Algorithms **2015**, *8*, 336-366; doi:10.3390/a8020336

OPEN ACCESS

algorithms

ISSN 1999-4893
www.mdpi.com/journal/algorithms

Article

MAKHA—A New Hybrid Swarm Intelligence Global Optimization Algorithm

Ahmed M.E. Khalil [1,†], Seif-Eddeen K. Fateen [1,2,†] and Adrián Bonilla-Petriciolet [3,*]

[1] Department of Chemical Engineering, Faculty of Engineering, Cairo University, Giza 12613, Egypt; E-Mails: ahmad.alsayed@eng1.cu.edu.eg (A.M.E.K.); sfateen@alum.mit.edu (S.-E.K.F.)

[2] Department of Petroleum and Energy Engineering, American University in Cairo, New Cairo 11835, Egypt

[3] Department of Chemical Engineering, Aguascalientes Institute of Technology, Aguascalientes 20256, Mexico

[†] These authors contributed equally to this work.

[*] Author to whom correspondence should be addressed; E-Mail: petriciolet@hotmail.com; Tel.: +52-449-910-5002 (ext. 127); Fax: +52-449-910-5002.

Academic Editor: George Karakostas

Received: 2 April 2015 / Accepted: 3 June 2015 / Published: 19 June 2015

Abstract: The search for efficient and reliable bio-inspired optimization methods continues to be an active topic of research due to the wide application of the developed methods. In this study, we developed a reliable and efficient optimization method via the hybridization of two bio-inspired swarm intelligence optimization algorithms, namely, the Monkey Algorithm (MA) and the Krill Herd Algorithm (KHA). The hybridization made use of the efficient steps in each of the two original algorithms and provided a better balance between the exploration/diversification steps and the exploitation/intensification steps. The new hybrid algorithm, MAKHA, was rigorously tested with 27 benchmark problems and its results were compared with the results of the two original algorithms. MAKHA proved to be considerably more reliable and more efficient in tested problems.

Keywords: global optimization; nature-inspired methods; monkey algorithm; krill herd algorithm; hybridization

1. Introduction

The use of stochastic global optimization methods has gained popularity in a wide variety of scientific and engineering applications as those methods have some advantages over deterministic optimization methods [1]. Those advantages include the lack of the need for a good initial guess and the ability to handle multi-modal and non-convex objective functions without the assumptions of continuity and differentiability.

Several stochastic methods have been proposed and investigated in challenging optimization problems using continuous variables. Such methods include simulated annealing, genetic algorithms, differential evolution, particle swarm optimization, harmony search, and ant colony optimization. In general, these methods may show different numerical performances and, consequently, the search for more effective and reliable stochastic global optimization methods is currently an active area of research. In particular, the Monkey Algorithm (MA) [2] and the Krill-Herd Algorithm (KHA) [3] are two new, nature-inspired stochastic optimization method that are gaining popularity in finding the global minimum of diverse science and engineering application problems. For example, MA and its variants were recently used for the power system optimization [4], the coordinated control of low frequency oscillation [5], and for finding optimal sensor placement in structural health monitoring [6–8]. KHA is a new method and has been used in network route optimization [9] and economic load dispatch [10].

Since the development of those two algorithms, some modifications have been proposed to improve their performance. The modifications often involved variations of the search rules or hybridization with other algorithms. For example, chaotic search methods were added to MA [11] and KHA [12–14] to improve their performance. MA modifications also included the use of new parameters that change their value during the optimization [11], changing the watch-jump process of MA to make use of information obtained by other monkeys [5], redesigning the MA steps to facilitate discrete optimization problems [15], and incorporating an asynchronous climb process [7]. Other KHA modifications ncluded the addition of local Lévy-flight move [16], adapting KHA to discrete optimization [9], better exchange of information between top krill during motion calculation [17], and hybridization of KHA with Harmony Search [18] and Simulated Annealing [19].

Hybridization is an enhancement in optimization algorithms in which operators from a certain algorithm are combined with other operators from another algorithm to produce more reliable and effective synergistic entity and get better results than that of the main parent algorithms. For example, SA (simulated annealing) is trajectory-based technique that is better at intensification or exploitation; it can detect the best solution with high probability in a confined search space. On the other hand, GA (Genetic algorithm) is regarded as population-based algorithm, which carrys out a diversification process and identifies promising regions of the search space [20,21]. An integration of both algorithms generated the SA-GA hybrid, which outperformed simple GA and a Monte Carlo search in terms of reliability and efficiency of the results. The improved genetic algorithm was implemented to optimize the weight of a pressure vessel under the burst pressure constraint [22]. It was contrived to cope with the phenomena of stagnation in earlier and later stages so that the ability of GA to escape entrapment in local minimum was used and wisely associated with SA's intensification behavior.

Other examples of hybrids display the enhancement in results better than their parent algorithms. Hybrid Evolutionary Firefly Algorithm (HEFA) is a combination of FA and DE (Differential evolution) in which population was initiated, fitness values were evaluated, population was sorted and then split to two halves, the fitter half follows the FA, while the worse half evolves with the DE. HEFA was able to outperform the parent algorithms and GA, but with a longer computation time than GA [23]. BF-PSO hybrid is composed of BFO (Bacterial Forage Optimization) and PSO (Particle Swarm optimization), which was formed to improve the BFO's ability to tackle multi-modal functions [24]. ACO has been hybridized with a Pseudo-Parallel GA (PPGA) for solving set of optimization problems, and PPGA-ACO obtained successfully the best minimum with minimal computational effort as compared to PPGA, GA, and neural networks [25]. The HS-BA (Harmony Search and Bees Algorithm hybrid) made the best results on eight out of 14 data instances of the University Course Timetabling Problem (UCTP), as compared to VNS (Variable Neighborhood Search), BA (Bees Algorithm), and TS (Tabu Search) [26]. The explorative ability of FA was enhanced by adding GA, and the hybrid was tested on a number of benchmarks and gained better results than standard FA and a number of PSO variants. However, it was often either outperformed by, or at best comparable to, FAs that use Gaussian distribution (Brownian motion) instead of Lévy flights, or use learning automata for parameter adaptation [27]. For further reading about hybrid algorithms, their methods, and strategies, please refer to the following: GA and BFO [28,29], PSO and SA [30], GA and SA [31], ACO and TS [32], and GA and PSO [33].

In this study, a new hybrid stochastic optimization method was developed, which uses features from the two algorithms, MA and KHA. The aim of this paper is to present the new algorithm and to evaluate its performance in comparison with the original algorithms. The remainder of this paper is divided as follows: Sections 2 and 3 introduce the Monkey Algorithm and the Krill Herd Algorithm, respectively. Section 4 introduces the proposed hybrid algorithm. The numerical experiments performed to evaluate the modification are presented in Section 5. The results of the numerical experiments are presented and discussed in Section 6. Finally, Section 7 summarizes the conclusions of this study.

2. The Monkey Algorithm (MA)

This algorithm [2] mimics the process in which monkeys climb mountains to reach the highest point. The climbing method consists of three main processes:

1) The climb process: In this exploitation process, monkeys search the local optimum solution extensively in a close range.
2) The watch-jump process: In this process, monkeys look for new solutions with objective value higher than the current ones. It is considered an exploitation and intensification method.
3) The somersault process: This process is for exploration and it prevents getting trapped in a local optimum. Monkeys search for new points in other search domains. In nature, each monkey attempts to reach the highest mountaintop, which corresponds to the maximum value of the objective function. The fitness of the objective function simulates the height of the mountaintop, while the decision variable vector is considered to contain the positions of the monkeys. Changing the sign of the objective function allows the algorithm to find the global minimum instead of the global maximum. The pseudo-code for this algorithm is shown in Figure 1.

```
Part I: Set parameters and initialize
        Randomly generate initial population of NP positions of the monkeys ($X_{ij}$)
        i=1,2,…..NP, and j=1,2,..NV
        Define low and high boundaries of $X_i$
        Assign values to parameters: a, b, c and d
Part II: Algorithm loops
        For I = 1: $I_{max}$ (max. iteration)
        Climb process loop
            For counter l= 1: Nc (number of cycles)
                Randomly generate vector $\Delta X(l)$
                Generate the pseudo-gradient of the objective function $f'(X_i)$
                Generate new monkeys' position $Y_i$ using a and $f'$
                Update $X_i$ with $Y_i$ if feasible
            End For $N_c$
        Watch-jump process
          Generate new $Y_i(I)$ from $(X_i-b, X_i+b)$
          If $-f(Y_i) \geq -f(X_i)$
                    Update $X_i(I)$ with $Y_i(I)$ if feasible (i.e., within limits)
          End If
        Apply climb process loop again
            Rank the positions and find current best solution so far
        Somersault process
          Estimate Pivot P(I) from $X_{ij}$, c and d
          Calculate $Y_i(I)$ around the pivot
          Update $X_i(I)$ with $Y_i(I)$ if feasible and repeat somersault until feasible
          Rank the positions and find current best solution
          If (the new monkey's position violates its boundary limits)
                    Use Search-based function to handle constraints
          End If
        End For I
        Return the highest mountaintop ($-f(X)$) and its position X
```

Figure 1. The pseudo-code of the Monkey Algorithm (MA).

There are different equations for the somersault process. In this study, the somersault jump steps were as follows:

a) Random generation of *α* from the somersault interval [c, d] where *c* and *d* governs the maximum distance that the monkey can somersault.

b) Create a pivot *P* by the following equation:

$$P_i = \frac{1}{NP-1} \sum_{l=1}^{NP} \left(\sum_{i=1}^{NP} X_{lj} - X_{ij} \right) \tag{1}$$

where $P_i = (P_1, P_2, ..., P_{NV})$, *NP* is the population number and *X* is the monkey position.

c) Get *y* (Monkey new position) from

$$Y_i = X_i + \alpha \left| P_i - X_{ij} \right| \tag{2}$$

d) Update X_i with Y_i if feasible (within boundary limits) or repeat until feasible.

3. The Krill Herd Algorithm (KHA)

This bio-inspired algorithm [3] simulates the herding behavior of krill individuals. The values of the objective function correspond to the krill movements, which represent the minimum distances of each individual krill from food and from the highest density of the herd. The krill motion involves three main mechanisms,

a) The movement induced by the presence of other individuals.
b) The foraging activity.
c) Random diffusion.

In addition, two adaptive genetic operators are used: Mutation and Crossover algorithms. In nature, when the predation action is made by predators, such as seals, penguins or sea birds, they remove krill individuals resulting in decreasing the krill density. Afterwards, the krill individuals increase their density and find food. So, the individual krill moves towards the best optimum solution as it searches for the highest density and food. The closer the distance to the highest density and food, the less value of the objective function is obtained. The objective function value of each individual krill is supposed to be an imaginary distance and contains a combination of the distance from food and from the highest density of the krill swarm. The individuals' variables of the function are considered to be time-dependent positions of an individual krill, which are governed by the three mentioned features along with the genetic operator. The pseudo-code for this algorithm is shown in Figure 2.

It is important to note that there are four types of KHA: (1) KHA without any genetic operators (KHA I); (2) KH with crossover operator (KHA II); (3) KHA with mutation operator (KHA III); and (4) KH with crossover and mutation operators (KHA IV). In this study, KHA IV was used in solving the benchmark problems.

```
Part I: Set parameters and initialize
        Define lower and upper bounds of X, and generate random population (NP) of the
        krill positions ($X_i$) within the variable limits
        Assign values to KHA parameters: $N_{max}$, $V_f$ and $D_{max}$, calculate others: $C_t$, $w_f$ and $w_N$
        Set time (t), induced motion ($N_i$), food foraging ($F_i$) and diffusion ($D_i$) to zero
        Calculate genetic operator parameters ($Cr$, $Mu$) using Eq.15 and 17
Part II: Algorithm loops
        Evaluate the fitness of krill individuals
        For I = 1: $I_{max}$ (max. iteration)
            Calculate time interval $\Delta t(I)$
            Induced motion step
              Evaluate local, target swarm density and direction of motion for each krill
              Calculate the distances between each 2 krill positions ($ds_{ij}$)
              Identify the induced motion sensing distance ($ds_i$)
                If $ds_{ij} < ds_i$
                    Calculate and update the induced motion $N_i(I)$
              End If
            Foraging motion step
              Generate center of food density $X^{food}(I)$ for the herd
              Calculate and update the foraging motion $F_i(I)$
            Physical diffusion step
              Calculate and update the diffusion motion $D_i(I)$
              Move the positions of the krill individuals after $t+\Delta t$ period
              Update the time (t)
              Evaluate the fitness of krill individuals
              Rank the krill positions according to minimum fitness
              Return the best krill position and its fitness so far
            Implement the genetic operator
              Apply crossover and mutation on each krill position
              Update the krill positions and check their limits
        End For I
        Obtain the best solution
```

Figure 2. The pseudo-code of the Krill Herd Algorithm (KHA).

4. MAKHA Hybrid Algorithm

MAKHA is a new hybrid algorithm, which combines some of the mechanisms and processes of MA and KHA to get a reliable algorithm with appreciated performance. The steps of both algorithms include exploration/diversification and exploitation/intensification features as follows. The exploration/diversification features of MA are the somersault process and the watch-jump process, while for KHA, they are the physical random diffusion and the genetic operators. On the other hand, the exploitation/intensification features of MA are the climb and the watch-jump process, while for KHA, they are the induced motion and the foraging activity.

Both algorithms attempt to balance between exploration/diversification and exploitation/ intensification features. MA has two exploration operators and two exploitation operators. The watch-jump process acts as both an exploration and an exploitation operator. The somersault operator is a high-performing diversification operator that makes a good use of the pivot function. Since MA is an exploration-dominant algorithm, the exploitation balance is brought to the algorithm by running the climb process twice per iteration. In each process, the MA algorithm uses a large number of cycles that reaches up to 2000 cycles in some problems. Increasing the number of cycles reduces the computational efficiency because it increases the number of function evaluations (NFE).

Even though KHA also has two exploration operators and two exploitation operators, its exploration component is not dominating because the physical random diffusion is a less efficient exploration operator than the somersault operator. Thus, the entrapment in local minima is more probable in KHA than in MA. The trapping problem can be addressed in the KHA by the use of two genetic operators (crossover and mutation), which appear in KHA IV algorithm. Since the foraging movement is a high-performing exploitation operator, KHA could be considered an exploitation-dominant algorithm.

An equal number of exploration and exploitation operators does not necessitate a balance between exploration and exploitation. The performance of operator is a critical factor. Assessing the performance of an operator can be done by replacing the exploration or exploitation operator in one algorithm with the same type of operator in the other algorithm. Testing the modified algorithms with benchmark problems can reveal whether or not the replaced operator was performing its function efficiently relative to the other operator.

To improve the performance of the algorithm such that the modified algorithm outperforms the two original algorithms, we aimed at using the best performing exploration and exploitation operators from the two algorithms. The hybrid algorithm, MAKHA, was constructed from the following processes:

1. The watch-jump process.
2. The foraging activity process.
3. The physical random diffusion process.
4. The genetic mutation and crossover process.
5. The somersault process.

The climb process, which consumes a high NFE, was not included in the hybrid algorithm. The random diffusion step was included in only one of MAKHA's variant as explained below.

MAKHA was implemented in two different ways: MAKHA I, which does not use random diffusion; and MAKHA II, which uses the random diffusion step. It was found, as shown in the Results

Section, that MAKHA I was more suitable for low-dimensional problems, while MAKHA II was better for the high-dimensional problems (NV = 50).

```
Part I: Set parameters and initialize
          Define lower and upper bounds of X
          Generate random population (NP) of the hybrid positions ($X_i$) within the
          feasible limits
          Assign values to KHA parameters: $V_f$ and $D_{max}$, calculate others: $C_t$ and $w_f$
          Assign values to MA parameters: b, c and d
          Set time (t), food foraging (Fi) and diffusion ($D_i$) to zero
          Calculate genetic operator parameters ($Cr$, $Mu$) using Eq.15 and 17
Part II: Algorithm loops
          Evaluate the fitness of the hybrid individuals
          For I = 1: $I_{max}$ (max. iteration)
           Watch-jump process
             Generate new $Y_i$(I) from ($X_{ij}-b$, $X_{ij}+b$)
             If $-f(Y_i) \geq -f(X_i)$
                          Update $X_i$(I) with $Y_i$(I) if feasible (i.e., within limits)
             End If
            Foraging motion step
              Generate center of food density $X^{food}$(I) for the hybrids
              Calculate and update the foraging motion $F_i$(I)
            Physical diffusion step
              Calculate and update the diffusion motion $D_i$(I)
              Move the positions of the hybrid individuals after t+$\Delta$t period
              Update the time t(I)
              Evaluate the fitness of the hybrid individuals H(I)
              Rank the hybrid positions according to minimum fitness
              Return the best hybrid position and its fitness so far
            Implement the genetic operator
              Apply crossover and mutation on each hybrid position
              Update the hybrid positions and check their limits
            Somersault process
              Estimate Pivot P(I) from $X_{ij}$, c and d
              Calculate $Y_i$(I) around the pivot
              Update $X_i$(I) with $Y_i$(I) if feasible and repeat somersault until feasible
              Rank the positions and find current best solution $X_{best}$(I)
         End For I
         Get the best solution
```

Figure 3. The pseudo-code of hybrid MAKHA.

The general pseudo-code for this algorithm is shown in Figure 3, while the equations used are as follows:

- *Initialization procedure:*
    - Random generation of population in which the positions of the hybrid agent (monkey/krill) are created randomly, $X_i = (X_{i1}, X_{i2}, \ldots, X_{i(NV)})$ where $i = 1$ to NP, which represents the number of hybrids, while NV represents the dimension of the decision variable vector.
- *The fitness evaluation and sorting:*
    - $H_i=f(X_i)$ where H stands for hybrid fitness and f is the objective function used.
- *The watch-jump process:*
    - Random generation of X_i from $(X_{ij} - b, X_{ij} + b)$ where b is the eyesight of the hybrid (monkey in MA) which indicates the maximal distance the hybrid can watch and $Y_i = (Y_{i1}, Y_{i2}, \ldots, Y_{i(NV)})$, which are the new hybrid positions.
    - If $-f(Y_i) \geq -f(X_i)$ then update X_i with Y_i if feasible (*i.e.*, within limits).
- *Foraging motion:*
    - Depends on food location and the previous experience about the location.
    - Calculate the food attractive β_i^{food} and the effect of best fitness so far β_i^{Best}

$$\beta_i^{food} = C^{food}\hat{H}_{i,food}\hat{X}_{i,food} \tag{3}$$

$$\beta_i^{Best} = \hat{H}_{i,ibest}\hat{X}_{i,ibest} \tag{4}$$

where C^{food} is the food coefficient, which decreases with time and is calculated from:

$$C^{food} = 2(1 - I / I_{max}) \tag{5}$$

where I is the iteration number and I_{max} is the maximum number of iterations.

- The center of food density is estimated from the following equation:

$$X^{food} = \frac{\sum_{i=1}^{NP} \frac{1}{H_i} X_i}{\sum_{i=1}^{NP} \frac{1}{H_i}} \tag{6}$$

and H_{ibest} is the best previously visited position.

- $\hat{H}$ and $\hat{X}$ are unit normalized values obtained from this general form:

$$\hat{X}_{i,j} = \frac{X_j - X_i}{\|X_j - X_i\| + \varepsilon} \tag{7}$$

$$\hat{H}_{i,j} = \frac{H_j - H_i}{H_{worst} - H_{best}} \tag{8}$$

where ε is a small positive number that is added to avoid singularities. H_{best} and H_{worst} are the best and the worst fitness values, respectively, of the hybrid agents so far. *H* stands for the hybrid fitness and was used as *K* symbol in krill herd method.

- The foraging motion is defined as

$$F_i = V_f \beta_i + w_f F_i^{old} \tag{9}$$

where V_f is the foraging speed, w_f is the inertia weight of the foraging motion in the range [0, 1], and F_i^{old} is the last foraging motion.

- *Physical diffusion:*

This is an exploration step that is used at high dimensional problem, then

$$\mathrm{D_i} = \mathrm{D}_{max}(1 - \mathrm{I}/\mathrm{I}_{max})\delta \tag{10}$$

where D_{max} is the maximum diffusion speed and δ is the random direction vector.

- *Calculate the time interval* Δt

$$\Delta t = C_t \sum_{L=1}^{NV} \left(UB_L - LB_L \right) \tag{11}$$

where C_t is constant.

- *The step for position is calculated through:*

$$\frac{dX_i}{dt} = F_i + D_i \tag{12}$$

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

where $\frac{dX_i}{dt}$ represents the velocity of the hybrid agent (Krill/Monkey).

- *Implementation of genetic operator:*

- *Crossover*

$$X_{i,m} = \begin{Bmatrix} X_{r,m}, & random < Cr \\ X_{i,m,} & otherwise \end{Bmatrix} \tag{14}$$

where $r \in \{1,2,...,i-1,i+1,...,NP\}$ and *Cr* is the crossover probability

$$C_r = 0.8 + 0.2\hat{K}_{i,best} \tag{15}$$

- *Mutation*

$$X_{i,m} = \begin{Bmatrix} X_{gbest,m} + \mu(X_{p,m} - X_{q,m}), & random < Mu \\ X_{i,m}, & otherwise \end{Bmatrix} \tag{16}$$

where μ is a random number, $p,q \in \{1,2,...,i-1,i+1,...,NP\}$ and *Mu* is the mutation probability:

$$Mu = 0.8 + 0.05\hat{H}_{i,best} \tag{17}$$

$$\hat{H}_{i,best} = (H_i - H_{gb}) / (H_{worst} - H_{gb}) \tag{18}$$

where H_{gb} is the best global fitness of the hybrid so far and X_{gbest} is its position.

- *The somersault process:*
  - α is generated randomly from [c, d] where c and d are somersault interval. Two different implementations of the somersault process can be used:

Somersault I

- Create the pivot P [2]:

$$P_i = \frac{1}{NP}\sum_{i=1}^{NP} X_{ij} \text{ where } P_i = (P_1, P_2, \ldots, P_{NV}) \tag{19}$$

$$Y_i = X_i + \alpha(P_i - X_{ij}) \tag{20}$$

- Update X_i with Y if feasible or repeat until feasible.

Somersault II

- Create a pivot P by this equation used in MA:

$$P_i = \frac{1}{NP-1}\sum_{l}^{NP}\left(\sum_{i=1}^{NP} X_{lj} - X_{ij}\right) \tag{21}$$

- Get Y (*i.e.*, the hybrid new position)

$$Y_i = X_i + \alpha\left|P_i - X_{ij}\right| \tag{22}$$

- Update X_i with Y if feasible and repeat until feasible.

In summary, MAKHA offers more exploration than KHA and more exploitation than MA.

5. Numerical Experiments

Twenty-seven classical benchmark functions were used to evaluate the performance of MAKHA as compared to the original MA and KHA. Tables 1 and 2 show the benchmark functions used along with their names, number of variables, variable limits and the value of the global minimum. Table 3 lists the parameters used in the three algorithms in which their values were set to give the best attainable results for each algorithm. A new parameter for MAKHA was defined as the midpoint between the lower boundary and the upper boundary of the decision variables X. It is calculated from this formula:

$$R = 0.5\sum_{L=1}^{NV}(UB_L - LB_L) \tag{23}$$

in which, according to the value R, certain values were assigned to MAKHA's parameters, as seen in Table 3.

Table 1. Benchmark functions used for testing the performance of MA, KHA and MAKHA.

Name	**Objective Function**
Ackley [34]	$f_1 = 20\left(1 - e^{-0.2\sqrt{0.5\left(x_1^2 + x_2^2\right)}}\right) - e^{0.5(\cos 2\pi x_1 + \cos 2\pi x_2)} + e^1$
Beale [35]	$f_2 = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
Bird [36]	$f_3 = \sin(x_1)e^{(1-\cos x_2)^2} + \cos(x_2)e^{(1-\sin x_1)^2} + (x_1 - x_2)^2$
Booth [35]	$f_4 = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
Bukin 6 [35]	$f_5 = 100\sqrt{\left\|x_2 - 0.01x_1^2\right\|} + 0.01\left\|x_1 + 10\right\|$
Carrom table [37]	$f_6 = -\left[\cos x_1 \cos x_2 e^{\left\|1-\sqrt{x_1^2+x_2^2}/\pi\right\|}\right]^2 / 30$
Cross-leg table [37]	$f_7 = -\left[\left\|\sin(x_1)\sin(x_2)e^{\left\|100-\sqrt{x_1^2+x_2^2}/\pi\right\|}\right\| + 1\right]^{-0.1}$
Generalized egg holder [35]	$f_8 = \sum_{i=1}^{m-1}\left\{\begin{array}{l} -(x_{i+1}+47)\sin\left(\sqrt{\left\|x_{i+1}+x_i/2+47\right\|}\right) + \\ \sin\left[\sqrt{\left\|x_i-(x_{i+1}+47)\right\|}\right](-x_i) \end{array}\right\}$
Goldstein–Price [38]	$f_9 = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))$ $(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$
Himmelblau [39]	$f_{10} = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2$
Levy 13 [40]	$f_{11} = \sin^2(3\pi x_1) + (x_1 - 1)^2\left[1 + \sin^2(3\pi x_2)\right] + (x_2 - 1)^2\left[1 + \sin^2(2\pi x_2)\right]$
Schaffer [37]	$f_{12} = 0.5 + \frac{sin^2\left[\sqrt{x_1^2 + x_2^2}\right] - 0.5}{\left[0.001(x_1^2 + x_2^2) + 1\right]^2}$
Zettl [41]	$f_{13})(x_1^2 + x_2^2 - 2x_1)^2 + \frac{x_1}{4}$
Helical valley [42]	$f_{14} = 100\left[(x_3 - 10\theta)^2 + \left(\sqrt{x_1^2 + x_2^2} - 1\right)^2\right] + x_3^2, 2\pi\theta = \tan^{-1}\left(\frac{x_1}{x_2}\right)$
Powell [43]	$f_{15} = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$
Wood [44]	$f_{16} = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 +$ $10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$
Extended Cube [45]	$f_{17} = \sum_{i=1}^{m-1} 100(x_{i+1} - x_i^3)^2 + (1 - x_i)^2$
Shekel 5* [46]	$f_{18} = -\sum_{i=1}^{M}(\sum_{j=1}^{4}(x_j - C_{ji})^2 + \beta_i)^{-1}$

Table 1. *Cont.*

Sphere [47]	$f_{19} = \sum_{i=1}^{m} x_i^2$
Hartman 6 * [48]	$f_{20} = -\sum_{i=1}^{4} \alpha_i \exp(-\sum_{j=1}^{6} A_{ij}(x_j - O_{ij})^2)$
Griewank [49]	$f_{21} = \frac{1}{4000}\left[\sum_{i=1}^{m}(x_i - 100)^2\right] - \left[\prod_{i=1}^{m}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right)\right] + 1$
Rastrigin [50]	$f_{22} = \sum_{i=1}^{m}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$
Rosenbrock [51]	$f_{23} = \sum_{i=1}^{m-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$
Sine envelope sine wave [37]	$f_{24} = \sum_{i=1}^{m-1}\left\{0.5 + \frac{sin^2\left[\sqrt{x_{i+1}^2 + x_i^2}\right] - 0.5}{\left[0.001\left(x_{i+1}^2 + x_i^2\right) + 1\right]^2}\right\}$
Styblinski–Tang [52]	$f_{25} = 0.5\sum_{i=1}^{m}(x_i^4 - 16x_i^2 + 5x_i)$
Trigonometric [53]	$f_{26} = \sum_{i=1}^{m}\left[m + i\,(1 - \cos x_i) - \sin x_i - \sum_{j=1}^{m}\cos x_j\right]^2$
Zacharov [54]	$f_{27} = \sum_{i=1}^{m} x_i^2 + \left(\sum_{i=1}^{m} 0.5\, i\, x_i\right)^2 + \left(\sum_{i=1}^{m} 0.5\, i\, x_i\right)^4$

* Shekel and Hartman parameters were obtained from [36].

Table 2. Decision variables, global optimum of benchmark functions and number of iterations used for testing the performance of MA, KHA, and MAKHA.

Objective Function	NV	Search Domain	Global Minimum	Iterations		
				MA	KHA	MAKHA
Ackley	2	[−35, 35]	0	25	3000	1000
Beale	2	[−4.5, 4.5]	0	25	3000	1000
Bird	2	[−2π, 2π]	−106.765	25	3000	1000
Booth	2	[−10, 10]	0	25	3000	1000
Bukin 6	2	[−15, 3]	0	25	3000	1000
Carrom table	2	[−10, 10]	−24.15681	25	3000	1000
Cross-leg table	2	[−10, 10]	−1	25	3000	1000
Generalized egg holder	2	[−512, 512]	−959.64	124	15,000	5000
Goldstein-Price	2	[−2, 2]	3	25	3000	1000
Himmelblau	2	[−5, 5]	0	25	3000	1000
Levy 13	2	[−10, 10]	0	25	3000	1000
Schaffer	2	[−100, 100]	0	199	15,000	8000
Zettl	2	[−5, 5]	−0.003791	25	3000	1000
Helical valley	3	[−1000, 1000]	0	25	3000	1000

Table 2. *Cont.*

Powell	4	[−1000, 1000]	0	50	6000	2000
Wood	4	[−1000, 1000]	0	25	3000	1000
Extended Cube	5	[−100, 100]	0	25	3000	1000
Shekel 5	4	[0, 10]	−10.1532	25	3000	1000
Sphere	5	[−100, 100]	0	75	9000	1000
Hartman 6	6	[0,1]	−3.32237	25	3000	1000
Griewank	50	[−600, 600]	0	124	15000	5000
Rastrigin	50	[−5.12, 5.12]	0	124	15000	5000
Rosenbrock	50	[−50, 50]	0	124	15000	5000
Sine envelope sine wave	50	[−100, 100]	0	124	15000	5000
Styblinski-Tang	50	[−5,5]	−1958.2995	124	15000	5000
Trigonometric	50	[−1000, 1000]	0	124	15000	5000
Zacharov	50	[−5,10]	0	25	3000	1000

Table 3. Selected values of the parameters used in the implementation of MA, KHA and MAKHA.

Method	**Condition**	**Parameter**	**Selected value**
		b	1
	$R \geq 100$	b	10
		c	−1
MA		d	1
	$R \geq 500$	c	−10
	$R \geq 500$	d	30
		N_C	30
		D_{max}	[0.002, 0.01]
		C_t	0.5
KHA		V_f	0.02
		N_{max}	0.01
		w_f and w_N	[0.1, 0.8]
		b	1
	$R < 2$	b	0.5*R
	$R \geq 100$	b	10
		c	−0.1
MAKHA I		d	0.1
		D_{max}	0
		C_t	0.5
		V_f	0.2
		w_f	0.1
	Somersault I is used		
		b	0.3*R
		c	−R
		d	R
MAKHA II (NV = 50)		D_{max}	[0.002, 0.01]
		C_t	0.5
		V_f	0.02
		w_f	[0.1, 0.8]
	Somersault II is used		

The twenty-seven problems constitute a comprehensive testing of the reliability and effectiveness of the hybrid algorithm. Thirteen functions have two variables only, yet some of them are very difficult to optimize. Surface plots of eight of the two-variable functions are shown in Figure 4. Each problem was solved 30 times by each of the three algorithms. The best value at each iteration was recorded and the means of the best values were calculated amongst the 30 run as a function of the iteration number. The plots of the mean best values *versus* the number of function for the three algorithms provide a clear comparison of both the reliability and the efficiency of the algorithms. Reliability is represented by how far the algorithm predictions of the minimum are from the known global minimum, while efficiency is represented by the number of function evaluations needed for the calculation of this best value.

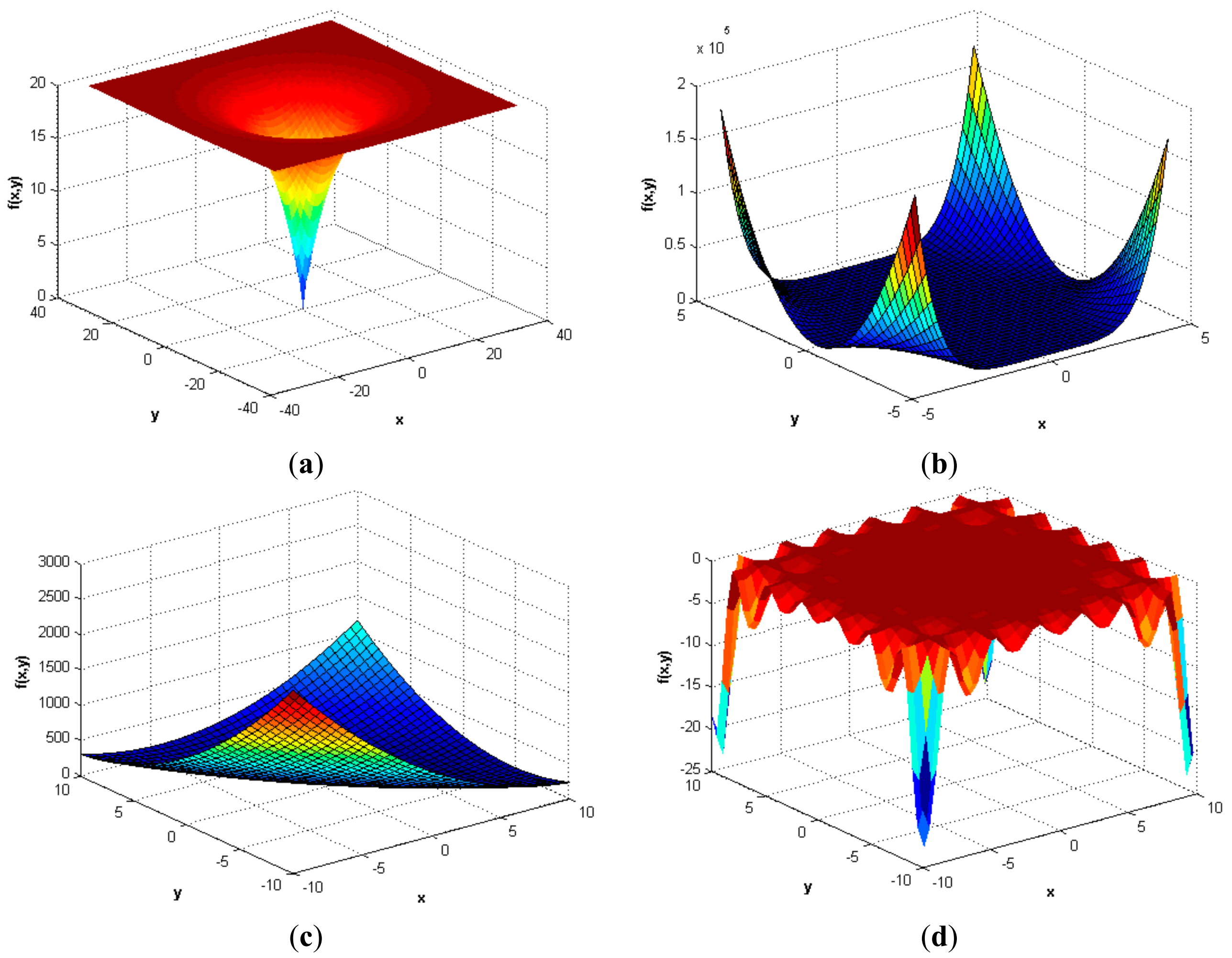
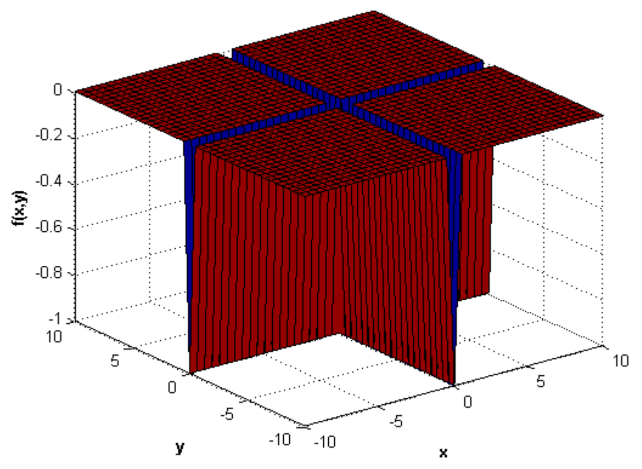
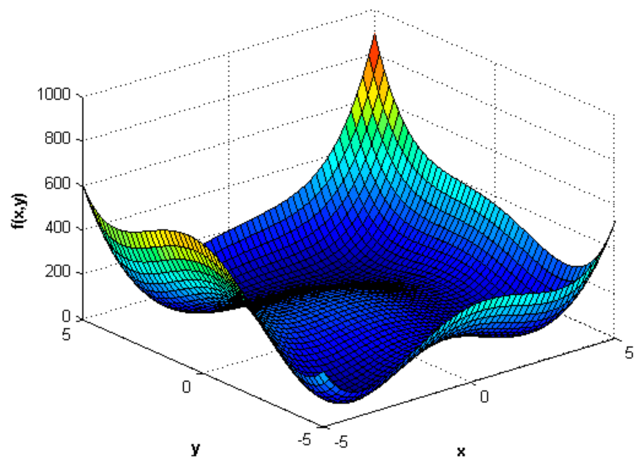


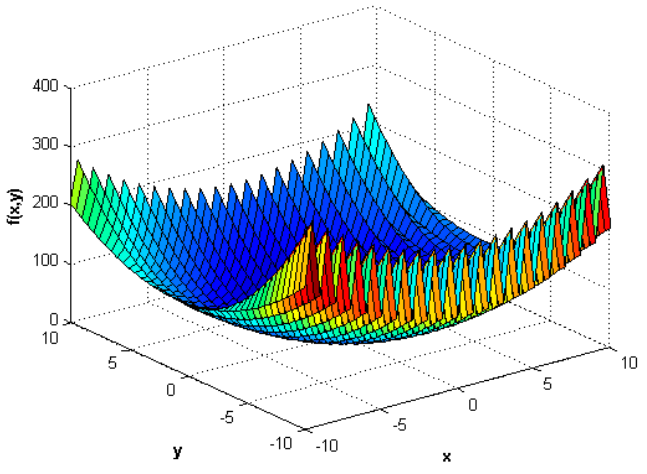
Figure 4. *Cont.*



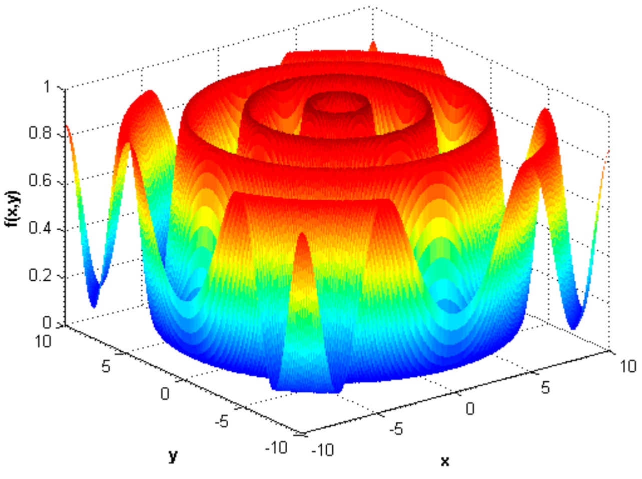
(e)



(f)



(g)



(h)

Figure 4. Surface plots of the two-variable benchmark functions used in this study: (**a**) Ackley, (**b**) Beale, (**c**) Booth, (**d**) Carrom table, (**e**) Cross-leg table, (**f**) Himmelblau, (**g**) Levy 13, and (**h**) Schaffer.

To complete the evaluation of the MAKHA in comparison with the original MA and KHA algorithms, we have employed the performance profile (*PP*) reported by Dolan *et al*. [55], who introduced PP as a tool for evaluating and comparing the performance of optimization software. In particular, PP has been proposed to represent compactly and comprehensively the data collected from a set of solvers for a specified performance metric. For instance, the number of function evaluations or computing time can be considered performance metrics for solver comparison. The PP plot allows visualization of the expected performance differences among several solvers and to compare the quality of their solutions by eliminating the bias of failures obtained in a small number of problems.

To introduce *PP*, consider n_s solvers (*i.e.*, optimization methods) to be tested over a set of n_{p} problems. For each problem p and solver s, the performance metric t_{ps} must be defined. In our study, reliability of the stochastic method in accurately finding the global minimum of the objective function is considered as the principal goal, and hence the performance metric is defined as

$$t_{ps} = f_{calc} - f^* \tag{24}$$

where f^* is the known global optimum of the objective function and f_{calc} is the mean value of that objective function calculated by the stochastic method over several runs. In our study, f_{calc} is calculated from 30 runs to solve each test problem by each solver; note that each run is different because of the random number seed used and the stochastic nature of the method. So, the focus is on the average performance of stochastic methods, which is desirable for comparison purposes.

For the performance metric of interest, the performance ratio, r_{ps}, is used to compare the performance on problem, p, by solver, s, with the best performance by any solver on this problem. This performance ratio is given by

$$r_{ps} = \frac{t_{ps}}{\min\left\{t_{ps} : 1 \le s \le n_s\right\}} \tag{25}$$

The value of r_{ps} is 1 for the solver that performs the best on a specific problem p. To obtain an overall assessment of the performance of solver$s$ on n_p problems, the following cumulative function for r_{ps} is used:

$$\rho_s(\varsigma) = \frac{1}{n_p} size\left\{p : r_{ps} \le \zeta\right\} \tag{26}$$

where $\rho(\varsigma)$ is the fraction of the total number of problems, for which solver s has a performance ratio r_{ps} within a factor of ς of the best possible ratio. The PP of a solver is a plot of $\rho_s(\varsigma)$ *versus* ς; it is a non-decreasing, piece-wise constant function, continuous from the right at each of the breakpoints.

To identify the best solver, it is only necessary to compare the values of $\rho_s(\varsigma)$ for all solvers and to select the highest one, which is the probability that a specific solver will "win" over the rest of solvers used. In our case, the PP plot compares how accurately the stochastic methods can find the global optimum value relative to one another, and so the term "win" refers to the stochastic method that provides the most accurate value of the global minimum in the benchmark problems used.

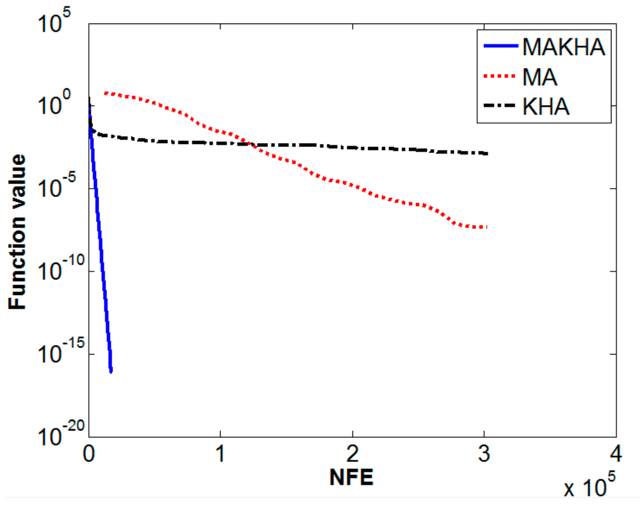
6. Results and Discussion

As stated, each of the numerical experiments was repeated 30 times with different random seeds for MAKHA and the original MA and KHA algorithms. Parameters used for stochastic algorithms are reported in Table 3. The objective function value at each iteration for each trial was recorded. The mean and the standard deviation of the function values were calculated at each iteration. The global optimum was considered to be obtained by the method if it finds a solution within a tolerance value of 10^{-10}. The progress of the mean values is presented in Figures 5–8 for each benchmark function and a brief discussion of those results follows.

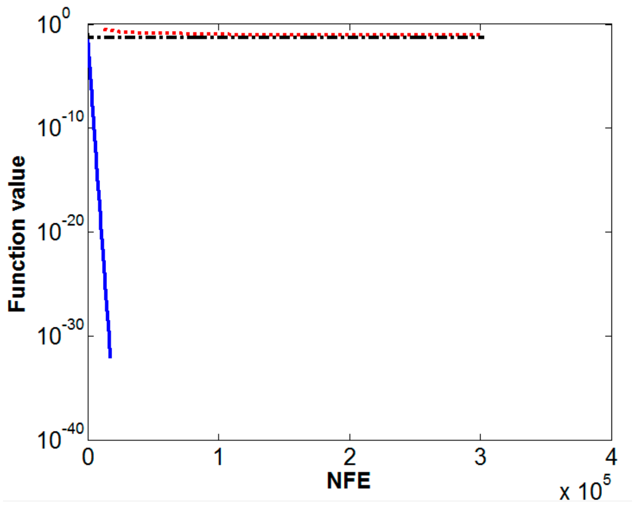
The Ackley function has one minimum only. The global optimum was obtained using MAKHA and in relatively small number of function evaluations as shown in Figure 5a. MA and KHA were not able to obtain satisfactorily the global minimum. The best value obtained by MA was still improving by the end of the run, whereas KHA results were not.

This significant improvement in performance was also clear with the Beale function (Figure 5b). The Beale function has one minimum only, which was only obtained satisfactory by MAKHA. The performance pattern for the three methods was different for the Bird function, as depicted in Figure 5c. MAKHA arrived at the global minimum almost instantaneously. KHA was trapped in a local minimum, while MA was approaching the global minimum, but did not reach it after 300,000 function evaluations. For the Booth function, both MAKHA and KHA arrived at the global minimum but with an improvement in orders of magnitude for the efficiency of MAKHA. On the other hand, MA failed to arrive at the global minimum, as shown in Figure 5d. For Bukin 6 function, none of the three methods obtained the global minimum within the used tolerance. However, MAKHA performed better than KHA, which performed better than MA, as shown in Figure 5e.

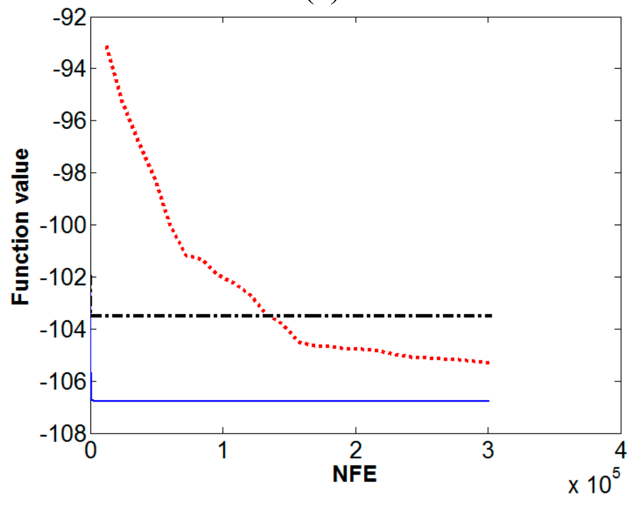
MAKHA and KHA performed similarly for the Carrom table function, as they obtained the global minimum almost instantaneously, as shown in Figure 5f. MA could not achieve the global minimum even after 300,000 function evaluations. For the Cross-leg table function, MAKHA was the only method to obtain the global minimum, as shown in Figure 5g. Note that MA performed better than KHA after many NFE. MAKHA obtained the global minimum for the Generalized Eggholder function almost instantaneously, as depicted in Figure 5h. The other two methods were not able to obtain the global minimum but MA's performance was significantly better than KHA's.



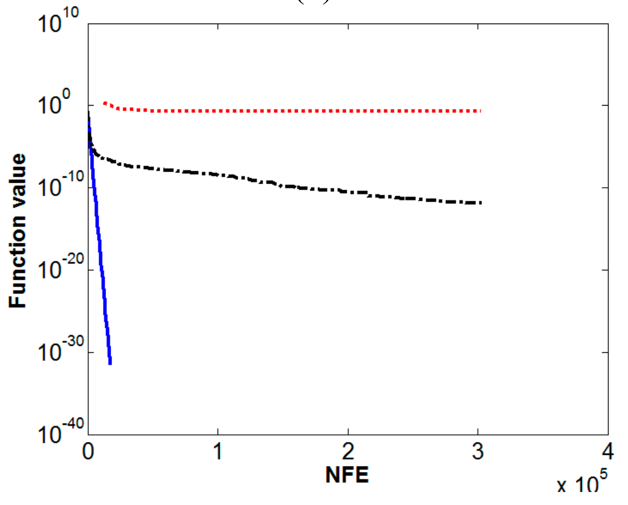
(a)



(b)

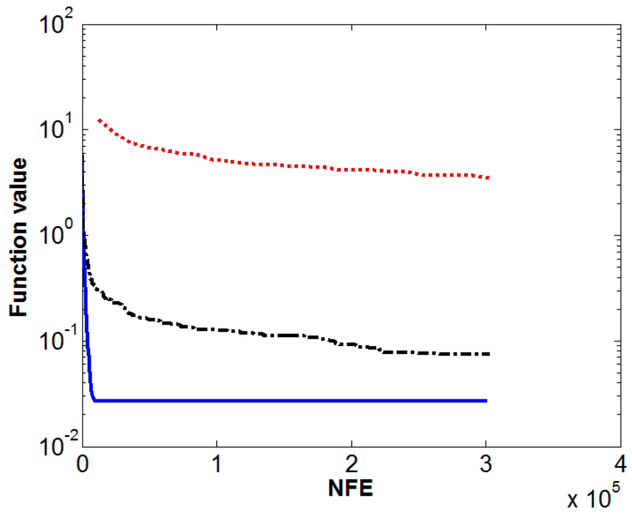


(c)

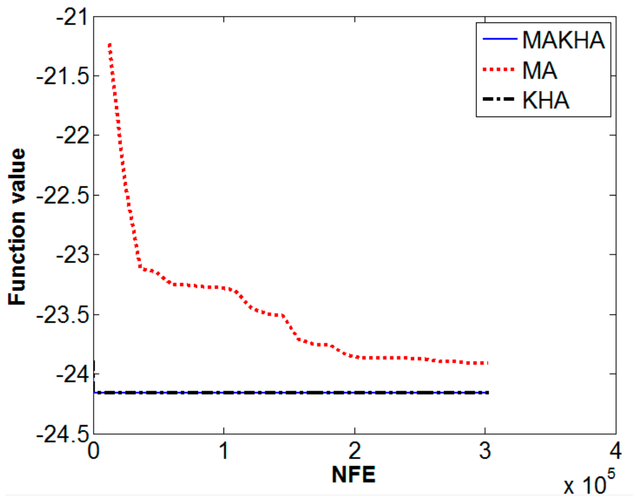


(d)

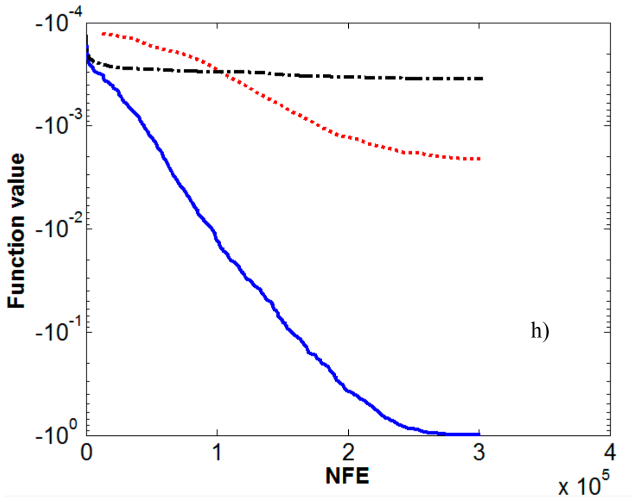
Figure 5. *Cont.*



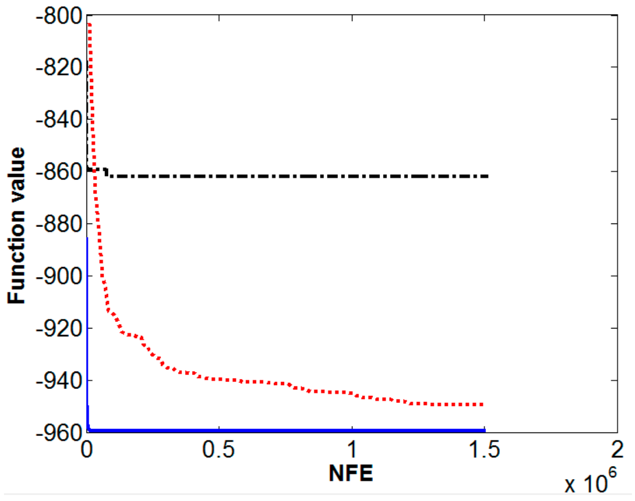
(e)



(f)



(g)

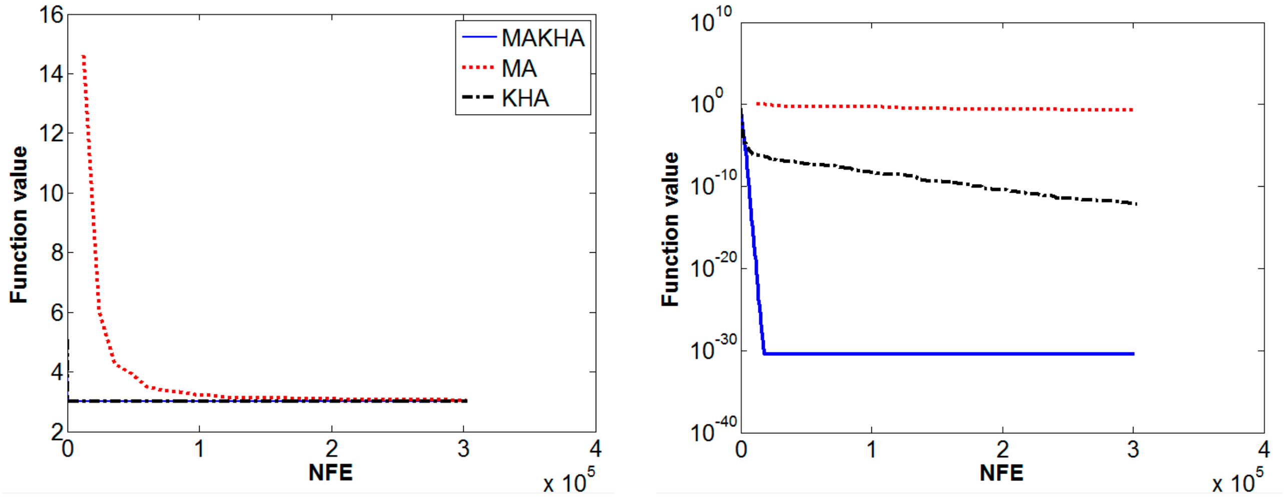


(h)

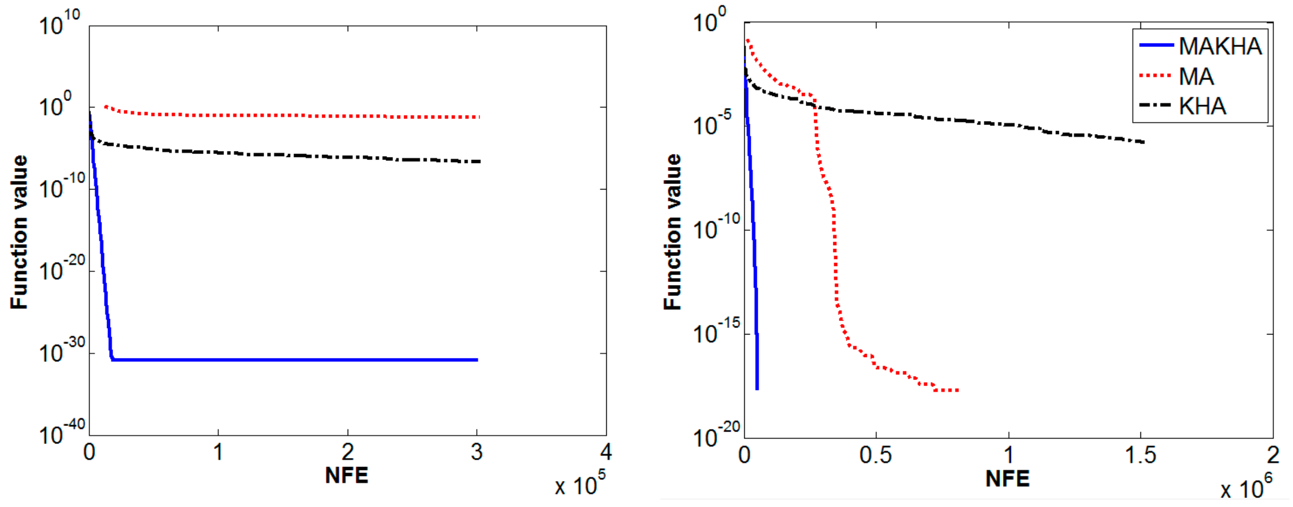
Figure 5. Evolution of mean best values for MA, KHA and MAKHA for: (**a**) Ackley, (**b**) Beale, (**c**) Bird (**d**) Booth, (**e**) Bukin6 (**f**) Carrom table, (**g**) Cross-leg table, and (**h**) Generalized Eggholder functions.

For the Goldstein-Price function, all three algorithms obtained the global minimum, as depicted in Figure 6a. However, MAKHA and KHA were orders-of-magnitude more efficient than MA. For the Himmelblau function, MAKHA obtained the global minimum at low NFE, KHA obtained it at high NFE, and MA was not able to obtain it after 300,000 NFE, as shown in Figure 6b. This performance was almost exactly repeated with the Levy 13 function, as shown in Figure 6c. As for the Schaffer function, both MAKHA and MA converged to the global minimum within the used tolerance. MAKHA was more efficient than MA in terms of the NFE required to obtain the global minimum, as shown in Figure 6d. KHA failed to converge to the global minimum for this particular function. Figure 6e shows the evolution pattern for the Zettl function. MAKHA and KHA obtained the global minimum within a small NFE, while MA obtained it after considerably more NFE. The Helical Valley function has three variables. The evolution of the mean best values of the three algorithms is reported in Figure 6f and results showed that the performance of MAKHA was better than the other two algorithms. MAKHA efficiently obtained the global minimum, while the other two could not, with

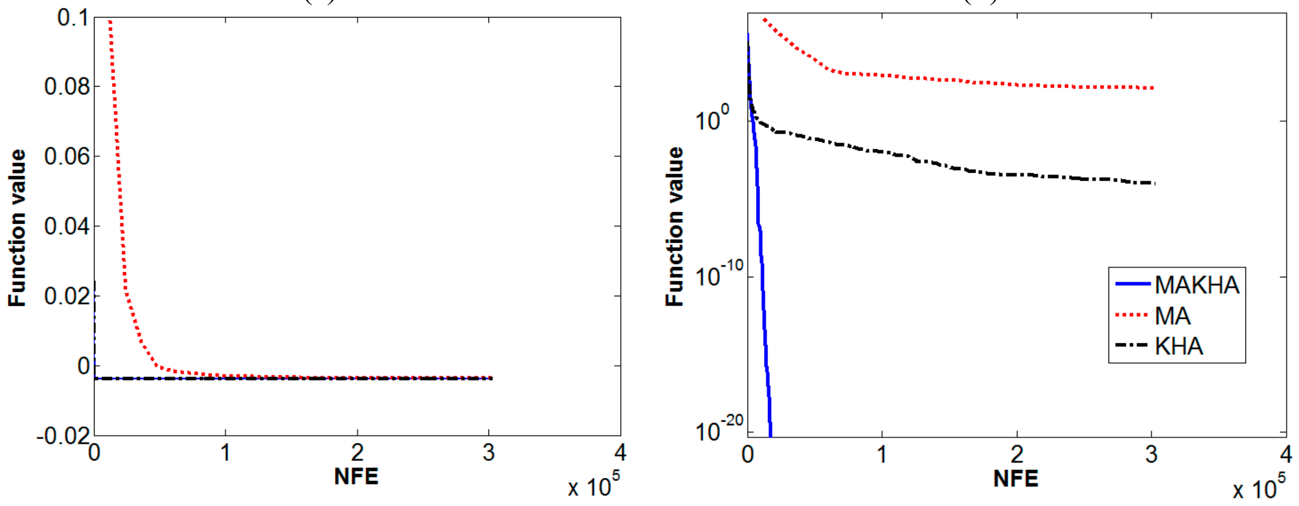
KHA performing better than MA. This convergence performance is almost exactly repeated with the two four-variable functions, the Powell function (Figure 6g) and with the Wood function (Figure 6h).



(a) **(b)**

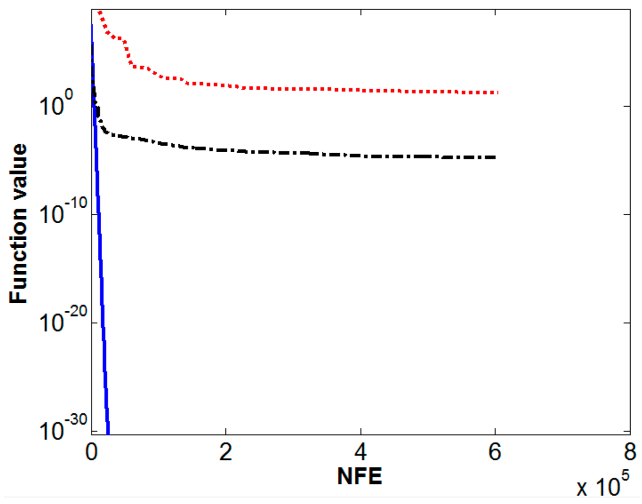


(c) **(d)**

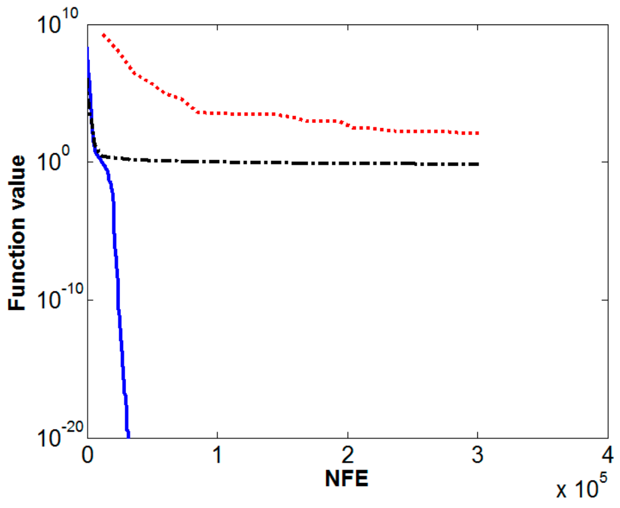


(e) **(f)**

Figure 6. *Cont.*



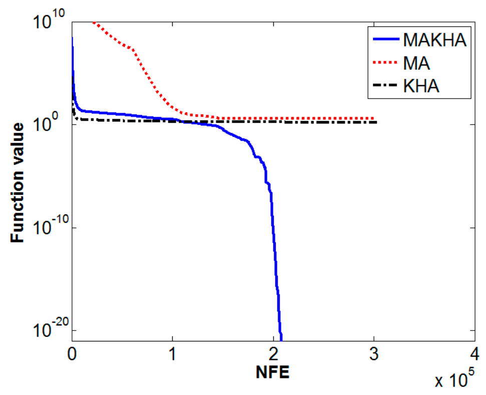
(g)



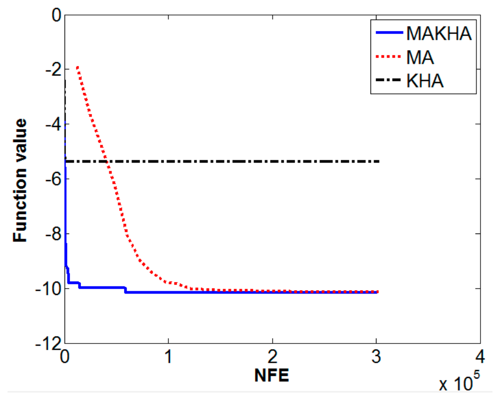
(h)

Figure 6. Evolution of mean best values for MA, KHA and MAKHA for: (**a**) Goldstein–Price (**b**) Himmelblau, (**c**) Levy 13, (**d**) Schaffer functions, (**e**) Zettl, (**f**) Helical Valley, (**g**) Powell and (**h**) Wood.

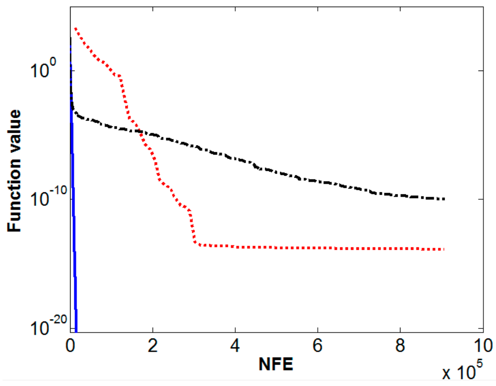
The evolution of the mean best values of the three algorithms for the five-variable functions, which are the Extended Cube, Shekel and Sphere functions, are reported in Figure 7a–c, respectively. For the Extended Cube function, MAKHA was able to obtain the global minimum, while the other two methods failed to converge to the global minimum. For the Shekel function, MAKHA obtained the global minimum very efficiently, as compared to MA, which obtained it at a higher NFE. KHA failed to converge to the global minimum for this function. A relative close pattern is repeated with the five-variable sphere function. Hartman function is a fifty-variable function and its results are depicted in Figure 7d. MAKHA and KHA converged to the global minimum efficiently, while MA failed to achieve it after 300,000 function evaluations. For the Griewank function, shown in Figure 7e, the three algorithms failed to converge to the global minimum. However, MAKHA performance was considerable better than the performance of the other two algorithms, which performed similarly. The Rastrigin function is one of the three functions in which MAKHA was not the top performer. The results for the Rastrigin function are reported in Figure 7f. The three methods did not obtain the global minimum. However, MA performed better than MAKHA, which in turn performed better than KHA. For the Rosenbrock function, whose results are shown in Figure 7g, the three algorithms did not obtain the global minimum, within the acceptable tolerance, with 1,500,000 function evaluations. However, the best value obtained by MAKHA is seven orders-of-magnitude better than that obtained by the other two algorithms. Sine Envelope Sine function is the second function in which MAKHA did not outperform MA, see Figure 7h. MA was the only method to achieve the global minimum. MAKHA's performance was significantly better than KHA's performance.



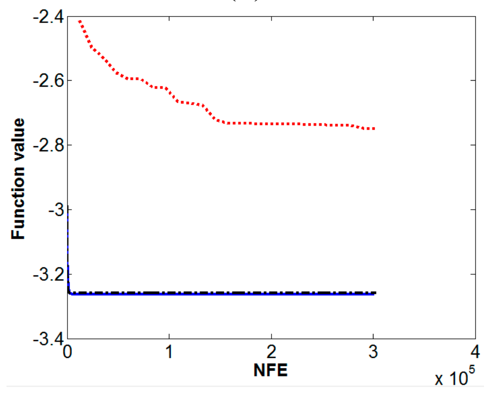
(**a**)



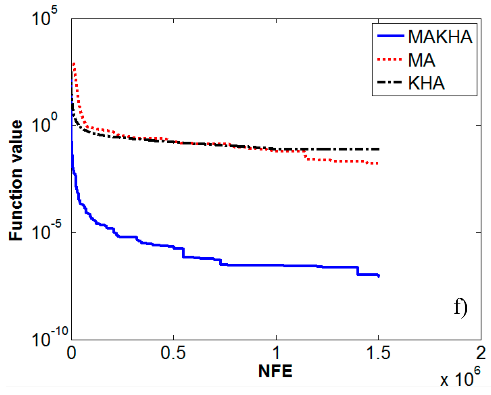
(**b**)



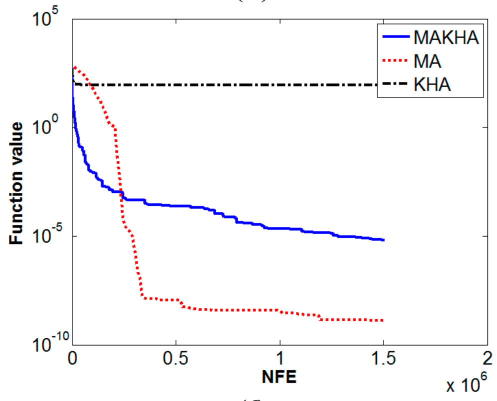
(**c**)



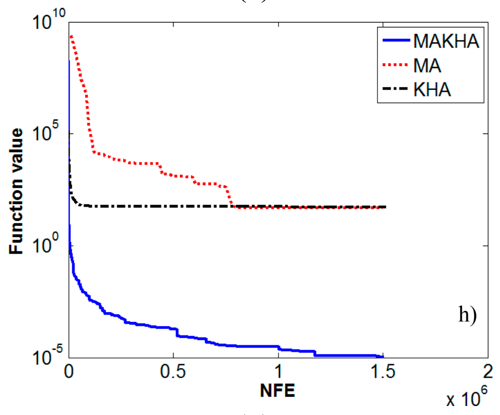
(**d**)



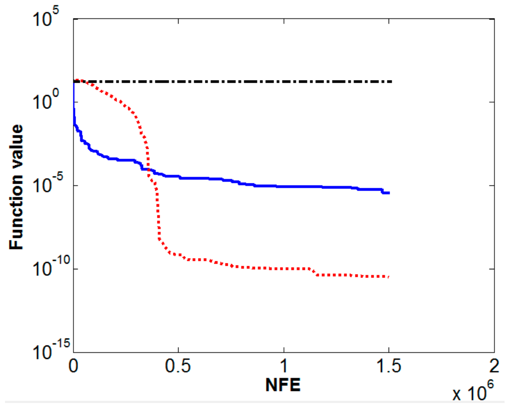
(**e**)



(**f**)



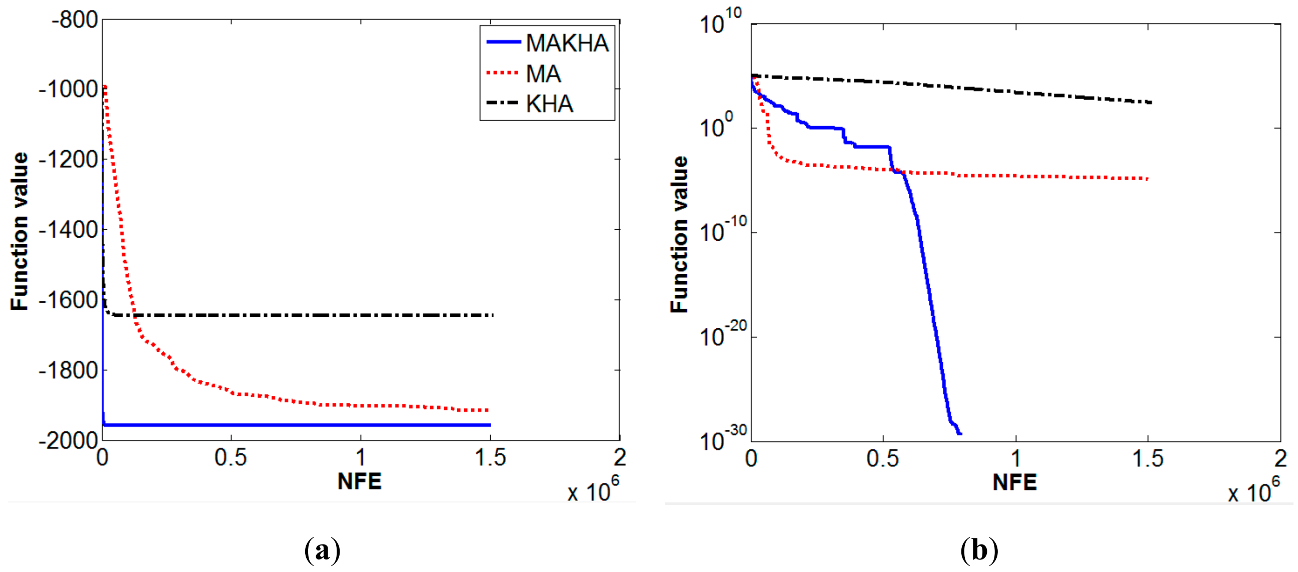
(**g**)

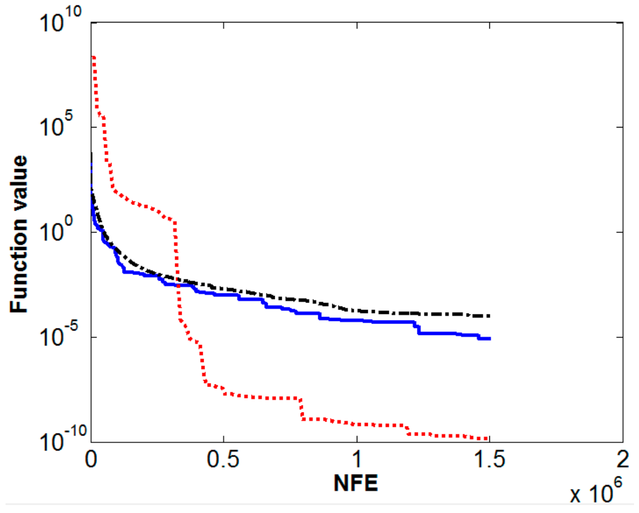


(**h**)

Figure 7. Evolution of mean best values for MA, KHA and MAKHA for: (**a**) Extended cube, (**b**) Shekel, (**c**) Sphere, (**d**) Hartman, (**e**) Griewank (**f**) Rastrigin, (**g**) Rosenbrock, and (**h**) Sine Envelope Sine functions.

Figure 8a–c shows the results of the mean best value obtained by the three algorithms for Styblinski–Tang, Trigonometric, and Zacharov functions, respectively. MAKHA was the only method to obtain the global minimum for Styblinski-Tang function and it did efficiently as measured by NFE. MA outperformed KHA for this particular function. The same pattern was obtained with the Trigonometric function as depicted in Figure 8b. The Zacharov function is the third function, which MAKHA did not outperform MA, as shown in Figure 8c.





(c)

Figure 8. Evolution of mean best values for MA, KHA and MAKHA for: (**a**) Styblinski-Tang, (**b**) Trigonometric and (**c**) Zacharov functions.

Table 4 shows a summary of the performance results for the twenty-seven benchmark problems. MAKHA has outperformed its parent algorithms in the majority of the benchmark problems studied. Best global values are shown in bold. The performance profiles reported in Figure 9 summarize the results of the MAKHA evaluation in comparison with the two original algorithms. MAKHA was the best algorithm in 24 out of the 27 cases considered. In several cases, the hybrid algorithm was the only algorithm that obtained the global minimum. Also, due to its efficient exploration and exploitation components, it converges to the global minimum with less NFE than the original two algorithms.

Table 4. Values of the mean minima (f_{calc}) and standard deviations (σ) obtained by the MAKHA, MA and KHA algorithms for the benchmark problems used in this study.*

	Numerical Performance of					
	MA		**KHA**		**MAKHA**	
Objective function	f_{calc}	σ	f_{calc}	σ	f_{calc}	σ
Ackley	4.8E−8	0	0.00129	0	**0**	0
Beale	0.084	0.125	0.0508	0.193	**0**	0
Bird	−105.326	1.45	−103.52	7.4	**−106.7645**	0
Booth	0.179	0.172	1.28E−12	0	**0**	0
Bukin 6	3.487	1.77	0.074	0.029	**0.0267**	0.0157
Carrom table	−23.9138	0.436	**−24.1568**	0	**−24.1568**	0
Cross-leg table	−0.002	4E−3	−0.00035	0	**−0.9985**	0
Generalized egg holder	−949.58	0	−862.1	0	**−959.641**	0
Goldstein-Price	3.051	0.055	**3**	0	**3**	0
Himmelblau	0.179	0.187	7.4E−13	0	**3.7E−31**	0
Levy 13	0.0616	0.08	2.1E−7	0	**1.35E−31**	0
Schaffer	**0**	0	1.7E−6	0	**0**	0
Zettl	−0.0037	1E−3	**−0.00379**	0	**−0.00379**	0
Helical valley	136	250	9.8E−5	3E−3	**0**	0
Powell	18.46	49	1.9E−5	0	**0**	0
Wood	113.6	256	0.698	1.6	**0**	0
Extended Cube	3.568	0.8	1.658	5.28	**0**	0
Shekel 5	−10.139	0.06	−5.384	3.1	**−10.1532**	0
Sphere	1.4E−14	0	1E−10	0	**0**	0
Hartman 6	−2.7499	0.2	−3.2587	0.06	**−3.2627**	0.06
Griewank	0.0165	0.032	0.0769	0.095	**8.4E−8**	0
Rastrigin	**1.3E-9**	0	89.38	48.38	6.47E−6	0
Rosenbrock	47.45	7.5	52.593	18.97	**1E−5**	0
Sine envelope sine wave	**3.3E−11**	0	16.107	5.94	3.46E−6	0
Styblinski-Tang	−1916.84	28.1	−1645.42	36.9	**−1958.31**	0
Trigonometric	1.35E−5	0	285.43	545.5	**0**	0
Zacharov	**1.46E−10**	0	1E−3	5.5E−3	8.4E−6	0

* Bold numbers represent the best global minimum value obtained for each function.

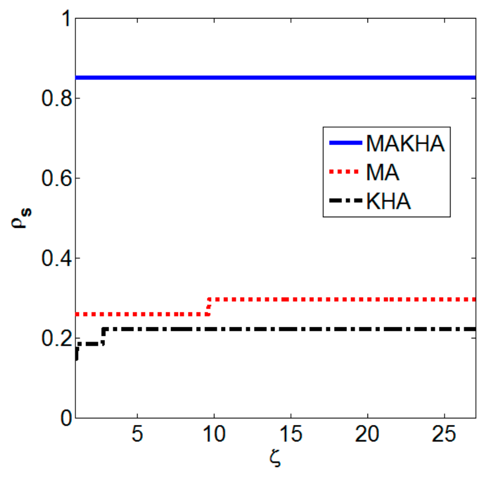


Figure 9. Performance profiles of the MAKHA, MA, and KHA methods for the global optimization of the twenty-seven benchmark problems used in this study.

The results of new proposed hybrid were promising that could be attributed to the combination of some of the mechanisms and processes of MA and KHA together to produce a reliable algorithm with appreciated performance. The procedures of both algorithms include exploration/diversification and exploitation/intensification features as follows:

(a) Exploration or diversification feature: The watch-jump process (MA), physical random diffusion (KHA), the somersault process (MA), and genetic operators (KHA).
(b) Exploitation or intensification feature: The climb process (MA), the watch-jump process (MA), the induced motion (KHA), and the foraging activity (KHA).

Both algorithms (MA and KHA) make a good compromise and balance between exploration/diversification and exploitation/intensification features. However, MAKHA uses the best setup of efficient operators, which are capable of producing the previous promising results.

7. Conclusions

In this paper, we propose a new hybrid algorithm, which is based on two bio-inspired swarm intelligence global stochastic optimization methods, the Monkey Algorithm and the Krill Herd Algorithm. The hybridization made use of the efficient components in each of the two original algorithms. It aimed to provide a better balance between exploration/diversification steps and exploitation/intensification steps to more efficiently and more reliably solve a wide range of problems in comparison with the its parent algorithms. This hybrid method was evaluated by attempting to find the global optimum of twenty-seven benchmark functions. The newly developed MAKHA algorithm led to improved reliability and effectiveness of the algorithm in the vast majority of the benchmark problems. In many cases, the global minimum could not be obtained via the original algorithms, but was easily obtained by the new method.

The authors are currently working on the improvement of MAKHA by decreasing the number of its parameters, and increasing its reliability and efficiency in solving difficult thermodynamic problems. The performance of MAKHA is compared to the performance of other algorithms thath have high reliability in solving these kinds of problems.

Supplementary Materials

Supplementary materials can be accessed at: http://www.mdpi.com/1999-4893/8/2/336/s1.

Acknowledgments

Authors acknowledge the support provided by the Cairo University (Egypt) and Instituto Tecnologico de Aguascalientes (Mexico).

Author Contributions

This research study was performed as a part of the Masters thesis of Ahmed M. E. Khalil at Cairo University. He suggested the hybrid algorithm to his supervisor, Seif-Eddeen K. Fateen, who contributed with ideas on implementation and evaluation of the algorithm. Adrian Bonilla-Petriciolet, a collaborator with Fateen on the use of stochastic global optimization, analyzed the data. Khalil and Fateen wrote the paper, while Bonilla-Petriciolet reviewed the manuscript and provided valuable comments to improve the paper.

Conflicts of Interest

The authors declare no conflict of interest.

Abbreviations

A	Hartman's recommended constants
a	Pseudo-gradient monkey step
b	Eyesight of the monkey (hybrid), which indicates the maximum distance the monkey (hybrid) can watch.
C	Shekel's recommended constants
C^{food}	Food coefficient
Cr	Crossover probability
C_t	Empirical and experimental Constants (Time constant)
c	Somersault interval
D_i	Physical diffusion of krill (hybrid) number i
D_{max}	Maximum diffusion speed
d	Somersault interval
ds_i	Sensing distance of the krill
ds_{ij}	Distance between each 2 krill positions
f	Objective function
F_i	Foraging motion
G	Global minimum
H	Fitness value of the hybrid in MAKHA
I, i, j and l	Counters for any value
K	Fitness value of the krill in KHA

M	Number of local minima in Shekel function
LB	Lower boundaries and low limit of decision variable
Mu	Mutation probability
m	Dimension of the problem, *i.e.*, number of variables.
N	Induced speed for KHA
N_c	Number of climb cycles
$N_{\max}$	Maximum induced speed
NP	Population size (number of points)
NV	Dimension of the problem, *i.e.*, number of variables.
n	A counter
n_p	Number of problems
n_s	Number of solvers
O	Hartman's recommended constants
P	Pivot value
R	The half range of boundaries between the lower boundary and the upper boundary of the decision variables (X)
r_{ps}	The performance ratio
T	Time taken by krill or hybrid
t_{ps}	Performance metric
UB	Upper boundaries and high limit of decision variable
V_f	Foraging speed
w_f or w_N	Inertia weight
X	Decision variable matrix
X^{food}	Centre of food density
x	Decision variable
Y	Decision variable matrix
Greek Letters	
α	Somersault interval random output.
β	Shekel's recommended constant
β^{food}	Food attractive factor
δ	Random direction vector
Δt	Incremental period of time
ε	Small positive number to avoid singularity
$\zeta\zeta$	The simulating value of r_{ps}
ζ_{max}	The maximum assumed value of rps
ρ	The cumulative probabilistic function of r_{ps} and the fraction of the total number of problems
σ	Standard deviation
$\varsigma\varsigma$	The counter of ρ points

References

1. Floudas, C.A.; Gounaris, C.E. A review of recent advances in global optimization. *J. Glob. Optim.* **2009**, *45*, 3–38.
2. Zhao, R.; Tang, W. Monkey algorithm for global numerical optimization. *J. Uncertain Syst.* **2008**, *2*, 165–176.
3. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845.
4. Ituarte-Villarreal, C.M.; Lopez, N.; Espiritu, J.F. Using the Monkey Algorithm for Hybrid Power Systems Optimization. *Procedia Comput. Sci.* **2012**, *12*, 344–349.
5. Aghababaei, M.; Farsangi, M.M. Coordinated Control of Low Frequency Oscillations Using Improved Monkey Algorithm. *Int. J. Tech. Phys. Probl. Eng.* **2012**, *4*, 13–17.
6. Yi, T.-H.; Li, H.-N.; Zhang, X.-D. A modified monkey algorithm for optimal sensor placement in structural health monitoring. *Smart Mater. Struct.* **2012**, *21*, doi:10.1088/0964-1726/21/10/105033.
7. Yi, T.-H.; Li, H.-N.; Zhang, X.-D. Sensor placement on Canton Tower for health monitoring using asynchronous-climb monkey algorithm. *Smart Mater. Struct.* **2012**, *21*, doi:10.1088/0964-1726/21/12/125023.
8. Yi, T.H.; Zhang, X.D.; Li, H.N. Modified monkey algorithm and its application to the optimal sensor placement. *Appl. Mech. Mater.* **2012**, *178*, 2699–2702.
9. Sur, C.; Shukla, A. Discrete Krill Herd Algorithm—A Bio-Inspired Meta-Heuristics for Graph Based Network Route Optimization. In *Distributed Computing and Internet Technology*; Springer: New York, NY, USA, 2014; pp. 152–163.
10. Mandal, B.; Roy, P.K.; Mandal, S. Economic load dispatch using krill herd algorithm. *Int. J. Electr. Power Energy Syst.* **2014**, *57*, 1–10.
11. Zheng, L. An improved monkey algorithm with dynamic adaptation. *Appl. Math. Comput.* **2013**, *222*, 645–657.
12. Saremi, S.; Mirjalili, S.M.; Mirjalili, S. Chaotic krill herd optimization algorithm. *Procedia Technol.* **2014**, *12*, 180–185.
13. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. A chaotic particle-swarm krill herd algorithm for global numerical optimization. *Kybernetes* **2013**, *42*, 962–978.
14. Gharavian, L.; Yaghoobi, M.; Keshavarzian, P. Combination of krill herd algorithm with chaos theory in global optimization problems. In Proceedings of the 2013 3rd Joint Conference of AI & Robotics and 5th RoboCup Iran Open International Symposium (RIOS), Tehran, Iran, 8 April 2013; IEEE: Piscataway, NJ, USA, 2013.
15. Wang, J.; Yu, Y.; Zeng, Y.; Luan, W. Discrete monkey algorithm and its application in transmission network expansion planning. In Proceedings of the Power and Energy Society General Meeting, Minneapolis, MN, USA, 25–29 July 2010; IEEE: Piscataway, NJ, USA, 2010.
16. Wang, G.; Guo, L.; Gandomi, A.H.; Cao, L. Lévy-flight krill herd algorithm. *Math. Probl. Eng.* **2013**, doi:10.1155/2013/682073.
17. Guo, L.; Wang, G.-G.; Gandomi, A.H.; Alavi, A.H.; Duan, H. A new improved krill herd algorithm for global numerical optimization. *Neurocomputing* **2014**, *138*, 392–402.

18. Wang, G.; Guo, L.; Wang, H.; Duan, H.; Liu, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2014**, *24*, 853–871.
19. Wang, G.-G.; Guo, L.H.; Gandomi, A.H.; Alavi, A.H.; Duan, H. Simulated annealing-based krill herd algorithm for global optimization. In *Abstract and Applied Analysis*; Hindawi Publishing Corporation: Cairo, Egypt, 2013.
20. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* **2003**, *35*, 268–308.
21. Lozano, M.; García-Martínez, C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Comput. Op. Res.* **2010**, *37*, 481–497.
22. Liu, P.-F.; Xu, P.; Han, S.-X.; Zheng, J.-Y. Optimal design of pressure vessel using an improved genetic algorithm. *J. Zhejiang Univ. Sci. A* **2008**, *9*, 1264–1269.
23. Abdullah, A.; Deris, S.; Mohamad, M.S.; Hashim, S.Z.M. A New Hybrid Firefly Algorithm for Complex and Nonlinear Problem. In *Distributed Computing and Artificial Intelligence*; Omatu, S.; Bersini, H., Corchado, J.M., Rodríguez, S., Pawlewski, P., Bucciarelli, E., Eds.; Springer: Berlin, Germany; Heidelberg, Germany, 2012; pp. 673–680.
24. Biswas, A.; Dasgupta, S.; Das, S.; Abraham, A. *Synergy of PSO and Bacterial Foraging Optimization—A Comparative Study on Numerical Benchmarks Innovations in Hybrid Intelligent Systems*; Corchado, E., Corchado, J., Abraham, A., Eds.; Springer: Berlin, Germany; Heidelberg, Germany, 2007; pp. 255–263.
25. Li, S.; Chen, H.; Tang, Z. Study of Pseudo-Parallel Genetic Algorithm with Ant Colony Optimization to Solve the TSP. *Int. J. Comput. Sci. Netw. Secur.* **2011**, *11*, 73–79.
26. Nguyen, K.; Nguyen, P.; Tran, N. A hybrid algorithm of Harmony Search and Bees Algorithm for a University Course Timetabling Problem. *Int. J. Comput. Sci. Issues* **2012**, *9*, 12–17.
27. Farahani, Sh.M.; Abshouri, A.A.; Nasiri, B.; Meybodi, M.R. Some hybrid models to improve Firefly algorithm performance. *Int. J. Artif. Intell.* **2012**, *8*, 97–117.
28. Kim, D.H.; Abraham, A.; Cho, J.H. A hybrid genetic algorithm and bacterial foraging approach for global optimization. *Inf. Sci.* **2007**, *177*, 3918–3937.
29. Kim, D.H.; Cho, J.H. A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems. *Int. J. Control Autom. Syst.* **2006**, *4*, 624–636.
30. Dehbari, S.; Rosta, A.P.; Nezhad, S.E.; Tavakkoli-Moghaddam, R. A new supply chain management method with one-way time window: A hybrid PSO-SA approach. *Int. J. Ind. Eng. Comput.* **2012**, *3*, 241–252.
31. Zahrani, M.S.; Loomes, M.J.; Malcolm, J.A.; Dayem Ullah, A.Z.M.; Steinhöfel, K.; Albrecht, A.A. Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing. *Comput. Op. Res.* **2008**, *35*, 2049–2070.
32. Huang, K.-L.; Liao, C.-J. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 1030–1046.
33. Shahrouzi, M. A new hybrid genetic and swarm optimization for earthquake accelerogram scaling. *Int. J. Optim. Civ. Eng.* **2011**, *1*, 127–140.

34. Bäck, T.; Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolut. Comput.* **1993**, *1*, 1–23.
35. Jamil, M.; Yang, X.-S. A literature survey of benchmark functions for global optimization problems. *Int. J. Math. Model. Numer. Optim.* **2013**, *4*, 150–194.
36. Mishra, S.K. Global optimization by differential evolution and particle swarm methods: Evaluation on some benchmark functions, Social Science Research Network, Rochester, NY, USA. Available online: http://ssrn.com/abstract=933827 (accessed on 3 April 2015).
37. Mishra, S.K. Some new test functions for global optimization and performance of repulsive particle swarm method, Social Science Research Network, Rochester, NY, USA. Available online: http://ssrn.com/abstract=926132 (accessed on 3 April 2015).
38. Goldstein, A.; Price, J. On descent from local minima. *Math. Comput.* **1971**, *25*, 569–574.
39. Himmelblau, D.M. *Applied Nonlinear Programming*; McGraw-Hill Companies: New York, NY, USA, 1972.
40. Ortiz, G.A. *Evolution Strategies (ES)*; Mathworks: Natick, MA, USA, 2012.
41. Schwefel, H.-P.P. *Evolution and Optimum Seeking: The Sixth Generation*; John Wiley & Sons: Hoboken, NJ, USA, 1993.
42. Fletcher, R.; Powell, M.J. A rapidly convergent descent method for minimization. *Comput. J.* **1963**, *6*, 163–168.
43. Fu, M.C.; Hu, J.; Marcus, S.I. Model-based randomized methods for global optimization. In Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, 24–28 July 2006.
44. Grippo, L.; Lampariello, F.; Lucidi, S. A truncated Newton method with nonmonotone line search for unconstrained optimization. *J. Optim. Theory Appl.* **1989**, *60*, 401–419.
45. Oldenhuis, R.P.S. *Extended Cube Function*; Mathworks: Natick, MA, USA, 2009.
46. Pintér, J. *Global Optimization in Action: Continuous and Lipschitz optimization: Algorithms, Implementations and Applications*; Springer Science & Business Media: Berlin, Germany; Heidelberg, Germany, 1995; Volume 6.
47. Schumer, M.; Steiglitz, K. Adaptive step size random search. *Autom. Control IEEE Trans.* **1968**, *13*, 270–276.
48. Hartman, J.K. Some experiments in global optimization. *Nav. Res. Logist. Q.* **1973**, *20*, 569–576.
49. Griewank, A.O. Generalized descent for global optimization. *J. Optim. Theory Appl.* **1981**, *34*, 11–39.
50. Rastrigin, L. *Systems of Extremal Control*; Nauka: Moscow, Russia, 1974.
51. Rosenbrock, H.H. An automatic method for finding the greatest or least value of a function. *Comput. J.* **1960**, *3*, 175–184.
52. Silagadze, Z. Finding two-dimensional peaks. *Phys. Part. Nucl. Lett.* **2007**, *4*, 73–80.
53. Dixon, L.C.W.; Szegö, G.P. (Eds.) *Towards Global Optimisation 2*; North-Holland Publishing: Amsterdam, The Netherlands, 1978.

54. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. A novel population initialization method for accelerating evolutionary algorithms. *Comput. Math. Appl.* **2007**, *53*, 1605–1614.
55. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213.