Neural Comput & Applic (2014) 24:853–871
DOI 10.1007/s00521-012-1304-8

ORIGINAL ARTICLE

Incorporating mutation scheme into krill herd algorithm for global numerical optimization

Gaige Wang · Lihong Guo · Heqi Wang · Hong Duan · Luo Liu · Jiang Li

Received: 2 August 2012 / Accepted: 7 December 2012 / Published online: 25 December 2012
© Springer-Verlag London 2012

Abstract Recently, Gandomi and Alavi proposed a robust meta-heuristic optimization algorithm, called Krill Herd (KH), for global optimization. To improve the performance of the KH algorithm, harmony search (HS) is applied to mutate between krill during the process of krill updating instead of physical diffusion used in KH. A novel hybrid meta-heuristic optimization approach HS/KH is proposed to solve global numerical optimization problem. HS/KH combines the exploration of harmony search (HS) with the exploitation of KH effectively, and hence, it can generate the promising candidate solutions. The detailed implementation procedure for this improved meta-heuristic method is also described. Fourteen standard benchmark functions are applied to verify the effects of these improvements, and it is demonstrated that, in most cases, the performance of this hybrid meta-heuristic method (HS/KH) is superior to, or at least highly competitive with, the standard KH and other population-based optimization methods, such as ACO, BBO, DE, ES, GA, HS, KH, PSO, and SGA. The effect of the HS/FA parameters is also analyzed.

G. Wang · L. Guo (✉) · H. Wang · L. Liu · J. Li
Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China
e-mail: guolh@ciomp.ac.cn

G. Wang
e-mail: gaigewang@163.com; gaigewang@gmail.com

G. Wang · L. Liu · J. Li
Graduate School of Chinese Academy of Sciences, Beijing 100039, China

H. Duan
School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China
e-mail: duanh0618@163.com

Keywords Global optimization problem · Krill herd (KH) · Harmony search (HS) · Multimodal function

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, optimization algorithms are applied to solve optimization problems. A simple classification way for optimization 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 commence 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. However, generally speaking, the final values, though slightly different, will converge to the same optimal solutions within a given accuracy. Generally, stochastic algorithms have two types: heuristic and meta-heuristic. Recently, nature-inspired meta-heuristic algorithms perform powerfully and efficiently in solving modern nonlinear numerical global optimization problems [1]. To some extent, all meta-heuristic algorithms strive for making balance between randomization (global search) and local search [2].

Inspired by nature, these strong meta-heuristic algorithms are applied to solve NP-hard problems such as task-resource assignment. Optimization algorithms cover all searching for extreme value problems. These kinds of meta-heuristic algorithms carry out on a population of solutions and always find best solutions. During the 1950s

and 1960s, computer scientists studied the possibility of conceptualizing evolution as an optimization tool and this generated a subset of gradient-free approaches named genetic algorithms (GAs) [3, 4]. Since then many other nature-inspired meta-heuristic algorithms have emerged, such as differential evolution (DE) [5–7], particle swarm optimization (PSO) [8–10], genetic programming (GP) [11, 12], biogeography-based optimization (BBO) [13, 14], bat algorithm (BA) [15, 16], cuckoo search (CS) algorithm [17–19], firefly algorithm (FA) [20–23], and more recently, the krill herd (KH) algorithm [24] that is based on simulating the herding behavior of krill individuals in nature.

Firstly presented by A. H. Gandomi and A. H. Alavi in 2012, krill herd (KH) is a new meta-heuristic optimization algorithm [24], inspired by the herding behavior of krill individuals. In KH algorithm, the objective function for the krill movement is determined by the minimum distances of each individual krill from food and from highest density of the herd. The time-dependent position of the krill individuals is comprised of three main components: (1) movement induced by other individuals (2) foraging motion, and (3) random physical diffusion. One of the remarkable advantages of KH algorithm is that the derivative information is not necessary because it uses a stochastic random search rather than a gradient search. The other important advantage of KH algorithm is its simplicity, therefore, it is very easy to implement. In principle, comparing with other population-based meta-heuristic algorithms such as particle swarm optimization (PSO) [8] and harmony search (HS) [25, 26], there is essentially only a single-parameter C_t (time interval) in KH (apart from the population size).

Firstly proposed by Geem et al. in 2001 [25], harmony search (HS) is a new meta-heuristic approach for minimizing possibly non-differentiable and nonlinear functions in continuous space. HS is inspired by behavior of musician's improvisation process, where every musician attempts to improve its tune so as to create optimal harmony in a real-world musical performance processes. HS algorithm originates in the similarity between engineering optimization and music improvisation, and the engineers search for a global optimal solution as determined by an objective function, just like the musicians strive for finding esthetic harmony as determined by esthetician. In music improvisation, each musician chooses any pitch within the feasible range, together producing one harmony vector. If all the pitches produce a good solution, that experience is reserved in each variable's memory, and the possibility of producing a better solution is also increased next time. Furthermore, this new approach requires few control variables, which makes HS easy to implement, more robust, and is very appropriate for parallel computation.

Harmony search is a powerful algorithm in exploration, but at times, it may trap into some local optima so that it cannot perform global search well. For krill herd, the search depends completely on random walks, so a fast convergence cannot be guaranteed. Firstly presented here, in order to increase the diversity of the population for KH, a main improvement of adding HS serving as mutation operator is made to the KH with the aim of speeding up convergence, thus making the approach more feasible for a wider range of practical applications while preserving the attractive characteristics of the basic KH. That is to say, we combine two approaches to propose a new hybrid meta-heuristic algorithm according to the principle of HS and KH, and then an improved KH method is used to search the optimal objective function value. The proposed approach is evaluated on fourteen standard benchmark functions that have ever been applied to verify optimization algorithms in continuous optimization problems. Experimental results demonstrate that the HS/KH performs more efficiently and accurately than basic KH, ACO, BBO, DE, ES, GA, HS, PSO, and SGA.

The structure of this paper is organized as follows: Sect. 2 describes global numerical optimization problem, the HS algorithm, and basic KH in brief. Our proposed approach HS/KH is presented in detail in Sect. 3. Subsequently, our method is evaluated through fourteen benchmark functions in Sect. 4. In addition, the HS/KH is also compared with ACO, BBO, DE, ES, GA, HS, KH, PSO, and SGA. Finally, Sect. 5 consists of the conclusion and proposals for future work.

2 Preliminary

To begin with, in this section, we will provide a brief background on the optimization problem, harmony search (HS), and krill herd (KH) algorithm.

2.1 Optimization problem

In computer science, mathematics, and management science, optimization (also called mathematical programming or mathematical optimization) means the selection of an optimal solution from some set of feasible alternatives. In general, an optimization problem includes minimizing or maximizing a function by systematically selecting input values from a given feasible set and calculating the value of the function. More generally, optimization consists of finding the optimal values of some objective function within a given domain, including a number of different types of domains and different types of objective functions.

A global optimization problem can be described as follows:

Given: a function f: $S \rightarrow R$ from some set S to the real numbers

Sought: a parameter x_0 in S such that $f(x_0) \leq f(x)$ for all x in S ("minimization") or such that $f(x_0) \geq f(x)$ for all x in S ("maximization").

Such a formulation is named a numerical optimization problem. Many theoretical and practical problems may be modeled in this general framework. In general, S is some subset of the Euclidean space R^n, often specified by a group of constraints, equalities, or inequalities that the components of S have to satisfy. The domain S of f is named the search space, while the elements of S are named feasible solutions or candidate solutions. In general, the function f is called an objective function, utility function (maximization), or cost function (minimization). An optimal solution is an available solution that is the extreme of (minimum or maximum) the objective function.

Conventionally, the standard formulation of an optimization problem is stated in accordance with minimization. In general, unless both the feasible region and the objective function are convex in a minimization problem, there may be more than one local minima. A global minimum x^* is defined as a point for which the following expression

$$f(x*) \leq f(x) \tag{1}$$

holds.

The branch of numerical analysis and applied mathematics that investigates deterministic algorithms that can guarantee convergence in limited time to the true optimal solution of a non-convex problem is called global numerical optimization problems. A variety of algorithms have been proposed to solve non-convex problems. Among them, heuristics algorithms can evaluate approximate solutions to some optimization problems, as described in introduction.

2.2 Harmony search

Firstly developed by Z. W. Geem et al. in 2001 [25], harmony search (HS) is a relatively new meta-heuristic optimization algorithm [27], and it is based on natural musical performance processes that occur when a musician searches for an optimal state of harmony [28]. The optimization operators of HS algorithm are specified as the harmony memory (HM), which stores the solution vectors which are all within the search space, as shown in Eq. (2), the harmony memory size (HMS), which specifies the number of solution vectors stored in the HM, the harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR) and the pitch adjustment bandwidth (bw).

$$\mathrm{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \ldots & x_D^1 & fitness(x^1) \\ x_1^2 & x_2^2 & \ldots & x_D^2 & fitness(x^2) \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \ldots & x_D^{\mathrm{HMS}} & fitness(x^{\mathrm{HMS}}) \end{array}\right] \tag{2}$$

In more detail, we can explain the HS algorithm with the help of discussing the improvisation process by a music player. When a player is improvising, he or she has 3 feasible options: (1) play any famous piece of music (a series of pitches in harmony) exactly from his or her memory as the HMCR; (2) play something similar to a known piece in player's memory (thus adjusting the pitch slightly); or (3) play totally new or random pitch from feasible ranges. If these three options are formalized for optimization, we have three corresponding components: use of harmony memory, pitch adjusting, and randomization. Similarly, when each decision variable chooses one value in the HS algorithm, it can apply one of the above three rules in the whole HS procedure. If a new harmony vector is better than the worst harmony vector in the HM, the new harmony vector replaces the worst harmony vector in the HM. This procedure is repeated until a stopping criterion is satisfied.

The use of harmony memory is significant as it is analogous to select the optimal fit individuals in the GA. This will make sure the best harmonies will be kept on to the new harmony memory. For the purpose of using this memory more effectively, we should properly set the value of the parameter HMCR[0, 1]. If this rate is very approaching to 1 (too high), almost all the harmonies are utilized in the harmony memory, then other harmonies are not explored well, resulting in potentially wrong solutions. If this rate is very close to 0 (extremely low), only few best harmonies are chosen and it may have a slow convergence rate. Therefore, generally, HMCR $= 0.7 \sim 0.95$.

To adjust the pitch slightly in the second component, an appropriate approach is to be applied to adjust the frequency efficiently. In theory, we can adjust the pitch linearly or nonlinearly, but in fact, linear adjustment is utilized. If x_{old} is the current solution (or pitch), then the new solution (pitch) x_{new} is generated by

$$x_{\mathrm{new}} = x_{\mathrm{old}} + \mathrm{bw}(2\varepsilon - 1) \tag{3}$$

where ε is a uniformly distributed random numbers in [0,1]. Here, bw is the bandwidth, controlling the local range of pitch adjustment. Actually, we can see that the pitch adjustment (3) is a random walk.

Pitch adjustment is similar to the mutation operator in GA. Also, we must appropriately set the parameter PAR to control the degree of the adjustment. If PAR nears 1 (too high), then the solution is always changing and the algorithm may not converge at all. If it is very close to 0 (too low), then there is very little change and the algorithm may be premature. Therefore, we use PAR $= 0.1 \sim 0.5$ in most simulations as usual.

For the purpose of increasing the diversity of the solutions, the randomization is needed in the third component. Although adjusting pitch has a similar role, it is confined to certain local pitch adjustment and thus corresponds to a local search. The usage of randomization can make the system move further to explore multifarious regions with

Algorithm 1 *Harmony search algorithm*

Begin

Step 1: *Set the parameters and initialize the HM.*

Step 2: *Evaluate the fitness for each individual in HM*

Step 3: ***while*** *the halting criteria is not satisfied* ***do***

for *d=1:D* ***do***

if *rand* < *HMCR* ***then*** *// memory consideration*

$x_{new}(d) = x_a(d)$ *where* $a \in (1,2,..,HMS)$

if rand < *PAR then* *// pitch adjustment*

$x_{new}(d) = x_{old}(d) + bw \times (2 \times rand - 1)$

endif

else *// random selection*

$x_{new}(d) = x_{\min,d} + rand \times (x_{\max,d} - x_{\min-d})$

endif

endfor *d*

Update the HM as $x_w = x_{new}$, *if* $f(x_{new}) < f(x_w)$ *(minimization objective)*

Update the best harmony vector found so far

Step 4: ***end while***

Step 5: *Post-processing the results and visualization.*

End.

high-solution diversity in order to search for the global optimal solution. In real-world engineering applications, HS has been applied to solve many optimization problems including linear antenna arrays, function optimization, flow shop scheduling, reliability problem, economic load dispatch, and others.

The mainframe of the basic HS algorithm can be described as shown in Algorithm 1. Where D is the number of decision variables. *rand* is a random real number in interval (0, 1) drawn from uniform distribution. From Algorithm 1, it is clear that there are only two control parameters in HS, which are HMCR and PAR.

2.3 Krill herd algorithm

Krill herd (KH) [24] is a novel meta-heuristic swarm intelligence optimization method for solving optimization problems, which is based on the simulation of the herding of the krill swarms in response to specific biological and environmental processes. The time-dependent position of an individual krill in two-dimensional surface is determined by three main actions described as follows:

I. Movement induced by other krill individuals;
II. Foraging action; and
III. Random diffusion

In KH, the Lagrangian model is used in a d-dimensional decision space as shown in Eq. (4).

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{4}$$

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 ith krill individuals.

2.3.1 Motion induced by other krill individuals

The direction of motion induced, α_i, is approximately evaluated 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 defined as follows:

$$N_i^{new} = N^{max}\alpha_i + \omega_n N_i^{old} \tag{5}$$

where

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target} \tag{6}$$

and N^{max} is the maximum induced speed, ω_n is the inertia weight of the motion induced in [0, 1], N_i^{old} is the last motion induced, α_i^{local} is the local effect provided by the neighbors, and α_i^{target} is the target direction effect provided by the best krill individual. According to the experimental values of the maximum induced speed, we set N^{max} to 0.01 (ms^{-1}) in our study.

2.3.2 Foraging motion

The foraging motion is influenced by the two main factors. One is the food location and the other one is the previous

experience about the food location. For the ith krill individual, this motion can be expressed as follows:

$$F_i = V_f \beta_i + \omega_f F_i^{\text{old}} \tag{7}$$

where

$$\beta_i = \beta_i^{\text{food}} + \beta_i^{\text{best}} \tag{8}$$

and V_f is the foraging speed, ω_f is the inertia weight of the foraging motion between 0 and 1, F_i^{old} is the last foraging motion, β_i^{food} is the food attractive, and β_i^{best} is the effect of the best fitness of the ith krill so far. In our study, we set V_f to 0.02 [24].

In KH, the virtual center of food concentration is approximately calculated according to the fitness distribution of the krill individuals, which is inspired from "*center of mass.*" The center of food for each iteration is formulated as follows:

$$X^{\text{food}} = \frac{\sum_{i=1}^{N} \frac{1}{K_i} X_i}{\sum_{i=1}^{N} \frac{1}{K_i}} \tag{9}$$

Therefore, the food attraction for the ith krill individual can be determined as follows:

$$\beta_i^{\text{food}} = C^{\text{food}} \widehat{K}_{i,\text{food}} \widehat{X}_{i,\text{food}} \tag{10}$$

where C^{food} is the food coefficient.

2.3.3 Physical diffusion

The physical diffusion of the krill individuals is considered to be a random process. This motion can be expressed in terms of a maximum diffusion speed and a random directional vector. It can be formulated as follows:

$$D_i = D^{\max} \delta \tag{11}$$

where $D^{\max}$ is the maximum diffusion speed, and δ is the random directional vector, and its arrays are random values in $[-1, 1]$. The better the position of the krill is, the less random the motion is. The effects of the motion induced by other krill individuals and foraging motion gradually decrease with increasing the time (iterations). Thus, another term Eq. (12) is added to Eq. (11). This term linearly decreases the random speed with the time and performs on the basis of a geometrical annealing schedule:

$$D_i = D^{\max} \left(1 - \frac{1}{\text{I}_{\max}}\right) \delta \tag{12}$$

2.3.4 Main procedure of the KH algorithm

In general, the defined motions frequently change the position of a krill individual toward the best fitness. The foraging motion and the motion induced by other krill individuals contain two global and two local strategies. These are working in parallel which make KH a powerful algorithm. 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 given by the following equation:

$$X_i(t + \Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt} \tag{13}$$

It should be noted that Δt is one of the most important constants and should be carefully set according to the optimization problem. This is because this parameter works as a scale factor of the speed vector.

In addition, to improve the performance of the KH, genetic reproduction mechanisms are incorporated into the algorithm. The introduced adaptive genetic reproduction mechanisms are crossover and mutation which are inspired from the classical DE algorithm.

Various krill-inspired algorithms can be developed by idealizing the motion characteristics of the krill individuals. Generally, the KH algorithm can be described by the following steps:

(i) Data structures: define the simple bounds, determination of algorithm parameter(s) and, etc.
(ii) Initialization: randomly create the initial population in the search space.
(iii) Fitness evaluation: evaluate each krill individual according to its position.
(iv) Motion calculation:

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

(v) Perform the genetic operators.
(vi) Updating: update the krill individual position in the search space.
(vii) Repeating: go to step iii until the stop criterion is reached.
(viii) End

The basic representation of the KH can be summarized as shown in Algorithm 2. More details about the three main motions and KH algorithm can be found in [24].

3 Our approach: HS/KH

Based on the introduction of HS and KH in previous section, we will explain how we combine the two approaches to form the proposed krill herd with harmony search (HS/KH) in this section, which modifies the solutions with poor fitness in order to add diversity of the population to improve the search efficiency.

Algorithm 2 *Krill herd algorithm*

Begin

Step 1: Initialization. *Set the generation counter G = 1; initialize the population P of NP krill individuals randomly and each krill corresponds to a potential solution to the given problem; set the foraging speed V_f, the maximum diffusion speed D^{max}, and the maximum induced speed N^{max}.*

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

Step 3: While *the termination criteria is not satisfied* ***or*** *G < MaxGeneration* ***do***

Sort the population/ krill from best to worst.

for *i=1:NP (all frill)* ***do***

Perform the following motion calculation.

Motion induced by the presence of other individuals

Foraging motion

Physical diffusion

Implement the genetic operators.

Update the krill individual position in the search space.

Evaluate each krill individual according to its position.

end for *i*

Sort the population/krill from best to worst and find the current best.

G = G+1.

Step 4: end while

Step 5: *Post-processing the results and visualization.*

End.

Algorithm 3 *The hybrid meta-heuristic algorithm of HS/KH*

Begin

Step 1: Initialization. *Set the generation counter t = 1; initialize the population P of NP krill individuals randomly and each krill corresponds to a potential solution to the given problem; set the foraging speed V_f, the maximum diffusion speed D^{max}, and the maximum induced speed N^{max}.*

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

Step 3: While *the termination criteria is not satisfied or t < MaxGeneration* ***do***

Sort the population/ krill from best to worst.

for *i=1:NP (all frill)* ***do***

Perform the following motion calculation.

Motion induced by the presence of other individuals

Foraging motion

Hybrid harmony search mutation operation

Update the krill individual position in the search space.

Evaluate each krill individual according to its position.

end for *i*

Sort the population/krill from best to worst and find the current best.

G = G+1.

t = t+1;

Step 4: end while

Step 5: *Post-processing the results and visualization;*

End.

For krill herd, as the search relies entirely on random walks, a fast convergence cannot be guaranteed. Described here for the first time, a main improvement of adding hybrid HS mutation operator is made to the KH, which is made with the aim of speeding up convergence, thus making the method more practical for a wider range of applications but without losing the attractive features of the original method.

The main improvement is to add HS serving as mutation operator in an attempt to increase diversity of the population to improve the search efficiency and speed up the convergence to optima. For the local search part, once a solution is selected among the current best solutions, a new solution for each krill is generated globally. And then, we mutate every element in x_i using HS. When ξ is larger than HMCR, *that is*, $\xi_1 \geq$ HMCR, the element j is updated randomly; while when $\xi_1 <$ HMCR, we update the element j in accordance with x_{r1}, moreover, hybrid HS mutation operator is applied to update the element j if $\xi_2 <$ PAR to increase diversity of the population to improve the search efficiency, as shown in Eq. (3), where ξ_1 and ξ_2 are two uniformly distributed random numbers in [0,1], r_1 is the integer number in [1, *NP*], and *NP* is population size. Through testing benchmarks in Sect. 4.2, it was found that setting the parameter of harmony memory consideration rate HMCR to 0.9 and pitch adjustment rate PAR to 0.1 produced the best results.

Based on above-mentioned analyses, the mainframe of the harmony search/krill herd (HS/KH) is presented in Algorithm 3.

4 Simulation experiments

In this section, we test the performance of the proposed metaheuristic HS/KH to global numerical optimization through a series of experiments conducted in benchmark functions.

To allow a fair comparison of running times, all the experiments were conducted 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 or HS tools were used in the following experiments.

4.1 General performance of HS/KH

In order to explore the benefits of HS/KH, in this section, we compared its performance on global numeric optimization problem with eight other population-based optimization methods, which are ACO, BBO, DE, ES, GA, HS, PSO, and SGA. Ant colony optimization (ACO) [29] is a swarm intelligence algorithm for solving optimization problems which is based on the pheromone deposition of ants. Biogeography-based optimization (BBO) [13, 30] is a new excellent powerful and efficient evolution algorithm developed for the global optimization inspired by the immigration and emigration of species between islands (or habitats) in search of more compatible islands. Differential evolution (DE) [5, 31] is a simple but excellent optimization method that uses the difference between two solutions to probabilistically adapt a third solution. An evolutionary strategy (ES) [32] is an algorithm that generally distributes equal importance to mutation and recombination, and that allows two or more parents to reproduce an offspring. A genetic algorithm (GA) [3] is a search heuristic that mimics the process of natural evolution. Particle swarm optimization (PSO) [8, 33] is also a swarm intelligence algorithm which is based on the swarm behavior of fish and bird schooling in nature. A stud genetic algorithm (SGA) [34] is a GA that uses the best individual at each generation for crossover. In addition, we must point out that, in [24], A. H. Gandomi and A. H. Alavi demonstrated that, comparing all the algorithms, the KH II (KH with crossover operator) has the best performance which confirms the robustness of the krill herd algorithm. Therefore, in our study, we use KH II as basic krill herd algorithm.

In all experiments, we will use the same parameters for HS, KH, and HS/KH that are the foraging speed $V_f = 0.02$, the maximum diffusion speed $D^{\max} = 0.005$, the maximum induced speed $N^{\max} = 0.01$, the harmony memory consideration rate HMCR = 0.9, and the pitch adjustment rate PAR = 0.1. In addition, the inertia weights (ω_n, ω_f) are equal to 0.9 at the beginning of the search to emphasize exploration. These two parameters are linearly decreased to 0.1 at the end to encourage exploitation. For ACO, BBO, DE, ES, GA, PSO, and SGA, we set the parameters as follows. For ACO, initial pheromone value $\tau_0 = 1E{-}6$, pheromone update constant $Q = 20$, exploration constant $q_0 = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay rate $\rho_l = 0.5$, pheromone sensitivity $s = 1$, and visibility sensitivity $\beta = 5$; for BBO, habitat modification probability = 1, immigration probability bounds per gene = [0, 1], step size for numerical integration of probabilities = 1, maximum immigration and migration rates for each island = 1, and mutation probability = 0.005; for DE, a weighting factor $F = 0.5$ and a crossover constant $CR = 0.5$; for the ES, the number of offspring $\lambda = 10$ produced in each generation, and standard deviation $\sigma = 1$ for changing solutions. For the GA, we used roulette wheel selection, single-point crossover with a crossover probability of 1, and a mutation probability of 0.01. For PSO, we set an inertial constant = 0.3, a cognitive constant = 1, and a social constant for swarm interaction = 1. For the SGA, we used single-point crossover with a crossover probability of 1, and a mutation probability of 0.01.

Well-defined problem sets are favorable for evaluating the performance of optimization methods proposed in this paper. Based on mathematical functions, benchmark functions can be applied as objective functions to perform such tests. The properties of these benchmark functions can be easily achieved from their definitions. Fourteen different benchmark functions are applied to verify our proposed meta-heuristic algorithm HS/KH.

The benchmark functions described in Table 1 are standard testing functions. The properties of the benchmark functions are given in Table 2. The modality property means the number of the best solutions in the search space. Unimodal benchmark functions only have an optimum, which is the global optimum. Multimodal benchmark functions have at least two optima in their search space, indicating that they have more than one local optimum

Table 1 Benchmark functions

No.	Name	Definition
F01	Ackley	$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}$
F02	Fletcher-Powell	$f(\vec{x}) = \sum_{i=1}^{n} (A_i - B_i)^2, A_i = \sum_{j=1}^{n} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$ $B_i = \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j)$
F03	Griewank	$f(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F04	Penalty #1	$f(\vec{x}) = \frac{\pi}{30}\left\{10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 \cdot \left[1 + 10 \sin^2(\pi y_{i+1})\right]. + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ y_i = 1 + 0.25(x_i + 1)$
F05	Penalty #2	$f(\vec{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})]. + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F06	Quartic *with noise*	$f(\vec{x}) = \sum_{i=1}^{n} (i \cdot x_i^4 + U(0, 1))$
F07	Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$
F08	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
F09	Schwefel 2.26	$f(\vec{x}) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin\left(\lvert x_i \rvert^{1/2}\right)$
F10	Schwefel 1.2	$f(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F11	Schwefel 2.22	$f(\vec{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F12	Schwefel 2.21	$f(\vec{x}) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$
F13	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F14	Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor$

In benchmark function F02, the matrix elements $\mathbf{a}_{n\times n}, \mathbf{b}_{n\times n} \in (-100, 100), \boldsymbol{\alpha}_{n\times 1} \in (-\pi, \pi)$ are drawn from uniform distribution [36]

In benchmark functions F04 and F05, the definition of the function $u(x_i,a,k,m)$ is as follows: $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$

Table 2 Properties of benchmark functions, *lb* denotes lower bound, *ub* denotes upper bound, *opt* denotes optimum point

No.	Function	lb	ub	opt	Continuity	Modality
F01	Ackley	−32.768	32.768	0	Continuous	Multimodal
F02	Fletcher-Powell	$-\pi$	π	0	Continuous	Multimodal
F03	Griewank	−600	600	0	Continuous	Multimodal
F04	Penalty #1	−50	50	0	Continuous	Multimodal
F05	Penalty #2	−50	50	0	Continuous	Multimodal
F06	Quartic *with noise*	−1.28	1.28	1	Continuous	Multimodal
F07	Rastrigin	−5.12	5.12	0	Continuous	Multimodal
F08	Rosenbrock	−2.048	2.048	0	Continuous	Unimodal
F09	Schwefel 2.26	−512	512	0	Continuous	Multimodal
F10	Schwefel 1.2	−100	100	0	Continuous	Unimodal
F11	Schwefel 2.22	−10	10	0	Continuous	Unimodal
F12	Schwefel 2.21	−100	100	0	Continuous	Unimodal
F13	Sphere	−5.12	5.12	0	Continuous	Unimodal
F14	Step	−5.12	5.12	0	Discontinuous	Unimodal

Table 3 Mean normalized optimization results in fourteen benchmark functions

	ACO	BBO	DE	ES	GA	HS	HSKH	KH II	PSO	SGA
F01	6.13	3.40	4.92	7.59	6.82	7.78	**1.00**	2.14	6.57	3.43
F02	9.53	**1.00**	3.73	10.09	4.26	8.90	1.23	26.63	7.85	1.19
F03	11.67	8.56	18.97	92.94	38.53	185.24	**1.00**	30.59	73.42	7.67
F04	1.3E7	2.6E3	6.6E4	7.9E6	1.6E5	1.3E7	**1.00**	1.3E6	1.4E6	60.51
F05	1.2E7	1.5E4	1.5E5	6.5E6	3.2E5	1.1E7	**1.00**	1.3E6	1.5E6	1.5E3
F06	1.5E3	142.47	674.06	2.1E4	1.4E3	2.1E4	**1.00**	4.6E3	4.6E3	54.16
F07	6.81	1.46	6.03	9.34	6.41	8.83	**1.00**	7.77	7.08	2.00
F08	65.73	4.07	9.52	82.72	16.36	57.63	**1.00**	22.50	20.18	3.93
F09	3.29	1.58	5.99	7.45	2.47	9.11	**1.00**	12.67	9.28	1.75
F10	3.56	2.06	4.84	5.56	3.78	5.08	**1.00**	11.95	3.97	3.20
F11	33.66	4.79	14.25	53.83	25.73	40.89	**1.00**	1.3E11	29.61	6.91
F12	6.06	6.52	7.73	9.22	8.00	9.58	**1.00**	2.21	7.80	5.56
F13	738.32	54.37	135.27	1.5E3	485.40	1.3E3	**1.00**	246.00	550.01	57.56
F14	65.12	31.51	74.94	500.52	140.94	731.38	**1.00**	121.61	296.96	28.18
Time	2.06	1.00	1.24	1.28	1.36	1.77	2.88	2.63	1.54	1.32

The values shown are the minimum objective function values found by each algorithm, averaged over 100 Monte Carlo simulations

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

Bold values indicate the best value achieved for each test problem

except the global optimum. More details of all the benchmark functions can be found in [35].

We set population size $NP = 50$, an elitism parameter $Keep = 2$, and maximum generation $Maxgen = 50$ for each algorithm. We ran 100 Monte Carlo simulations of each algorithm on each benchmark function to get representative performances. Tables 3 and 4 illustrate the results of the simulations. Table 3 shows the average minima found by each algorithm, averaged over 100 Monte Carlo runs. Table 4 shows the absolute best minima found by each algorithm over 100 Monte Carlo runs. In other words, Table 3 shows the average performance of each algorithm, while Table 4 shows the best performance of each algorithm. The best value achieved for each test problem is shown in bold. Note that the normalizations in the tables are based on different scales, so values cannot be compared between the two tables. Each of the functions in this study has 20 independent variables (*i.e.*, $d = 20$).

From Table 3, we see that, on average, HS/KH is the most effective at finding objective function minimum on thirteen of the fourteen benchmarks (F01, F03–F14). BBO is the second most effective, performing best on the other one benchmark (F02) when multiple runs are made. Table 4 shows that HS/KH performs the best on twelve of

Table 4 Best normalized optimization results in fourteen benchmark functions

	ACO	BBO	DE	ES	GA	HS	HSKH	KH II	PSO	SGA
F01	13.68	6.29	11.18	18.24	13.21	19.02	**1.00**	3.24	15.70	6.10
F02	14.07	1.20	6.92	22.74	3.67	13.61	**1.00**	47.24	14.07	1.36
F03	5.84	3.50	9.95	60.03	4.14	126.53	**1.00**	18.80	34.92	3.08
F04	**1.00**	14.24	2.2E3	1.5E7	9.37	2.5E7	3.32	3.9E6	2.0E4	2.67
F05	**1.00**	9.4E17	1.1E21	1.8E23	1.0E18	3.1E23	1.5E16	3.8E22	2.3E22	3.9E16
F06	1.3E4	671.45	9.2E3	1.6E5	9.8E3	4.3E5	**1.00**	8.5E4	5.0E4	244.00
F07	12.10	2.09	11.23	16.24	8.53	16.76	**1.00**	12.09	13.21	2.67
F08	61.63	2.91	9.75	85.32	12.24	49.89	**1.00**	24.13	14.21	2.89
F09	7.16	2.10	14.85	17.10	2.04	23.91	**1.00**	33.70	21.64	1.47
F10	2.95	1.20	6.65	6.91	2.50	6.57	**1.00**	5.77	4.88	3.44
F11	87.38	8.44	37.82	112.69	46.82	122.87	**1.00**	1.2E5	60.91	12.79
F12	7.07	6.76	12.22	16.02	8.52	15.15	**1.00**	2.50	11.44	5.75
F13	3.9E3	273.60	994.91	8.5E3	2.7E3	1.1E4	**1.00**	1.6E3	2.2E3	221.34
F14	302.67	106.33	372.00	2.4E3	377.67	4.3E3	**1.00**	702.00	1.9E3	93.00

The values shown are the minimum objective function values found by each algorithm

The values are normalized so that the minimum in each row is 1.00. These are the absolute best minima found by each algorithm

Bold values indicate the best value achieved for each test problem

the fourteen benchmarks (F01–F03, F06–F14). ACO is the second most effective, performing the best on the benchmarks F04 and F05 when multiple runs are made. In addition, statistical analysis on these values obtained by the ten methods on fourteen benchmark functions based on the Friedman's test [37] reveals that the differences in the obtained average and best function minima are statistically significant ($p = 1.72 \times 10^{-17}$ and $p = 8.19 \times 10^{-17}$, respectively) at the confidence level of 5 %.

Furthermore, the computational requirements of the ten optimization methods were similar. We collected the average computational time of the optimization methods as applied to the 14 benchmarks discussed in this section. The results are shown in Table 3. BBO was the quickest optimization method. HS/KH was the *tenth* fastest of the ten algorithms. However, it should be noted that in the vast majority of real-world applications, it is the fitness function evaluation that is by far the most expensive part of a population-based optimization algorithm.

Further, convergence graphs of ACO, BBO, DE, ES, GA, HS, HS/KH, KH, PSO, and SGA are shown in Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14, which represent the process of optimization. The values shown in Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14 are the mean objective function optimum achieved from 100 Monte Carlo simulations, which are the true objective function values, not normalized. In addition, it is notable that the global optima of the benchmarks (F02, F04, F05, and F11) are illustrated in the form of the semi-logarithmic convergence plots. Also, we use KH short for KH II in the legend of the Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14.

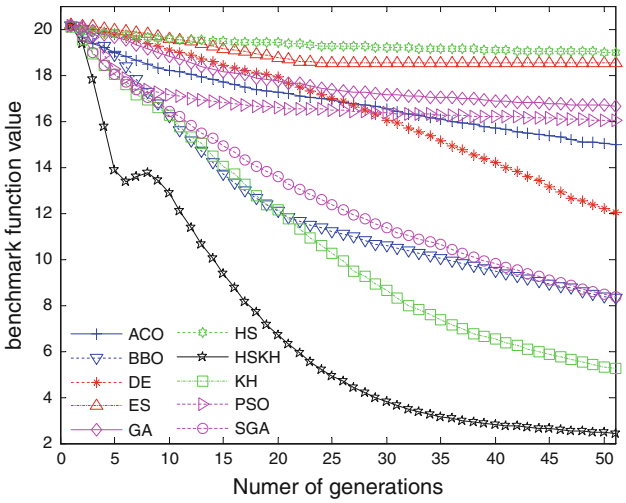

Fig. 1 Comparison of the performance of the different methods for the F01 Ackley function

Figure 1 shows the results obtained for the ten methods when the F01 Ackley function is applied. This is a multimodal function with a narrow global minimum basin ($F01_{min} = 0$) and many minor local optima. From Fig. 1, clearly, we can draw the conclusion that HS/KH 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 HS/KH has a stably faster convergence rate than other algorithms. All methods clearly outperform the standard HS algorithm.

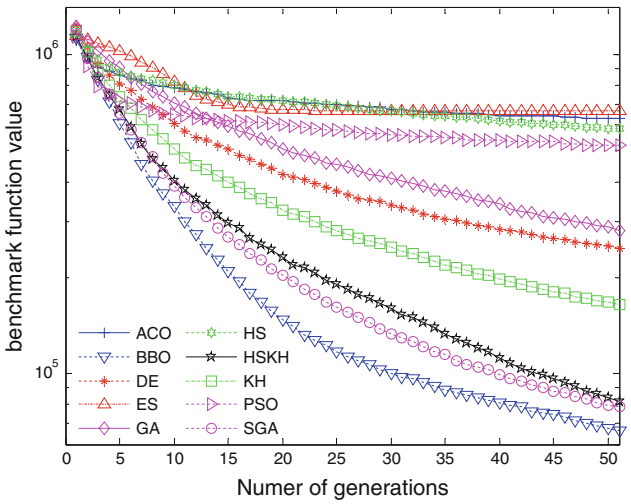

Fig. 2 Comparison of the performance of the different methods for the F02 Fletcher-Powell function

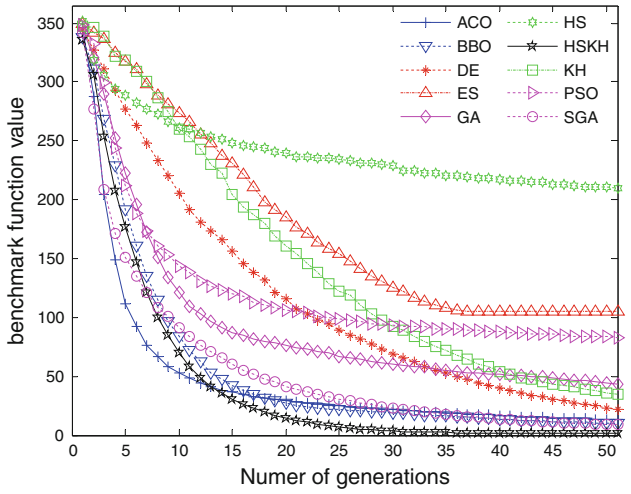

Fig. 3 Comparison of the performance of the different methods for the F03 Griewank function

Figure 2 illustrates the optimization results for F02 Fletcher-Powell function. From Fig. 2, clearly, we can draw the conclusion that BBO is significantly superior to the other algorithms including HS/KH during the process of optimization, while SGA performs the second best in this multimodal benchmark function. Unfortunately, HS/KH only performs the third for this case. Moreover, KH II is inferior to HS/KH and ranks 4 out of the ten methods.

Figure 3 shows the optimization results for F03 Griewank function, which has many local minima but a global minimum of $F03_{\min} = 0$. F03 has a $\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right)$ component causing linkages among dimensions thus making it difficult to reach the global optimum. An interesting phenomenon of F03 is that it is more difficult for lower dimensions than higher dimensions [38]. From Table 3 and

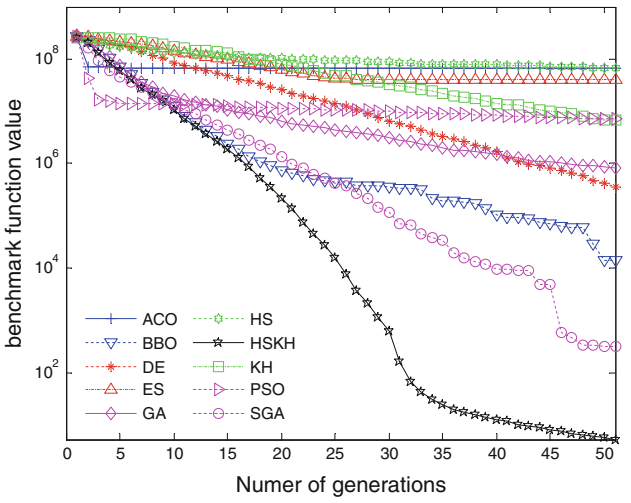

Fig. 4 Comparison of the performance of the different methods for the F04 Penalty #1 function

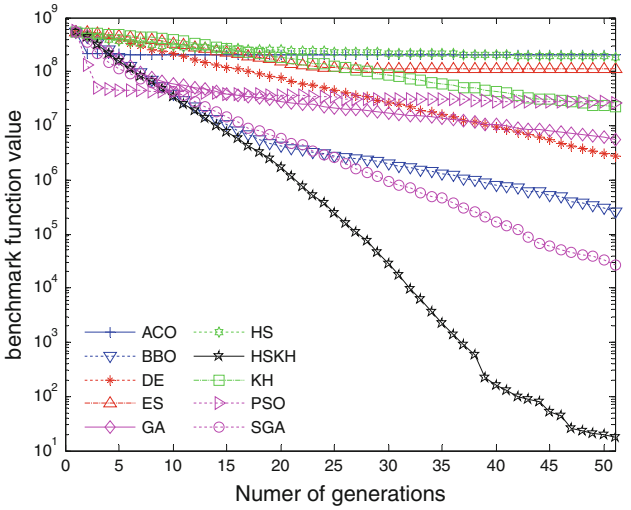

Fig. 5 Comparison of the performance of the different methods for the F05 Penalty #2 function

Fig. 3, we can see that HS/KH performs the best in this multimodal function. By carefully looking at the Fig. 3, SGA and ACO show a faster convergence rate initially than HS/KH; however, they are outperformed by HS/KH after 13 generations. For other algorithms, although slower, ACO, BBO, and SGA eventually find the global minimum close to HS/KH, while DE, ES, GA, HS, KH II, and PSO fail to search the global minimum within the limited iterations.

Figure 4 shows the results for F04 Penalty #1 function. From Fig. 4, apparently, HS/KH outperforms all other methods in this example. Here, PSO shows a much faster convergence rate initially; however, it seems to be attracted to sub-minima as the distance from the global minimum increases slightly. Although slower, SGA performs the second best at finding the global minimum.

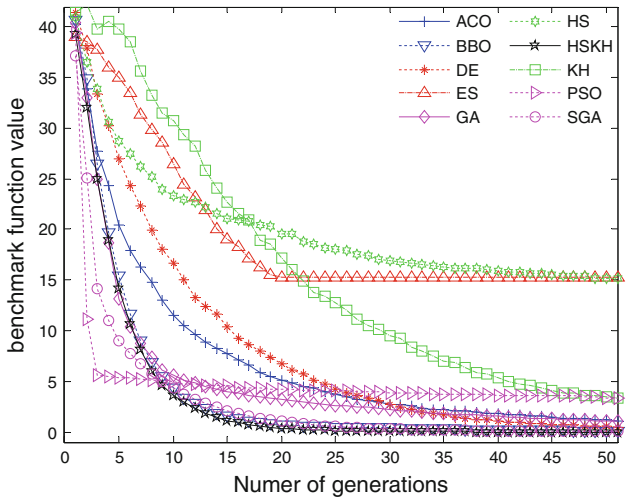

Fig. 6 Comparison of the performance of the different methods for the F06 Quartic (*with noise*) function

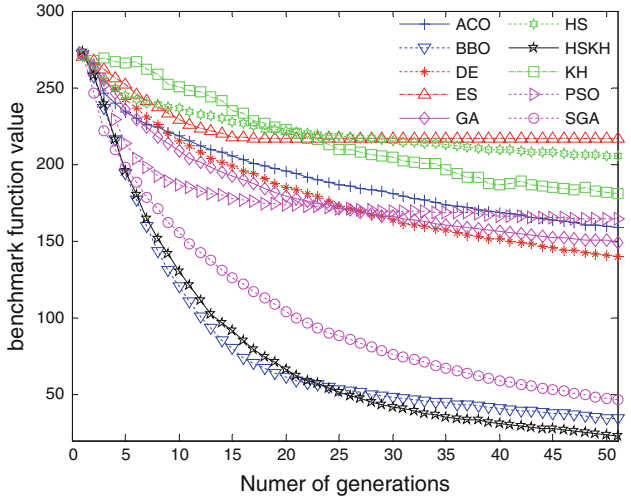

Fig. 7 Comparison of the performance of the different methods for the F07 Rastrigin function

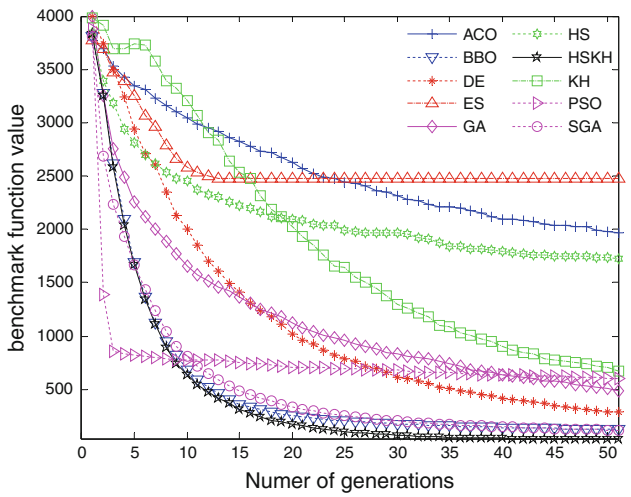

Fig. 8 Comparison of the performance of the different methods for the F08 Rosenbrock function

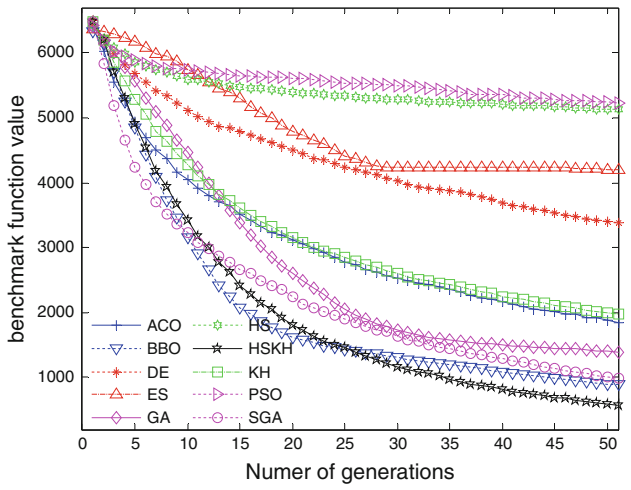

Fig. 9 Comparison of the performance of the different methods for the F09 Schwefel 2.26 function

Figure 5 shows the performance achieved for F05 Penalty #2 function. For this multimodal function, very similar to F04 Penalty #1 function as shown in Fig. 4, HS/KH significantly outperforms all other methods during the process of optimization. By carefully looking at Fig. 4, we can see that in the beginning of the optimization process, PSO converges faster than HS/KH while HS/KH is able to improve its solution steadily for a long run.

Figure 6 shows the results achieved for the ten methods when using the F06 Quartic (*with noise*) function. For this case, the figure shows that there is little difference between the performance of BBO, HS/KH, and SGA. From Table 3 and Fig. 6, we can conclude that HS/KH performs the best in this multimodal function. Through carefully looking at Fig. 6, PSO and SGA have a fast convergence initially toward the known minimum, as the procedure proceeds HS/KH gets closer and closer to the minimum, while PSO comes into being premature and traps into the local minimum; eventually they are outperformed by HS/KH after 8 generations. ACO, DE, ES, GA, HS, KH II, and PSO do not manage to succeed in this benchmark function within maximum number of generations, showing a wide range of obtained results. At last, BBO and SGA converge to the value that is very close to HS/KH's.

Figure 7 shows the optimization results for the F07 Rastrigin function, which is a complex multimodal problem with a unique global minimum of $F07_{min} = 0$ and a large number of local optima. When attempting to solve F07, methods may easily trap into a local optimum. Hence, a method capable of maintaining a larger diversity is likely

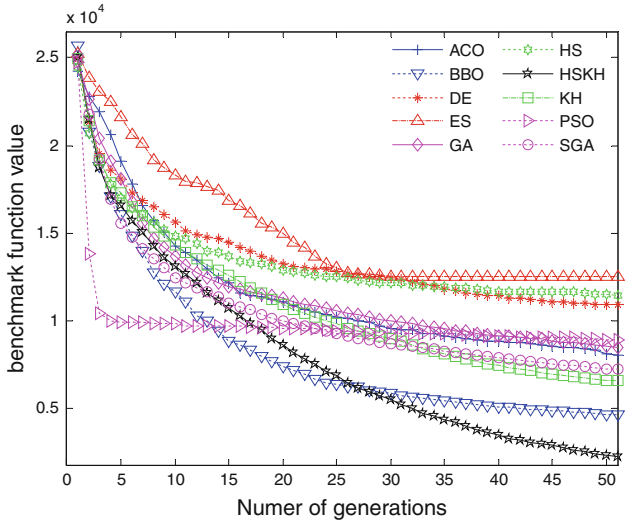

Fig. 10 Comparison of the performance of the different methods for the F10 Schwefel 1.2 function

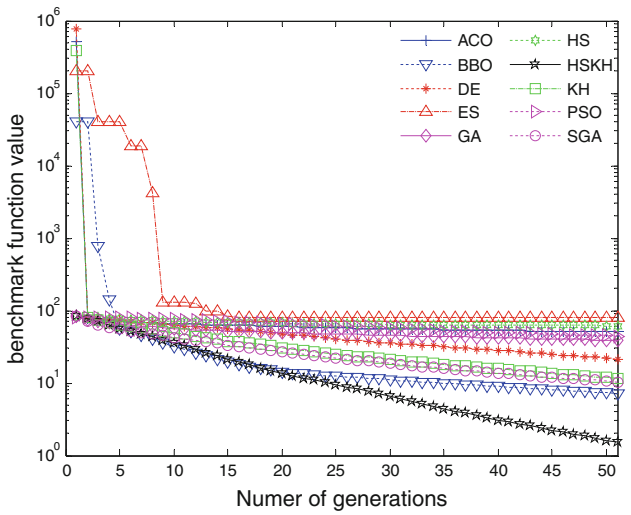

Fig. 11 Comparison of the performance of the different methods for the F11 Schwefel 2.22 function

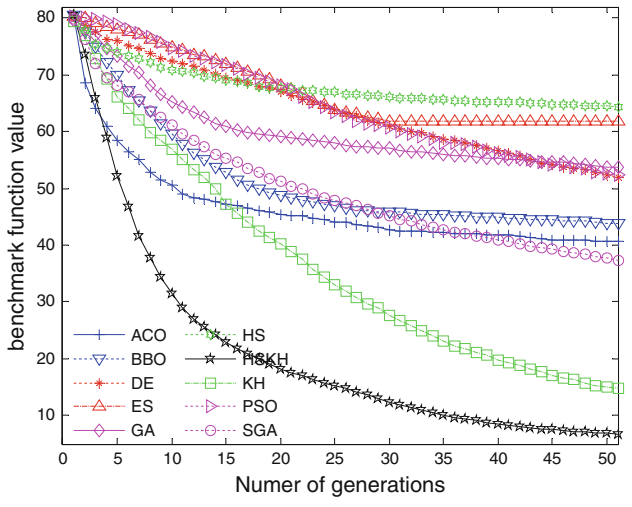

Fig. 12 Comparison of the performance of the different methods for the F12 Schwefel 2.21 function

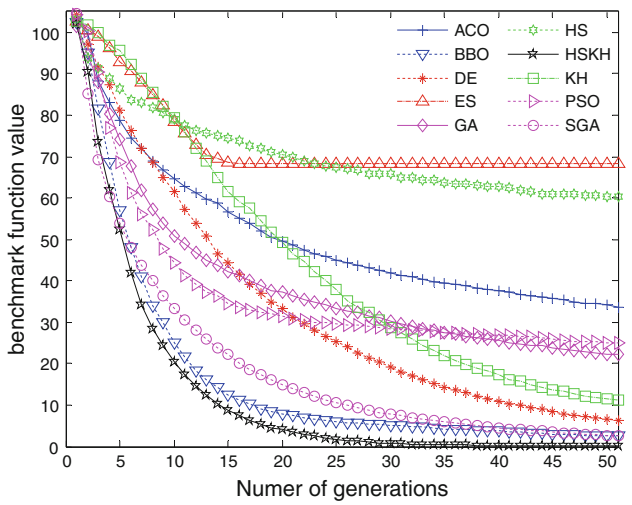

Fig. 13 Comparison of the performance of the different methods for the F13 Sphere function

to produce better results. At first glance, it is clear that though HS/KH is outperformed by BBO within 25 generations, HS/KH has the fastest convergence rate at finding the global minimum and significantly outperforms all other approaches. Further, BBO and SGA works very well because the BBO and SGA have ranks of 2, 3, respectively, among ten algorithms.

Figure 8 shows the results for F08 Rosenbrock function. It can be also treated as a multimodal problem. It has a narrow valley from the perceived local optima to the global optimum. For this case, analogous to the F06 Quartic (*with noise*) function as shown in Fig. 6, we can conclude: (1) the figure shows that there is little difference between the performance of BBO, HS/KH, and SGA; (2) PSO shows a much faster convergence rate initially than HS/KH; however, it is outperformed by HS/KH after 7 generations and is premature to the local minimum. From Table 3 and Fig. 8, we can see that HS/KH is superior to the other algorithms during the optimization process in this benchmark function.

Figure 9 shows the equivalent results for the F09 Schwefel 2.26 function. This is a multimodal function with a global minimum of $F09_{\min} = 0$. The complexity of F09 is due to its deep local optima being far from the global optimum. From Fig. 9, very apparently, though HS/KH is outperformed by BBO and SGA within 23 and 9 generations, respectively, it has the stable convergence rate at finding the global minimum and significantly outperforms all other approaches in this multimodal benchmark function.

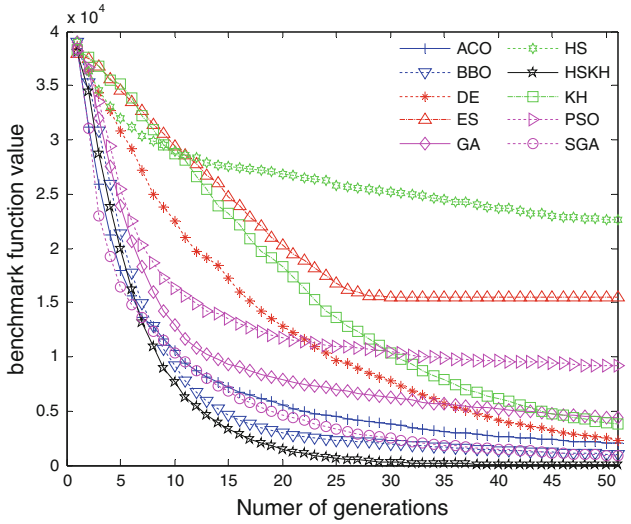

Fig. 14 Comparison of the performance of the different methods for the F14 Step function

Figure 10 shows the results for F10 Schwefel 1.2 function. From Fig. 10, for this relative simple unimodal benchmark function, we can see that HS/KH performs far better than other algorithms in the optimization process. PSO shows a faster convergence rate initially than HS/KH; however, it is outperformed by HS/KH after 17 generations. For other algorithms, BBO works very well, because it ranks 2 among ten methods.

Figure 11 shows the results for F11 Schwefel 2.22 function. From Fig. 11, very clearly, though HS/KH performs equally with BBO within 20 generations, it has the stable convergence rate at finding the global minimum and significantly outperforms all other approaches in this unimodal benchmark function. At last, HS/KH reaches the optimal solution significantly superiorly to other algorithms. BBO is only inferior to HS/KH and performs the second best in this unimodal function.

Figure 12 shows the results for F12 Schwefel 2.21 function. At first glance, it is clear that HS/KH has the fastest convergence rate finding the global minimum in the whole optimization progress. HS/KH reaches the optimal solution significantly superiorly to other algorithms. KH II is only inferior to HS/KH and performs the second best in this unimodal function.

Figure 13 shows the results for F13 Sphere function, which is also known as *de Jong*'s function, and has a unique global minimum of $F13_{min} = 0$, and therefore, it is easy to solve. From Table 3 and Fig. 13, HS/KH has the fastest convergence rate at finding the global minimum and outperforms all other approaches. Looking carefully at Fig. 13, for BBO and SGA, we can see that BBO has a faster convergence rate than SGA, but SGA does finally converge to the value of BBO that is very close to HS/KH's. 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. This highlights

Table 5 Best normalized optimization results in fourteen benchmark functions with different HMCR

HMCR											
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	2.51	2.51	2.51	2.18	2.16	2.00	1.19	1.37	1.95	**1.00**	2.37
F02	2.01E5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F03	74.32	101.26	8.27	9.01	9.21	9.02	9.08	9.12	5.36	**1.00**	9.18
F04	3.47E6	3.45	5.26E4	6.03	2.79	2.69	2.35	2.39	2.89	**1.00**	2.78
F05	3.15E6	9.32	1.01	1.05E4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F06	2.30E6	4.14E3	5.15E3	3.40E3	2.55E3	1.95E3	521.36	119.36	16.89	**1.00**	2.79E3
F07	12.03	12.22	11.36	1.16	1.14	8.86	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F08	79.25	1.22	15.32	1.26	1.26	1.25	30.15	**1.00**	**1.00**	**1.00**	**1.00**
F09	360.25	348.52	1.25	12.63	1.24	1.56	1.54	189.25	**1.00**	**1.00**	**1.00**
F10	956.85	20.87	1.63E3	27.63	2.13	2.79	2.16	2.48	826.96	**1.00**	**1.00**
F11	10.14	1.59	6.98	4.56	8.13	4.26	3.16	3.78	1.58	**1.00**	2.85
F12	5.12	**1.00**	2.36	1.25	5.85	1.52	1.63	1.85	1.47	1.47	5.13
F13	158.32	265.32	158.25	198.23	25.36	85.26	39.15	28.96	15.22	**1.00**	63.25
F14	978.23	9.68	7.86	**1.00**	**1.00**	10.59	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	0	2	1	2	3	2	4	5	6	**13**	7

The numbers shown are the best results found after 100 Monte Carlo simulations of HS/KH algorithm

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

Bold values indicate the best value achieved for each test problem

Table 6 Mean normalized optimization results in fourteen benchmark functions with different HMCR

	HMCR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	**1.00**	**1.00**
F02	1.45E3	13.56	12.35	14.26	7.36	8.52	7.26	9.25	**1.00**	1.56	8.75
F03	10.22	3.65E4	1.05	1.02	1.02	1.05	1.08	1.08	**1.00**	**1.00**	1.06
F04	4.65E6	3.89E4	4.25E4	1.12E4	2.65E3	2.21E3	250.36	156.23	1.05	**1.00**	4.36E3
F05	1.05E7	8.36E4	7.96E4	5.25E4	6.36E4	2.05E4	1.15E4	3.23E3	4.25	**1.00**	3.63E4
F06	**1.00**	3.25E4	356.25	345.25	3.26E4	1.18	1.26	1.27	1.26	1.22	1.18
F07	9.66	4.24E6	10.35	269.85	306.59	3.27E4	1.06	1.08	**1.00**	**1.00**	1.26
F08	89.26	3.53E4	2.89E4	308.25	258.36	287.25	2.98E4	1.05	1.06	**1.00**	1.48
F09	228.36	7.8E6	2.69	10.23	285.36	256.45	248.96	2.63E4	1.26	**1.00**	2.69
F10	285.49	4.36E4	1.25E6	1.56E4	76.24	189.23	156.32	148.74	1.31E4	**1.00**	2.33
F11	3.78	**1.00**	4.63E4	1.28	15.69	178.49	314.94	318.15	315.56	3.6E4	1.17
F12	2.89	2.69E4	3.85E4	**1.00**	3.0E4	158.31	158.91	185.46	275.34	248.56	3.0E4
F13	2.96	8.95	8.9E6	3.5E6	**1.00**	9.63	185.49	198.25	285.14	285.21	289.12
F14	156.1	4.67E5	1.05E4	4.52E3	**1.00**	3.58E3	15.23	19.54	15.32	34.85	36.25
	1	1	0	1	2	0	0	0	3	**8**	1

The numbers shown are the best results found after 100 Monte Carlo simulations of HS/KH algorithm

The values are normalized so that the minimum in each row is 1.00. These are the absolute best minima found by each algorithm

Bold values indicate the best value achieved for each test problem

Table 7 Best normalized optimization results in fourteen benchmark functions with different PAR

	PAR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F02	4.41E3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F03	**1.00**	2.36	1.56	1.58	1.59	1.25	1.99	1.96	1.58	1.58	1.06
F04	**1.00**	2.36	258.66	3.69	3.85	3.45	3.85	3.69	3.58	3.54	3.21
F05	4.25	**1.00**	1.98	8.67E3	1.95	1.95	1.95	1.95	1.95	1.95	1.96
F06	**1.00**	10.25	22.36	36.89	5.63	24.69	19.25	36.89	4.56	7.99	15.36
F07	12.23	**1.00**	1.89	3.58	3.15	32.86	3.58	3.45	3.26	3.89	3.56
F08	6.52	**1.00**	1.58	**1.00**	**1.00**	**1.00**	13.85	**1.00**	**1.00**	**1.00**	**1.00**
F09	125.96	4.85	1.65	**1.00**	1.98	1.96	1.95	356.22	1.95	1.63	1.95
F10	1.25E3	1.58	**1.00**	6.48	3.68	3.98	3.52	3.56	4.5E3	3.58	3.62
F11	2.9E3	**1.00**	4.4E3	4.6E3	2.5E3	4.7E3	4.2E3	4.4E3	4.8E3	2.2E3	4.9E3
F12	1.3E3	**1.00**	3.3E3	352.25	600.36	370.25	389.52	345.23	348.25	354.12	2.6E3
F13	1.03	7.66	3.89	3.52	5.62	5.25	4.69	3.25	2.10	1.35	**1.00**
F14	135.25	**1.00**	1.58	3.69	3.58	5.65	3.56	3.58	3.51	3.98	3.52
	4	**8**	3	4	3	3	2	3	3	3	4

The numbers shown are the best results found after 100 Monte Carlo simulations of HS/KH algorithm

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

Bold values indicate the best value achieved for each test problem

the lack of the diversity of the population in the standard algorithm, meaning KH II cannot take advantage of the symmetry of this function as HS/KH can.

Figure 14 shows the results for F14 Step function. Apparently, HS/KH shows the fastest convergence rate at finding the global minimum and significantly outperforms

Table 8 Mean normalized optimization results in fourteen benchmark functions with different PAR

	PAR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F02	42.36	2.58	4.56	5.23	4.56	9.25	5.69	2.56	1.05	1.98	**1.00**
F03	**1.00**	1.2E4	1.25	1.25	1.23	1.25	1.27	1.55	1.26	1.85	1.89
F04	2.26	56.36	8.9 E3	1.05	**1.00**	1.06	1.07	1.08	**1.00**	**1.00**	**1.00**
F05	256.36	1.45	158.63	7.9E3	1.56	1.05	9.56	63.56	1.25	1.85	**1.00**
F06	**1.00**	5.26E6	5..2E4	3.4E4	5.1E6	526.33	569.25	585.63	578.69	578.26	589.15
F07	5.65	3.48	**1.00**	136.32	58.26	1.3E4	1.25	1.26	1.89	1.56	1.64
F08	15.36	58.65	8.5E3	106.32	85.63	58.65	8.7E3	1.08	1.08	**1.00**	1.25
F09	62.25	158.61	1.85	1.65	65.23	68.56	38.12	5.6E3	1.25	1.36	**1.00**
F10	125.25	1.04	2.52	1.9E3	16.32	25.36	24.12	16.69	1.7E3	**1.00**	1.05
F11	96.36	**1.00**	7.8E3	125.96	85.22	1.3E4	1.1E4	1.6E4	6.9E3	1.4E6	125.36
F12	6.32	8.5E3	8.6E3	**1.00**	8.4E3	158.36	85.36	105.39	96.56	58.32	8.4E3
F13	**1.00**	25.96	1.3E3	15.56	6.25	4.25	1.4E3	585.65	665.23	605.32	359.61
F14	35.96	1.52	1.56	35.69	**1.00**	5.3E3	78.25	115.25	56.25	65.85	58.51
	4	2	2	2	3	1	1	1	2	4	**5**

The numbers shown are the best results found after 100 Monte Carlo simulations of HS/KH algorithm

The values are normalized so that the minimum in each row is 1.00. These are the absolute best minima found by each algorithm

Bold values indicate the best value achieved for each test problem

all other approaches. Looking carefully at Fig. 14, for BBO and SGA, they perform approximately equally and are only inferior to HS/KH.

From above analyses about the Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14, we draw the conclusion that the HS/KH's performance is superior to or at least quite competitive with the other nine acclaimed state-of-the-art population-based algorithms. In general, BBO and SGA are only inferior to the HS/KH. Further, benchmarks F04, F05, F06, F08, and F10 illustrate that PSO has a much faster convergence rate initially, while later it converges slower and slower to the true objective function value. At last, we must point out, in [13], Simon compared BBO with seven state-of-the-art EAs over fourteen benchmark functions and a real-world sensor selection problem. The results demonstrated the good performance of BBO. It is also indirectly demonstrated that our proposed hybrid metaheuristic method HS/KH is a more powerful and efficient optimization algorithm than other population-based optimization algorithms.

4.2 Influence of control parameter

The choice of the control parameters is of vital importance for different problems. To compare the different effects on the parameter of the harmony memory consideration rate HMCR and pitch adjustment rate PAR, we ran 100 Monte Carlo simulations of HS/KH algorithm on the above problem to get the best performances. All other parameter settings are kept unchanged (unless noted otherwise in the following paragraph). The results are recorded in Tables 5, 6, 7 and 8 after 100 Monte Carlo runs. Among them, Tables 5 and 7 show the best minima found by HS/KH algorithm over 100 Monte Carlo runs. Tables 6 and 8 show the average minima found HS/KH algorithm, averaged over 100 Monte Carlo runs. In other words, Tables 5, 7 and Tables 6, 8 show the best and average performance of HS/KH algorithm, respectively. In each table, the last row is the total number of functions on which HS/KH performs the best with specific parameters.

4.2.1 Harmony memory consideration rate (HMCR)

Tables 5 and 6 recorded the results performed on the benchmark problem with the harmony memory consideration rate HMCR = 0, 0.1, 0.2, … , 0.9, 1.0, and fixed pitch adjustment rate PAR = 0.1. From Tables 5 and 6, obviously, it can be seen that (1) for the three benchmark functions F02, F05, and F14, HS/KH performs slightly differently, that is to say, these three benchmark functions are insensitive to the parameter HMCR. (2) For benchmark function F12, HS/KH performs better on smaller HMCR (<0.5). (3) However, for other functions, HS/KH performs better on bigger HMCR (>0.5). In sum, HS/KH performs the best when HMCR is equal or very close to 0.9. So, we set HMCR = 0.9 in other experiments. In addition,

statistical analysis on these values obtained by the HS/KH with harmony memory consideration rate HMCR on fourteen benchmark functions based on the Friedman's test reveals that the differences in the obtained average and best function minima across various chaotic maps are statistically significant ($p = 8.1 \times 10^{-16}$ and $p = 5.6 \times 10^{-16}$, respectively) at the confidence level of 5 %.

4.2.2 Pitch adjustment rate (PAR)

Tables 7 and 8 recorded the results performed in the benchmark problem with the pitch adjustment rate PAR = 0, 0.1, 0.2, …, 0.9, 1.0 and fixed harmony memory consideration rate HMCR = 0.9. From Table 7, we can recognize that the function values for HS/KH vary little with the increase in PAR, and HS/KH reaches optimum/minimum in most benchmarks when PAR is equal or very close to 0.1. Whereas, looking at numbers in Table 8, the numbers are very complex and in disorder. Considering comprehensively, our aim is to get the best performance, so we set PAR = 0.1 in other experiments. In addition, statistical analysis on these values obtained by the HS/KH with 11 different pitch adjustment rate PAR on fourteen benchmark functions based on the Friedman's test reveals that the differences in the obtained average and best function minima across various pitch adjustment rate PAR are statistically significant ($p = 7.9 \times 10^{-16}$ and $p = 4.9 \times 10^{-16}$, respectively) at the confidence level of 5 %.

With respect to the HS in the HS/KH and the original HS, the effects of the two main parameters of the HS namely HMCR and HMS were investigated by many researchers [39]. Based on those observations, a large value for HMCR (*i.e.* approaching to 1.00) is generally used except for problems with a very low dimensionality for which a small value of HMCR is recommended. This conclusion coincides with the experimental results conducted in Sect. 4.2.

5 Discussion

For all of the standard benchmark functions that have been considered, the HS/KH has been demonstrated to perform better than or be equal to the standard KH and other acclaimed state-of-the-art population-based algorithms with the HS/KH performing significantly better in some functions. The HS/KH performs excellently and efficiently because of its ability to simultaneously carry out a local search, still searching globally at the same time. It succeeds in doing this due to the local search via krill herd algorithm and global search via harmony search algorithm concurrently. A similar behavior may be performed in the PSO by using multiswarm from a particle population initially [40]. However, HS/KH's advantages include performing simply and easily and only have two parameters to regulate. The work carried out here demonstrates the HS/KH to be robust over all kinds of benchmark functions.

Benchmark evaluation is a good way for verifying the performance of the meta-heuristic algorithms, but it also has limitations. First, we did not make any special effort to tune the optimization algorithms in this section. Different tuning parameter values in the optimization algorithms might result in significant differences in their performance. Second, real-world optimization problems may not have much of a relationship to benchmark functions. Third, benchmark tests might result in different conclusions whether the grading criteria or problem setup change. In our work, we examined the mean and best results obtained with a certain population size and after a certain number of generations. However, we might arrive at different conclusions whether (for example) we change the generation limit, or look at how many generations it takes to reach a certain function value, or whether we change the population size. In spite of these caveats, the benchmark results shown here are promising for HS/KH and indicate that this novel method might be able to find a niche among the plethora of population-based optimization algorithms.

The proposed method has strong optimization ability, but it is not perfect and its running time is a little too long. We note that CPU time is a bottleneck to the implementation of many population-based optimization algorithms. If an algorithm cannot converge fast, it will be impractical, since it would take too long to find an optimal or sub-optimal solution. HS/KH does not seem to require an unreasonable amount of computational effort; of the ten optimization algorithms compared in this paper, HS/KH was the *tenth* fastest. How to accelerate the HS/KH's convergence speed still deserves further scrutiny. Therefore, besides deeply studying the algorithm, we will continue improving the method and optimize code to reduce running time.

In this work, fourteen benchmark functions are used to evaluate the performance of our approach; we will test our approach on more problems, such as the high-dimensional ($d \geq 20$) CEC 2010 test suit [41] and the real-world problems. Moreover, we will compare HS/KH with other EAs. In addition, we only consider the unconstrained function optimization in this work. Our future work consists on adding the diversity rules into HS/KH for constrained optimization problems, such as constrained real-parameter optimization CEC 2010 test suit [42].

6 Conclusion and future work

This paper proposed a hybrid meta-heuristic HS/KH method for optimization problem. We improved the KH by

combining harmony search (HS) algorithm and evaluate the HS/KH on multimodal numerical optimization problems. A novel type of KH model has been presented, and an improvement is applied to mutate between krill using harmony search algorithm during the process of krill updating. Using the original configuration of the krill herd algorithm, we generate the new harmonies based on the newly generated krill each iteration after krill's position has been updated. The new harmony vector substitutes the newly generated krill only if it has better fitness. This selection scheme is rather greedy which often overtakes original HS and KH. The HS/KH attempts to take merits of the KH and the HS in order to avoid all krill getting trapped in inferior local optimal regions. The HS/KH enables the krill to have more diverse exemplars to learn from as the krill are updated each iteration and also form new harmonies to search in a larger search space. This new method can speed up the global convergence rate without losing the strong robustness of the basic KH. From the analysis of the experimental results, we observe that the proposed HS/KH makes good use of the information in past solutions more effectively to generate better quality solutions frequently when compared to the other population-based optimization algorithms such as ACO, BBO, DE, ES, GA, HS, KH, PSO, and SGA. Based on the results of the ten approaches on the test problems, we can conclude that the HS/KH significantly improves the performances of the HS and the KH on most multimodal and unimodal problems. Further, the novel configuration of HS/KH does not introduce additional complex operations beyond the original KH and HS. In addition, the HS/KH is simple and easy to implement.

In the field of optimization, there are many issues that are worthy of further study, and efficient optimization method should be developed depending on the analysis of specific engineering problems. Our future work will focus on the two issues. On the one hand, we would apply our proposed approach HS/KH to solve practical engineering optimization problems, such as structural optimization, geometric optimization, resource-constrained project scheduling, economic load dispatch problem, and PID control. And, obviously, HS/KH can be a promising method for real-world engineering optimization problems. On the other hand, in this study, HS is applied to improve the performance of the KH algorithm, and there are many new excellent meta-heuristic optimization methods and technology can be used, for example, teaching–learning-based optimization, eagle strategy, bat algorithm, cuckoo search, and firefly algorithm. Therefore, we would adopt these new strategies and develop new meta-hybrid approach to solve optimization problem.

Acknowledgments This work was supported by State Key Laboratory of Laser Interaction with Material Research Fund under Grant No. SKLLIM0902-01 and Key Research Technology of Electric-discharge Non-chain Pulsed DF Laser under Grant No. LXJJ-11-Q80.

References

1. Gandomi AH, Yang XS, Talatahari S, Alavi AH (2013) Meta-heuristic applications in structures and infrastructures. Elsevier, London, UK
2. Yang XS, Gandomi AH, Talatahari S, Alavi AH (2013) Meta-heuristics in water. Geotechnical and Transport Engineering, Elsevier
3. Goldberg DE (1998) Genetic algorithms in search. Optimization and Machine learning, Addison-Wesley
4. Zhao M, Ren J, Ji L, Fu C, Li J, Zhou M (2012) Parameter selection of support vector machines and genetic algorithm based on change area search. Neural Comput Appl 21(1):1–8. doi:10.1007/s00521-011-0603-9
5. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
6. Gandomi AH, Yang X-S, Talatahari S, Deb S (2012) Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Comput Math Appl 63(1):191–200. doi:10.1016/j.camwa.2011.11.010
7. Khazraee S, Jahanmiri A, Ghorayshi S (2011) Model reduction and optimization of reactive batch distillation based on the adaptive neuro-fuzzy inference system and differential evolution. Neural Comput Appl 20(2):239–248. doi:10.1007/s00521-010-0364-x
8. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceeding of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948
9. Chen D, Zhao C, Zhang H (2011) An improved cooperative particle swarm optimization and its application. Neural Comput Appl 20(2):171–182. doi:10.1007/s00521-010-0503-4
10. Talatahari S, Kheirollahi M, Farahmandpour C, Gandomi AH (2012) A multi-stage particle swarm for optimum design of truss structures. Neural Comput Appl. doi:10.1007/s00521-012-1072-5
11. Gandomi AH, Alavi AH (2012) A new multi-gene genetic programming approach to nonlinear system modeling. Part II: Geotechnical and Earthquake Engineering Problems. Neural Comput Appl 21 (1):189–201
12. Gandomi AH, Alavi AH (2011) Multi-stage genetic programming: a new strategy to nonlinear system modeling. Inf Sci 181(23):5227–5239. doi:10.1016/j.ins.2011.07.026
13. Simon D (2008) Biogeography-based optimization. IEEE Trans Evolut Comput 12(6):702–713
14. Wang G, Guo L, Duan H, Liu L, Wang H (2012) Dynamic deployment of wireless sensor networks by biogeography based optimization algorithm. J Sens Actuat Netw 1(2):86–96. doi:10.3390/jsan1020086
15. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483
16. Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2012) Bat algorithm for constrained optimization tasks. Neural Comput Appl. doi:10.1007/s00521-012-1028-9
17. Gandomi AH, Yang X-S, Alavi AH (2012) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput Ger. doi:10.1007/s00366-011-0241-y
18. Gandomi AH, Talatahari S, Yang XS, Deb S (2012) Design optimization of truss structures using cuckoo search algorithm. Struct Des Tall Spec. doi:10.1002/tal.1033
19. Wang G, Guo L, Duan H, Wang H, Liu L, Shao M (2012) A hybrid meta-heuristic DE/CS algorithm for UCAV three-dimension path planning. Sci World J 2012:1–11. doi:10.1100/2012/583973
20. Yang X-S, Sadat Hosseini SS, Gandomi AH (2012) Firefly algorithm for solving non-convex economic dispatch problems

with valve loading effect. Appl Soft Comput 12(3):1180–1186. doi:10.1016/j.asoc.2011.09.017
21. Gandomi AH, Yang X-S, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336. doi:10.1016/j.compstruc.2011.08.002
22. Talatahari S, Gandomi AH, Yun GJ (2012) Optimum design of tower structures using Firefly Algorithm. Struct Des Tall Spec
23. Wang G, Guo L, Duan H, Liu L, Wang H (2012) A modified firefly algorithm for UCAV path planning. Int J Hybrid Inf Technol 5(3):123–144
24. Gandomi AH, Alavi AH (2012) Krill Herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simulat 17(12):4831–4845. doi:10.1016/j.cnsns.2012.05.010
25. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2): 60–68. doi:10.1177/003754970107600201
26. Wang G, Guo L, Duan H, Wang H, Liu L, Shao M (2012) Hybridizing Harmony Search with Biogeography based Optimization for Global Numerical Optimization. J Comput Theor Nanosci
27. Yang X-S (2011) Optimization Algorithms. In: Koziel S, Yang X-S (eds) Computational Optimization, Methods and Algorithms, vol 356. Studies in Computational Intelligence. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, pp. 13–31. doi: 10.1007/978-3-642-20859-1_2
28. Zhao SZ, Suganthan PN, Pan Q-K, Fatih Tasgetiren M (2011) Dynamic multi-swarm particle swarm optimizer with harmony search. Expert Syst Appl 38(4):3735–3742. doi:10.1016/j.eswa.2010.09.032
29. Dorigo M, Stutzle T (2004) Ant Colony Optimization. MIT Press, Cambridge
30. Duan H, Zhao W, Wang G, Feng X (2012) Test-sheet composition using AHP and TS/BBO. Math Probl Eng 2012:1–22. doi:10.1155/2012/712752
31. Wang G, Guo L, Duan H, Liu L, Wang H, Shao M (2012) Path planning for uninhabited combat aerial vehicle using Hybrid Meta-Heuristic DE/BBO algorithm. Adv Sci Eng Med 4(6): 550–564. doi:10.1166/asem.2012.1223
32. Beyer H (2001) The theory of evolution strategies. Springer, New York
33. Gandomi AH, Yun GJ, Yang X-S, Talatahari S (2013) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simulat 18(2):327–340. doi:10.1016/j.cnsns.2012.07.017
34. Khatib W, Fleming P (1998) The stud GA: A mini revolution? In: Eiben A, Back T, Schoenauer M, Schwefel H (eds) Proceeding of the 5th International Conference on Parallel Problem Solving from Nature (1998) Parallel problem solving from nature. Springer-Verlag, London, pp 683–691
35. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3(2):82–102
36. Fletcher R, Powell MJD (1963) A rapidly convergent descent method for minimization. Comput J 6(2):163–168
37. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92. doi:citeulike-article-id:7471117
38. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evolut Comput 10(3):281–295. doi:10.1109/tevc.2005.857610
39. Omran MGH, Mahdavi M (2008) Global-best harmony search. Appl Math Comput 198(2):643–656. doi:10.1016/j.amc.2007.09.004
40. Brits R, Engelbrecht A, Van den Bergh F (2007) Locating multiple optima using particle swarm optimization. Appl Math Comput 189(2):1859–1883
41. Tang K, Li X, Suganthan PN, Yang Z, Weise T (2010) Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Nature Inspired Computation and Applications Laboratory, USTC
42. Mallipeddi R, Suganthan P (2010) Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Nanyang Technological University, Singapore