symmetry

MDPI

Article

An Improved Butterfly Optimization Algorithm for Engineering Design Problems Using the Cross-Entropy Method

Guocheng Li [1,*], Fei Shuang [2], Pan Zhao [1] and Chengyi Le [3,*]

[1] School of Finance and Mathematics, West Anhui University, Lu'an 237012, China
[2] Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA
[3] School of Economics and Management, East China Jiaotong University, Nanchang 330013, China
* Correspondence: liguocheng@wxc.edu.cn (G.L.); ncycy@126.com (C.L.);
Tel.: +86-564-330-5031 (G.L.); +86-791-8704-5037 (C.L.)

Received: 4 July 2019; Accepted: 9 August 2019; Published: 14 August 2019

Abstract: Engineering design optimization in real life is a challenging global optimization problem, and many meta-heuristic algorithms have been proposed to obtain the global best solutions. An excellent meta-heuristic algorithm has two symmetric search capabilities: local search and global search. In this paper, an improved Butterfly Optimization Algorithm (BOA) is developed by embedding the cross-entropy (CE) method into the original BOA. Based on a co-evolution technique, this new method achieves a proper balance between exploration and exploitation to enhance its global search capability, and effectively avoid it falling into a local optimum. The performance of the proposed approach was evaluated on 19 well-known benchmark test functions and three classical engineering design problems. The results of the test functions show that the proposed algorithm can provide very competitive results in terms of improved exploration, local optima avoidance, exploitation, and convergence rate. The results of the engineering problems prove that the new approach is applicable to challenging problems with constrained and unknown search spaces.

Keywords: global optimization; meta-heuristic; butterfly optimization algorithm; cross-entropy method; engineering design problems

1. Introduction

Real-world engineering design optimization problems are very challenging to find the global optimum of a highly complex and multiextremal objective function, involving many different decision variables under complex constraints [1,2]. A general engineering design optimization problem can be stated as follows:

$$\min f(x), x = (x_1, x_2, \cdots, x_n)^T \in R^n, \tag{1}$$

subject to

$$g_j(x) \leq 0, j = 1, 2, \cdots, p, \tag{2}$$

$$h_k(x) = 0, k = 1, 2, \cdots, q, \tag{3}$$

$$lb_i \leq x_i \leq ub_i, i = 1, 2, \cdots, n, \tag{4}$$

where n is the number of search space dimensions, $g_j(x)$ and $h_k(x)$are the jth inequality constraint and kth equality constraint, respectively. lb_i and ub_i represent the lower and upper bound of the value of x_i.

Most of the constraints of the global optimization problem are nonlinear. Such nonlinearity often results in a multimodal response landscape [3]. Due to their high complexity, nonlinearity,

and multimodality, traditional optimization methods, such as gradient-based optimization algorithms, are no longer suitable for this problem [4]. Many researchers have developed various derivative-free global optimization methods for it. In general, these methods can be divided into two classes: deterministic methods and stochastic meta-heuristic algorithms [4,5]. The former, such as the Hill-Climbing [6], Newton–Raphson [7], and DIRECT algorithm [8], can obtain the same optimal results based on the same set of initial values. However, this behavior results in local optima entrapment and a loss of reliability in finding the global optimum, since most real engineering design problems have extremely high numbers of local solutions [9]. The latter, such as the Genetic Algorithm (GA) [10,11], Particle Swarm Optimization (PSO) [12], Cuckoo Search (CS) [13], Bat Algorithm (BA) [14], Grey Wolf Optimizer (GWO) [15], Forest Optimization Algorithm (FOA) [16,17], Ant Lion Optimizer (ALO) [9], Whale Optimization Algorithm (WOA) [18], Crow Search Algorithm (CSA) [19], Salp Swarm Algorithm (SSA) [20], and Butterfly Optimization Algorithm (BOA) [21,22], mostly benefit from their simplicity, flexibility, and stochastic operators, which makes them different from deterministic methods [4], and they have become very popular optimization techniques for such optimization problems in recent years.

Despite the merits of nature-inspired meta-heuristic algorithms, it has been pointed out that most of them are unable to guarantee global convergence [23]. At the same time, the No Free Lunch (NFL) theorem has proved that none of these methods are able to solve all global optimization problems [24]. They perform very well when dealing with certain optimization problems, but fail in most cases. An excellent meta-heuristic algorithm has two symmetric search capabilities: local search and global search, and achieves a proper balance between exploration and exploitation to effectively avoid it falling into a local, and find the global optimum. In order to enhance global search capabilities, many hybrid meta-heuristic algorithms have been developed by combining meta-heuristics with exact algorithms or other meta-heuristics for solving more complicated optimization problems [25–29].

Inspired by the idea of global random search algorithms [5,30], and based on co-evolution, this paper explores an improved butterfly optimization algorithm (BOA) using the cross-entropy method, which is a global stochastic optimization method based on a statistical model. The original BOA was proposed by Arora and Singh [21] in 2015, and was inspired by the food foraging behavior of butterflies, while the cross-entropy (CE) method was developed by Rubinstein [31] in 1997 to estimate the probability of rare events in complex random networks. Since the BOA has a tendency to prematurely converge to local optima [32], this paper embeds the CE method into the BOA to obtain a good balance between exploration and exploitation and improve the BOA's global search capability.

The rest of this paper is structured as follows. In Section 2, the BOA and CE method are briefly introduced, and a study of the improved BOA is presented in Section 3. The results and a discussion of benchmark functions are provided in Section 4. Section 5 solves three classical engineering design problems. In Section 6, conclusions and future research directions are presented.

2. Methods

2.1. Butterfly Optimization Algorithm

The BOA is a new nature-inspired meta-heuristic algorithm which was inspired by the food foraging behavior of butterflies [21,22]. In this method, the characteristics of butterflies are ideally summarized as follows:

1. All butterflies are assumed to emit some fragrance that makes the butterflies attractive to each other;

2. Each butterfly moves randomly or toward the best butterfly—i.e., the one which emits the most fragrance;

3. The stimulus intensity of a butterfly is determined by the landscape of the objective function.

In the BOA, the perceived magnitude of the fragrance (f) is defined as a function of the stimulus physical intensity as follows:

$$f = cI^a, \tag{5}$$

where $c \in [0, \infty]$ is the sensory modality; I is the stimulus intensity, which is associated with the encoded objective function; and $a \in [0, 1]$ is the power exponent dependent on modality, which represents the varying degree of fragrance absorption.

There are two key steps in the BOA: global search and local search. The former can make the butterflies move toward the best butterfly, which can be represented as

$$x_i^{t+1} = x_i^t + (r^2 \times g^* - x_i^t) \times f_i, \tag{6}$$

where x_i^t is the position of the ith butterfly at time t. Here, g^* represents the current best position. f_i represents the fragrance of the ith butterfly, and r is a random number in $[0, 1]$.

The latter is implemented through a local random walk, which can be represented as

$$x_i^{t+1} = x_i^t + (r^2 \times x_j^t - x_k^t) \times f_i, \tag{7}$$

where x_j^t and x_k^t are the positions of the jth and kth butterflies in the solution space, respectively, and r is a random number in $[0, 1]$.

Additionally, a switch probability p is used in the BOA to switch between a common global search and an intensive local search. Based on the above, the original BOA can be implemented with pseudo-code as shown in Algorithm 1:

Algorithm 1: Butterfly Optimization Algorithm

Begin
Objective function $f(x)$, $x = (x_1, x_2, \ldots, x_d)^T$, here d represents the number of dimensions.
Generate initial population P containing n butterflies $pop_i (i = 1, 2, \ldots, n)$.
Stimulus I_i intensity at is pop_i determined by the fitness value $f(pop_i)$.
Define sensor modality c, power exponent a, and switch probability p.
while stop criteria not met **do**
  for each butterfly in the population P **do**
    Calculate fragrance f using Equation (5).
  end for
  Evaluate and rank the population P, and find the best butterfly.
  for each butterfly in the population P **do**
    Generate a random number $r \sim U[0, 1]$.
    if $(r < p)$
      Implement global search using Equation (6).
    else
      Implement local search using Equation (7).
    end if
  end for
  Update the value of the power exponent a.
end while
Output the best solution and optimal value.
End

BOA shows better performance compared to some other optimization algorithms [21,22] and has attracted the attention of many researchers due to its simplicity and interesting nature-inspired interpretations as an optimization approach for global optimization problems and for various real-life applications [32,33].

2.2. The Cross-Entropy (CE) Method

Based on Monte Carlo technology, Rubinstein [31] developed the CE method in 1997 and uses the cross-entropy or Kullback–Leibler divergence to measure the distance between two sampling distributions, solve an optimization problem by minimizing this distance, and obtain the optimal parameters of probability distribution. The CE method has good global search capability, excellent adaptability, and strong robustness. It is frequently used for many complex optimization problems such as continuous multi-extremal optimization [34], multi-objective optimization [35], combination optimization [36,37], vehicle routing problems [38], energy-efficient radio resource management [39], and complex optimization problems from other fields [40–43].

A general optimization problem can be stated as follows:

$$\min_{x \in \mathcal{X}} S(x), \tag{8}$$

where $\mathcal{X}$ is a finite set of states, and S is a real-valued performance function on $\mathcal{X}$.

Next, we can construct a probability distribution estimation problem associated with the above problem, and the auxiliary problem can be defined as follows:

$$l(\gamma) = P_u(S(X) \leq \gamma) = E_u[I_{\{S(X) \leq \gamma\}}], \tag{9}$$

where P_u is the probability measure under which the random state X has some probability density function $f(x;u)$ on $\mathcal{X}$; E_u is the corresponding expectation operator; γ represents a threshold parameter; and I represents the indicator function, whose value is 1, if $S(X) \leq \gamma$, and 0, otherwise. The importance sampling approach is used to reduce the sample size in the CE method. Consequently, Equation (9) can be rewritten as follows:

$$l(\gamma) = \frac{1}{N} \sum_{i=1}^{N} I_{S(X) \leq \gamma} \frac{f(x^i;v)}{g(x^i)}, \tag{10}$$

where x^i represents a random sample from $f(x;v)$ with importance sampling density $g(x)$. the Kullback–Leibler divergence, i.e., the cross-entropy is introduced to measure the distance between two sampling distributions for obtaining the optimal importance sampling density. Thus, the optimal density $g^*(x)$ can be found by minimizing the Kullback–Leibler divergence, which is equivalent to solving the following optimization problem [34]:

$$\min_{v} \frac{1}{N} \sum_{i=1}^{N} I_{S(X) \leq \gamma} \ln f(x^i;v). \tag{11}$$

In order to improve the convergence speed of the CE method, adaptive smoothing $\hat{v}_k$ is demoted by $\widetilde{v}$, as follows:

$$\hat{v}_{k+1} = \alpha \widetilde{v} + (1-\alpha)\hat{v}_k, \tag{12}$$

where $0 \leq \alpha \leq 1$ is a smoothing parameter.

The CE method for optimization problems is described in Algorithm 2.

Algorithm 2: Cross-entropy (CE) for optimization problems

Begin
Set $t = 0$. Initialize the probability parameter $\hat{v}_k$.
while stop criteria for CE not met **do**
Generate $S_1, S_2, \ldots, S_N \sim_{iid} f(x; \hat{v}_k)$. Evaluate and rank the sample S.
Solve the problem given in Equation (11) based on the sample S. Denote the solution by $\widetilde{v}$.
Update the parameter $\hat{v}_k$ using $\widetilde{v}$.
Set $t = t + 1$.
end while
Output the best solution and optimal value.
End

3. Hybrid BOA-CE Method

This section presents the details of the improved algorithm, named BOA-CE. A meta-heuristic method should have two main functions—exploration and exploitation—and should try to find a proper balance between them for better performance [44]. The nature-inspired BOA has shown the advantages of excellent local search capability and fast convergence; however, it tends to fall into a local optimum rather than finding the global optimum [21,22,32,33]. In order to improve the global search capability of the BOA, and, based on a co-evolutionary technique, this paper proposes the BOA-CE algorithm by embedding the CE method into it. The new method contains BOA and CE optimization operators, and co-updates the BOA population P and the CE sample S using co-evolutionary technology in each iteration. The CE operator obtains the initial probability parameters using the BOA population P in order to improve its convergence rate while the BOA operator updates its population P and the best butterfly using the elite sample S_e of the CE operator to enhance the population diversity. The pseudo-code of the BOA-CE hybrid algorithm is described in Algorithm 3. Figure 1 presents the flow chart of the BOA-CE algorithm, and the co-evolutionary process between BOA operator and CE operator can be clearly seen from Figure 1.

In addition, as an example, we use the BOA-CE algorithm to find the global optimum of the 2D Sphere function in order to more clearly describe the co-evolutionary process of the BOA operator and the CE operator:

$$f(x) = {x_1}^2 + {x_2}^2, \quad -100 \le x_1, x_2 \le 100, \tag{13}$$

which has a global minimum $f^* = 0$ at $x^* = (0,0)^T$. Now, let us use the BOA-CE algorithm to find the optima and this process is summarized as follows: (1) the values of the parameters of the BOA operator are $c = 0.01$, $a = 0.1$, $p = 0.8$, the population size $n = 40$, and the maximum number of iteration $t_1 = 50$, while the values of the parameters of the CE operator are $\alpha = 0.8$, the sample size $N = 80$, the elite sample size $Ne = 30$, and the maximum number of iteration $t_2 = 2$; (2) initializing the population of the BOA operator P, calculating $f_i(x) = x_{i1}^2 + x_{i2}^2, i = 1, 2, \cdots, 40$, and finding the current best $f^* = 13.5036$ and $x^* = (-3.5586, -0.9164)^T$; (3) generating a rand number r, and updating x_i with $x_i^{new} = x_i^{old} + (r^2 \times f^* - x_i^{old}) \times f_i$ if $r \le p$ and $x_i^{new} = x_i^{old} + (r^2 \times x_j^{old} - x_k^{old}) \times f_i$ otherwise for each butterfly in the P; (4) calculating the mean $\mu = (-4.9571, -11.4528)$ and standard $\sigma = (60.2592, 53.9977)$ deviation of P, and using them to initialize the probability parameter $v = (-4.9571, -11.4528, 60.2592, 53.9977)$; (5) generating a sample S by the probability parameter v, and calculating $f_i(x) = x_{i1}^2 + x_{i2}^2, i = 1, 2, \cdots, 80$, and finding the new current best $f^* = 4.4290$ and $x^* = (1.8693, 0.9668)^T$; (6) updating the probability parameter v using the elite sample, and regenerating a sample S using the new parameter v, and finding the new current best $f^* = 2.4542$ and $x^* = (0.8951, 1.2857)^T$; (7) co-updating the population P using the sample S and the population P; (8) repeating the above operations until the termination condition ($t_1 > 50$) is

met, and outputting the approximate optimal solution $x^* = (1.1154 \times 10^{-31}, -2.5727 \times 10^{-32})^T$ and function value $f^* = 1.3104 \times 10^{-62}$. Figure 2 clearly describes the of co-updating the current best f^* by the BOA operator and CE operator.

Algorithm 3: BOA-CE algorithm

Begin
Objective function $f(x)$, $x = (x_1, x_2, \ldots, x_d)^T$; here, d represents the number of dimensions.
Generate initial population P containing n butterflies $pop_i (i = 1, 2, \ldots, n)$.
Stimulus I_i intensity at is pop_i determined by the fitness value $f(pop_i)$.
Define sensor modality c, power exponent a, and switch probability p.
while stop criteria not met **do**
for each butterfly in the population P **do**
Calculate fragrance f using Equation (5).
end for
Evaluate,rank P, and find the best butterfly.
for each butterfly in the population P **do**
Generate a random number $r \sim U[0, 1]$.
if $(r < p)$
Implement global search using Equation (6).
else
Implement local search using Equation (7).
end if
end for
Update the value of the power exponent a.
Evaluate and rank the population P.
Initialize the probability parameter $\hat{v}_k$ using the population P.
while stop criteria for CE not met **do**
Generate $S_1, S_2, \ldots, S_N \sim_{iid} f(x; \hat{v}_k)$, evaluate the sample S.
Rank the population P and the sample S together, co-update P and S, update the best butterfly.
Calculate the probability parameter $\widetilde{v}$ by the elite sample S_e.
Update the probability parameters $\hat{v}_k$ via Equation (12).
end while
end while
Output the best solution and optimal value.
End

The BOA-CE hybrid algorithm has two main operators: BOA and CE. The BOA operator has two inner loops for the butterflies population size n_1, and one outer loop for iteration t_1. While the CE operator has one inner loops for the sample size n_2, and two outer loops for iterations t_1 and t_2, respectively. Therefore, for our proposed hybrid algorithm (BOA-CE), the time complexity is $O(2n_1t_1 + n_2t_1t_2)$. It is linear in terms of t_1t_2, which represents the total number of iterations in the BOA-CE.

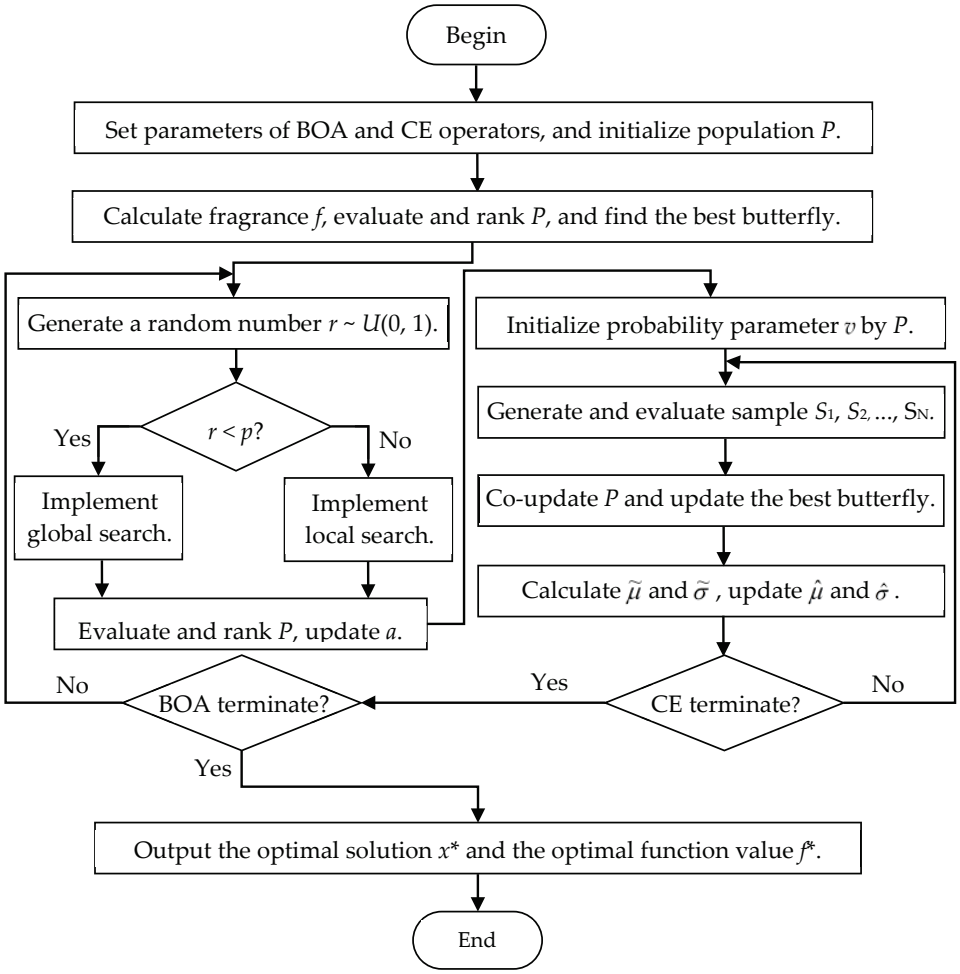

Figure 1. The flow chart of the BOA-CE algorithm.

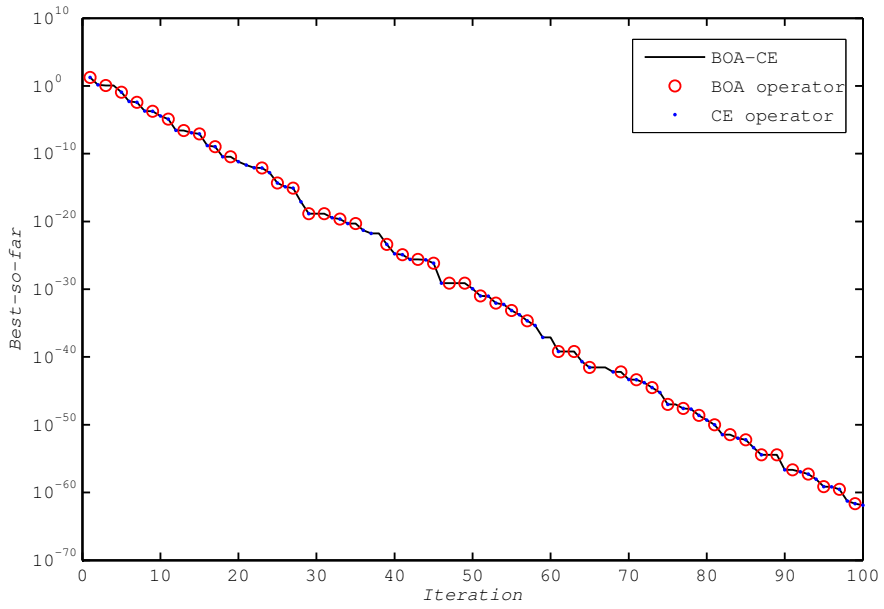

Figure 2. The BOA operator and CE operator co-update the current best f^* of the 2D Sphere function.

4. Experiment and Results

In this section, a total of 19 test functions [45,46] with different characteristics were employed to benchmark the performance of the proposed BOA-CE algorithm from different perspectives [9,15,16,20,28]. The test functions can be divided into three classes: unimodal (Table 1), multimodal (Table 2), and composite functions (Table 3). Generally, unimodal functions are employed to benchmark the exploitation and convergence of a method, while multimodal functions are selected to evaluate the performance of exploration and local optima avoidance [9,20]. In contrast, composite functions can be used to evaluate the combined performance of exploration and exploitation.

Table 1. Unimodal benchmark functions.

Function	Dim	Range	F_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0

Table 2. Multi-modal benchmark functions.

Function	Dim	Range	F_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	−12,569.487
$F_9(x) = \sum_{i=1}^{n-1} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1}) + (y_n + 1)^2]\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

To verify the results, the proposed BOA-CE algorithm was compared against a number of well-known and recent algorithms: Genetic Algorithm (GA) [10], PSO [12], BA [14], GWO [15], CSA [19], SSA [20], and BOA [31]. The variants were coded in the MATLAB R2018b, running on a PC with an Intel Core i7-8700 processor (Gainesville, FL, USA), a 3.19 GHz CPU, and 16 GB of RAM. To provide a fair comparison, the following test experimental conditions and settings were used: (1) the population size of the BOA operator in the BOA-CE algorithm was set to 40, while the CE operator's sample size was 80. The maximum number of iterations of the BOA operator and CE operator were 500 and 2, respectively; (2) the other method's population sizes were 100 for fair comparison, and their maximum number of iterations was 1000; (3) all of the other parameters in each algorithm for comparison were as the same as those used in the original references [10,12,14,15,19,20,31]. Each of the algorithms was run 30 times on each of the test functions, and the results are shown in Tables 4–6. The average value and standard deviation of the best solution obtained by each method is given to compare their overall performance. The winner (best value) is highlighted in bold.

Table 3. Composite benchmark functions.

Function [1]	**Dim**	**Range**	F_{min}
F_{14}(CF1) $f_1, f_2, \ldots, f_{10}$ = Sphere Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1], [\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1/100, 5/100, \ldots, 5/100]$	10	$[-5, 5]$	0
F_{15}(CF2) $f_1, f_2, \ldots, f_{10}$ = Griewank's Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1], [\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1/100, 5/100, \ldots, 5/100]$	10	$[-5, 5]$	0
F_{16}(CF3) $f_1, f_2, \ldots, f_{10}$ = Griewank's Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1], [\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [1, 1, \ldots, 1]$	10	$[-5, 5]$	0
F_{17}(CF4) f_1, f_2 = Ackley's Function, f_3, f_4 = Rastrigin's Function, f_5, f_6 = Weierstrass Function, f_7, f_8 = Griewank's Function, f_9, f_{10} = Sphere Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1]$, $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [\frac{5}{32}, \frac{5}{32}, 1, 1, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}]$	10	$[-5, 5]$	0
F_{18}(CF5) f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley'sGriewank's Function, f_9, f_{10} = Sphere Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1]$, $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [\frac{1}{5}, \frac{1}{5}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}]$	10	$[-5, 5]$	0
F_{19}(CF6) f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley'sGriewank's Function, f_9, f_{10} = Sphere Function, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [1, 1, \ldots, 1]$, $[\lambda_1, \lambda_2, \ldots, \lambda_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$, $[\sigma_1, \sigma_2, \ldots, \sigma_{10}] = [0.1 \times \frac{1}{5}, 0.2 \times \frac{1}{5}, 0.3 \times 0.4 \times \frac{5}{0.5}, 0.5 \times \frac{5}{0.5},$ $0.6 \times \frac{5}{100}, 0.7 \times \frac{5}{100}, 0.8 \times \frac{5}{32}, 0.9 \times \frac{5}{32}, 1 \times \frac{5}{100}, \frac{5}{100}]$	10	$[-5, 5]$	0

[1] Mathematical formulation of Sphere, Ackley, Rastrigin, Weierstrass, and Griewank can be found in Appendix A A1.

4.1. Results of the Algorithms Using Unimodal Test Functions

The results of the application of the algorithms to unimodal test functions are shown in Table 4. The best results are highlighted in bold. From Table 4, it can be found that: (1) the proposed method outperforms GA, PSO, BA, CSA, SSA, and BOA for the majority of the test functions; and (2) BOA-CE and GWO each provide the best results for three of these problems. This is due to the co-evolutionary technology that are adopted between the BOA and CE operators to enhance exploitation.

The progress of the average best value over 30 runs for some of the benchmark functions, such as F1, F2, F6, and F7, is illustrated in Figure 3. The figure clearly shows that the proposed BOA-CE algorithm rapidly converges towards the optimum and exploits it accurately. This benefit is mainly due to the algorithm's excellent exploitation.

4.2. Results of the Algorithms Using Multimodal Test Functions

The results of the application of the algorithms to multimodal test functions are shown in Table 5. Table 5 shows that the BOA-CE method outperforms the other approaches in the majority of the test cases. Multimodal test functions are employed to benchmark the performance in terms of exploration and local optima avoidance. Therefore, these results prove that the BOA-CE algorithm has an excellent exploration which helps it to explore the promising regions of the search space. Additionally, the local optima avoidance of the improved method is also excellent, since it is able to avoid all of the local optima and find the global optimum for the majority of the multimodal test functions.

Table 4. Results of unimodal benchmark functions.

Fun.	Metr.	GA	PSO	BA	GWO	CSA	SSA	BOA	BOA-CE
F_1	Mean	1.44×10^{-07}	1.43×10^{-15}	1.46×10^{-04}	1.97×10^{-90}	1.20×10^{-03}	6.70×10^{-09}	1.57×10^{-14}	$\mathbf{1.26 \times 10^{-95}}$
	SD	4.32×10^{-07}	3.42×10^{-15}	7.98×10^{-04}	5.36×10^{-90}	5.64×10^{-04}	1.23×10^{-09}	7.69×10^{-16}	$\mathbf{9.78 \times 10^{-96}}$
F_2	Mean	1.48×10^{-02}	7.50×10^{-07}	$2.57 \times 10^{+01}$	$\mathbf{2.92 \times 10^{-52}}$	7.16×10^{-01}	6.00×10^{-06}	9.14×10^{-12}	3.60×10^{-47}
	SD	6.05×10^{-02}	3.30×10^{-06}	$3.93 \times 10^{+01}$	$\mathbf{3.35 \times 10^{-52}}$	4.51×10^{-01}	1.33×10^{-06}	1.37×10^{-12}	2.50×10^{-47}
F_3	Mean	5.20×10^{-01}	$2.30 \times 10^{+00}$	$2.42 \times 10^{+03}$	$\mathbf{1.74 \times 10^{-26}}$	$8.30 \times 10^{+00}$	8.33×10^{-10}	1.60×10^{-14}	2.44×10^{-09}
	SD	1.95×10^{-01}	$1.18 \times 10^{+00}$	$2.18 \times 10^{+03}$	$\mathbf{5.09 \times 10^{-26}}$	$4.74 \times 10^{+00}$	3.05×10^{-10}	9.73×10^{-16}	1.83×10^{-10}
F_4	Mean	1.88×10^{-01}	2.01×10^{-01}	$3.56 \times 10^{+01}$	$\mathbf{1.74 \times 10^{-21}}$	$1.35 \times 10^{+00}$	1.23×10^{-05}	1.09×10^{-11}	8.82×10^{-07}
	SD	4.63×10^{-01}	6.33×10^{-02}	6.27×10^{-13}	$\mathbf{2.81 \times 10^{-21}}$	6.84×10^{-01}	1.99×10^{-06}	6.72×10^{-13}	1.51×10^{-07}
F_5	Mean	$1.87 \times 10^{+01}$	$4.28 \times 10^{+01}$	$1.47 \times 10^{+02}$	$2.62 \times 10^{+01}$	$4.19 \times 10^{+01}$	$\mathbf{1.26 \times 10^{+01}}$	$2.89 \times 10^{+01}$	$2.79 \times 10^{+01}$
	SD	$2.82 \times 10^{+01}$	$3.07 \times 10^{+01}$	$2.71 \times 10^{+02}$	6.30×10^{-01}	$2.82 \times 10^{+01}$	$2.86 \times 10^{+01}$	$\mathbf{3.95 \times 10^{-02}}$	2.56×10^{-01}
F_6	Mean	1.56×10^{-07}	1.47×10^{-15}	1.06×10^{-14}	1.10×10^{-01}	9.65×10^{-04}	4.74×10^{-10}	$5.05 \times 10^{+00}$	**0**
	SD	2.37×10^{-07}	2.20×10^{-15}	5.40×10^{-14}	1.58×10^{-01}	4.19×10^{-04}	1.44×10^{-10}	5.34×10^{-01}	**0**
F_7	Mean	3.79×10^{-01}	3.22×10^{-02}	1.25×10^{-01}	3.99×10^{-04}	9.87×10^{-03}	2.06×10^{-03}	6.68×10^{-04}	$\mathbf{3.29 \times 10^{-04}}$
	SD	9.79×10^{-02}	1.12×10^{-02}	4.30×10^{-02}	1.65×10^{-04}	4.50×10^{-03}	1.22×10^{-03}	3.25×10^{-04}	$\mathbf{1.52 \times 10^{-04}}$

Table 5. Results of multi-modal benchmark functions.

Fun.	Metr.	GA	PSO	BA	GWO	CSA	SSA	BOA	BOA-CE
F_8	Mean	**−10, 486.64**	−6500.75	−3525.02	−6941.03	−7294.61	−2824.35	−4246.74	−4017.15
	SD	626.18	819.29	**192.65**	641.14	801.93	211.32	350.36	323.25
F_9	Mean	$1.33 \times 10^{+00}$	3.24×10^{-01}	$1.35 \times 10^{+02}$	$3.45 \times 10^{+01}$	$1.87 \times 10^{+01}$	$1.21 \times 10^{+01}$	$\mathbf{3.03 \times 10^{-14}}$	$1.01 \times 10^{+01}$
	SD	$1.23 \times 10^{+00}$	$1.78 \times 10^{+00}$	$3.38 \times 10^{+01}$	$1.14 \times 10^{+01}$	$7.82 \times 10^{+00}$	$5.86 \times 10^{+00}$	$\mathbf{1.46 \times 10^{-13}}$	$3.46 \times 10^{+01}$
F_{10}	Mean	1.76×10^{-04}	1.10×10^{-14}	$1.64 \times 10^{+01}$	1.89×10^{-08}	$2.56 \times 10^{+00}$	4.24×10^{-01}	1.01×10^{-11}	$\mathbf{4.44 \times 10^{-15}}$
	SD	7.59×10^{-05}	3.11×10^{-15}	$2.30 \times 10^{+00}$	1.72×10^{-08}	6.99×10^{-01}	6.81×10^{-01}	1.10×10^{-12}	**0**
F_{11}	Mean	1.23×10^{-03}	8.70×10^{-04}	$1.22 \times 10^{+02}$	1.29×10^{-02}	2.89×10^{-02}	2.47×10^{-01}	5.59×10^{-16}	**0**
	SD	3.41×10^{-03}	2.68×10^{-03}	$5.54 \times 10^{+01}$	1.45×10^{-02}	1.34×10^{-02}	1.46×10^{-01}	5.64×10^{-16}	**0**
F_{12}	Mean	5.19×10^{-02}	1.35×10^{-02}	$1.82 \times 10^{+01}$	$\mathbf{9.96 \times 10^{-18}}$	6.25×10^{-01}	8.53×10^{-02}	3.84×10^{-01}	2.02×10^{-05}
	SD	1.12×10^{-01}	9.37×10^{-03}	$5.84 \times 10^{+00}$	$\mathbf{1.13 \times 10^{-17}}$	5.39×10^{-01}	2.11×10^{-01}	1.13×10^{-01}	8.76×10^{-05}
F_{13}	Mean	5.09×10^{-03}	1.51×10^{-01}	$3.85 \times 10^{+01}$	1.10×10^{-03}	1.20×10^{-02}	7.32×10^{-04}	$2.24 \times 10^{+00}$	$\mathbf{1.35 \times 10^{-32}}$
	SD	1.23×10^{-02}	1.43×10^{-01}	$1.62 \times 10^{+01}$	3.35×10^{-03}	1.51×10^{-02}	2.79×10^{-030}	4.04×10^{-01}	$\mathbf{5.57 \times 10^{-48}}$

Table 6. Results of composite benchmark functions.

Fun.	Metr.	GA	PSO	BA	GWO	CSA	SSA	BOA	BOA-CE
F_{14}	Mean	46.67	70.00	70.00	45.90	26.67	43.33	270.28	**23.33**
	SD	50.74	95.23	83.67	71.84	44.98	67.89	58.82	**43.02**
F_{15}	Mean	57.91	128.09	139.61	115.17	116.59	**27.77**	262.27	104.76
	SD	62.53	76.97	95.29	82.14	61.98	**17.39**	110.66	72.26
F_{16}	Mean	154.18	172.96	348.41	160.34	240.86	195.18	374.19	**105.92**
	SD	42.63	75.64	111.53	48.84	66.57	**38.64**	57.17	59.17
F_{17}	Mean	305.43	378.64	452.73	360.09	424.94	319.45	602.76	**264.51**
	SD	35.95	128.17	114.34	105.96	92.62	**27.58**	85.08	96.10
F_{18}	Mean	53.09	92.41	85.22	56.75	**9.49**	14.10	130.84	71.00
	SD	79.02	136.96	150.90	62.29	**17.69**	29.53	59.39	47.42
F_{19}	Mean	627.64	765.17	745.07	698.91	**497.13**	55.26	819.40	790.79
	SD	192.16	191.11	198.03	201.69	**17.93**	152.72	112.13	153.62

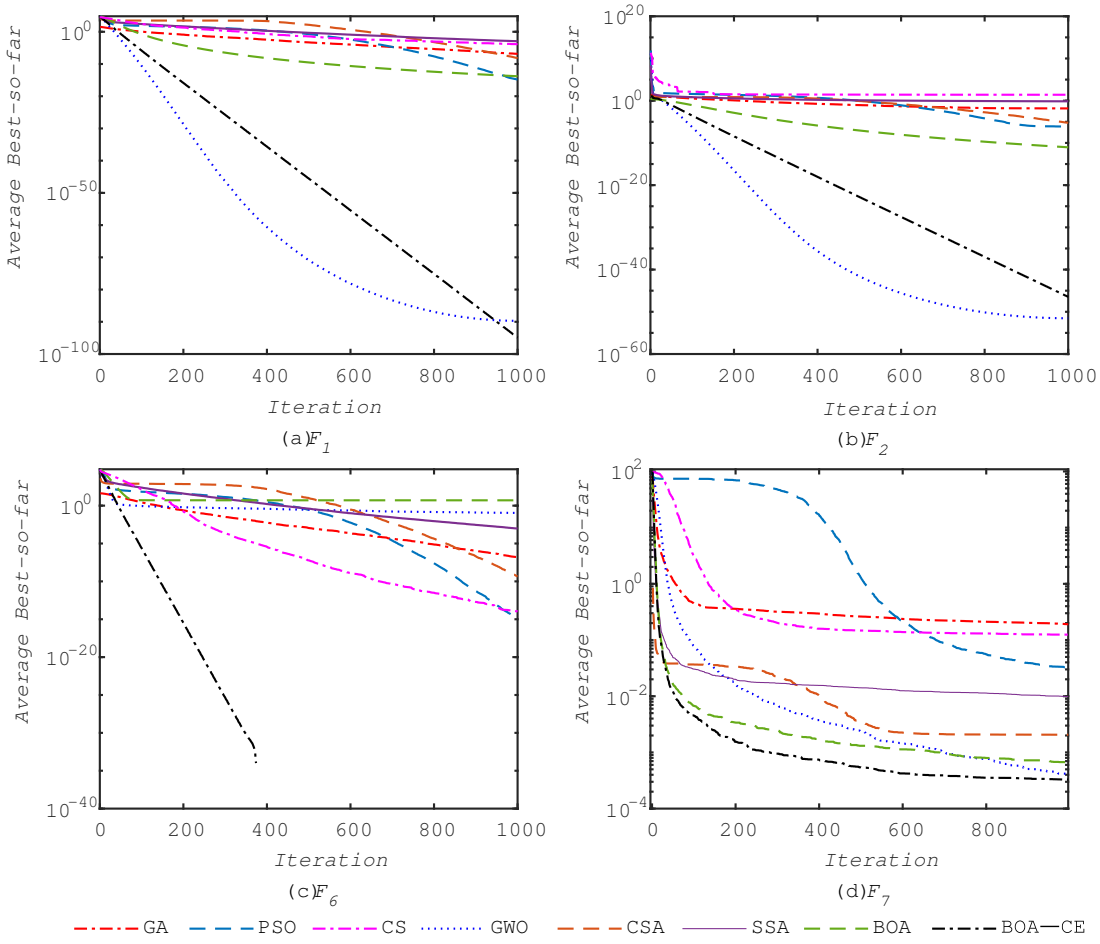

Figure 3. Convergence of algorithms when applied to some unimodal benchmark functions.

Figure 4 shows the convergence curves of the algorithms when applied to some of the multimodal test functions, such as F10, F11, F12, and F13. The figure shows that the BOA-CE algorithm has the fastest convergence for the majority of the multimodal test functions.

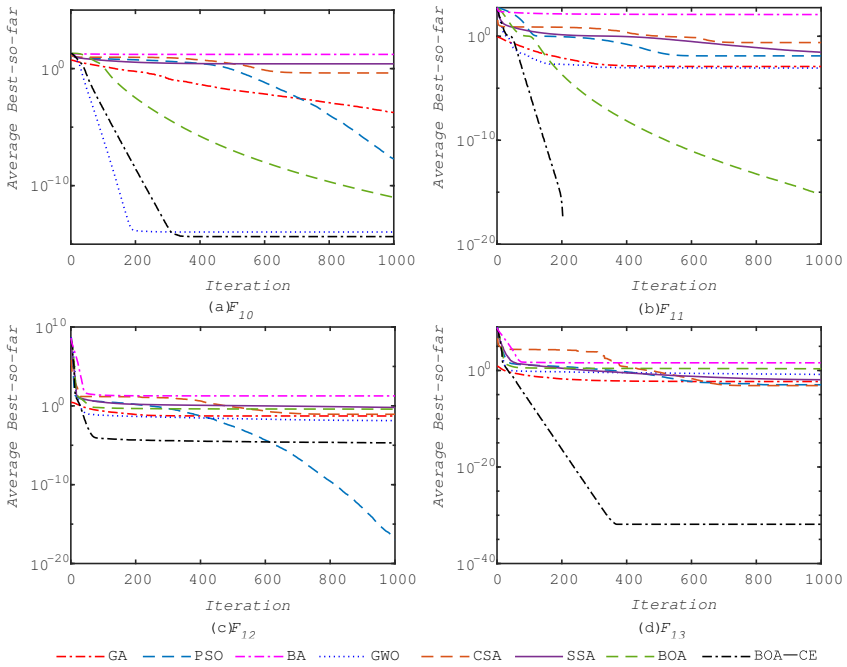

Figure 4. Convergence of algorithms on some unimodal benchmark functions.

4.3. Results of the Algorithms on Composite Test Functions

The results of composite test functions are reported in Table 6. From Table 6, we can find that the BOA-CE algorithm outperforms others in most of the composite test functions. Considering the characteristics of composite test functions and these results, it may be stated that the BOA-CE algorithm appropriately balances exploration and exploitation to focus on the high-performance areas of the search space.

The convergence curves of the methods for some of the composite test functions are shown in Figure 5. Figure 5 clearly shows that the BOA-CE algorithm has the fastest convergence on the composite test functions.

In addition, Table 7 presents the time consumption of all the algorithms to solve the 19 benchmark functions. From Table 7, it can be seen that the least total time cost is PSO to solve all of the benchmark functions, followed by CSA, and the BOA-CE is ranked third, which is better than BOA.

The results of three test functions show that the local optima avoidance of the improved method was enhanced by embedding the cross-entropy method. This global stochastic optimization method enables the BOA-CE algorithm to achieve a proper balance between exploration and exploitation, and enhance its global search capability. Furthermore, the new method is particularly outstanding in solving multimodal function optimization problems.This provides a new method for solving complex engineering design optimization problems.

4.4. Analysis of the Hybrid BOA-CE Algorithm

The main reasons for the excellent performance of the proposed BOA-CE algorithm in solving global optimization problems may be stated as follows:

- The BOA, which mimics the food foraging behavior of butterflies in nature, has the advantage of a fast convergence rate. Based on co-updating, the BOA-CE uses the excellent individuals obtained by the BOA operator to update the CE operator's probability parameters during the iterative process, which speeds up the convergence rate of the CE operator.
- The CE method is a global stochastic optimization method based on statistical model, and has the advantages of randomness, adaptability, and robustness, which bring a good population

diversity to the BOA operator so that it can effectively avoid falling into a local optimum and improve its global search capability.

- The BOA-CE algorithm employs a co-evolutionary technique to co-update the BOA operator's population and the CE operator's probability parameters. This enables the improved method to obtain an appropriate balance between exploration and exploitation and have more superior performance in terms of exploitation, exploration, and local optima avoidance when solving complex optimization problems.

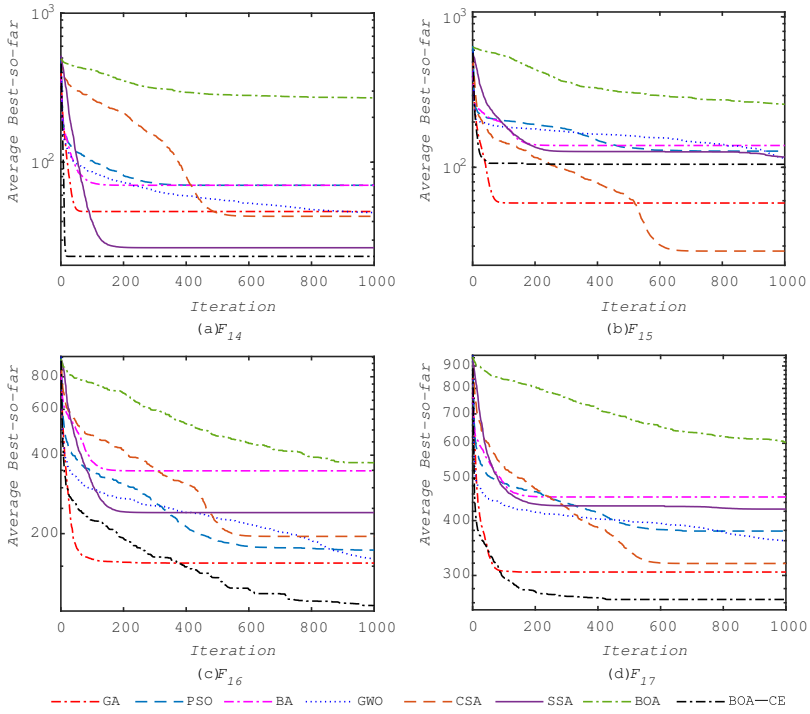

Figure 5. Convergence of algorithms when applied to some composite benchmark functions.

Table 7. Time consumption of BOA-CE solving 19 benchmark functions.

Fun.	GA	PSO	BA	GWO	CSA	SSA	BOA	BOA-CE
F_1	8.96	0.20	0.33	0.55	0.10	0.66	0.46	0.41
F_2	8.80	0.21	0.35	0.57	0.11	0.43	0.49	0.42
F_3	10.75	0.76	2.52	1.13	2.02	0.56	1.61	1.15
F_4	8.44	0.20	0.41	0.55	0.14	0.38	0.46	0.40
F_5	9.11	0.27	0.59	0.62	0.29	0.45	0.59	0.51
F_6	8.57	0.20	0.34	0.55	0.10	0.38	0.45	0.40
F_7	9.29	0.54	0.91	1.14	0.58	0.52	0.84	0.90
F_8	8.72	0.31	0.87	0.65	0.19	0.44	0.68	0.62
F_9	8.81	0.26	0.41	0.57	0.15	0.40	0.58	0.48
F_{10}	8.66	0.26	0.43	0.59	0.17	0.41	0.55	0.45
F_{11}	8.97	0.31	0.69	0.64	0.34	0.47	0.66	0.54
F_{12}	10.14	1.17	1.62	1.52	1.16	0.91	2.45	1.67
F_{13}	10.10	1.17	1.62	1.52	1.17	0.91	2.46	1.66
F_{14}	264.89	249.55	256.03	304.78	260.24	266.09	263.94	265.80
F_{15}	261.13	232.07	261.98	300.36	258.01	255.73	263.67	259.32
F_{16}	259.28	210.24	269.61	250.40	263.24	258.70	259.05	252.60
F_{17}	313.15	212.06	287.05	273.28	282.19	283.91	286.06	282.38
F_{18}	293.31	262.41	289.38	272.57	286.17	286.81	288.63	286.88
F_{19}	287.42	172.65	278.74	288.92	282.58	297.25	291.06	284.31
Total	1798.50	1344.82	1653.88	1700.91	1638.95	1655.41	1664.70	1640.89
Ranking	8	1	5	7	2	6	4	3

5. Using the BOA-CE Algorithm for Classical Engineering Design Problems

In this section, the BOA-CE algorithm was tested on three classical constrained engineering design problems: a tension/compression spring, a welded beam, and a pressure vessel [9,15,19,20,28,31,47–52]. Constraint handling is a challenging task for a method when solving these problems. Penalty functions are divided into different types: static, dynamic, annealing, adaptive, co-evolutionary, and death penalty [47]. For the sake of simplicity, we equipped the BOA-CE algorithm with a death penalty function to handle constraints. Table 8 shows the BOA-CE parameters chosen to solve these design problems, where N is the population sizes of the BOA and CE operator, and $Iter_{max}$ represents the maximum number of iterations.

Table 8. Parameters of the BOA-CE algorithm for solving design problems.

Design Problem	BOA Operator		CE Operator	
	N	$Iter_{max}$	N	$Iter_{max}$
Tension/compression spring	100	50	100	50
Welded beam	100	100	100	50
Pressure vessel	100	100	100	50

5.1. Tension/Compression Spring Design

The objective of this problem is to minimize the weight of the tension/compression spring shown in Figure 6. It contains three design variables: the wire diameter (d), mean coil diameter (D), and number of active coils (N), which are subject to one linear and three nonlinear inequality constraints on shear stress, surge frequency, and deflection. The optimization problem is formulated as follows:

$$\min f(x) = (x_3 + 2)x_2x_1^2, \tag{14}$$

subject to

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0, \tag{15}$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0, \tag{16}$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \tag{17}$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0, \tag{18}$$

where $0.05 \leq x_1 \leq 2.00$, $0.25 \leq x_2 \leq 1.30$, and $2.00 \leq x_3 \leq 15.00$.

Figure 6. The tension/compression spring design problem.

This problem has been popular among researchers and has been optimized using various meta-heuristic algorithms. To solve this problem, Coello employed GA [48], He and Wang employed PSO [49], Gandomi et al. used BA [50], Mirjalili employed GWO [15] and SSA [20], Lee and Geem [51]

used HS, and Askarzadeh [19] used CSA. The best weight values are highlighted in bold. Table 9 shows a comparison of the results obtained using the BOA-CE algorithm and those obtained using the other algorithms. It can be seen that the BOA-CE and CSA algorithms find a design with the minimum weight for this problem and outperform all other algorithms.

Table 9. A comparison of results for the tension/compression spring design problem.

Algorithm	Optimum Variables			Optimum Weight
	d	***D***	***N***	
GA [48]	0.051480	0.351661	11.632201	0.0127048
PSO [49]	0.051728	0.357644	11.244543	0.012675
BA [50]	0.051690	0.356730	11.288500	0.012670
GWO [15]	0.051690	0.356737	11.288850	0.012666
HS [51]	0.051154	0.349871	12.076432	0.012671
CSA [19]	0.051689	0.356717	11.289012	**0.012665**
SSA [20]	0.051207	0.345215	12.004032	0.012676
BOA-CE	0.051618	0.355004	11.390144	**0.012665**

5.2. Welded Beam Design

The objective of this problem is to minimize the fabrication cost of the welded beam shown in Figure 7. It contains four design variables: the thickness of the weld (h), length of the clamped bar (l), height of the bar (t), and thickness of the bar (b), which are subject to two linear and five nonlinear inequality constraints on shear stress, bending stress in the beam, buckling load, and end deflection of the beam. The optimization problem can be stated as follows:

$$\min f(x) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2), \tag{19}$$

subject to

$$g_1(x) = \tau(x) - \tau_{max} \leq 0, \tag{20}$$

$$g_2(x) = \sigma(x) - \sigma_{max} \leq 0, \tag{21}$$

$$g_3(x) = \delta(x) - \delta_{max} \leq 0, \tag{22}$$

$$g_4(x) = x_1 - x_4 \leq 0, \tag{23}$$

$$g_5(x) = P - P_c(x) \leq 0, \tag{24}$$

$$g_6(x) = 0.125 - x_1 \leq 0, \tag{25}$$

$$g_7(x) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0, \tag{26}$$

where $\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{P}{\sqrt{2}x_1x_2}$, $\tau'' = \frac{MR}{J}$, $M = P(L + \frac{x_2}{2})$, $R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}$, $J = 2\{\sqrt{2}x_1x_2[\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2]\}$, $\sigma(x) = \frac{6PL}{x_4x_3^2}$, $\delta(x) = \frac{6PL^3}{Ex_3^3x_4}$, $P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$, $P = 6000$ lb, $L = 14$ in, $\sigma_{max} = 0.25$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{max} = 30{,}000$ psi, $0.1 \leq x_1, x_4 \leq 2$, and $0.1 \leq x_2, x_3 \leq 10$.

The welded beam design problem has been tackled using many meta-heuristic algorithms, such as GA [48], PSO [49], BA [50], GWO [15], HS [51], WOA [18], and CSA [19]. The optimization results using the BOA-CE algorithm are compared with those from the literature in Table 10. The minimum cost is highlighted in bold. The table shows that BOA-CE outperforms all other algorithms except CSA, with only a slight difference in its result.

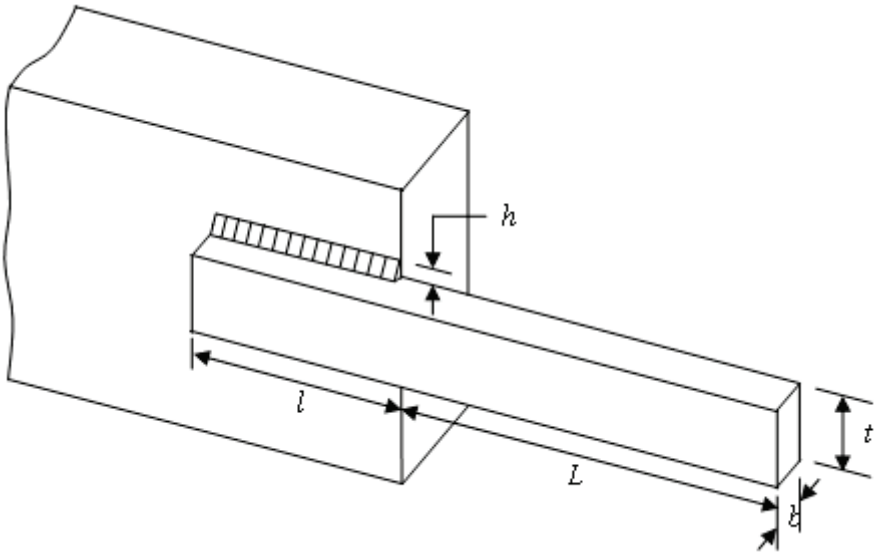

Figure 7. The welded beam design problem.

Table 10. A comparison of results for the tension/compression spring design problem.

Algorithm	Optimum Variables				Optimum Cost
	h	*l*	*t*	*b*	
GA [48]	0.205986	3.471328	9.020224	0.206480	1.728226
PSO [49]	0.202369	3.544214	9.048210	0.205723	1.731485
BA [50]	0.2015	3.562	9.0414	0.2057	1.7312
GWO [15]	0.205676	3.478377	9.03681	0.205778	1.72624
HS [51]	0.2442	6.2231	8.2915	0.2443	2.3807
WOA [18]	0.205396	3.484293	9.037426	0.206276	1 .730499
CSA [19]	0.205730	3.470489	9.036624	0.205730	**1.724852**
BOA-CE	0.205730	3.470481	9.036611	0.205730	1.724854

5.3. Pressure Vessel Design

The objective of this problem is to minimize the total cost of a pressure vessel considering the cost of material, forming and welding shown in Figure 8. There are two discrete and two continuous design variables: thickness of the shell (T_s), thickness of the head (T_h), inner radius (R) and length of the cylindrical section of the vessel (L), not including the head, which are subjected to three linear and one nonlinear inequality constraints. The thicknesses of the variables are integer multiples of 0.0625 inches. This optimization problem can be mathematically formulated as follows:

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{27}$$

subject to

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0, \tag{28}$$

$$g_2(x) = -x_3 + 0.00954x_4 \leq 0, \tag{29}$$

$$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0, \tag{30}$$

$$g_4(x) = x_4 - 240 \leq 0, \tag{31}$$

where $0 < x_1, x_2 \leq 99$, $0 < x_3, x_4 \leq 200$.

In the past, this problem is solved by GA [48], PSO [49], BA [50], GWO [15], HS [51], WOA [18], and DE [52]. Table 11 shows optimization results of BOA-CE are compared with those found by

other methods. It can be seen that BOA-CE and BA outperform all other algorithms except GWO. However the value of T_h obtained by GWO is not an integer multiple of 0.0625 inches and thus does not satisfy the constraint.

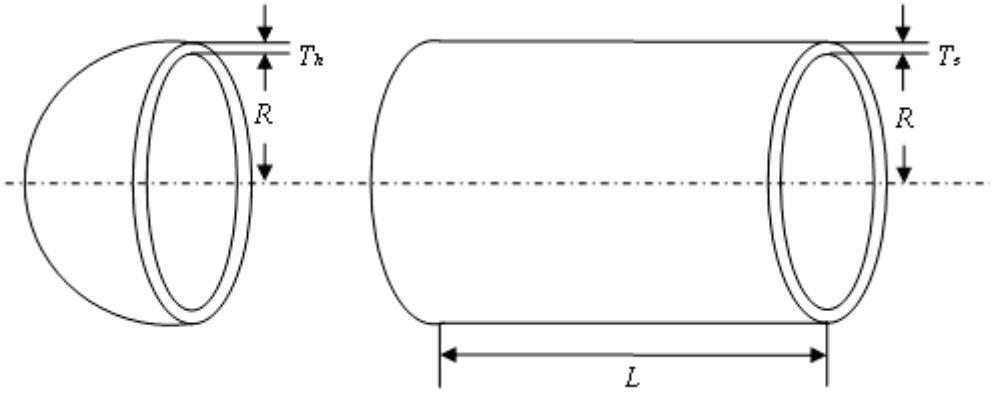

Figure 8. The tension/compression spring design problem.

Table 11. A comparison of results for the tension/compression spring design problem.

Algorithm	Optimum Variables				Optimum Cost
	T_s	T_h	R	L	
GA [48]	0.8125	0.4375	42.097398	176.654050	6059.9463
PSO [49]	0.8125	0.4375	42.091266	176.746500	6061.0777
BA [50]	0.8125	0.4375	42.0984456	176.636596	**6059.7143**
GWO [15]	0.8125	0.4345	42.089181	176.758731	6051.5639
HS [[51]	1.1250	0.6250	58.2789	43.7549	7198.433
WOA [18]	0.8125	0.4375	42.0982699	176.638998	6059.7410
DE [52]	0 .8125	0.4375	42.098411	176.63769	6059 .7341
BOA-CE	0.8125	0.4375	42.0984456	176.6365958	**6059.7143**

6. Conclusions

In order to improve the global search ability of the BOA, an improved BOA algorithm, named BOA-CE, was constructed by embedding the CE method into the BOA using a co-evolutionary technique. A total of 19 test functions were used to evaluate the performance of the new method in terms of its exploration, exploitation, local optima avoidance, and convergence rate. The results of the test functions demonstrated that the proposed algorithm can effectively avoid falling into a local optima and has excellent local and global search capacity. This is mainly due to the co-evolutionary technique, which enables the new method to obtain an appropriate balance between exploration and exploitation and has more superior performance when solving complex function optimization problems. Furthermore, the paper also considered the solving of three classical engineering problems using the hybrid algorithm. The comparative results show that the BOA-CE algorithm provides very competitive results when solving real problems with unknown search spaces. In future research, a discrete version of the BOA-CE algorithm will be constructed to solve combinatorial optimization problems.

Author Contributions: G.L. designed the algorithm, implemented all experiments, and wrote the manuscript. F.S. revised the manuscript. P.Z. conducted the literature review and analyzed the data. C.L. revised the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 71761012), the Natural Science Foundation of Anhui Province (Grant No. 1808085MG224), and the Natural Science Foundation of West Anhui University (Grant No. WXZR201818).

Acknowledgments: The authors are grateful for the support provided by the Risk Management and Financial Engineering Lab at the University of Florida, Gainesville, FL 32611, USA.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

See Table A1.

Table A1. Mathematical formulation of the primitive functions in Table 3.

Name	Formulation
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + e$
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Weierstrass	$f(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}[a^k\cos(2\pi b^k(x_i+0.5))]\right) - D\sum_{k=0}^{k_{max}}[a^k\cos(2\pi b^k\cdot 0.5)], a = 0.5, b = 3, k_{max} = 20$
Rastrigin	$f(x) = \sum_{i=1}^{D-1}[x_i^2 - 10\cos\left(2\pi x_i\right) + 10]$

D: dimensions.

References

1. Hu, X.; Eberhart, R.C.; Shi, Y. Engineering optimization with particle swarm. In Proceedings of the 2003 IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, 26–26 April 2003; pp. 53–57.
2. Sergeyev, Y.D.; Kvasov, D.E. A deterministic global optimization using smooth diagonal auxiliary functions. *Commun. Nonlinear Sci. Numer. Simul.* **2015**, *21*, 99–111. [CrossRef]
3. Lera, D.; Sergeyev, Y.D. GOSH: Derivative-free global optimization using multi-dimensional space-filling curves. *J. Glob. Optim.* **2018**, *71*, 193–211. [CrossRef]
4. Sergeyev, Y.D.; Kvasov, D.E.; Mukhametzhanov, M.S. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Sci. Rep.* **2018**, *8*, 453. [CrossRef] [PubMed]
5. Zilinskas, A.; Zhigljavsky, A. Stochastic Global Optimization: A Review on the Occasion of 25 Years of Informatica. *Informatica* **2016**, *27*, 229–256. [CrossRef]
6. Goldfeld, S.M.; Quandt, R.E.; Trotter, H.F. Maximization by quadratic hill-climbing. *Econometrica* **1966**, *34*, 541–551. [CrossRef]
7. Abbasbandy, S. Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method. *Appl. Math. Comput.* **2003**, *145*, 887–893. [CrossRef]
8. Jones, D.R.; Perttunen, C.D.; Stuckman, B.E. Lipschitzian optimization without the Lipschitz constant. *J. Optim. Theory Appl.* **1993**, *79*, 157–181. [CrossRef]
9. Mirjalili, S. The Ant Lion Optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
10. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
11. Wieczorek, L.; Ignaciuk, P. Continuous Genetic Algorithms as Intelligent Assistance for Resource Distribution in Logistic Systems. *Data* **2018**, *3*, 68. [CrossRef]
12. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
13. Yang, X.S.; Deb, S. Engineering Optimisation by Cuckoo Search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
14. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
15. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
16. Ghaemi, M.; Feizi-Derakhshi, M.R. Forest Optimization Algorithm. *Expert Syst. Appl.* **2014**, *41*, 6676–6687. [CrossRef]
17. Naz, M.; Zafar, K.; Khan, A. Ensemble Based Classification of Sentiments Using Forest Optimization Algorithm. *Data* **2019**, *4*, 76. [CrossRef]
18. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
19. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]

20. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
21. Arora, S.; Singh, S. Butterfly algorithm with Lèvy Flights for global optimization. In Proceedings of the 2015 International Conference on Signal Processing, Computing and Control (ISPCC), Waknaghat, India, 24–26 September 2015; pp. 220–224.
22. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
23. Yang, X.S.; Deb, S. Cuckoo search: Recent advances and applications. *Neural Comput. Appl.* **2014**, *24*, 169–174. [CrossRef]
24. Wolpert, D.H.; Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
25. Lai, X.S.; Zhang, M.Y. An Efficient Ensemble of GA and PSO for Real Function Optimization. In Proceedings of the 2009 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, China, 8–11 August 2009; pp. 651–655.
26. Mirjalili, S.; Hashim, S.Z.M. A New Hybrid PSOGSA Algorithm for Function Optimization. In Proceedings of the 2010 International Conference on Computer and Information Application (2010 ICCIA), Tianjin, China, 3–5 December 2010; pp. 374–377.
27. 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*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 673–680.
28. He, X.S.; Ding, W.J.; Yang, X.S. Bat algorithm based on simulated annealing and Gaussian perturbations. *Neural Comput. Appl.* **2014**, *25*, 459–468. [CrossRef]
29. Mafarja, M.M.; Mirjalili, S. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [CrossRef]
30. Pepelyshev, P.; Zhigljavsky, A.; Zilinskas, A. Performance of global random search algorithms for large dimensions. *J. Glob. Optim.* **2018**, *71*, 57–71. [CrossRef]
31. Rubinstein, R.Y. Optimization of Computer Simulation Models with Rare Events. *Eur. J. Oper. Res.* **1997**, *99*, 89–112. [CrossRef]
32. Arora, S.; Singh, S. An Improved Butterfly Optimization Algorithm for Global Optimization. *Adv. Sci.* **2017**, *8*, 711–717. [CrossRef]
33. Arora, S.; Singh, S. Node Localization in Wireless Sensor Networks Using Butterfly Optimization Algorithm. *Arab. J. Sci. Eng.* **2017**, *42*, 3325–3335. [CrossRef]
34. Kroese, D.P.; Portsky, S.; Rubinstein, R.Y. The Cross-Entropy Method for Continuous Multi-extremal Optimization. *Methodol. Comput. Appl. Probab.* **2006**, *8*, 383–407. [CrossRef]
35. Bekker, J.; Aldrich, C. The cross-entropy method in multi-objective optimisation: An assessment. *Eur. J. Oper. Res.* **2011**, *211*, 112–121. [CrossRef]
36. Rubinstein, R.Y. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodol. Comput. Appl. Probab.* **1999**, *1*, 127–190. [CrossRef]
37. Rubinstein, R.Y.; Kroese, D.P. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*; Springer: New York, NY, USA, 2004.
38. Chepuri, K.; Homem-De-Mello, T. Solving the vehicle routing problem with stochastic demands using the cross-entropy method. *Ann. Oper. Res.* **2005**, *134*, 153–181. [CrossRef]
39. Yu, J; Konaka, S.; Akutagawa, M.; Zhang, Q. Cross-Entropy-Based Energy-Efficient Radio Resource Management in HetNets with Coordinated Multiple Points. *Information* **2016**, *7*, 3. [CrossRef]
40. Joseph, A.G.; Bhatnagar, S. An online prediction algorithm for reinforcement learning with linear function approximation using cross entropy method. *Mach. Learn.* **2018**, *107*, 1385–1429. [CrossRef]
41. Peherstorfer, B.; Kramer, B.; Willcox, K. Multifidelity preconditioning of the cross-entropy method for rare event simulation and failure probability estimation. *SIAM/ASA J. Uncertain. Quantif.* **2018**, *6*, 737–761. [CrossRef]
42. Wang, Y.; Yang, H.; Qin, K. The Consistency between Cross-Entropy and Distance Measures in Fuzzy Sets. *Symmetry* **2019**, *11*, 386. [CrossRef]
43. Pramanik, S.; Dalapati, S.; Alam, S.; Smarandache, F.; Roy, T.K. NS-Cross Entropy-Based MAGDM under Single-Valued Neutrosophic Set Environment. *Information* **2018**, *9*, 37. [CrossRef]
44. Eiben, A.E.; Schipper, C.A. On evolutionary exploration and exploitation. *Fund. Inform.* **1998**, *35*, 35–50.

45. Yao, X.; Liu, Y.; Lin, G.M. Evolutionary Programming Made Faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
46. Liang, J.; Suganthan, P.; Deb, K. Novel composition test functions for numerical global optimization. In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, Pasadena, CA, USA, 8–10 June 2005; pp. 68–75.
47. Coello, C.C.A.; Mezura, M.E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inform.* **2002**, *16*, 193–203. [CrossRef]
48. Coello, C.C.A. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Comput. Ind.* **2000**, *41*, 113–127. [CrossRef]
49. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [CrossRef]
50. Gandomi, A.H.; Yang, X.S.; Alavi, A.H.; Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Comput. Appl.* **2013**, *22*, 1239–1255. [CrossRef]
51. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
52. Huang, F.; Wang, L.; He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 340–356. [CrossRef]