Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 9167414, 27 pages
https://doi.org/10.1155/2018/9167414

Research Article

Modified Backtracking Search Optimization Algorithm Inspired by Simulated Annealing for Constrained Engineering Optimization Problems

Hailong Wang,[1] Zhongbo Hu ,[1] Yuqiu Sun,[1] Qinghua Su,[1] and Xuewen Xia[2]

[1]*School of Information and Mathematics, Yangtze University, Jingzhou, Hubei 434023, China*
[2]*School of Software, East China Jiaotong University, Nanchang, Jiangxi 330013, China*

Correspondence should be addressed to Zhongbo Hu; huzbdd@126.com

Received 10 October 2017; Accepted 20 December 2017; Published 13 February 2018

Academic Editor: Silvia Conforto

The backtracking search optimization algorithm (BSA) is a population-based evolutionary algorithm for numerical optimization problems. BSA has a powerful global exploration capacity while its local exploitation capability is relatively poor. This affects the convergence speed of the algorithm. In this paper, we propose a modified BSA inspired by simulated annealing (BSAISA) to overcome the deficiency of BSA. In the BSAISA, the amplitude control factor (F) is modified based on the Metropolis criterion in simulated annealing. The redesigned F could be adaptively decreased as the number of iterations increases and it does not introduce extra parameters. A self-adaptive ε-constrained method is used to handle the strict constraints. We compared the performance of the proposed BSAISA with BSA and other well-known algorithms when solving thirteen constrained benchmarks and five engineering design problems. The simulation results demonstrated that BSAISA is more effective than BSA and more competitive with other well-known algorithms in terms of convergence speed.

1. Introduction

Optimization is an essential research objective in the fields of applied mathematics and computer sciences. Optimization algorithms mainly aim to obtain the global optimum for optimization problems. There are many different kinds of optimization problems in real world. When an optimization problem has a simple and explicit gradient information or requires relatively small budgets of allowed function evaluations, the implementation of classical optimization techniques such as mathematical programming often could achieve efficient results [1]. However, many real-world engineering optimization problems may have complex, nonlinear, or nondifferentiable forms, which make them difficult to be tackled by using classical optimization techniques. The emergence of metaheuristic algorithms has overcome the deficiencies of classical optimization techniques to some extent, as they do not require gradient information and have the ability to escape from local optima. Metaheuristic algorithms are mainly inspired from a variety of natural phenomena and/or biological social behavior. Among these metaheuristic algorithms, swarm intelligence algorithms and evolutionary algorithms perhaps are the most attractive [2]. Swarm intelligence algorithms [3] generally simulate the intelligence behavior of swarms of creatures, such as particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], cuckoo search (CS) [6], and the artificial bee colony (ABC) algorithm [7]. These types of algorithms generally are developed by inspirations from a series of complex behavior processes in swarms with mutual cooperation and self-organization, in which "cooperation" is their core concept. The evolutionary algorithms (EAs) [8, 9] are inspired by the mechanism of nature evolution, in which "evolution" is the key idea. Examples of EAs include genetic algorithm (GA) [10], differential evolution (DE) [11–14], covariance matrix adaptation evolution strategy (CMAES) [15], and the backtracking search optimization algorithm (BSA) [16].

BSA is an iterative population-based EA, which was first proposed by Civicioglu in 2013. BSA has three basic genetic operators: selection, mutation, and crossover. The

main difference between BSA and other similar algorithms is that BSA possesses a memory for storing a population from a randomly chosen previous generation, which is used to generate the search-direction matrix for the next iteration. In addition, BSA has a simple structure, which makes it efficient, fast, and capable of solving multimodal problems. BSA has only one control parameter called the *mix-rate*, which significantly reduces the sensitivity of the initial values to the algorithm's parameters. Due to these characteristics, in less than 4 years, BSA has been employed successfully to solve various engineering optimization problems, such as power systems [17–19], induction motor [20, 21], antenna arrays [22, 23], digital image processing [24, 25], artificial neural networks [26–29], and energy and environmental management [30–32].

However, BSA has a weak local exploitation capacity and its convergence speed is relatively slow. Thus, many studies have attempted to improve the performance of BSA and some modifications of BSA have been proposed to overcome the deficiencies. From the perspective of modified object, the modifications of BSA can be divided into the following four categories. It is noted that we consider classifying the publication into the major modification category if it has more than one modification:

(i) Modifications of the initial populations [33–38]

(ii) Modifications of the reproduction operators, including the mutation and crossover operators [39–47]

(iii) Modifications of the selection operators, including the local exploitation strategy [48–51]

(iv) Modifications of the control factor and parameter [52–57].

The research on controlling parameters of EAs is one of the most promising areas of research in evolutionary computation; even a little modification of parameters in an algorithm can make a considerable difference [58]. In the basic BSA, the value of amplitude control factor (F) is the product of three and the standard normal distribution random number (i.e., $F = 3 \cdot \text{randn}$), which is often too large or too small according to its formulation. This may give BSA a powerful global exploration capability at the early iterations; however, it also affects the later exploitation capability of BSA. Based on these considerations, we focus mainly on the influence of the amplitude control factor (F) on the BSA, that is, the fourth of the categories defined above. Duan and Luo [52] redesigned an adaptive F based on the fitness statistics of population at each iteration. Wang et al. [53] and Tian et al. [54] proposed an adaptive F based on Maxwell-Boltzmann distribution. Askarzadeh and dos Santos Coelho [55] proposed an adaptive F based on Burger's chaotic map. Chen et al. [56] redesigned an adaptive F by introducing two extra parameters. Nama et al. [57] proposed a new F to adaptively change in the range of $(0.45, 1.99)$ and new mix-rate to randomly change in the range of $(0, 1)$. These modifications of F have achieved good effects in the BSA.

Different from the modifications of F in BSA described above, a modified version of BSA (BSAISA) inspired by simulated annealing (SA) is proposed in this paper. In the BSAISA, F based on iterations is redesigned by learning a characteristic where SA can probabilistically accept a higher energy state and the acceptance probability decreases with the decrease in temperature. The redesigned F can adaptively decrease as the number of iterations increases without introducing extra parameters. This adaptive variation tendency provides an efficient tradeoff between early exploration and later exploitation capability. We verified the effectiveness and competitiveness of BSAISA in simulation experiments using thirteen constrained benchmarks and five engineering design problems in terms of convergence speed.

The remainder of this paper is organized as follows. Section 2 introduces the basic BSA. As the main contribution of this paper, a detailed explanation of BSAISA is presented in Section 3. In Section 4, we present two sets of simulation experiments in which we implemented BSAISA and BSA to solve thirteen constrained optimization and five engineering design problems. The results are compared with those obtained by other well-known algorithms in terms of the solution quality and function evaluations. Finally, we give our concluding remarks in Section 5.

2. Backtracking Search Optimization Algorithm (BSA)

BSA is a population-based iterative EA. BSA generates trial populations to take control of the amplitude of the search-direction matrix which provides a strong global exploration capability. BSA equiprobably uses two random crossover strategies to exchange the corresponding elements of individuals in populations and trial populations during the process of crossover. Moreover, BSA has two selection processes. One is used to select population from the current and historical populations; the other is used to select the optimal population. In general, BSA can be divided into five processes: initialization, selection I, mutation, crossover, and selection II [16].

2.1. Initialization. BSA generates initial population P and initial old population oldP using

$$\begin{aligned} P_{i,j} &\sim U\left(\text{low}_j, \text{up}_j\right), \\ \text{oldP}_{i,j} &\sim U\left(\text{low}_j, \text{up}_j\right), \end{aligned} \tag{1}$$

where $P_{i,j}$ and $\text{oldP}_{i,j}$ are the jth individual elements in the problem dimension (D) that falls in ith individual position in the population size (N), respectively, low_j and up_j mean the lower boundary and the upper boundary of the jth dimension, respectively, and U is a random uniform distribution.

2.2. Selection I. BSA's selection I process is the beginning of each iteration. It aims to reselect a new oldP for calculating the search direction based on population P and historical population oldP. The new oldP is reselected through the "if-then" rule in

$$\text{if } a < b \text{ then oldP} := P \mid a, b \sim U(0, 1), \tag{2}$$

where $\coloneqq$ is the update operation; a and b represent random numbers between 0 and 1. The update operation (see (2)) ensures BSA has a memory. After oldP is reselected, the order of the individuals in oldP is randomly permuted by

$$\text{oldP} \coloneqq \text{permuting}\,(\text{oldP})\,. \tag{3}$$

2.3. Mutation. The mutation operator is used for generating the initial form of trial population M with

$$M = P + F \cdot (\text{oldP} - P)\,, \tag{4}$$

where F is the amplitude control factor of mutation operator used to control the amplitude of the search direction. The value $F = 3 \cdot \text{randn}$, where $\text{randn} \sim N(0, 1)$, and N is standard normal distribution.

2.4. Crossover. In this process, BSA generates the final form of trial population T. BSA equiprobably uses two crossover strategies to manipulate the selected elements of the individuals at each iteration. Both the strategies generate different binary integer-valued matrices (map) of size $N \cdot D$ to select the elements of individuals that have to be manipulated.

Strategy I uses the mix-rate parameter (mix-rate) to control the numbers of elements of individuals that are manipulated by using $\lceil \text{mix-rate} \cdot \text{rand} \cdot D \rceil$, where mix-rate = 1. Strategy II manipulates the only one randomly selected element of individuals by using randi(D), where $\text{randi} \sim U(0, D)$.

The two strategies equiprobably are employed to manipulate the elements of individuals through the "if-then" rule: if $\text{map}_{n,m} = 1$, where $n \in \{1, 2, \ldots, N\}$ and $m \in \{1, 2, \ldots, D\}$, then, T is updated with $T_{n,m} \coloneqq P_{n,m}$.

At the end of crossover process, if some individuals in T have overflowed the allowed search space limits, they will need to be regenerated by using (1).

2.5. Selection II. In BSA's selection II process, the fitness values in P_i and T_i are compared, used to update the population P based on a greedy selection. If T_i have a better fitness value than P_i, then, T_i is updated to be P_i. The population P is updated by using

$$P_i = \begin{cases} T_i, & \text{if fitness}\,(T_i) < \text{fitness}\,(P_i)\,, \\ P_i, & \text{else.} \end{cases} \tag{5}$$

If the best individual of P (P_{best}) has a better fitness value than the global minimum value obtained, the global minimizer will be updated to be P_{best}, and the global minimum value will be updated to be the fitness value of P_{best}.

3. Modified BSA Inspired by SA (BSAISA)

As mentioned in the introduction of this paper, the research work on the control parameters of an algorithm is very meaningful and valuable. In this paper, in order to improve BSA's exploitation capability and convergence speed, we propose a modified version of BSA (BSAISA) where the redesign of F is inspired by SA. The modified details of BSAISA are described in this section. First, the structure of the modified F is described, before we explain the detailed design principle for the modified F inspired by SA. Subsequently, two numerical tests are used to illustrate that the redesigned F improves the convergence speed of the algorithm. We introduce a self-adaptive ε-constrained method for handling constraints at the end of this section.

3.1. Structure of the Adaptive Amplitude Control Factor F. The modified F is a normally distributed random number, where its mean value is an exponential function and its variance is equal to 1. In BSAISA, we redesign the adaptive F to replace the original version using

$$F_i \sim N\left(\exp\left(\frac{-1/\left|\Delta I_i\right|}{1/G}\right), 1\right), \tag{6}$$

where i is the index of individuals, F_i is the adaptive amplitude control factor that corresponds to the ith individual, $|\Delta I_i|$ is the absolute value of the difference between the objective function values of P_i and T_i (individual differences), N is the normal distribution, and G is the current iteration.

According to (6), the exponential function (mean value) decreases dynamically with the change in the number of iterations (G) and the individual differences ($|\Delta I_i|$). Based on the probability density function curve of the normal distribution, the modified F can be decreased adaptively as the number of iterations increases. Another characteristic of the modified F is that there are not any extra parameters.

3.2. Design Principle of the Modified Amplitude Control Factor F. The design principle of the modified F is inspired by the Metropolis criterion in SA. SA is a metaheuristic optimization technique based on physical behavior in nature. SA based on the Monte Carlo method was first proposed by Metropolis et al. [59] and it was successfully introduced into the field of combinatorial optimization for solving complex optimization problems by Kirkpatrick et al. [60].

The basic concept of SA derives from the process of physical annealing with solids. An annealing process occurs when a metal is heated to a molten state with a high temperature; then it is cooled slowly. If the temperature is decreased quickly, the resulting crystal will have many defects and it is just metastable; even the most stable crystalline state will be achieved at all. In other words, this may form a higher energy state than the most stable crystalline state. Therefore, in order to reach the absolute minimum energy state, the temperature needs to be decreased at a slow rate. SA simulates this process of annealing to search the global optimal solution in an optimization problem. However, accepting only the moves that lower the energy of system is like extremely rapid quenching; thus SA uses a special and effective acceptance method, that is, Metropolis criterion, which can probabilistically accept the hill-climbing moves (the higher energy moves). As a result, the energy of the system evolves into a Boltzmann distribution during the process of the simulated annealing. From this angle of view,

it is no exaggeration to say that the Metropolis criterion is the core of SA.

The Metropolis criterion can be expressed by the physical significance of energy, where the new energy state will be accepted when the new energy state is lower than the previous energy state, and the new energy state will be probabilistically accepted when the new energy state is higher than the previous energy state. This feature of SA can escape from being trapped in local minima especially in the early stages of the search. It can also be described as follows.

(i) If $\Delta E = E_j - E_i \leq 0$, then the new state j is accepted and the energy with the displaced atom is used as the starting point for the next step, where E represents the energy of the atom. Both i and j are the states of atoms, and j is the next state of i.

(ii) If $\Delta E > 0$, then calculate the probability of $P(\Delta E) = \exp(-\Delta E/kT_c)$, and generate a random number θ, which is a uniform distribution over $(0, 1)$, where k is Boltzmann's constant (in general, $k = 1$) and T_c is the current temperature. If $P(\Delta E) > \theta$, then the new energy will be accepted; otherwise, the previous energy is used to start the next step.

Analysis 1. The Metropolis criterion states that SA has two characteristics: (1) SA can probabilistically accept the higher energy and (2) the acceptance probability of SA decreases as the temperature decreases. Therefore, SA can reject and jump out of a local minimum with a dynamic and decreasing probability to continue exploiting the other solutions in the state space. This acceptance mechanism can enrich the diversity of energy states.

Analysis 2. As shown in (4), F is used to control the amplitude of population mutation in BSA, thus F is an important factor for controlling population diversity. If F is excessively large, the diversity of the population will be too high and the convergence speed of BSA will slow down. If F is excessively small, the diversity of the population will be reduced so it will be difficult for BSA to obtain the global optimum and it may be readily trapped by a local optimum. Therefore, adaptively controlling the amplitude of F is a key to accelerating the convergence speed of the algorithm and maintaining its the population diversity.

Based on Analyses 1 and 2, it is clear that if F can dynamically decrease, the convergence speed of BSA will be able to accelerate while maintaining the population diversity. On the other hand, SA possesses this characteristic that its acceptance probability can be dynamically reduced. Based on these two considerations, we propose BSAISA with a redesigned F, which is inspired by SA. More specifically, the new F (see (6)) is redesigned by learning the formulation ($P(\Delta E)$) of acceptance probability, and its formulation has been shown in the previous subsection.

For the two formulas of the modified F and $P(\Delta E)$, the individual difference ($|\Delta I_i|$) of a population or the energy difference (ΔE) of a system will decrease as the number of iterations increases in an algorithm, and the temperature of SA tends to decrease, while the iteration of BSA tends to increase. As a result, one can observe the correspondence between modified F and $P(\Delta E)$ where the reciprocal of individual difference ($1/|\Delta I_i|$) corresponds to the energy difference (ΔE) of SA, and the reciprocal of current iteration ($1/G$) corresponds to the current temperature (T_c) of SA. In this way, the redesigned F can be decreased adaptively as the number of iterations increases.

3.3. Numerical Analysis of the Modified Amplitude Control Factor F. In order to verify that the convergence speed of the basic BSA is improved with the modified F, two types (unimodal and multimodal) of unconstrained benchmark functions are used to test the changing trends in F and population variances and best function values as the iterations increases, respectively. The two functions are Schwefel 1.2 and Rastrigin, and their detailed information is provided in [61]. The two functions and the user parameters including the populations (N), dimensions (D), and maximum iterations (Max) are shown in Table 1. Three groups of test results are compared in the tests including (1) the comparative curves of the mean values of the modified F and original F for Schwefel 1.2 and Rastrigin, (2) the comparative curves of the mean values of BSA and BSAISA population variances for two functions, and (3) the convergence curves of BSA and BSAISA for two functions. They are depicted in Figures 1, 2, and 3, respectively. Based on Figures 1–3, two results can be observed as follows.

(1) According to the trends of F in Figure 1, both the original F and modified F are subject to changes from the normal distribution. The mean value of the original F does not tend to decrease as the number of iterations increases. By contrast, the mean value of the modified F exhibits a clear and fluctuating downward trend as the number of iterations increases.

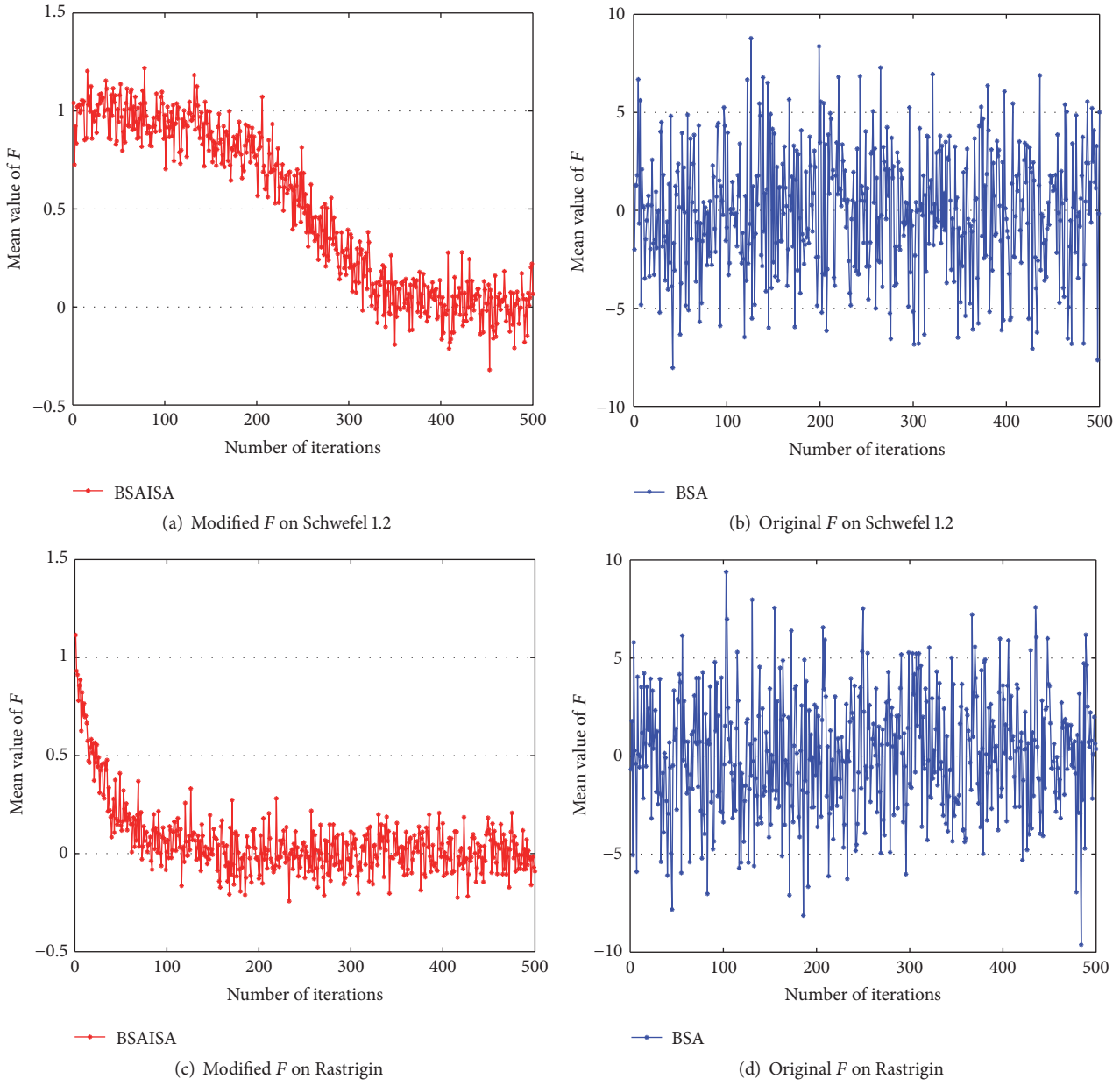
(2) According to Figure 2, the population variances of BSAISA and BSA both exhibit a clear downward trend as the number of iterations increases. The population variances of BSAISA and BSA are almost same during the early iterations. This illustrates that the modified F does not reduce the population diversity in the early iterations. In the middle and later iterations, the population variances of BSAISA decrease more quickly than that of BSA. This illustrates that the modified F improves the convergence speed. As can be seen from Figure 3, as the number of iterations increases, the best objective function value of BSAISA drops faster than that of BSA. This shows that the modified F improves the convergence speed of BSA. Moreover, BSAISA can find a more accurate solution at the same computational cost.

Summary. Based on the design principle and numerical analysis of the modified F, the modified F exhibits an overall fluctuating and downward trend, which matches with the concept of the acceptance probability in SA. In particular, during the early iterations, the modified F is relatively large. This allows BSAISA to search in a wide region, while maintaining the population diversity of BSAISA. As the number of iterations decreases, the modified F gradually exhibits a decreasing trend. This accelerates the convergence speed of BSAISA. In the later iterations, the modified F is relatively small. This enables BSAISA to fully search in the

Table 1: Two benchmark functions and the corresponding populations, dimensions, and max iterations.

Name	Objective function	Range	N	D	Max
Schwefel 1.2	$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]$	100	30	500
Rastrigin	$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	$[-5.12, 5.12]$	100	30	500

Note. "N" means populations, "D" means dimensions, and "Max" means maximum iterations.



(a) Modified F on Schwefel 1.2

(b) Original F on Schwefel 1.2

(c) Modified F on Rastrigin

(d) Original F on Rastrigin

Figure 1: Comparisons of modified F and original F for Schwefel 1.2 and Rastrigin.

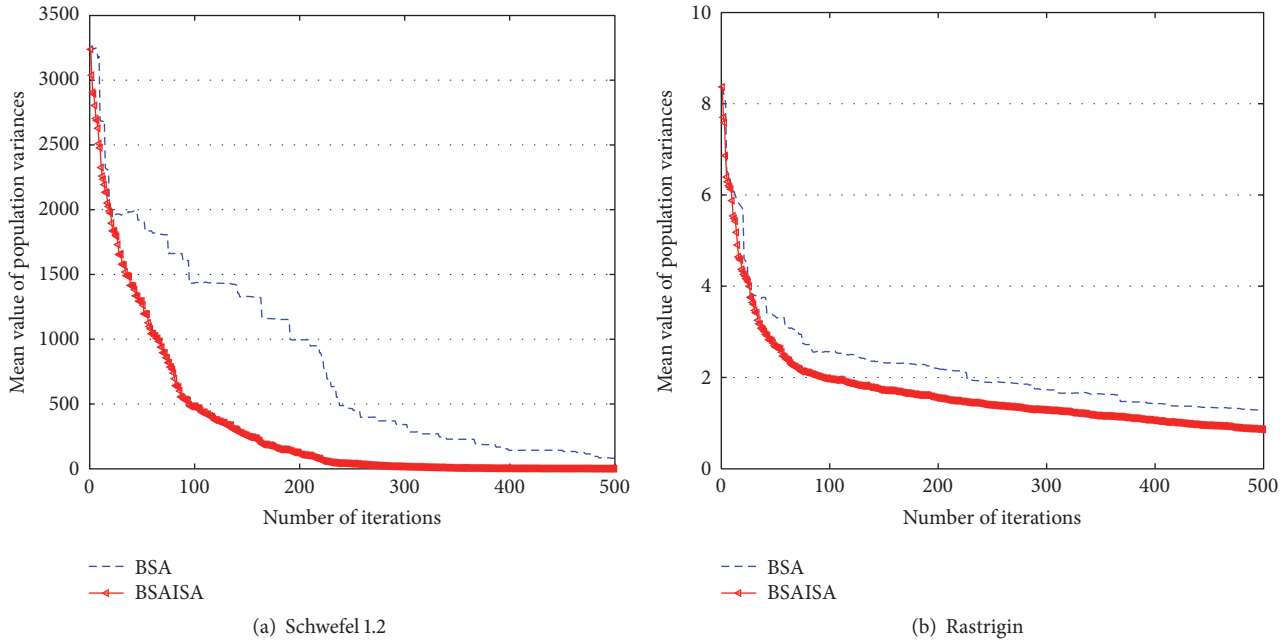


Figure 2: The mean values of population variances versus number of iterations using BSA and BSAISA for two functions.

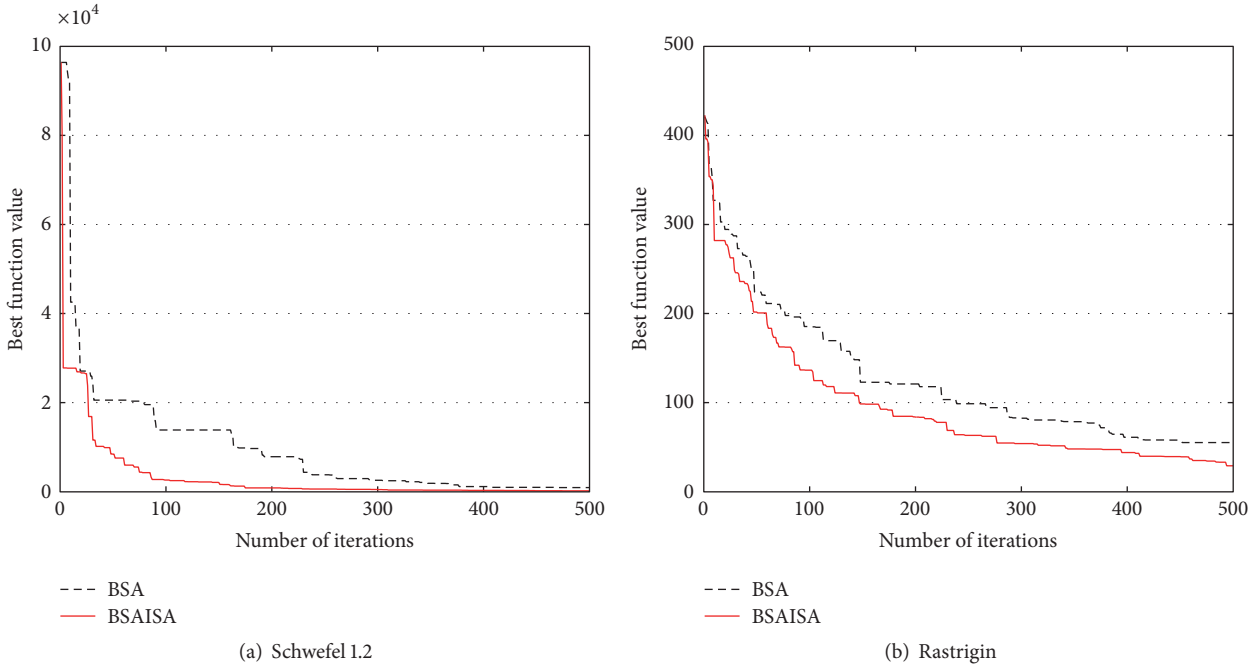


Figure 3: Convergence curves of BSA and BSAISA for two functions.

local region. Therefore, it can be concluded that BSAISA can adaptively control the amplitude of population mutation to change its local exploitation capacity. This may improve the convergence speed of the algorithm. Moreover, the modified F does not introduce extra parameters, so it does not increase the sensitivity of BSAISA to the parameters.

3.4. A Self-Adaptive ε-Constrained Method for Handling Constraints. In general, a constrained optimization problem can be mathematically formulated as a minimization problem, as follows:

$$\begin{aligned} \min \quad & f(X) \\ \text{subject to:} \quad & g_i(X) \le 0, \quad i = 1, 2, \ldots, m, \\ & h_j(X) = 0, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{7}$$

where $X = (x_1, x_2, \ldots, x_D) \in R^D$ is a D-dimensional vector, m is the total number of inequality constraints, and n is the total number of equality constraints. The equality constraints are transformed into inequality constraints by using $|h_j(X)| - \delta \leq 0$, where δ is a very small degree of violation, and $\delta = 1E-4$ in this paper. The maximization problems are transformed into minimization problems using $-f(X)$. The constraint violation $\varphi(X)$ is given by

$$\varphi(X) = \sum_{i=1}^{m} \max(0, g_i(X)) + \sum_{j=1}^{n} \max\left(0, \left|h_j(X)\right| - \delta\right). \quad (8)$$

Several constraint-handling methods have been proposed previously, where the five most commonly used methods comprise penalty functions, feasibility and dominance rules (FAD), stochastic ranking, ε-constrained methods, and multiobjectives concepts. Among these five methods, the ε-constrained method is relatively effective and used widely. Zhang et al. [62] proposed a self-adaptive ε-constrained method (SAε) to combine with the basic BSA for constrained problems. It has been verified that the SAε has a stronger search efficiency and convergence than the fixed ε-constrained method and FAD. In this paper, the SAε is used to combine with BSAISA for constrained optimization problems, which comprises the following two rules: (1) if the constraint violations of two solutions are smaller than a given ε value or two solutions have the same constraint violations, the solution with a better objective function value is preferred and (2) if not, the solution with a smaller constraint violation is preferred. SAε could be expressed by the following equations:

$$\text{the better one} = f(X_1) \begin{cases} f(X_1) < f(X_2), & \text{if } \varphi(X_1),\ \varphi(X_2) \leq \varepsilon, \\ f(X_1) < f(X_2), & \text{if } \varphi(X_1) = \varphi(X_2), \\ \varphi(X_1) \leq \varphi(X_2), & \text{otherwise}, \end{cases} \quad (9)$$

where ε is a positive value that represents a tolerance related to constraint violation. The self-adaptive ε value is formulated as the following equation:

$$\varepsilon_0 = \varphi\left(P_0^{\theta}\right), \quad (10)$$

$$\varepsilon_1(0) = \varepsilon_0, \quad (11)$$

$$\varepsilon_2(t) = \begin{cases} \varphi\left(T_t^{\theta}\right), & \text{if } \varepsilon_0 > \text{Th1}, \\ \varepsilon_0, & \text{else}, \end{cases} \quad (12)$$

$$\varepsilon_1(t) = \begin{cases} \varepsilon_2(t), & \text{if } \varepsilon_2(t) > \text{Th2},\ \varepsilon_2(t) < \varepsilon_1(t-1), \\ \varepsilon_1(t-1), & \text{else}, \end{cases} \quad (13)$$

$$\varepsilon(t) = \begin{cases} \varepsilon_1(t)\left(1 - \dfrac{t}{T_c}\right)^{\text{cp}}, & \text{if } t \leq T_c \\ 0, & \text{else}, \end{cases} \quad (14)$$

where t is the number of the current iterations. $\varphi(P_0^{\theta})$ is the constraint violation of the top θth individual in the initial population. $\varphi(T_t^{\theta})$ is the constraint violation of the top θth individual in the trial population at the current iteration. cp and T_c are control parameters. Th1 and Th2 are threshold values. εt is related to the iteration t and functions $\varepsilon_1(t)$ and $\varepsilon_2(t)$.

Firstly, ε_0 is set as $\varphi(P_0^{\theta})$. If the initial value ε_0 is bigger than Th1, $\varphi(T_t^{\theta})$ will be assigned to $\varepsilon_2(t)$; otherwise ε_0 will be assigned to $\varepsilon_2(t)$. Then, if $\varepsilon_2(t) < \varepsilon_1(t-1)$ and $\varepsilon_2(t)$ is bigger than Th2, $\varepsilon_2(t)$ will be assigned to $\varepsilon_1(t)$; otherwise $\varepsilon_1(t-1)$ will be assigned to $\varepsilon_1(t)$. Finally, εt is updated as (14). The detailed information of SAε can be acquired from [62], and the related parameter settings of SAε (the same as [62]) are presented in Table 2.

To illustrate the changing trend of the self-adaptive ε value vividly, BSAISA with SAε is used to solve a well-known benchmark constrained function G10 in [61]. The related parameters are set as $N = 30$, $\theta = 0.3N$, cp $= 5$, $T_c = 2333$, Th1 $= 10$, and Th2 $= 2$. The changing trend of ε value is shown in Figure 4. Three sampling points, that is, $\varepsilon(500) = 0.6033$, $\varepsilon(1000) = 0.1227$, and $\varepsilon(2000) = 1.194E-4$, are marked in Figure 4. As shown in Figure 4, it can be observed that ε value declines very fast at first. After it is smaller than about 2, it declines as an exponential way. This changing trend of ε value could help algorithm to sufficiently search infeasible domains near feasible domains.

The pseudocode for BSAISA is showed in Pseudocode 1. In Pseudocode 1, the modified adaptive F is shown in lines (14)–(16). When BSAISA deals with constrained optimization problems, the code in line (8) and line (40) in Pseudocode 1 should consider objective function value and constraint violation simultaneously, and SAε is applied to choose a better solution or best solution in line (42) and lines (47)-(48).

4. Experimental Studies

In this section, two sets of simulation experiments were executed to evaluate the effectiveness of the proposed BSAISA. The first experiment set performed on 13 well-known benchmark constrained functions taken from [63] (see Appendix A). These thirteen benchmarks contain different properties as shown in Table 3, including the number of variables (D), objective function types, the feasibility ratio (ρ), constraint types and number, and the number of active constraints in the optimum solution. The second experiment

Table 2: The parameter setting of SAε.

$T_c = 0.2 * T_{\max}$	$\theta = 0.3 * N$	cp = 5	Th1 = 10	Th2 = 2

Note. "$T_{\max}$" means the maximum iterations; "N" means populations.

Table 3: Characters of the 13 benchmark functions.

Fun.	D	Type	ρ (%)	LI	NI	LE	NE	Active
g01	13	Quadratic	0.0003	9	0	0	0	6
g02	20	Nonlinear	99.9973	1	1	0	0	1
g03	10	Nonlinear	0.0000	0	0	0	1	1
g04	5	Quadratic	27.0079	0	6	0	0	2
g05	4	Nonlinear	0.0000	2	0	0	3	3
g06	2	Nonlinear	0.0057	0	2	0	0	2
g07	10	Quadratic	0.0003	3	5	0	0	6
g08	2	Nonlinear	0.8581	0	2	0	0	0
g09	7	Nonlinear	0.5199	0	4	0	0	2
g10	8	Linear	0.0020	3	3	0	0	3
g11	2	Quadratic	0.0973	0	0	0	1	1
g12	3	Quadratic	4.7679	0	9^3	0	0	0
g13	5	Nonlinear	0.0000	0	0	1	2	3

Note. "D" is the number of variables. "ρ" represents feasibility ratio. "LI," "NI," "LE," and "NE" represent linear inequality, nonlinear inequality, linear equality, and nonlinear equality, respectively. "Active" represents the number of active constraints at the global optimum.

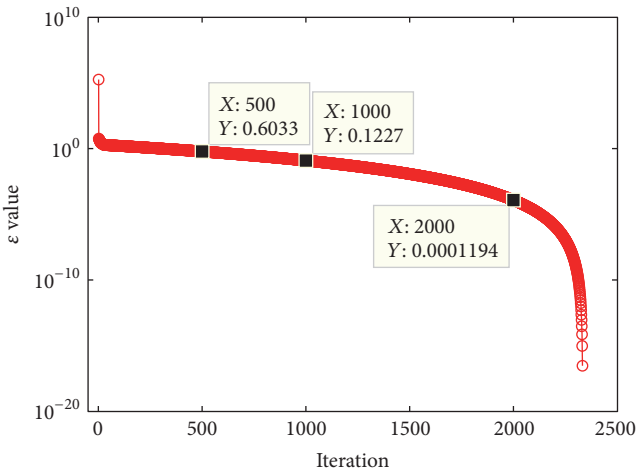


Figure 4: Plot of ε value with iteration.

is conducted on 5 engineering constrained optimization problems chosen from [64] (see Appendix B). These five problems are the three-bar truss design problem (TTP), pressure vessel design problem (PVP), tension/compression spring design problem (TCSP), welded beam design problem (WBP), and speed reducer design problem (SRP), respectively. These engineering problems include objective functions and constraints of various types and natures (quadratic, cubic, polynomial, and nonlinear) with various number of design variables (continuous, integer, mixed, and discrete).

The recorded experimental results include the best function value (Best), the worst function value (Worst), the mean function value (Mean), the standard deviation (Std), the best solution (variables of best function value), the corresponding constraint value, and the number of function evaluations (FEs). The number of function evaluations can be considered as a convergence rate or a computational cost.

In order to evaluate the performance of BSAISA in terms of convergence speed, the FEs are considered as the best FEs corresponding to the obtained best solution in this paper. The calculation of FEs are the product of population sizes (N) and the number of iterations (Ibest) at which the best function value is first obtained (i.e., FEs = $N * $ Ibest). For example, if 2500 is the maximum number of iterations for one minimization problem, $f(1999) = 0.0126653$, $f(2000) = 0.0126652$, $f(2500) = f(2000) = 0.0126652$, the Ibest value should be 2000. However, BSAISA needs to evaluate the initial historical population (oldP), so its actual FEs should be plus N (i.e., FEs = $N * $ Ibest + N).

4.1. Parameter Settings. For the first experiment, the main parameters for 13 benchmark constrained functions are the same as the following: population size (N) is set as 30; the maximum number of iterations ($T_{\max}$) is set as 11665. Therefore, BSAISA's maximum number of function evaluations (MFEs) should equal to 34,9980 (nearly 35,0000). The 13 benchmarks were executed by using 30 independent runs.

For the 5 real-world engineering design problems, we use slightly different parameter settings since each problem has different natures, that is, TTP ($N = 20$, $T_{\max} = 1000$), PVP ($N = 20$, $T_{\max} = 3000$), TCSP ($N = 20$, $T_{\max} = 3000$), WBP ($N = 20$, $T_{\max} = 3000$), SRP ($N = 20$, $T_{\max} = 2000$). The 6 engineering problems were performed using 50 independent runs.

The user parameters of all experiments are presented in Table 4. Termination condition may be the maximum

Input: ObjFun, N, D, max-iteration, mix-rate, $\text{low}_{1:D}$, $\text{up}_{1:D}$
Output: globalminimum, globalminimizer
//rand $\sim U(0,1)$, randn $\sim N(0,1)$, w = randint($\cdot$), randint $\sim U(\cdot)$ $w \in \{1,2,\ldots,\cdot\}$
(1) function BSAISA(ObjFun, D, N, max-iteration, low, up) // **Initialization**
(2) globalminimum = inf
(3) **for** i **from** 1 **to** N **do**
(4) **for** j **from** 1 **to** D **do**
(5) $P_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$ // Initialization of population P
(6) $\text{oldP}_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$ // Initialization of oldP
(7) **end**
∗**(8)** fitnessP_i = ObjFun(P_i) fitnessoldP_i = ObjFun(oldP_i) // Initial-fitness values of P and oldP
(9) **end**
(10) **for** *iteration* **from** 1 **to** *max-iteration* **do** // **Selection-I**
(11) **if** $(a < b \mid a, b \sim U(0,1))$ **then** oldP $\coloneqq P$ **end**
(12) oldP $\coloneqq$ permuting(oldP)
(13) **Generation of Trail-Population**
•(14) $\Delta I_i = \text{abs}(\text{ObjFun}(P_i) - \text{ObjFun}(\text{oldP}_i))$ // **Modified F based on SA**
•(15) G = iteration // **Modified F based on SA**
•(16) $F_i \sim N\left(\exp\left(-\dfrac{1/\Delta I_i}{1/G}\right), 1\right)$ // **Modified F based on SA**
(17) $M = P + F \cdot (\text{oldP} - P)$ // **Mutation**
(18) $\text{map}_{1:N,1:D} = 1$ // Initial-map is an N-by-D matrix of ones
(19) **if** $(c < d \mid c, d \sim U(0,1))$ **then** // **Crossover**
(20) **for** i **from** 1 **to** N **do**
(21) $\text{map}_{i,u(1:\lceil \text{mix-rate} \cdot \text{rand} \cdot D \rceil)} = 0 \mid u = \text{permuting}(1,2,3,\ldots,D)$
(22) **end**
(23) **else**
(24) **for** i **from** 1 **to** N **do**, $\text{map}_{i,\text{randi}(D)} = 0$, **end**
(25) **end**
(26) $T \coloneqq M$ //Generation of Trial Population, T
(27) **for** i **from** 1 **to** N **do**
(28) **for** j **from** 1 **to** D **do**
(29) **if** $\text{map}_{i,j} = 1$ **then** $T_{i,j} \coloneqq P_{i,j}$
(30) **end**
(31) **end**
(32) **for** i **from** 1 **to** N **do**
(33) **for** j **from** 1 **to** D **do**
(34) **if** $(T_{i,j} < \text{low}_{i,j})$ **or** $(T_{i,j} > \text{up}_{i,j})$ **then**
(35) $T_{i,j} = \text{rand} \cdot (\text{up}_j - \text{low}_j) + \text{low}_j$
(36) **end**
(37) **end**
(38) **end**
(39) **end**
∗**(40)** fitness T_i = ObjFun(T_j) // **Seletion-II**
(41) **for** i **from** 1 **to** N **do**
∗**(42)** **if** fitness $T_i <$ fitness P_i **then**
(43) fitness P_i = fitness T_i
(44) $P_i = T_i$
(45) **end**
(46) **end**
∗**(47)** fitness $P_{\text{best}} = \min(\text{fitness } P) \mid \text{best} \in \{1,2,3,\ldots,N\}$
∗**(48)** **if** fitness $P_{\text{best}} <$ globalminimum **then** // Export globalminimum and globalminimizer
(49) globalminimum $\coloneqq$ fitness P_{best}
(50) globalminimizer $\coloneqq P_{\text{best}}$
(51) **end**
(52) **end**

Pseudocode 1: Pseudocode of BSAISA.

Table 4: User parameters used for all experiments.

Problem	G01–G13	TTP	PVP	TCSP	WBP	SRP
N	30	20	20	20	20	20
$T_{\max}$	11665	1000	3000	3000	3000	2000
Runs	30	50	50	50	50	50

Note. The expression "runs" denotes the number of independent runs.

number of iterations, CPU time, or an allowable tolerance value between the last result and the best known function value for most metaheuristics. In this paper, the maximum number of iterations is considered as termination condition. All experiments were conducted on a computer with a Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz and 4 GB RAM.

4.2. Simulation on Constrained Benchmark Problems. In this section, BSAISA and BSA are performed on the 13 benchmarks simultaneously. Their statistical results obtained from 30 independent runs are listed in Tables 5 and 6, including the best known function value (Best Known) and the obtained Best/Mean/Worst/Std values as well as the FEs. The best known values for all the test problems are derived from [62]. The best known values found by algorithms are highlighted in *bold*. From Tables 5 and 6, it can be seen that BSAISA is able to find the known optimal function values for G01, G04, G05, G06, G08, G09, G011, G012, and G013; however, BSA fails to find the best known function value on G09 and G13. For the rest of the functions, BSAISA obtains the results very close to the best known function values. Moreover, BSAISA requires fewer FEs than BSA on G01, G03, G04, G05, G06, G07, G08, G09, G011, and G012. Although the FEs of BSAISA are slightly worse than that of BSA for G02, G10, and G13, BSAISA finds more accurate best function values than BSA for G02, G03, G07, G09, G10, and G13.

To further compare BSAISA and BSA, the function value convergence curves of 13 functions that have significant differences have been plotted, as shown in Figures 5 and 6. The horizontal axis represents the number of iterations, while the vertical axis represents the difference of the objective function value and the best known value. For G02, G04, G06, G08, G11, and G12, it is obvious that the convergence speed of BSAISA is faster than that of BSA, and its best function value is also better than that of BSA. For G01, G03, G05, G07, G09, G10, and G13, it can be observed that the objective function values of BSAISA and BSA fluctuate in the early iterations, and they decrease as the number of iterations increases during the middle and late stages. This illustrates that both the algorithms are able to escape the local optimum under different degrees, and the convergence curve of BSAISA drops still faster than that of BSA. Therefore, the experiment results demonstrate that the convergence speed of the basic BSA is improved with our modified F.

In order to further verify the competitiveness of BSAISA in aspect of convergence speed, we compared BSAISA with some classic and state-of-the-art approaches in terms of best function value and function evaluations. The best function value and the corresponding FEs of each algorithm on 13 benchmarks are presented in Table 7, where the optimal results are in *bold* on each function. These compared algorithms are listed below:

(1) Stochastic ranking (SR) [63]

(2) Filter simulated annealing (FSA) [65]

(3) Cultured differential evolution (CDE) [66]

(4) Agent based memetic algorithm (AMA) [64]

(5) Modified artificial bee colony (MABC) algorithm [67]

(6) Rough penalty genetic algorithm (RPGA) [68]

(7) BSA combined self-adaptive ε constrained method (BSA-SAε) [62].

To compare these algorithms synthetically, a simple evaluation mechanism is used. It can be explained as the best function value (Best) is preferred, and the function evaluations (FEs) are secondary. More specifically, (1) if one algorithm has a better *Best* than those of others on a function, there is no need to consider *FEs* and the algorithm is superior to other algorithms on this function. (2) If two or more algorithms have found the optimal *Best* on a function, the algorithm with the lowest *FEs* is considered as the winner on the function. (3) Record the number of winners and the number of the optimal function values for each algorithm on the set of benchmarks, and then give the sort for all algorithms.

From Table 7, it can be observed that the rank of these 8 algorithms is as follows: BSAISA, CDE, BSA-SAε, MABC, SR, RPGA, FSA, and AMA. Among the 13 benchmarks, BSAISA wins on 6 functions and it is able to find the optimal values of 10 functions. This is better than all other algorithms, thus BSAISA ranks the first. The second algorithm CDE performs better on G02, G07, G09, and G10 than BSAISA but worse on G01, G03, G08, G11, G12, and G13. BSA-SAε obtains the optimal function values of 10 functions but requires more function evaluations than BSAISA and CDE, so it should rank the third. MABC ranks the fourth. It obtains the optimal function values of 7 functions, which are fewer in number than those of the former three algorithms. Both SR and RPGA have found the same number of the optimal function values, while the former is the winner on G04, so SR is slightly better than RPGA. As for the last two algorithms, FSA and AMA just perform well on three functions, while FSA is the winner on G06, so FSA is slightly better than AMA.

Based on the above comparison, it can be concluded that BSAISA is effective and competitive in terms of convergence speed.

4.3. Simulation on Engineering Design Problems. In order to assess the optimization performance of BSAISA in real-world engineering constrained optimization problems, 5 well-known engineering constrained design problems including three-bar truss design, pressure vessel design, tension/compression spring design, welded beam design, and speed reducer design are considered in the second experiment.

Table 5: The statistical results of BSAISA for 13 constrained benchmarks.

Fun.	Known optimal	Best	Mean	Worst	Std	FEs
G01	−15	**−15**	−15	−15.000000	$8.08E-16$	84,630
G02	−0.803619	−0.803599	−0.787688	−0.758565	$1.14E-02$	349,500
G03	−1.000500	−1.000498	−1.000481	−1.000441	$1.35E-05$	58,560
G04	−30665.538672	**−30665.538672**	−30665.538672	−30665.538672	$1.09E-11$	121,650
G05	5126.496714	**5126.496714**	5126.496714	5126.496714	$5.85E-13$	238,410
G06	−6961.813876	**−6961.813876**	−6961.813876	−6961.813876	$1.85E-12$	89,550
G07	24.306209	24.307381	24.400881	24.758205	$1.02E-01$	15,060
G08	−0.0958250	**−0.0958250**	−0.086683	−0.027263	$2.37E-02$	30,930
G09	680.630057	**680.630057**	680.633025	680.680043	$9.11E-03$	347,760
G10	7049.248021	7049.249056	7081.241789	7326.853581	$6.24E+01$	346,980
G11	0.749900	**0.749900**	0.749900	0.749900	$1.13E-16$	87,870
G12	−1	**−1**	−1	−1	0	5430
G13	0.0539415	**0.0539415**	0.1030000	0.4594033	$9.80E-02$	349,800

Note. "Known optimal" denotes the best known function values in the literatures. "*Bold*" means the algorithm has found the best known function values. The same as Table 6.

Table 6: The statistical results of the basic BSA on 13 constrained benchmarks.

Fun.	Known optimal	Best	Mean	Worst	Std	FEs
G01	−15	**−15**	−15	−15.000000	$6.60E-16$	99,300
G02	−0.803619	−0.803255	−0.792760	−0.749326	$9.10E-03$	344,580
G03	−1.000500	−1.000488	−0.998905	−0.990600	$2.66E-03$	348,960
G04	−30665.538672	**−30665.538672**	−30665.538672	−30665.538672	$1.11E-11$	272,040
G05	5126.496714	**5126.496714**	5144.041363	5275.384724	$3.99E+01$	299,220
G06	−6961.813876	**−6961.813876**	−6961.813876	−6961.813876	$1.85E-12$	111,450
G07	24.306209	24.307607	24.344626	24.399896	$1.93E-02$	347,250
G08	−0.0958250	**−0.0958250**	−0.0958250	−0.0958250	$2.82E-17$	73,440
G09	680.630057	680.630058	680.630352	680.632400	$5.63E-04$	348,900
G10	7049.248021	7049.278543	7053.573853	7080.192700	$7.28E+00$	340,020
G11	0.749900	**0.749900**	0.749900	0.749900	$1.13E-16$	113,250
G12	−1	**−1**	−1	−1	0	13,590
G13	0.0539415	0.0539420	0.1816986	0.5477657	$1.50E-01$	347,400

4.3.1. Three-Bar Truss Design Problem (TTP). The three-bar truss problem is one of the engineering minimization test problems for constrained algorithms. The best feasible solution is obtained by BSAISA at x = (0.788675, 0.408248) with the objective function value $f(x)$ = 263.895843 using 8940 FEs. The comparison of the best solutions obtained from BSAISA, BSA, differential evolution with dynamic stochastic selection (DEDS) [69], hybrid evolutionary algorithm (HEAA) [70], hybrid particle swarm optimization with differential evolution (POS-DE) [71], differential evolution with level comparison (DELC) [72], and mine blast algorithm (MBA) [73] is presented in Table 8. Their statistical results are listed in Table 9.

From Tables 8 and 9, BSAISA, BSA, DEDS, HEAA, PSO-DE, and DELC all reach the best solution with the corresponding function value equal to 263.895843 except MBA with 263.895852. However, BSAISA requires the lowest FEs (only 8940) among all algorithms. Its Std value is better than BSA, DEDS, HEAA, PSO-DE, and MBA except DELC. These comparative results indicate that BSAISA outperforms other algorithms in terms of computational cost and robustness for this problem.

Figure 7 depicts the convergence curves of BSAISA and BSA for the three-bar truss design problem, where the value of $F(x^*)$ on the vertical axis equals 263.895843. As shown in Figure 7, BSA achieves the global optimum at about 700 iterations, while BSAISA only reaches the global optimum at about 400 iterations. It can be concluded that the convergence speed of BSAISA is faster than that of BSA for this problem.

4.3.2. Pressure Vessel Design Problem (PVP). The pressure vessel design problem has a nonlinear objective function with three linear and one nonlinear inequality constraints and two discrete and two continuous design variables. The values of the two discrete variables (x_1, x_2) should be the integer multiples of 0.0625. The best feasible solution is obtained by BSAISA at x = (0.8750, 0.4375, 42.0984, 176.6366) with the objective function value $f(x)$ = 6059.7143 using 31,960 FEs.

Table 7: Comparison of the best values and FEs obtained by BSAISA and other algorithms.

Alg.	BSAISA	SR	FSA	CDE	AMA	MABC	RPGA	BSA-SAε
Fun.	Best (FEs)	Best (FEs)	Best (FEs)	Best (FEs)	Best (FEs)	Best (FEs)	Best (FEs)	Best (FEs)
G01	**−15** (**84,630**)	**−15.000** (148,200)	−14.993316 (205,748)	**−15.000000** (100,100)	**−15.000** (350,000)	**−15.000** (350,000)	**−15.000** (350,000)	**−15.000000** (350,000)
G02	**−0.8036** (349,500)	−0.8035 (217,200)	−0.7549 (227,832)	**−0.8036** (**100,100**)	−0.8035 (350,000)	**−0.8036** (350,000)	**−0.8036** (350,000)	**−0.8036** (350,000)
G03	**−1.000498** (**58,560**)	−1.000 (229,200)	−1.0000015 (314,938)	−0.995413 (100,100)	−1.000 (350,000)	−1.000 (350,000)	−1.000 (350,000)	**−1.000498** (350,000)
G04	**−30665.539** (121,650)	**−30665.539** (**88,200**)	−30665.538 (86,154)	**−30665.539** (100,100)	−30665.538 (350,000)	**−30665.539** (350,000)	**−30665.539** (350,000)	**−30665.539** (350,000)
G05	5126.497 (238,410)	5126.497 (51,600)	5126.4981 (47,661)	5126.571 (100,100)	5126.512 (350,000)	**5126.487** (**350,000**)	5126.544 (350,000)	5126.497 (350,000)
G06	**−6961.814** (89,550)	**−6961.814** (118,000)	**−6961.814** (**44,538**)	**−6961.814** (100,100)	−6961.807 (350,000)	**−6961.814** (350,000)	**−6961.814** (350,000)	**−6961.814** (350,000)
G07	24.307 (15,060)	24.307 (143,000)	24.311 (404,501)	**24.306** (**100,100**)	24.315 (350,000)	24.324 (350,000)	24.333 (350,000)	**24.306** (350,000)
G08	**−0.095825** (**30,930**)	**−0.095825** (76,200)	**−0.095825** (56,476)	**−0.095825** (100,100)	**−0.095825** (350,000)	**−0.095825** (350,000)	**−0.095825** (350,000)	**−0.095825** (350,000)
G09	**680.630057** (347,760)	680.630 (111,400)	680.63008 (324,569)	**680.630057** (**100,100**)	680.645 (350,000)	680.631 (350,000)	680.631 (350,000)	680.6301 (350,000)
G10	7049.249 (346,980)	7054.316 (128,400)	7059.864 (243,520)	**7049.248** (**100,100**)	7281.957 (350,000)	7058.823 (350,000)	7049.861 (350,000)	7049.278 (350,000)
G11	**0.749900** (**87,870**)	0.750 (11,400)	0.749999 (23,722)	**0.749900** (100,100)	0.750 (350,000)	0.750 (350,000)	0.749 (350,000)	**0.749900** (350,000)
G12	**−1** (**5430**)	**−1.000000** (16,400)	**−1.000000** (59,355)	**−1.000000** (100,100)	**−1.000** (350,000)	**−1.000** (350,000)	NA	**−1.000000** (350,000)
G13	**0.0539415** (**349,800**)	0.053957 (69,800)	0.0539498 (120,268)	0.056180 (100,100)	0.053947 (350,000)	0.757 (350,000)	NA	**0.0539415** (350,000)
Nu.	6 + 10	1 + 5	1 + 3	4 + 10	0 + 3	1 + 7	0 + 5	0 + 10
RK	1	5	7	2	8	4	6	3

Note. "NA" means not available. The same as Tables 8, 10, 11, 12, 13, 14, 15, and 17. "RK" represents the comprehensive ranking of each algorithm on the set of benchmarks. "Nu." represents the sum of the number of winners and the number of the optimal function values for each algorithm on the set of benchmarks.

Table 8: Comparison of best solutions for the three-bar truss design problem.

Method	DEDS	HEAA	PSO-DE	DELC	MBA	BSA	BSAISA
X_1	0.788675	0.788680	0.788675	0.788675	0.788675	0.788675	0.788675
X_2	0.408248	0.408234	0.408248	0.408248	0.408560	0.408248	0.408248
$g_1(X)$	$1.77E-08$	NA	$-5.29E-11$	NA	$-5.29E-11$	$-3.23E-12$	0
$g_2(X)$	−1.464102	NA	−1.463748	NA	−1.463748	−1.464102	−1.464102
$g_3(X)$	−0.535898	NA	−0.536252	NA	−0.536252	−0.535898	−0.535898
$f(X)$	**263.895843**	**263.895843**	**263.895843**	**263.895843**	263.895852	**263.895843**	**263.895843**

Table 9: Comparison of statistical results for the three-bar truss design problem.

Method	Worst	Mean	Best	Std	FEs
DEDS	263.895849	263.895843	**263.895843**	$9.7E-07$	15,000
HEAA	263.896099	263.895865	**263.895843**	$4.9E-05$	15,000
PSO-DE	263.895843	263.895843	**263.895843**	$4.5E-10$	17,600
DELC	263.895843	263.895843	**263.895843**	$4.3E-14$	10,000
MBA	263.915983	263.897996	263.895852	$3.93E-03$	13,280
BSA	263.895845	263.895843	**263.895843**	$2.64E-07$	13,720
BSAISA	263.895843	263.895843	**263.895843**	$5.75E-13$	**8940**

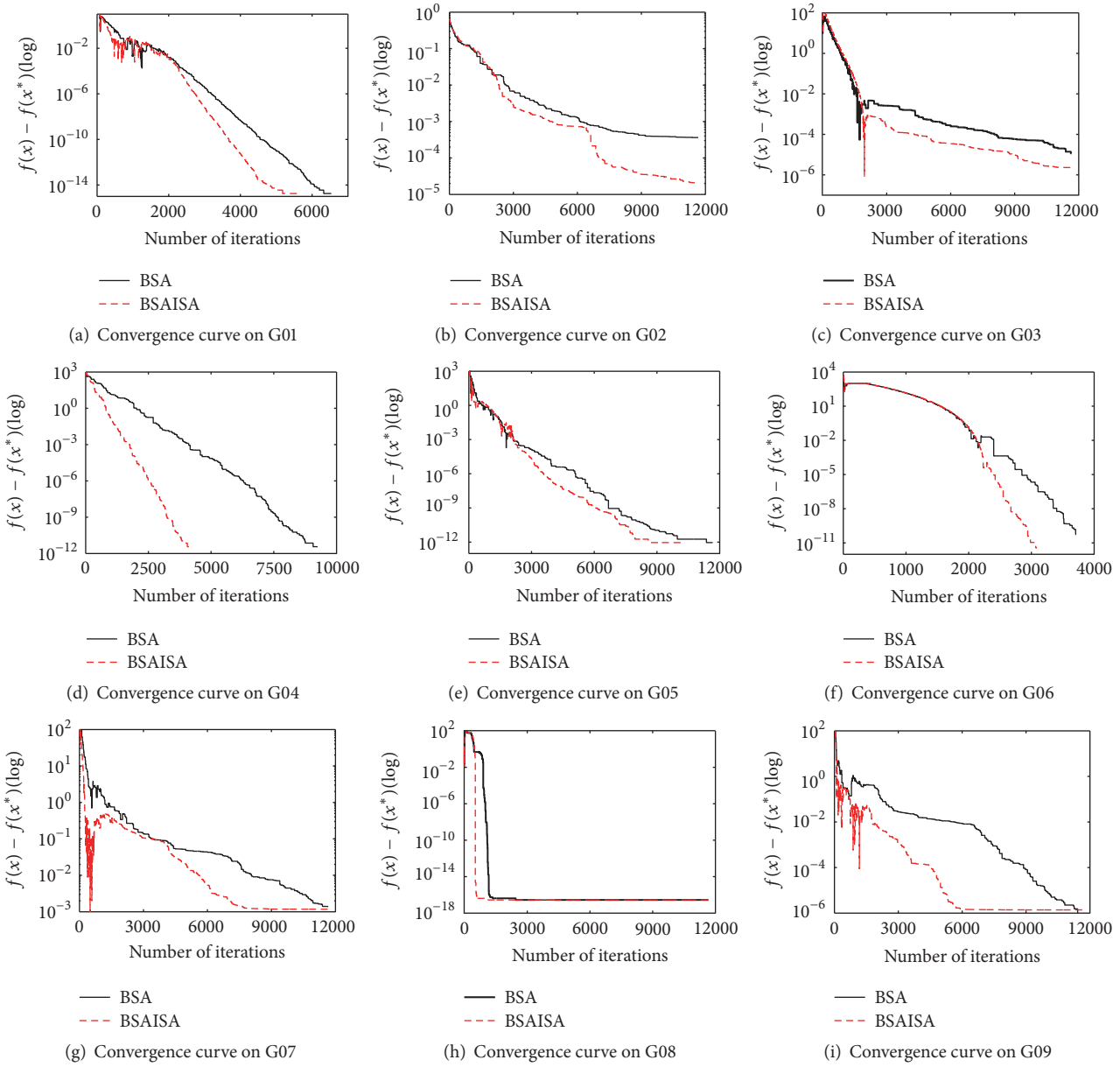


Figure 5: The convergence curves of the first 9 functions by BSAISA and BSA.

For this problem, BSAISA is compared with nine algorithms: BSA, BSA-SAε [62], DELC, POS-DE, genetic algorithms based on dominance tournament selection (GA-DT) [73], modified differential evolution (MDE) [74], coevolutionary particle swarm optimization (CPSO) [75], hybrid particle swarm optimization (HPSO) [76], and artificial bee colony algorithm (ABC) [77]. The comparison of the best solutions obtained by BSAISA and other reported algorithms is presented in Table 10. The statistical results of various algorithms are listed in Table 11.

As shown Table 10, the obtained solution sets of all algorithms satisfy the constraints for this problem. BSAISA, BSA-SAε, ABC, DELC, and HPSO find the same considerable good objective function value 6059.7143, which is slightly worse than MDE's function value 6059.7143. It is worth mentioning that MBA's best solution was obtained at $x = (0.7802, 0.3856, 40.4292, 198.4964)$ with $f(x) = 5889.3216$ and the corresponding constraint values equal to $g_i(x) = (0, 0, -86.3645, -41.5035)$ in [78]. Though MBA finds a far better function value than that of MDE, its obtained variables (i.e., 0.7802 and 0.3856) are not integer multiples of 0.0625. So they are not listed in Table 10 to ensure a fair comparison. From Table 11, except for MDE with the function value of 6059.7016, BSAISA offers better function value results compared to GA-DT, CPSO, ABC, and BSA. Besides that, BSAISA is far superior to other algorithms in terms of FEs. Unfortunately, the obtained Std value of BSAISA is relatively poor compared with others for this problem.

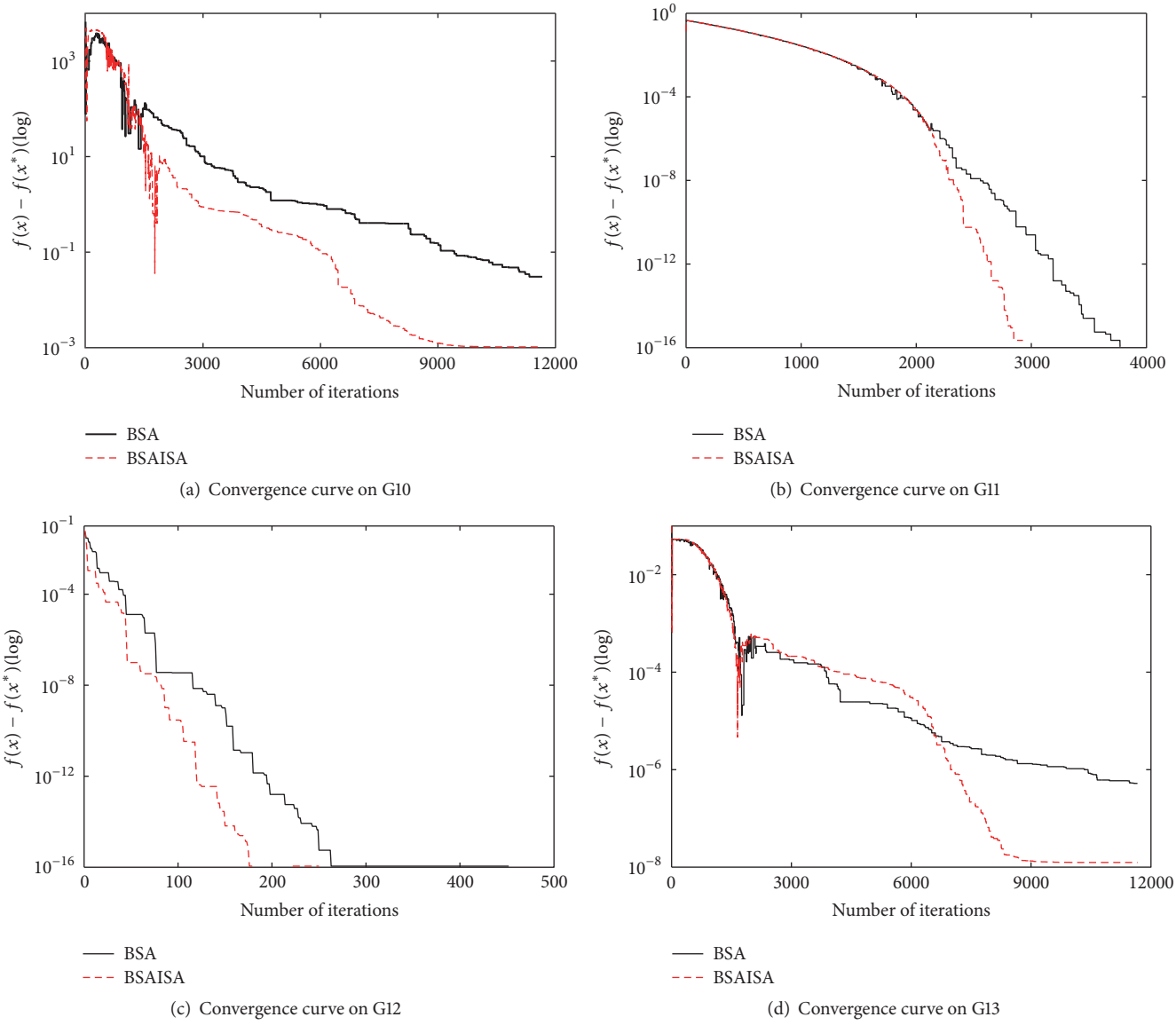


Figure 6: The convergence curves of the latter 4 functions by BSAISA and BSA.

Table 10: Comparison of best solutions for the pressure vessel design problem.

Method	GA-DT	MDE	CPSO	HPSO	DELC	ABC	BSA-SAε	BSA	BSAISA
X_1	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125	0.8125
X_2	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375	0.4375
X_3	42.0974	42.0984	42.0913	42.0984	42.0984	42.0984	42.0984	42.0984	42.0984
X_4	176.6540	176.6360	176.7465	176.6366	176.6366	176.6366	176.6366	176.6366	176.6366
$g_1(X)$	$-2.01E-03$	0	$-1.37E-06$	NA	NA	0	$-9.5E-10$	$-3.69E-08$	0
$g_2(X)$	$-3.58E-02$	-0.035881	$-3.59E-04$	NA	NA	-0.035881	$-3.59E-2$	-0.035881	-0.035881
$g_3(X)$	-24.7593	-0.0000	-118.7687	NA	NA	-0.000226	$-1.2E-4$	-0.095446	0
$g_4(X)$	-63.3460	-63.3639	-63.2535	NA	NA	-63.363	-63.363	-63.2842	-63.3634
$f(X)$	6059.9463	**6059.7017**	6061.0777	6059.7143	6059.7143	6059.7143	6059.7143	6059.7150	6059.7143

Table 11: Comparison of statistical results for the pressure vessel design problem.

Method	Worst	Mean	Best	Std	FEs
GA-DT	6469.3220	6177.2533	6059.9463	130.9297	80,000
MDE	6059.7017	6059.7017	**6059.7017**	$1.0E-12$	24,000
CPSO	6363.8041	6147.1332	6061.0777	86.45	30,000
HPSO	6288.6770	6099.9323	6059.7143	86.20	81,000
DELC	6059.7143	6059.7143	6059.7143	$2.1E-11$	30,000
PSO-DE	6059.7143	6059.7143	6059.7143	$1.0E-10$	42,100
ABC	NA	6245.3081	6059.7147	$2.05E+02$	30,000
BSA-SAε	6116.7804	6074.3682	6059.7143	$1.71E+01$	80,000
BSA	6771.5969	6221.2861	6059.7150	$2.03E+02$	60,000
BSAISA	7198.0054	6418.1935	6059.7143	$3.04E+02$	**16,320**

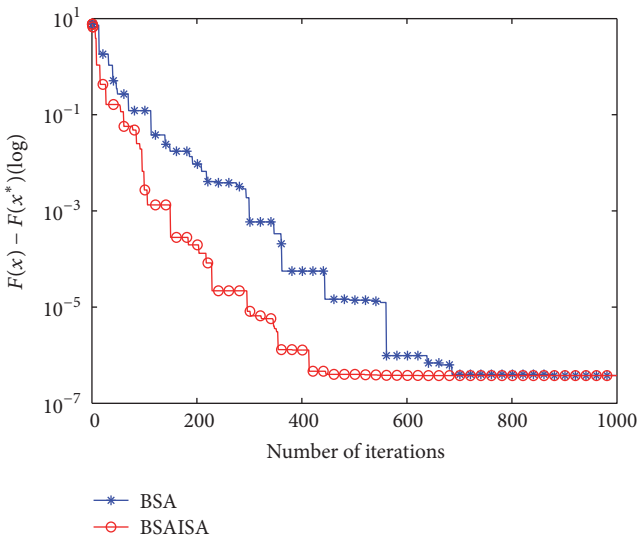


Figure 7: Convergence curves of BSAISA and BSA for the three-bar truss design problem.

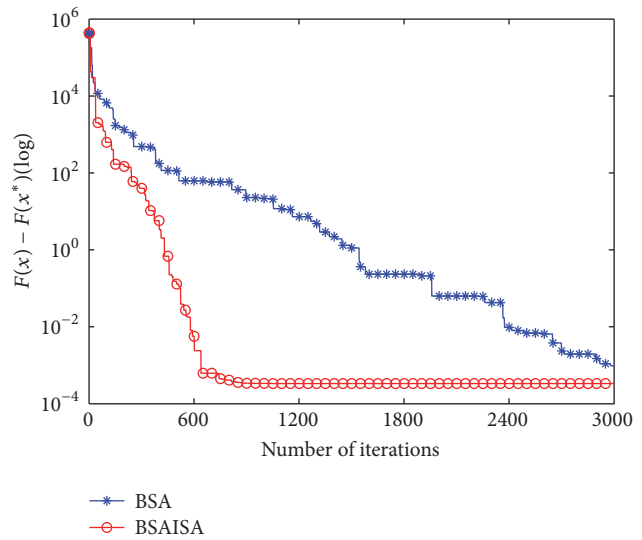


Figure 8: Convergence curves of BSAISA and BSA for the pressure vessel design problem.

Figure 8 describes the convergence curves of BSAISA and BSA for the pressure vessel design problem, where the value of $F(x^*)$ on the vertical axis equals 6059.7143. As shown in Figure 8, BSAISA is able to find the global optimum at about 800 iterations and obtains a far more accurate function value than that of BSA. Moreover, the convergence speed of BSAISA is much faster than that of BSA.

4.3.3. Tension Compression Spring Design Problem (TCSP). This design optimization problem has three continuous variables and four nonlinear inequality constraints. The best feasible solution is obtained by BSAISA at x = (0.051687, 0.356669, 11.291824) with $f(x)$ = 0.012665 using 9440 FEs. This problem has been solved by other methods as follows: GA-DT, MDE, CPSO, HPSO, DEDS, HEAA, DELC, POS-DE, ABC, MBA, BSA-SAε, and Social Spider Optimization (SSOC) [79]. The comparison of the best solutions obtained from various algorithms is presented in Table 12. Their statistical results are listed in Table 13.

From Tables 12 and 13, the vast majority of algorithms can find the best function value 0.012665 for this problem, while GA-DT and CPSO fail to find it. With regard to the computational cost (FEs), BSAISA only requires 9440 FEs when it reaches the global optimum, which is superior to all other algorithms except MBA with 7650 FEs. However, the Worst and Mean and Std values of BSAISA are better than those of MBA. Consequently, for this problem, it can be concluded that BSA has the obvious superiority in terms of FEs over all other algorithms except MBA. Moreover, BSAISA has a stronger robustness when compared with MBA alone.

Figure 9 depicts the convergence curves of BSAISA and BSA for the tension compression spring design problem, where the value of $F(x^*)$ on the vertical axis equals 0.012665. From Figure 9 it can be observed that both BSAISA and BSA fall into a local optimum in the early iterations but they are able to successfully escape from the local optimum. However, the convergence speed of BSAISA is obviously faster than that of BSA.

Table 12: Comparison of best solutions for the tension compression spring design problem.

Method	X_1	X_2	X_3	$g_1(X)$	$g_2(X)$	$g_3(X)$	$g_4(X)$	$f(X)$
GA-DT	0.051989	0.363965	10.890522	$-1.3E-05$	$-2.1E-05$	−4.061338	−0.722698	0.012681
MDE	0.051688	0.356692	11.290483	−0.000000	−0.000000	−4.053734	−0.727747	**0.012665**
CPSO	0.051728	0.357644	11.244543	$-8.45E-04$	$-1.26E-05$	−4.051300	−0.727090	0.012675
HPSO	0.051706	0.357126	11.265083	NA	NA	NA	NA	**0.012665**
DEDS	0.051689	0.356718	11.288965	NA	NA	NA	NA	**0.012665**
HEAA	0.051690	0.356729	11.288294	NA	NA	NA	NA	**0.012665**
DELC	0.051689	0.356718	11.288966	NA	NA	NA	NA	**0.012665**
ABC	0.051749	0.358179	11.203763	−0.000000	−0.000000	−4.056663	−0.726713	**0.012665**
MBA	0.051656	0.35594	11.344665	0	0	−4.052248	−0.728268	**0.012665**
SSOC	0.051689	0.356718	11.288965	NA	NA	NA	NA	**0.012665**
BSA-SAε	0.051989	0.356727	11.288425	$-7.70E-09$	$-3.30E-09$	−4.054	−0.728	**0.012665**
BSA	0.051694	0.356845	11.281488	$-1.05E-07$	$-1.77-08$	−4.054037	−0.727640	**0.012665**
BSAISA	0.051687	0.356669	11.291824	$-6.38E-10$	$-1.53E-09$	−4.053689	−0.727763	**0.012665**

Table 13: Comparison of statistical results for the tension compression spring design problem.

Method	Worst	Mean	Best	Std	FEs
GA-DT	0.012973	0.012742	0.012681	$5.90E-05$	80,000
MDE	0.012674	0.012666	**0.012665**	$2.0E-6$	24,000
CPSO	0.012924	0.012730	0.012675	$5.20E-05$	23,000
HPSO	0.012719	0.012707	**0.012665**	$1.58E-05$	81,000
DEDS	0.012738	0.012669	**0.012665**	$1.25E-05$	24,000
HEAA	0.012665	0.012665	**0.012665**	$1.4E-09$	24,000
DELC	0.012666	0.012665	**0.012665**	$1.3E-07$	20,000
PSO-DE	0.012665	0.012665	**0.012665**	$1.2E-08$	24,950
ABC	NA	0.012709	**0.012665**	0.012813	30,000
MBA	0.012900	0.012713	**0.012665**	$6.30E-05$	**7650**
SSOC	0.012868	0.012765	**0.012665**	$9.29E-05$	25,000
BSA-SAε	0.012666	0.012665	**0.012665**	$1.62E-07$	80,000
BSA	0.012669	0.012666	**0.012665**	$7.24E-07$	43,220
BSAISA	0.012668	0.012666	**0.012665**	$4.90E-07$	9440

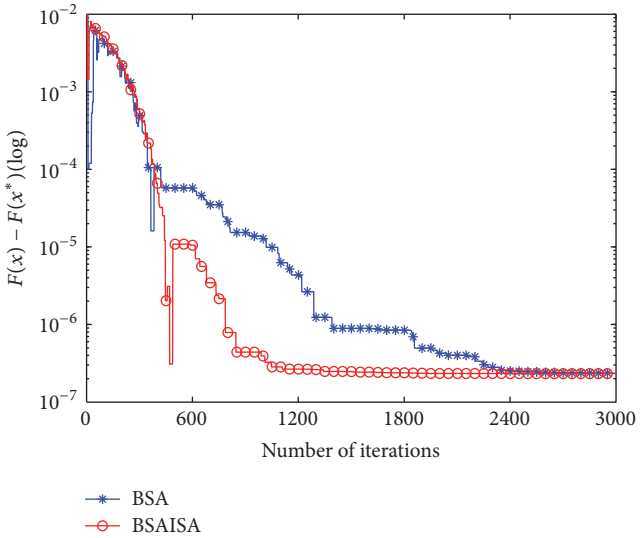


Figure 9: Convergence curves of BSAISA and BSA for the tension compression spring design problem.

4.3.4. Welded Beam Design Problem (WBP). The welded beam problem is a minimum cost problem with four continuous design variables and subject to two linear and five nonlinear inequality constraints. The best feasible solution is obtained by BSAISA at x = (0.205730, 3.470489, 9.036624, 0.205730) with the objective function value $f(x)$ = 1.724852 using 29,000 FEs.

For this problem, BSAISA is compared with many well-known algorithms as follows: GA-DT, MDE, CPSO, HPSO, DELC, POS-DE, ABC, MBA, BSA, BSA-SAε, and SSOC. The best solutions obtained from BSAISA and other well-known algorithms are listed in Table 14. The comparison of their statistical results is presented in Table 15.

From Tables 14 and 15, except that the constraint value of PSO is not available, the obtained solution sets of all algorithms satisfy the constraints for the problem. Most of algorithms including BSAISA, BSA, BSA-SAε, MDE, HPSO, DELC, POS-DE, ABC, and SSOC are able to find the best function value 1.724852, while GA-DT and CPSO and MBA fail to find it. It should be admitted that DELC is superior

Table 14: Comparison of best solutions for the welded beam design problem.

Method	GA-DT	MDE	CPSO	HPSO	ABC	MBA	BSA-SAε	BSA	BSAISA
$X_1(h)$	0.205986	0.205730	0.202369	0.205730	0.205730	0.205729	0.205730	0.205730	0.205730
$X_2(l)$	3.471328	3.470489	3.544214	3.470489	3.470489	3.470493	3.470489	3.470489	3.470489
$X_3(t)$	9.020224	9.036624	9.04821	9.036624	9.036624	9.036626	9.036624	9.036624	9.036624
$X_4(b)$	0.206480	0.205730	0.205723	0.205730	0.205730	0.205729	0.205730	0.205730	0.205730
$g_1(X)$	−0.074092	−0.000335	−12.839796	NA	0.000000	−0.001614	$-1.55E-10$	$-5.32E-07$	0
$g_2(X)$	−0.266227	−0.000753	−1.247467	NA	−0.000002	−0.016911	$-4.30E-09$	$-9.02E-06$	0
$g_3(X)$	$-4.95E-04$	−0.000000	$-1.49E-03$	NA	0.000000	$-2.40E-07$	$-1.55E-15$	$-7.86E-12$	$-5.55E-17$
$g_4(X)$	−3.430044	−3.432984	−3.429347	NA	−3.432984	−3.432982	−3.4330	−3.432984	−3.432984
$g_5(X)$	−0.080986	−0.080730	−0.079381	NA	−0.080730	−0.080729	$-8.07E-02$	−0.080730	−0.080730
$g_6(X)$	−0.235514	−0.235540	−0.235536	NA	−0.235540	−0.235540	−0.2355	−0.235540	−0.235540
$g_7(X)$	−58.666440	−0.000882	−11.681355	NA	0.000000	−0.001464	$-1.85E-10$	$-1.13E-07$	$-5.46E-12$
$f(X)$	1.728226	**1.724852**	1.728024	**1.724852**	**1.724852**	1.724853	**1.724852**	**1.724852**	**1.724852**

Table 15: Comparison of statistical results for the welded beam design problem.

Method	Worst	Mean	Best	Std	FEs
GA-DT	1.993408	1.792654	1.728226	$7.47E-02$	80,000
MDE	1.724854	1.724853	**1.724852**	NA	24,000
CPSO	1.782143	1.748831	1.728024	$1.29E-02$	24,000
HPSO	1.814295	1.749040	**1.724852**	$4.00E-02$	81,000
DELC	1.724852	1.724852	**1.724852**	$4.1E-13$	**20,000**
PSO-DE	1.724852	1.724852	**1.724852**	$6.7E-16$	66,600
ABC	NA	1.741913	**1.724852**	$3.1E-02$	30,000
MBA	1.724853	1.724853	1.724853	$6.94E-19$	47,340
SSOC	1.799332	1.746462	**1.724852**	$2.57E-2$	25,000
BSA-SAε	1.724852	1.724852	**1.724852**	$8.11E-10$	80,000
BSA	1.724854	1.724852	**1.724852**	$2.35E-07$	45,480
BSAISA	1.724854	1.724852	**1.724852**	$2.96E-07$	29,000

to all other algorithms in terms of FEs and robustness for this problem. On the other hand, except for DELC, MDE, and SSOC with FEs of 2000, 2400, and 2500, respectively, BSAISA requires fewer FEs than the remaining algorithms (excluding algorithms that do not reach the best solution). When considering the comparison of the Std values for this problem, MBA exhibits its powerful robustness and BSAISA performs better than most algorithms except MBA, DELC, PSO-DE, BSA, and BSA-SAε.

It is worth mentioning that from [74] the Worst, Mean, Best, and Std value of MDE are given as 1.724854, 1.724853, 1.724852, and $1.0E-15$, respectively. However, the corresponding values of DELC equal 1.724852, 1.724852, 1.724852, and $4.1E-13$, respectively, where its Worst and Mean values are smaller than those of MDE while its Std is bigger than those of MDE. So we consider that the Std of MDE is probably an error data for this problem, and we replace it with NA in Table 15.

Figure 10 depicts the convergence curves of BSAISA and BSA for the welded beam design problem, where the value of $F(x^*)$ on the vertical axis equals 1.724852. Figure 10 shows the convergence speed of BSAISA is faster than that of BSA remarkably.

4.3.5. Speed Reducer Design Problem (SRP). This speed reducer design problem has eleven constraints and six continuous design variables $(x_1, x_2, x_4, x_5, x_6, x_7)$ and one integer variable (x_3). The best solution obtained from BSAISA is x = (3.500000, 0.7000000, 17, 7.300000, 7.715320, 3.350215, 5.286654) with $f(x)$ = 2994.471066 using 15,860 FEs. The comparison of the best solutions obtained by BSAISA and other well-known algorithms is given in Table 16. The statistical results of BSAISA, BSA, MDE, DEDS, HEAA, DELC, POS-DE, ABC, MBA, and SSOC are listed in Table 17.

As shown in Tables 16 and 17, the obtained solution sets of all algorithms satisfy the constraints for this problem. BSAISA, BSA, DEDS, and DELC are able to find the best function value 2994.471066 while the others do not. Among the four algorithms, DEDS, DELC, and BSA require 30000, 30000, and 25640 FEs, respectively. However, BSAISA requires only 15,860 FEs when it reaches the same best function value. MBA fails to find the best known function value; thus BSAISA is better than MBA in this problem, even though MBA has lower FEs. As for the comparison of the Std, among the four algorithms that achieve the best known function value, BSAISA is worse than the others. However, one thing that should be mentioned is that the main

Table 16: Comparison of best solutions for the speed reducer design problem.

Method	MDE	DEDS	DELC	HEAA	POS-DE	MBA	BSA	BSAISA
X_1	3.500010	3.500000	3.500000	3.500023	3.500000	3.500000	3.500000	3.500000
X_2	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000	0.700000
X_3	17	17	17	17.000013	17.000000	17.000000	17	17
X_4	7.300156	7.300000	7.300000	7.300428	7.300000	7.300033	7.300000	7.300000
X_5	7.800027	7.715320	7.715320	7.715377	7.800000	7.715772	7.715320	7.715320
X_6	3.350221	3.350215	3.350215	3.350231	3.350215	3.350218	3.350215	3.350215
X_7	5.286685	5.286654	5.286654	5.286664	5.286683	5.286654	5.286654	5.286654
$f(X)$	2996.356689	**2994.471066**	**2994.471066**	2994.499107	2996.348167	2994.482453	**2994.471066**	**2994.471066**

Table 17: Comparison of statistical results for the speed reducer design problem.

Method	Worst	Mean	Best	Std	FEs
MDE	2996.390137	2996.367220	2996.356689	$8.2E-03$	24,000
DEDS	2994.471066	2994.471066	**2994.471066**	$3.58E-12$	30,000
HEAA	2994.752311	2994.613368	2994.499107	$7.0E-02$	40,000
DELC	2994.471066	2994.471066	**2994.471066**	$1.9E-12$	30,000
PSO-DE	2996.348204	2996.348174	2996.348167	$6.4E-06$	54,350
ABC	NA	2997.058412	2997.058412	0	30,000
MBA	2999.652444	2996.769019	2994.482453	1.56	6300
SSOC	2996.113298	2996.113298	2996.113298	$1.34E-12$	25,000
BSA	2994.471066	2994.471066	**2994.471066**	$9.87E-11$	25,640
BSAISA	2994.471095	2994.471067	**2994.471066**	$5.40E-06$	**15,860**

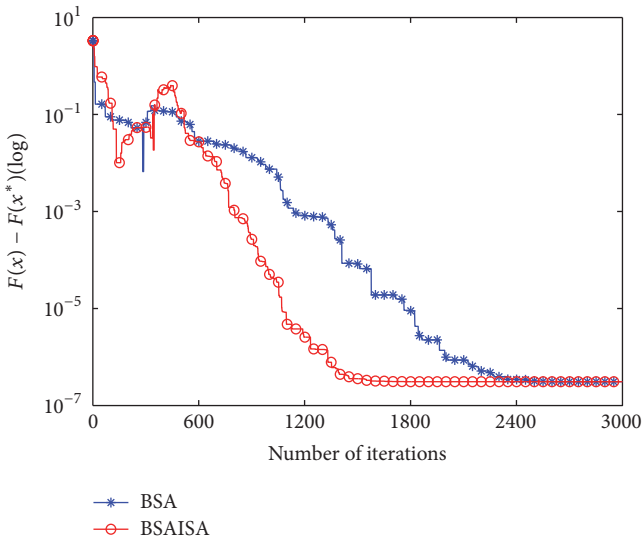


Figure 10: Convergence curves of BSAISA and BSA for the welded beam design problem.

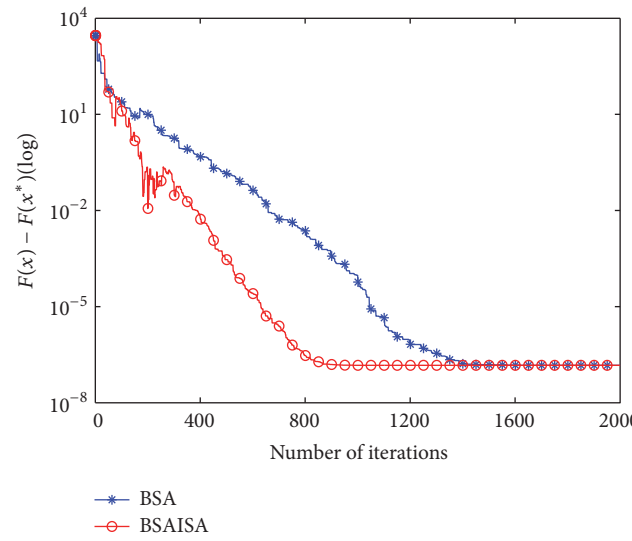


Figure 11: Convergence curves of BSAISA and BSA for the speed reducer design problem.

purpose of the experiment is to compare the convergence speed between BSAISA and other algorithms. From this point of view, it can be concluded that BSAISA has a better performance than other algorithms in terms of convergence speed.

Figure 11 depicts the convergence curves of BSAISA and BSA for the speed reducer design problem, where the value of $F(x^*)$ on the vertical axis equals 2994.471066. Figure 11 shows that the convergence speed of BSAISA is faster than that of BSA.

4.4. Comparisons Using Sign Test. Sign Test [80] is one of the most popular statistical methods used to determine whether two algorithms are significantly different. Recently, Miao et

Table 18: Comparisons between BSAISA and other algorithms in Sign Tests.

BSAISA-methods	+	≈	−	Total	p value
SR	11	1	1	13	0.006
FSA	12	0	1	13	0.003
CDE	8	0	5	13	0.581
AMA	13	0	0	13	0.000
MABC	10	2	1	13	0.012
RPGA	11	0	0	11	0.001
BSA-SAε	8	4	1	13	0.039

Note. The columns of "+," "≈," and "−" indicate the number of functions where BSAISA performs significantly better than, almost the same as, or significantly worse than the compared algorithm, respectively. "p value" denotes the probability value supporting the null hypothesis.

al. [81] utilized Sign Test method to analyze the performances between their proposed modified algorithm and the original one. In this paper, the two-tailed Sign Test with a significance level 0.05 is adopted to test the significant differences between the results obtained by different algorithms, and the test results are given in Table 18. The values of Best and FEs are two most important criterions for the evaluations of algorithms in our paper; they therefore should be chosen as the objectives of the Sign Test. The signs "+," "≈," and "−" represent, respectively, the fact that our BSAISA performs significantly better than, almost the same as, or significantly worse than the algorithm it is compared to. The null hypothesis herein is that the performances between BSAISA and one of the others are not significantly differential.

As shown in Table 18, the p values of supporting the null hypothesis of Sign Test for six pairs of algorithms (BSAISA-SR, BSAISA-FSA, BSAISA-AMA, BSAISA-MABC, BSAISA-RPGA, and BSAISA-BSA-SAε) are 0.006, 0.003, 0.000, 0.012, 0.001, and 0.039, respectively, and thereby we can reject the null hypothesis. This illustrates that the optimization performance of the proposed BSAISA is significantly better than those of the six algorithms. The p value of BSAISA-CDE is equal to 0.581, which shows that we cannot reject the null hypothesis. However, according to the related sign values ("+," "≈," and "−") from Table 18, BSAISA is slightly worse than CDE on 5 problems but wins on another 8 problems, which illustrates that the proposed BSAISA has a relatively excellent competitiveness compared with the CDE. Generally, the statistical p values and sign values validate that BSAISA has the superiority compared to the other well-known algorithms on the constrained optimization problems.

On the one hand, all experimental results suggest that the proposed method improves the convergence speed of BSA. On the other hand, the overall comparative results of BSAISA and other well-known algorithms demonstrate that BSAISA is more effective and competitive for constrained and engineering optimization problems in terms of convergence speed.

5. Conclusions and Future Work

In this paper, we proposed a modified version of BSA inspired by the Metropolis criterion in SA (BSAISA). The Metropolis criterion may probabilistically accept a higher energy state and the acceptance probability can decrease as the temperature decreases, which motivated us to redesign the amplitude control factor F so it can adaptively decrease as the number of iterations increases. The design principle and numerical analysis of the redesigned F indicate that the change in F could accelerate the convergence speed of the algorithm by improving the local exploitation capability. Furthermore, the redesigned F does not introduce extra parameters. We successfully implemented BSAISA to solve some constrained optimization and engineering design problems. The experimental results demonstrated that BSAISA has a faster convergence speed than BSA and it can efficiently balance the capacity for global exploration and local exploitation. The comparisons of the results obtained by BSAISA and other well-known algorithms demonstrated that BSAISA is more effective and competitive for constrained and engineering optimization problems in terms of convergence speed.

This paper suggests that the proposed BSAISA has a superiority in terms of convergence speed or computational cost. The downside of the proposed algorithm is, of course, that its robustness does not show enough superiority. So our future work is to further research into the robustness of BSAISA on the basis of current research. Niche technique is able to effectively maintain population diversity of evolutionary algorithms [82, 83]. How to combine BSAISA with niche technology to improve robustness of the algorithm may deserve to be studied in the future.

Appendix

A. Constrained Benchmark Problems

A.1. Constrained Problem 01

$$\begin{aligned} \min \quad & f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\ \text{subject to:} \quad & g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \\ & \leq 0 \\ & g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \\ & \leq 0 \end{aligned}$$

$$
\begin{aligned}
& g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \\
& \leq 0 \\
& g_4(x) = -8x_1 + x_{10} \leq 0 \\
& g_5(x) = -8x_2 + x_{11} \leq 0 \\
& g_6(x) = -8x_3 + x_{13} \leq 0 \\
& g_7(x) = -2x_4 - x_5 + x_{10} \leq 0 \\
& g_8(x) = -2x_6 - x_7 + x_{11} \leq 0 \\
& g_9(x) = -2x_8 - x_9 + x_{12} \leq 0 \\
& 0 \leq x_i \leq 1 \quad (i = 1, \ldots, 9) \\
& 0 \leq x_i \leq 100 \quad (i = 10, 11, 12) \\
& 0 \leq x_{13} \leq 1.
\end{aligned} \tag{A.1}
$$

A.2. Constrained Problem 02

$$
\begin{aligned}
\max \quad & f(x) \\
& = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \\
\text{subject to:} \quad & g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\
& g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \\
& n = 20, \ 0 \leq x_i \leq 10, \ i = 1, \ldots, n.
\end{aligned} \tag{A.2}
$$

A.3. Constrained Problem 03

$$
\begin{aligned}
\max \quad & f(x) = \left(\sqrt{n}\right)^n \cdot \prod_{i=1}^{n} x_i \\
\text{subject to:} \quad & h(x) = \sum_{i=1}^{n} x_i^2 = 0 \\
& n = 10, \ 0 \leq x_i \leq 1, \ i = 1, \ldots, n.
\end{aligned} \tag{A.3}
$$

A.4. Constrained Problem 04

$$
\begin{aligned}
\min \quad & f(x) \\
& = 5.3578547x_3^2 + 0.8356891x_1x_5 \\
& \quad + 37.293239x_1 - 40792.141 \\
\text{subject to:} \quad & g_1(x) \\
& = 85.334407 + 0.0056858x_2x_5 \\
& \quad + 0.0006262x_1x_4 - 0.0022053x_3x_5 \\
& \quad - 92 \leq 0 \\
& g_2(x) \\
& = -85.334407 - 0.0056858x_2x_5 \\
& \quad - 0.0006262x_1x_4 + 0.0022053x_3x_5 \\
& \leq 0 \\
& g_3(x) \\
& = 80.51249 + 0.0071317x_2x_5 \\
& \quad + 0.0029955x_1x_2 + 0.0021813x_3^2 \\
& \quad - 110 \leq 0 \\
& g_4(x) \\
& = -80.51249 - 0.0071317x_2x_5 \\
& \quad - 0.0029955x_1x_2 - 0.0021813x_3^2 \\
& \quad + 90 \leq 0 \\
& g_5(x) \\
& = 9.300961 + 0.0047026x_3x_5 \\
& \quad + 0.0012547x_1x_3 + 0.0019085x_3x_4 \\
& \quad - 25 \leq 0 \\
& g_6(x) \\
& = -9.300961 - 0.0047026x_3x_5 \\
& \quad - 0.0012547x_1x_3 - 0.0019085x_3x_4 \\
& \quad + 20 \leq 0 \\
& 78 \leq x_1 \leq 102, \\
& 33 \leq x_2 \leq 45 \\
& 27 \leq x_i \leq 45, \quad i = 3, 4, 5.
\end{aligned} \tag{A.4}
$$

A.5. Constrained Problem 05

$$
\begin{aligned}
\min \quad & f(x) \\
& = 3x_1 + 0.000001x_1^3 + 2x_2 \\
& \quad + \left(\frac{0.000002}{3}\right) x_2^3
\end{aligned}
$$

$$
\begin{aligned}
\text{subject to:}\quad & g_1(x) = -x_4 + x_3 - 0.55 \le 0 \\
& g_2(x) = -x_3 + x_4 - 0.55 \le 0 \\
& h_3(x) \\
& = 1000 \sin(-x_3 - 0.25) \\
& \quad + 1000 \sin(-x_4 - 0.25) + 894.8 \\
& \quad - x_1 = 0 \\
& h_4(x) \\
& = 1000 \sin(x_3 - 0.25) \\
& \quad + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 \\
& \quad - x_2 = 0 \\
& h_5(x) \\
& = 1000 \sin(x_4 - 0.25) \\
& \quad + 1000 \sin(x_4 - x_3 - 0.25) \\
& \quad + 1294.8 = 0 \\
& 0 \le x_1, x_2 \le 1200, \\
& -0.55 \le x_3, x_4 \le 0.55.
\end{aligned} \tag{A.5}
$$

A.6. Constrained Problem 06

$$
\begin{aligned}
\min\quad & f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \\
\text{subject to:}\quad & g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \\
& \le 0 \\
& g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \\
& \le 0 \\
& 13 \le x_1 \le 100, \\
& 0 \le x_2 \le 100.
\end{aligned} \tag{A.6}
$$

A.7. Constrained Problem 07

$$
\begin{aligned}
\min\quad & f(x) \\
& = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 \\
& \quad + (x_3 - 10)^2 + 4(x_4 - 5)^2 \\
& \quad + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\
& \quad + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 \\
& \quad + (x_{10} - 7)^2 + 45 \\
\text{subject to:}\quad & g_1(x) \\
& = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0 \\
& g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0 \\
& g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \\
& \le 0 \\
& g_4(x) \\
& = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 \\
& \quad - 7x_4 - 120 \le 0 \\
& g_5(x) \\
& = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \\
& \le 0 \\
& g_6(x) \\
& = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 \\
& \quad - 6x_6 \le 0 \\
& g_7(x) \\
& = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 \\
& \quad - x_6 - 30 \le 0 \\
& g_8(x) \\
& = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \\
& \le 0 \\
& -10 \le x_i \le 10, \quad i = 1, \ldots, 10.
\end{aligned} \tag{A.7}
$$

A.8. Constrained Problem 08

$$
\begin{aligned}
\min\quad & f(x) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \\
\text{subject to:}\quad & g_1(x) = x_1^2 - x_2 + 1 \le 0 \\
& g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0 \\
& 0 \le x_i \le 10, \quad i = 1, 2.
\end{aligned} \tag{A.8}
$$

A.9. Constrained Problem 09

$$
\begin{aligned}
\min\quad & f(x) \\
& = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 \\
& \quad + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 \\
& \quad - 4x_6 x_7 - 10x_6 - 8x_7
\end{aligned}
$$

$$\begin{aligned}
\text{subject to:}\quad & g_1(x) \\
& = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \\
& \le 0 \\
& g_2(x) \\
& = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \\
& \le 0 \\
& g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \\
& \le 0 \\
& g_4(x) \\
& = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 \\
& \quad - 11x_7 \le 0 \\
& -10 \le x_i \le 10, \quad i = 1, 2, 3, 4, 5, 6, 7.
\end{aligned} \tag{A.9}$$

A.10. Constrained Problem 10

$$\begin{aligned}
\min\quad & f(x) = x_1 + x_2 + x_3 \\
\text{subject to:}\quad & g_1(x) = -1 + 0.0025\left(x_4 + x_6\right) \le 0 \\
& g_2(x) = -1 + 0.0025\left(x_5 + x_7 - x_4\right) \\
& \le 0 \\
& g_3(x) = -1 + 0.01\left(x_8 - x_5\right) \le 0 \\
& g_4(x) \\
& = -x_1x_6 + 833.33252x_4 + 100x_1 \\
& \quad - 83333.333 \le 0 \\
& g_5(x) \\
& = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \\
& \le 0 \\
& g_6(x) \\
& = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \\
& \le 0 \\
& 100 \le x_1 \le 10000 \\
& 1000 \le x_i \le 10000, \quad i = 2, 3 \\
& 10 \le x_i \le 1000, \quad i = 4, 5, 6, 7, 8.
\end{aligned} \tag{A.10}$$

A.11. Constrained Problem 11

$$\begin{aligned}
\min\quad & f(x) = x_1^2 + \left(x_2 - 1\right)^2 \\
\text{subject to:}\quad & h(x) = x_2 - x_1^2 = 0 \\
& -1 \le x_i \le 1, \quad i = 1, 2.
\end{aligned} \tag{A.11}$$

A.12. Constrained Problem 12

$$\begin{aligned}
\max\quad & f(x) \\
& = \frac{\left(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2\right)^2}{100} \\
\text{subject to:}\quad & g(x) \\
& = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \\
& \le 0 \\
& p, q, r = 1, \ldots, 9, \\
& 0 \le x_i \le 10, \quad i = 1, 2, 3.
\end{aligned} \tag{A.12}$$

A.13. Constrained Problem 13

$$\begin{aligned}
\min\quad & f(x) = e^{x_1x_2x_3x_4x_5} \\
\text{subject to:}\quad & h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 \\
& = 0 \\
& h_2(x) = x_2x_3 - 5x_4x_5 = 0 \\
& h_3(x) = x_1^3 + x_2^3 + 1 = 0 \\
& -2.3 \le x_i \le 2.3, \quad i = 1, 2 \\
& -3.2 \le x_i \le 3.2, \quad i = 3, 4, 5.
\end{aligned} \tag{A.13}$$

B. Engineering Design Problems

B.1. Three-Bar Truss Design Problem

$$\begin{aligned}
\min\quad & f(x) = \left(2\sqrt{2}x_1 + x_2\right) \times l \\
\text{subject to:}\quad & g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}\mathrm{P} - \sigma \le 0 \\
& g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}\mathrm{P} - \sigma \le 0 \\
& g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}\mathrm{P} - \sigma \le 0 \\
& 0 \le x_i \le 1, \quad i = 1, 2 \\
& l = 100\,\mathrm{cm}, \\
& \mathrm{P} = 2\,\mathrm{kN/cm^2}, \\
& \sigma = 2\,\mathrm{kN/cm^2}.
\end{aligned} \tag{B.1}$$

B.2. Pressure Vessel Design Problem

$$\begin{aligned}
\min \quad & f(x) \\
& = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\
& \quad + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
\text{subject to:} \quad & g_1(x) = -x_1 + 0.0193x_3 \le 0 \\
& g_2(x) = -x_2 + 0.00954x_3 \le 0 \\
& g_3(x) \\
& = -\pi x_3^2x_4 - \left(\frac{4}{3}\right)\pi x_3^3 + 1296000 \le 0 \\
& g_4(x) = x_4 - 240 \le 0 \\
& 0 \le x_i \le 100, \quad i = 1, 2 \\
& 10 \le x_i \le 200, \quad i = 3, 4.
\end{aligned} \tag{B.2}$$

B.3. Tension/Compression Spring Design Problem

$$\begin{aligned}
\min \quad & f(x) = (x_3 + 2)\, x_2x_1^2 \\
\text{subject to:} \quad & g_1(x) = \frac{-x_2^3x_3}{(71785x_1^4)} + 1 \le 0 \\
& g_2(x) \\
& = \frac{\left(4x_2^2 - x_1x_2\right)}{\left(12566\left(x_2x_1^3 - x_1^4\right)\right)} + \frac{1}{\left(5108x_1^2\right)} \\
& \quad - 1 \le 0 \\
& g_3(x) = \frac{-140.45x_1}{\left(x_2^2x_3\right)} + 1 \le 0 \\
& g_4(x) = \frac{(x_1 + x_2)}{1.5} - 1 \le 0 \\
& 0.05 \le x_1 \le 2.00 \\
& 0.25 \le x_2 \le 1.30 \\
& 2.00 \le x_3 \le 15.00.
\end{aligned} \tag{B.3}$$

B.4. Welded Beam Design Problem

$$\begin{aligned}
\min \quad & f(x) \\
& = 1.10471x_1^2x_2 \\
& \quad + 0.04811x_3x_4\left(14 + x_2\right) \\
\text{subject to:} \quad & g_1(x) = \tau(x) - \tau_{\max} \le 0 \\
& g_2(x) = \sigma(x) - \sigma_{\max} \le 0 \\
& g_3(x) = x_1 - x_4 \le 0 \\
& g_4(x) \\
& = 0.10471x_1^2 + 0.04811x_3x_4\left(14 + x_2\right) \\
& \quad - 5 \le 0 \\
& g_5(x) = 0.125 - x_1 \le 0 \\
& g_6(x) = \delta(x) - \delta_{\max} \le 0 \\
& g_7(x) = P - P_c(x) \le 0 \\
& 0.1 \le x_i \le 2, \quad i = 1, 4 \\
& 0.1 \le x_i \le 10, \quad i = 2, 3,
\end{aligned} \tag{B.4}$$

where

$$\begin{aligned}
& \tau(x) = \sqrt{\left(\tau'\right)^2 + 2\tau'\tau''\frac{x_2}{2R} + \left(\tau''\right)^2}, \\
& \qquad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \ \tau'' = \frac{MR}{J} \\
& M = P\left(L + \frac{x_2}{2}\right), \\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \\
& J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
& \sigma(x) = \frac{6PL}{x_4x_3^2}, \\
& \delta(x) = \frac{4PL^3}{Ex_3^3x_4}, \\
& P_c(x) = \frac{4.013E\sqrt{\left(x_3^2x_4^6/36\right)}}{L^2} \times \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\
& P = 6000\,\text{lb}, \\
& L = 14\,\text{in}, \\
& E = 30 \times 10^6\,\text{psi}, \\
& G = 12 \times 10^6\,\text{psi} \\
& \tau_{\max} = 13600\,\text{psi}, \\
& \sigma_{\max} = 30000\,\text{psi}, \\
& \delta_{\max} = 0.25\,\text{in}.
\end{aligned} \tag{B.5}$$

B.5. Speed Reducer Design Problem

$$\begin{aligned}
\min \quad & f(x) \\
& = 0.7854 x_1 x_2^2 \left(3.3333 x_3^2 + 14.9334 x_3 - 43.0934\right) \\
& \quad - 1.508 x_1 \left(x_6^2 + x_7^2\right) + 7.4777 \left(x_6^3 + x_7^3\right) \\
& \quad + 0.7854 \left(x_4 x_6^2 + x_5 x_7^2\right) \\
\text{subject to:} \quad & g_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0 \\
& g_2(x) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0 \\
& g_3(x) = \frac{1.93 x_4^3}{x_2 x_6^4 x_3} - 1 \le 0 \\
& g_4(x) = \frac{1.93 x_5^3}{x_2 x_7^4 x_3} - 1 \le 0 \\
& g_5(x) = \frac{\left[\left(745 x_4 / (x_2 x_3)\right)^2 + 16.9 \times 10^6\right]^{1/2}}{110 x_6^3} - 1 \\
& \le 0 \\
& g_6(x) = \frac{\left[\left(745 x_5 / (x_2 x_3)\right)^2 + 157.5 \times 10^6\right]^{1/2}}{85 x_7^3} - 1 \\
& \le 0 \\
& g_7(x) = \frac{x_2 x_3}{40} - 1 \le 0 \\
& g_8(x) = \frac{5 x_2}{x_1} - 1 \le 0 \\
& g_9(x) = \frac{x_1}{12 x_2} - 1 \le 0 \\
& g_{10}(x) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0 \\
& g_{11}(x) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0,
\end{aligned} \tag{B.6}$$

where

$$\begin{aligned}
& 2.6 \le x_1 \le 3.6, \\
& 0.7 \le x_2 \le 0.8, \\
& 17 \le x_3 \le 28 \\
& 7.3 \le x_4, x_5 \le 8.3, \\
& 2.9 \le x_6 \le 3.9, \\
& 5.0 \le x_7 \le 5.5.
\end{aligned} \tag{B.7}$$

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no. 61663009), and the State Key Laboratory of Silicate Materials for Architectures (Wuhan University of Technology, SYSJJ2018-21).

References

[1] P. Posik, W. Huyer, and L. Pal, "A comparison of global search algorithms for continuous black box optimization," *Evolutionary Computation*, vol. 20, no. 4, pp. 509–541, 2012.

[2] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm Intelligence and Evolutionary Algorithms: Performance versus speed," *Information Sciences*, vol. 384, pp. 34–85, 2017.

[3] S. Das and A. Konar, "A swarm intelligence approach to the synthesis of two-dimensional IIR filters," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 8, pp. 1086–1096, 2007.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[6] X. S. Yang and S. Deb, "Cuckoo search via LΘvy flights," in *Proceedings of the In World Congress on Nature Biologically Inspired Computing*, pp. 210–214, NaBIC, 2009.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep., Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[8] J. Q. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[9] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Adaptive Configuration of evolutionary algorithms for constrained optimization," *Applied Mathematics and Computation*, vol. 222, pp. 680–711, 2013.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Oxford, UK, 1975.

[11] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] Z. Hu, Q. Su, X. Yang, and Z. Xiong, "Not guaranteeing convergence of differential evolution on a class of multimodal functions," *Applied Soft Computing*, vol. 41, pp. 479–487, 2016.

[13] Q. Su and Z. Hu, "Color image quantization algorithm based on self-adaptive differential Evolution," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 231916, 8 pages, 2013.

[14] Z. Hu, Q. Su, and X. Xia, "Multiobjective image color quantization algorithm based on self-adaptive hybrid differential evolution," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 2450431, 12 pages, 2016.

[15] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.

[16] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[17] A. El-Fergany, "Optimal allocation of multi-type distributed generators using backtracking search optimization algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 1197–1205, 2015.

[18] M. Modiri-Delshad and N. A. Rahim, "Multi-objective backtracking search algorithm for economic emission dispatch problem," *Applied Soft Computing*, vol. 40, pp. 479–494, 2016.

[19] S. D. Madasu, M. L. S. S. Kumar, and A. K. Singh, "Comparable investigation of backtracking search algorithm in automatic generation control for two area reheat interconnected thermal power system," *Applied Soft Computing*, vol. 55, pp. 197–210, 2017.

[20] J. A. Ali, M. A. Hannan, A. Mohamed, and M. G. M. Abdolrasol, "Fuzzy logic speed controller optimization approach for induction motor drive using backtracking search algorithm," *Measurement*, vol. 78, pp. 49–62, 2016.

[21] M. A. Hannan, J. A. Ali, A. Mohamed, and M. N. Uddin, "A Random Forest Regression Based Space Vector PWM Inverter Controller for the Induction Motor Drive," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 2689–2699, 2017.

[22] K. Guney, A. Durmus, and S. Basbug, "Backtracking search optimization algorithm for synthesis of concentric circular antenna arrays," *International Journal of Antennas and Propagation*, vol. 2014, Article ID 250841, 11 pages, 2014.

[23] R. Muralidharan, V. Athinarayanan, G. K. Mahanti, and A. Mahanti, "QPSO versus BSA for failure correction of linear array of mutually coupled parallel dipole antennas with fixed side lobe level and VSWR," *Advances in Electrical Engineering*, vol. 2014, Article ID 858290, 7 pages, 2014.

[24] M. Eskandari and Ö. Toygar, "Selection of optimized features and weights on face-iris fusion using distance images," *Computer Vision and Image Understanding*, vol. 137, article no. 2225, pp. 63–75, 2015.

[25] U. H. Atasever, P. Civicioglu, E. Besdok, and C. Ozkan, "A new unsupervised change detection approach based on DWT image fusion and backtracking search optimization algorithm for optical remote sensing data," in *Proceedings of the ISPRS Technical Commission VII Mid-Term Symposium 2014*, pp. 15–18, October 2014.

[26] S. K. Agarwal, S. Shah, and R. Kumar, "Classification of mental tasks from EEG data using backtracking search optimization based neural classifier," *Neurocomputing*, vol. 166, pp. 397–403, 2015.

[27] L. Zhang and D. Zhang, "Evolutionary cost-sensitive extreme learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 12, pp. 3045–3060, 2016.

[28] F. Zou, D. Chen, S. Li, R. Lu, and M. Lin, "Community detection in complex networks: Multi-objective discrete backtracking search optimization algorithm with decomposition," *Applied Soft Computing*, vol. 53, pp. 285–295, 2017.

[29] C. Zhang, J. Zhou, C. Li, W. Fu, and T. Peng, "A compound structure of ELM based on feature selection and parameter optimization using hybrid backtracking search algorithm for wind speed forecasting," *Energy Conversion and Management*, vol. 143, pp. 360–376, 2017.

[30] C. Lu, L. Gao, X. Li, and P. Chen, "Energy-efficient multi-pass turning operation using multi-objective backtracking search algorithm," *Journal of Cleaner Production*, vol. 137, pp. 1516–1531, 2016.

[31] M. Akhtar, M. A. Hannan, R. A. Begum, H. Basri, and E. Scavino, "Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization," *Waste Management*, vol. 61, pp. 117–128, 2017.

[32] M. S. Ahmed, A. Mohamed, T. Khatib, H. Shareef, R. Z. Homod, and J. A. Ali, "Real time optimal schedule controller for home energy management system using new binary backtracking search algorithm," *Energy and Buildings*, vol. 138, pp. 215–227, 2017.

[33] S. O. Kolawole and H. Duan, "Backtracking search algorithm for non-aligned thrust optimization for satellite formation," in *Proceedings of the 11th IEEE International Conference on Control and Automation (IEEE ICCA '14)*, pp. 738–743, June 2014.

[34] Q. Lin, L. Gao, X. Li, and C. Zhang, "A hybrid backtracking search algorithm for permutation flow-shop scheduling problem," *Computers & Industrial Engineering*, vol. 85, pp. 437–446, 2015.

[35] J. Lin, "Oppositional backtracking search optimization algorithm for parameter identification of hyperchaotic systems," *Nonlinear Dynamics*, vol. 80, no. 1-2, pp. 209–219, 2015.

[36] Q. L. Xu, N. Guo, and L. Xu, "Opposition-based backtracking search algorithm for numerical optimization problems," in *Proceedings of the In International Conference on Intelligent Science and Big Data Engineering*, pp. 223–234, 2015.

[37] X. Yuan, B. Ji, Y. Yuan, R. M. Ikram, X. Zhang, and Y. Huang, "An efficient chaos embedded hybrid approach for hydrothermal unit commitment problem," *Energy Conversion and Management*, vol. 91, pp. 225–237, 2015.

[38] X. Yuan, X. Wu, H. Tian, Y. Yuan, and R. M. Adnan, "Parameter identification of nonlinear muskingum model with backtracking search algorithm," *Water Resources Management*, vol. 30, no. 8, pp. 2767–2783, 2016.

[39] S. Vitayasak and P. Pongcharoen, "Backtracking search algorithm for designing a robust machine layout," *WIT Transactions on Engineering Sciences*, vol. 95, pp. 411–420, 2014.

[40] S. Vitayasak, P. Pongcharoen, and C. Hicks, "A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm," *International Journal of Production Economics*, vol. 190, pp. 146–157, 2017.

[41] M. Li, H. Zhao, and X. Weng, "Backtracking search optimization algorithm with comprehensive learning strategy," *Journal of Systems Engineering and Electronics*, vol. 37, no. 4, pp. 958–963, 2015 (Chinese).

[42] W. Zhao, L. Wang, Y. Yin, B. Wang, and Y. Wei, "An improved backtracking search algorithm for constrained optimization problems," in *Proceedings of the International Conference on Knowledge Science, Engineering and Management*, pp. 222–233, Springer International Publishing, 2014.

[43] L. Wang, Y. Zhong, Y. Yin, W. Zhao, B. Wang, and Y. Xu, "A hybrid backtracking search optimization algorithm with differential evolution," *Mathematical Problems in Engineering*, vol. 2015, Article ID 769245, p. 16, 2015.

[44] S. Das, D. Mandal, R. Kar, and S. P. Ghoshal, "Interference suppression of linear antenna arrays with combined Backtracking Search Algorithm and Differential Evolution," in *Proceedings of the 3rd International Conference on Communication and Signal Processing (ICCSP '14)*, pp. 162–166, April 2014.

[45] S. Das, D. Mandal, R. Kar, and S. P. Ghoshal, "A new hybridized backscattering search optimization algorithm with differential evolution for sidelobe suppression of uniformly excited concentric circular antenna arrays," *International Journal of RF and*

Microwave Computer-Aided Engineering, vol. 25, no. 3, pp. 262–268, 2015.

[46] S. Mallick, R. Kar, D. Mandal, and S. P. Ghoshal, "CMOS analogue amplifier circuits optimisation using hybrid backtracking search algorithm with differential evolution," *Journal of Experimental and Theoretical Artificial Intelligence*, pp. 1–31, 2015.

[47] D. Chen, F. Zou, R. Lu, and P. Wang, "Learning backtracking search optimisation algorithm and its application," *Information Sciences*, vol. 376, pp. 71–94, 2017.

[48] A. F. Ali, "A memetic backtracking search optimization algorithm for economic dispatch problem," *Egyptian Computer Science Journal*, vol. 39, no. 2, pp. 56–71, 2015.

[49] Y. Wu, Q. Tang, L. Zhang, and X. He, "Solving stochastic two-sided assembly line balancing problem via hybrid backtracking search optimization algorithm," *Journal of Wuhan University of Science and Technology (Natural Science Edition)*, vol. 39, no. 2, pp. 121–127, 2016 (Chinese).

[50] Z. Su, H. Wang, and P. Yao, "A hybrid backtracking search optimization algorithm for nonlinear optimal control problems with complex dynamic constraints," *Neurocomputing*, vol. 186, pp. 182–194, 2016.

[51] S. Wang, X. Da, M. Li, and T. Han, "Adaptive backtracking search optimization algorithm with pattern search for numerical optimization," *Journal of Systems Engineering and Electronics*, vol. 27, no. 2, Article ID 7514428, pp. 395–406, 2016.

[52] H. Duan and Q. Luo, "Adaptive backtracking search algorithm for induction magnetometer optimization," *IEEE Transactions on Magnetics*, vol. 50, no. 12, pp. 1–6, 2014.

[53] X. J. Wang, S. Y. Liu, and W. K. Tian, "Improved backtracking search optimization algorithm with new effective mutation scale factor and greedy crossover strategy," *Journal of Computer Applications*, vol. 34, no. 9, pp. 2543–2546, 2014 (Chinese).

[54] W. K. Tian, S. Y. Liu, and X. J. Wang, "Study and improvement of backtracking search optimization algorithm based on differential evolution," *Application Research of Computers*, vol. 32, no. 6, pp. 1653–1662, 2015.

[55] A. Askarzadeh and L. dos Santos Coelho, "A backtracking search algorithm combined with Burger's chaotic map for parameter estimation of PEMFC electrochemical model," *International Journal of Hydrogen Energy*, vol. 39, pp. 11165–11174, 2014.

[56] X. Chen, S. Y. Liu, and Y. Wang, "Emergency resources scheduling based on improved backtracking search optimization algorithm," *Computer Applications and Software*, vol. 32, no. 12, pp. 235–238, 2015 (Chinese).

[57] S. Nama, A. K. Saha, and S. Ghosh, "Improved backtracking search algorithm for pseudo dynamic active earth pressure on retaining wall supporting c-Φ backfill," *Applied Soft Computing*, vol. 52, pp. 885–897, 2017.

[58] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter Control in Evolutionary Algorithms: Trends and Challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.

[59] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[60] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[61] D. Karaboga and B. Akay, "A comparative study of artificial Bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[62] C. Zhang, Q. Lin, L. Gao, and X. Li, "Backtracking Search Algorithm with three constraint handling methods for constrained optimization problems," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7831–7845, 2015.

[63] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[64] A. S. B. Ullah, R. Sarker, D. Cornforth, and C. Lokan, "AMA: A new approach for solving constrained real-valued optimization problems," *Soft Computing*, vol. 13, no. 8-9, pp. 741–762, 2009.

[65] A. R. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," *Journal of Global Optimization*, vol. 35, no. 4, pp. 521–549, 2006.

[66] R. L. Becerra and C. A. Coello, "Cultured differential evolution for constrained optimization," *Computer Methods Applied Mechanics and Engineering*, vol. 195, no. 33–36, pp. 4303–4322, 2006.

[67] D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, no. 3, pp. 3021–3031, 2011.

[68] C.-H. Lin, "A rough penalty genetic algorithm for constrained optimization," *Information Sciences*, vol. 241, pp. 119–137, 2013.

[69] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, 2008.

[70] Y. Wang, Z. X. Cai, Y. R. Zhou, and Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Structural and Multidisciplinary Optimization*, vol. 37, no. 4, pp. 395–413, 2009.

[71] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing*, vol. 10, no. 2, pp. 629–640, 2010.

[72] L. Wang and L.-P. Li, "An effective differential evolution with level comparison for constrained engineering design," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 947–963, 2010.

[73] C. A. C. Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Advanced Engineering Informatics*, vol. 16, no. 3, pp. 193–203, 2002.

[74] E. Mezura-Montes, C. A. C. Coello, and J. Vela'zquez-Reyes, "Increasing successful offspring and diversity in differential evolution for engineering design," in *Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture (ACDM '06)*, pp. 131–139, 2006.

[75] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 1, pp. 89–99, 2007.

[76] Q. He and L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," *Applied Mathematics and Computation*, vol. 186, no. 2, pp. 1407–1422, 2007.

[77] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1001–1014, 2012.

[78] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, no. 5, pp. 2592–2612, 2013.

[79] E. Cuevas and M. Cienfuegos, "A new algorithm inspired in the behavior of the social-spider for constrained optimization," *Expert Systems with Applications*, vol. 41, no. 2, pp. 412–425, 2014.

[80] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[81] Y. Miao, Q. Su, Z. Hu, and X. Xia, "Modified differential evolution algorithm with onlooker bee operator for mixed discrete-continuous optimization," *SpringerPlus*, vol. 5, no. 1, article no. 1914, 2016.

[82] E. L. Yu and P. N. Suganthan, "Ensemble of niching algorithms," *Information Sciences*, vol. 180, no. 15, pp. 2815–2833, 2010.

[83] M. Li, D. Lin, and J. Kou, "A hybrid niching PSO enhanced with recombination-replacement crowding strategy for multimodal function optimization," *Applied Soft Computing*, vol. 12, no. 3, pp. 975–987, 2012.