is only inferior to CKH, and performs the second best in this unimodal function.

Figure 5 shows the results for F13 Sphere function, which is also known as de Jong’s function, and has a unique global minimum of $F13_{\text{min}} = 0$, therefore it is easy to solve. From Figure 5, CKH has the fastest convergence rate at finding the global minimum and outperforms all other approaches. Looking carefully at Figure 5, for DE and PSO, we can see that PSO has a faster convergence rate than DE, but DE does finally converge to the value of CKH, while PSO not. KH II does not manage to succeed in this relatively simple problem within the maximum number of iterations, showing a wide range of obtained results.

Figure 6 shows the results for F21 Shekel 1 function. From Figure 6, clearly, we can draw the conclusion that, CKH is significantly superior to the other algorithms during the process of optimization, while DE performs the second best in this multimodal benchmark function.

Figure 7 shows the results for F22 Shekel 2 function. Apparently, from Figure 7, analogous to the F21 Shekel 1 function as shown in Figure 6, CKH shows the fastest convergence rate at finding the global minimum and significantly outperforms all other approaches. Looking carefully at Figure 7, for other algorithms, DE performs the second best in this multimodal benchmark function and it is only inferior to CKH and reaches the minimum that is very close to CKH’s. Further, KH II works very well because the KH II ranks 3, among eleven algorithms. For other algorithms, ACO, BA, CS, ES, HS, PBIL, and PSO do not manage to succeed in this benchmark function within maximum number of generations, showing a wide range of obtained results.

From above-analyses about Figures 2–7, we can draw the conclusion that the CKH’s performance is superior to or at least quite competitive with the other ten acclaimed state-of-the-art population-based algorithms. In general, DE and KH II are only inferior to the CKH. Further, benchmark F07 also illustrates that PSO has a much faster convergence rate initially, while later it converges slower and slower to the true objective function value.

4.3. Discussion

For all the standard benchmark functions discussed in this section, the CKH has been demonstrated to perform better than or at least highly competitive with the standard KH and other acclaimed state-of-the-art population-based methods. The advantages of CKH involve performing simply and easily, and have no parameter to tune. The work conducted here proves the CKH to be robust, powerful and efficient over all kinds of benchmark functions.

Benchmark testing is a good way for evaluating the performance of the metaheuristic methods, but it is also not perfect and has some limitations. First, we did not do much work painstakingly to carefully regulate the optimization methods in this section. In general, different tuning parameter values in the optimization methods might lead to significant differences in their performance. Second, practical optimization problems may have little of a relationship to benchmark functions. Third, benchmark tests may come to quite different conclusions if the grading criteria or problem setup change. In our present study, we investigated the mean and best results achieved with some population size and after some number of iterations. However, we might draw different conclusions if (for example) we change the population size, or look at how many population size it needs to arrive at a certain function value, or if we change the iteration. Regardless of these caveats, the benchmark results shown here are prospective for CKH, and indicate that this novel method might be capable of finding a niche among the plethora of population-based optimization methods.

It should be noted that CPU time is a bottleneck to the implementation of many population-based optimization methods. If a method can’t converge fast, it will be impractical and infeasible, since it would take too long to find an optimal or near-optimal solution. CKH seems not to require an unreasonable amount of computational requirements; of the eleven optimization methods compared in this paper, CKH was the fifth fastest. How to speed up the CKH’s convergence still deserves further scrutiny.

In our present work, 24 benchmark functions are used to verify the performance of our method; we will test our proposed method on more problems, such as the high-dimensional ($d \geq 20$) CEC 2010 test suit (Tang, Li, Suganthan, Yang, & Weise, 2010) and the practical engineering problems. Moreover, we will compare CKH with other
optimization methods. In addition, we only consider the unconstrained function optimization in this study. Our future work consists on adding the diversity rules into CKH for constrained optimization problems, such as constrained real-parameter optimization CEC 2010 test suit (Mallipeddi & Suganthan, 2010).

5. Conclusion and future work

Chaos has been introduced to the standard KH to develop a novel improved metaheuristic chaotic KH method for optimization problem. Thirteen different chaotic maps have been investigated to tune the time interval, $\Delta t$, of the KH. By comparing different chaotic KH variants, the algorithm that uses the Sine map as its $\Delta t$ is the best chaotic KH (CKH). The results have also revealed the significant improvement of the new method with the application of deterministic chaotic signals instead of constant time interval value. This new method can speed up the global convergence rate without losing the strong robustness of the standard KH. From the analysis of the experimental results, we observe that the tuned KH clearly improves the reliability of the global optimality and they also improve the quality of the solutions. Based on the results of the eleven approaches on the test problems, we can conclude that the CKH significantly improves the performances of the KH on most multimodal and unimodal problems. In addition, the CKH is simple and implements easily.

Further, it can be observed that, comparing with other population-based optimization methods the main advantage of the CKH is that there is a fewer number of parameters. All methods have method-dependent parameters, and the performance of a method is mainly influenced by these parameters. Therefore, it is often one of the most difficult tasks to find optimal values for parameters in an optimization method. In this paper, we provided a strategy to solve this challenging issue by using chaotic maps to replace these parameters. In CKH, there is only one parameter, $\Delta t$, to be tuned using chaotic maps.

In the field of numerical optimization, there are many issues that are worthy of further study, and some more efficient optimization methods should be developed depending on the analysis of specific engineering problem. Our future work will focus on the two issues. On the one hand, we would apply our proposed method CKH to solve practical engineering optimization problems, and, obviously, CKH can be a promising method for real-world engineering optimization problems. On the other hand, we would develop more new metaheuristic method to solve optimization problem.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Future Generation Computer Systems, 85, 129–145. doi:10.1016/j.future.2018.03.020

Abdel-Basset, M., & Zhou, Y. (2018). An elite opposition-flower pollination algorithm for a 0-1 knapsack problem. International Journal of Bio-Inspired Computation, 11(1), 46–53. doi:10.1504/ijbic.2018.090080

Alatas, B. (2010). Chaotic harmony search algorithms. Applied Mathematics and Computation, 216(9), 2687–2699. doi:10.1016/j.amc.2010.03.114

Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization, 31(4), 633–672. doi:10.1007/s10898-004-9972-2

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Pittsburgh, PA: Carnegie Mellon University.

Beyer, H. (2001). The theory of evolution strategies. New York: Springer.

Bolaji, A. L. A., Al-Betar, M. A., Awadallah, M. A., Khader, A. T., & Abualigah, L. M. (2016). A comprehensive review: Krill herd algorithm (KH) and its applications. Applied Soft Computing, 49, 437–446. doi:10.1016/j.asoc.2016.08.041

Chen, S., Chen, R., Wang, G.-G., Gao, J., & Sangaiah, A. K. (2018). An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. Computers & Electrical Engineering, 67, 596–607

Cui, Z., Sun, B., Wang, G.-G., Xue, Y., & Chen, J. (2017). A novel oriented cuckoo search algorithm to improve DV-hop performance for cyber-physical systems. Journal of Parallel and Distributed Computing, 103, 42–52. doi:10.1016/j.jpdc.2016.10.011

Dorigo, M., & Stutzle, T. (2004). Ant colony optimization. Cambridge: MIT Press.

Duan, H., Zhao, W., Wang, G., & Feng, X. (2012). Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO. Mathematical Problems in Engineering, 2012, 1–22.

Feng, Y., & Wang, G.-G. (2018). Binary moth search algorithm for discounted 0-1 knapsack problem. IEEE Access, 6, 10708–10719. doi:10.1109/ACCESS.2018.2809445

Feng, Y., Wang, G.-G., & Wang, L. (2018a). Solving random-ized time-varying knapsack problems by a novel global firefly algorithm. Engineering with Computers, 34(3), 621–635. doi:10.1007/s00366-017-0562-6

Feng, Y., Yang, J., Wu, C., Lu, M., & Zhao, X.-J. (2018b). Solving 0-1 knapsack problems by chaotic monarch butterfly optimization algorithm. Memetic Computing, 10(2), 135–150. doi:10.1007/s12293-016-0211-4

Gandomi, A. H., & Alavi, A. H. (2011). Multi-stage genetic programming: A new strategy to nonlinear system modeling.
Talatahari, S., Farahmand Azar, B., Sheikholeslami, R., & Gandomi, A. H. (2012). Imperialist competitive algorithm combined with chaos for global optimization. Communications in Nonlinear Science and Numerical Simulation, 17(3), 1312–1319. doi:10.1016/j.cnsns.2011.08.021

Tan, Y. (2015). Fireworks algorithm—a novel swarm intelligence optimization method. Berlin: Springer-Verlag.

Tang, K., Li, X., Suganthan, P. N., Yang, Z., & Weise, T. (2010). Benchmark functions for the CEC2010 special session and competition on large scale global optimization. Hefei: Nature Inspired Computation and Applications Laboratory,USTC.

Wang, G.-G. (2018). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing, 10(2), 151–164. doi:10.1007/s12293-016-0212-3

Wang, G.-G., Cai, X., Cui, Z., Min, G., & Chen, J. (2017). High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. IEEE Transactions on Emerging Topics in Computing. doi:10.1109/TETC.2017.2703784

Wang, G.-G., Chu, H. E., & Mirjalili, S. (2016a). Three-dimensional path planning for UAV using an improved bat algorithm. Aerospace Science and Technology, 49, 231–238. doi:10.1016/j.ast.2015.11.040

Wang, G.-G., Deb, S., & Coelho, L. (2018a). Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. International Journal of Bio-Inspired Computation, 12(1), 1–22. doi:10.1504/IJBIC.2015.10004283

Wang, G.-G., Deb, S., Coelho, L. D. S. (2015a). Elephant herding optimization. 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015), Bali, Indonesia, December 7–9. IEEE, pp. 1–5. doi:10.1109/ISCBI.2015.8

Wang, G.-G., Deb, S., & Cui, Z. (2015b). Monarch butterfly optimization. Neural Computing & Applications. doi:10.1007/s00521-015-1923-y

Wang, G.-G., Deb, S., Gandomi, A. H., Zhang, Z., & Alavi, A. H. (2016d). Chaotic cuckoo search. Soft Computing, 20(9), 3349–3362. doi:10.1007/s00500-015-1726-1

Wang, G.-G., Deb, S., Gao, X.-Z., & Coelho, L. (2016c). A new metaheuristic optimization algorithm motivated by elephant herding behavior. International Journal of Bio-Inspired Computation, 8(6), 394–409. doi:10.1504/IJBIC.2016.10002274

Wang, G.-G., Deb, S., Zhao, X., & Cui, Z. (2018c). A new monarch butterfly optimization with an improved crossover operator. Operational Research, 18(3), 731–755.

Wang, G.-G., Gandomi, A. H., & Alavi, A. H. (2013a). A chaotic particle-swarm krill herd algorithm for global numerical optimization. Kybernetes, 42(6), 962–978. doi:10.1108/K-11-2012-0108

Wang, G.-G., Gandomi, A. H., & Alavi, A. H. (2014a). Stud krill herd algorithm. Neurocomputing, 128, 363–370. doi:10.1016/j.neucom.2013.08.031

Wang, G.-G., Gandomi, A. H., & Alavi, A. H. (2014b). An effective krill herd algorithm with migration operator in biogeography-based optimization. Applied Mathematical Modelling, 38(9–10), 2454–2462. doi:10.1016/j.apm.2013.10.052

Wang, G.-G., Gandomi, A. H., Alavi, A. H., & Deb, S. (2016a). A multi-stage krill herd algorithm for global numerical optimization. International Journal on Artificial Intelligence Tools, 25(2), 1550030. doi:10.1142/s021821301550030x

Wang, G.-G., Gandomi, A. H., Alavi, A. H., & Deb, S. (2016g). A hybrid method based on krill herd and quantum-behaved particle swarm optimization. Neural Computing & Applications, 27(4), 989–1006. doi:10.1007/s00521-015-1914-z

Wang, G.-G., Gandomi, A. H., Alavi, A. H., & Gong, D. (2019). A comprehensive review of krill herd algorithm: Variants, hybrids and applications. Artificial Intelligence Review, 51(1), 119–148. doi:10.1007/s10462-017-9559-1

Wang, G.-G., Gandomi, A. H., Zhao, X., & Chu, H. E. (2016e). Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Computing, 20(1), 273–285. doi:10.1007/s00500-014-1502-7

Wang, G., & Guo, L. (2013). A novel hybrid bat algorithm with harmony search for global numerical optimization. Journal of Applied Mathematics, 2013, 1–21.

Wang, G., Guo, L., Duan, H., Liu, L., & Wang, H. (2012c). A modified firefly algorithm for UAV path planning. International Journal of Hybrid Information Technology, 5(3), 123–144.

Wang, G., Guo, L., Duan, H., Liu, L., & Wang, H. (2012d). Path planning for UAV using bat algorithm with mutation. The Scientific World Journal, 2012, 1–15.

Wang, G., Guo, L., Duan, H., Liu, L., & Wang, H. (2012e). Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm. Journal of Sensor and Actuator Networks, 1(2), 86–96. doi:10.3390/jsan1020086

Wang, G., Guo, L., Duan, H., Liu, L., & Wang, H. (2012f). The model and algorithm for the target threat assessment based on Elman AdaBoost strong predictor. Acta Electronica Sinica, 40(5), 901–906.

Wang, G., Guo, L., Duan, H., Liu, L., Wang, H., & Shao, M. (2012a). Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm. Advanced Science, Engineering and Medicine, 4(6), 550–564. doi:10.1166/asem.2012.1223

Wang, G., Guo, L., Duan, H., Wang, H., Liu, L., & Shao, M. (2012f). A hybrid meta-heuristic DE/CS algorithm for UAV three-dimension path planning. The Scientific World Journal, 2012, 1–11.

Wang, G., Guo, L., Duan, H., Wang, H., Liu, L., & Shao, M. (2013b). Hybridizing harmony search with biogeography based optimization for global numerical optimization. Journal of Computational and Theoretical Nanoscience, 10(10), 2318–2328.

Wang, G., Guo, L., Wang, H., Duan, H., Liu, L., & Li, J. (2014c). Incorporating mutation scheme into krill herd algorithm for global numerical optimization. Neural Computing & Applications, 24(3–4), 853–871. doi:10.1007/s00521-012-1304-8

Wang, G.-G., Lu, M., Dong, Y.-Q., & Zhao, X.-J. (2016b). Self-adaptive extreme learning machine. Neural Computing & Applications, 27(2), 291–303. doi:10.1007/s00521-015-1874-3

Wang, G.-G., & Tan, Y. (2019). Improving metaheuristic algorithms with information feedback models. IEEE Transactions on Cybernetics, 49(2), 542–555. doi:10.1109/TCYB.2017.2780274

Wang, H., & Yi, J.-H. (2018). An improved optimization method based on krill herd and artificial bee colony with information exchange. Memetic Computing, 10(2), 177–198. doi:10.1007/s12293-017-0241-6

Yang, X.-S. (2010a). Nature-inspired metaheuristic algorithms (2nd ed.). From: Luniver Press.

Yang, X. S. (2010b). Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation, 2(2), 78–84. doi:10.1504/IJBIC.2010.032124
Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4), 330–343.

Yang, X. S., & Gandomi, A. H. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations*, 29(5), 464–483. doi:10.1108/02644401211235834

Yang, D., Li, G., & Cheng, G. (2007). On the efficiency of chaos optimization algorithms for global optimization. *Chaos, Solitons & Fractals*, 34(4), 1366–1375. doi:10.1016/j.chaos.2006.04.057

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2), 82–102. doi:10.1109/4235.771163

Yi, J.-H., Deb, S., Dong, J., Alavi, A. H., & Wang, G.-G. (2018b). An improved NSGA-III algorithm with adaptive mutation operator for big data optimization problems. *Future Generation Computer Systems*, 88, 571–585. doi:10.1016/j.future.2018.06.008

Yi, J.-H., Wang, J., & Wang, G.-G. (2016). Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem. *Advances in Mechanical Engineering*, 8(1), 1–13.

Yi, J.-H., Xing, L.-N., Wang, G.-G., Dong, J., Vasilakos, A. V., Alavi, A. H., & Wang, L. (2018a). Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Information Sciences*. doi:10.1016/j.ins.2018.10.005

Zhang, Y., Gong, D., Hu, Y., & Zhang, W. (2015). Feature selection algorithm based on bare bones particle swarm optimization. *Neurocomputing*, 148, 150–157. doi:10.1016/j.neucom.2012.09.049

Zhang, Y., Song, X.-F., & Gong, D.-W. (2017). A return-cost-based binary firefly algorithm for feature selection. *Information Sciences*, 418-419, 561–574. doi:10.1016/j.ins.2017.08.047

Zhang, J.-W., & Wang, G.-G. (2012). Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 203(1), 88–93. doi:10.4028/www.scientific.net/AMM.203.88

Zou, D.-X., Deb, S., & Wang, G.-G. (2018). Solving IIR system identification by a variant of particle swarm optimization. *Neural Computing & Applications*, 30(3), 685–698. doi:10.1007/s00521-016-2338-0

Zou, D.-X., Wang, G.-G., Pan, G., & Qi, H. (2016). A modified simulated annealing algorithm and an excessive area model for the floorplanning with fixed-outline constraints. *Frontiers of Information Technology & Electronic Engineering*, 17(11), 1228–1244. doi:10.1631/FITEE.1500386

Zou, D.-X., Wang, G.-G., Sangaiah, A. K., & Kong, X. (2017). A memory-based simulated annealing algorithm and a new auxiliary function for the fixed-outline floorplanning with soft blocks. *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-017-0661-7