Hindawi
Discrete Dynamics in Nature and Society
Volume 2019, Article ID 6523435, 24 pages
https://doi.org/10.1155/2019/6523435

Research Article

Constraint Consensus Based Artificial Bee Colony Algorithm for Constrained Optimization Problems

Liling Sun, Yuhan Wu, Xiaodan Liang, Maowei He, and Hanning Chen

School of Computer Science and Technology, Tiangong University, Tianjin 300387, China

Correspondence should be addressed to Xiaodan Liang; lxdtjpu@163.com

Received 21 May 2019; Revised 23 August 2019; Accepted 3 October 2019; Published 25 December 2019

Academic Editor: Ewa Pawluszewicz

Over the last few decades, evolutionary algorithms (EAs) have been widely adopted to solve complex optimization problems. However, EAs are powerless to challenge the constrained optimization problems (COPs) because they do not directly act to reduce constraint violations of constrained problems. In this paper, the robustly global optimization advantage of artificial bee colony (ABC) algorithm and the stably minor calculation characteristic of constraint consensus (CC) strategy for COPs are integrated into a novel hybrid heuristic algorithm, named ABCCC. CC strategy is fairly effective to rapidly reduce the constraint violations during the evolutionary search process. The performance of the proposed ABCCC is verified by a set of constrained benchmark problems comparing with two state-of-the-art CC-based EAs, including particle swarm optimization based on CC (PSOCC) and differential evolution based on CC (DECC). Experimental results demonstrate the promising performance of the proposed algorithm, in terms of both optimization quality and convergence speed.

1. Introduction

Optimization problems occur in many disciplines, for example, in engineering [1], physical sciences [2], social sciences [3], and commerce [4]. In real-world applications, optimization is carried out under certain physical limitations, with limited resources. If these limitations can be quantified as equality or inequality constraints on the variables, then a constrained optimization problem can be formulated whose solution leads to an optimal solution that satisfies the limitations imposed [5].

Constraint optimization problem, where the objective functions are optimized under given constraints, isvery important and frequently appears in practical applications. The general constrained optimization problem with inequality, equality, upper bound, and lower bound constraints is defined as follow:

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & g_j(x) \le 0, \quad j = 1, 2, \ldots, q, \\ & h_j(x) = 0, \quad j = q+1, \ldots, m, \\ & l_i \le x_i \le u_i, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ are n-dimensional variables, $f(x)$ is an objective function, $g(x)$ is inequality constraint, $h(x)$ is equality constraint, and j is the number of inequality or equality constraints. The values l_i and u_i are the lower bound and the upper bound of x_i, respectively. The upper and lower bounds define the search space. The inequality and equality constraints define the feasible region. Therefore, the point in the feasible domain or the boundary is a feasible point, otherwise it is not feasible.

Due to the complexity of constraint optimization problems, traditional evolutionary algorithms are difficult to solve. Meanwhile, evolutionary algorithms are search techniques mostly based on unconstrained optimization problems, the optimum solution may not be accurate when utilizing these to solve constrained optimization problems. For this reason, it is necessary to combine a suitable constraint handling technique to handle constraint optimization problems. Researchers have proposed various kinds of constraint handling techniques in recent years. The specific content will be discussed in the section below.

For constrained problem, it is widely acceptable that, during the search process, the ability of considering the fitness and feasibility improvement is beneficial for the optimizer to find out the feasible optimal solution more effectively and efficiently than the existing search operators. For this purpose, a

novel artificial bee colony based on constrained consensus strategy (ABCCC) is elaborated. Artificial bee colony (ABC) algorithm proposed by Karaboga is a latest heuristic algorithm, which is inspired by the foraging behavior of honey bees for numerical optimization problems [6]. Compared with differential evolution (DE) and particle swarm optimization (PSO), ABC algorithm has two distinct advantages: (1) ABC is very good in terms of the local and the global optimization. (2) ABC is flexible, robust and simple to use. It can be used efficiently in the optimization of multimodal and multi-variable problems. For driving individuals moving towards the feasible region, the constraint consensus (CC) strategy is employed into ABC algorithm. The CC strategy has the following advantages: (1) CC strategy is simple to implement and that has been proved effective in the past tests [7]. (2) CC has minor calculation and best calculating stability. To effectively combine ABC algorithm and CC strategy, the treatment of infeasible individuals is re-designed. That is, some of the infeasible individuals are evolved by CC strategy, the remaining infeasible ones and feasible individuals are reproduced by ABC algorithm. To verify the characteristics of ABCCC, 10-*D,* and 30-*D* CEC2017 benchmark functions are adopted to test its performance. The experiment shows that, in most cases, ABCCC converges faster and has an exact precision in terms of convergence accuracyand it can consider the problem of population diversity well. The proposed ABCCC is highly competitive than DECC and PSOCC and this algorithm is utilized to compare with famous constraint algorithms, CALSHADE [8] and UDE [9]. The experimental results show that ABCCC outperforms the other algorithms. Many variants of ABC algorithm, such as CABC [10] and GABC [11], have been proposed by researchers in recent years. The proposed ABCCC algorithm is also compared with CABCCC and GABCCC. ABCCC algorithm performs better than the other two algorithms.

This paper is organized as follows. After the Introduction, Section 2 elaborates the constraint handling techniques in detail. Section 3 states the related works before describing the proposed algorithm in Section 4. Detail of the proposed new algorithm is in Section 4. The experimental results and analysis are presented in Section 5. Finally, the conclusion is given in Section 6.

2. Constraint Handling Techniques

A great deal of work has already been undertaken on constraint handling techniques. The constraint handling techniques can be divided into four categories: (1) penalty functions; (2) special representations and operators; (3) separation of constraints and objectives; (4) hybrid methods. Substances of each category are briefly described below.

2.1. Penalty Functions. This method was originally proposed by Richard Courant in the 1940s and was later expanded by Alice and David [12]. It is one of the main approaches often used by researchers. The idea of penalty functions is to transform a constrained optimization problem into an unconstrained one by adding (or subtracting) a certain value to the objective function based on the amount of constraint violation present in a certain solution [13]. The general formulation of the penalty function is

$$f(x) = \begin{cases} f(x) & \text{if all constraints are feasible,} \\ f(x) \pm \text{ penalty} & \text{otherwise,} \end{cases} \tag{2}$$

where

$$\text{penalty} = \sum_{j}^{q} \Psi_j \times G_j + \sum_{j}^{m} \Phi_j \times H_j, \tag{3}$$

where G_j and H_j are the functions of the inequality constraints $g_i(x)$, and the equality constraints $h_j(x)$, respectively. ψ_j and Φ_j are the positive constants that are normally called "penalty factors".

There are different types of the penalty functions, such as: (1) static penalty [14]; (2) dynamic penalty [15]; (3) death penalty [16]; and (4) adaptive penalty [17]. The main problem with penalty functions is the "ideal" penalty factor which will be adopted. If the penalty is too high, the evolutionary algorithms will be pushed inside the feasible region quickly. On the other hand, if the penalty is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function [18].

In recent years, some modern constraint handling approaches, which use penalty functions deserve special consideration, since they are highly competitive. Runarsson and Yao [19] proposed SR method to achieve a balance, between objective function and the overall constraint violation, stochastically. An interesting aspect of the approach is that it doesn't require the definition of a penalty factor. Instead, it requires a user-defined parameter called p_f, which determines the balance between the objective function and the penalty function. The basic form of the method is presented in Table 1.

Table 1: Stochastic ranking.

if (no constraint violation or rand < p_f)
  rank based on the objective value only
else
  rank based on the constraint violation only
end

The main advantage of SR is its simplicity. However, it can't guarantee any expected diversity measures since its ranking is stochastic.

The ε-constraint handling method was proposed in [20] in which the relaxation of the constraints is controlled by using the ε parameter.

$$\varepsilon(k) = \begin{cases} \varepsilon(0)\left(1 - k/T_c\right)^{cp} & 0 < k < T_c, \\ 0 & k \geq T_c, \end{cases} \tag{4}$$

where k is the generation counter and T_c is the control generation. The recommended parameter ranges are $T_c \in [0.1\,T_{\max}, 0.8\,T_{\max}]$ and $cp \in [2, 10]$ [20]. The ε level comparisons are basically defined as a lexicographic order, in which the violation precedes the objective value, because the feasibility of a point x is more important than the minimization of its objective value [21].This method has the ability to

maintain reasonable diversity. However, extra parameters have been a negative issue for this method.

2.2. Special Representations and Operators. Some researchers have developed special representation schemes to tackle a certain (particularly difficult) problem for which a generic representation scheme might not be appropriate.

A more intriguing idea is to transform the whole feasible region into a different shape that is easier to explore. The most important approaches designed along these lines are the "homomorphous maps" [22]. This approach performs a homomorphous mapping between an n-dimensional cube and a feasible search space (either convex or non-convex). The main idea of this approach is to transform the original problem into another function that is easier to optimize by an evolutionary algorithm.

The Homomorphous Maps (HM) was the most competitive constraint handling approach for some time. However, the implementation of the algorithm is complex, and the experiments reported required a high number of fitness function evaluations [23].

2.3. Separation of Constraints and Objectives. Unlike penalty functions which combine the value of the objective function and the constraints of a problem to assign fitness, these approaches handle constraints and objectives separately.

Superiority of feasible points [24, 25], the idea of this technique is to assign always a higher fitness to feasible solutions. This approach considers solutions under the feasibility rules when selecting individuals. Feasible ones are always considered better than the infeasible ones. Two infeasible individuals are compared based on their constraint violations. Then the feasible ones are compared based on their objective function values only. The main disadvantage of this approach is the losses in diversity between solutions.

In [26], a multi-objective method with local and global search operators was proposed to solve constrained optimization problems. In this method, each constraint is treated as an objective to be optimized. One of the main drawbacks of this approach is the extra number of parameters and the time needed.

2.4. Hybrid Methods. Within this category, some methods may be considered when coupled with another technique.

Chen et al. [27] proposed a hybrid EA that integrates the penalty function method with the primal-dual method. This approach is based on sequential minimization of the Lagrangian method.

Bernardino et al. [28] proposed a GA hybridized with an AIS. The idea is to adopt as antigens some feasible solutions and evolve (in an inner GA) the antibodies (i.e., the infeasible solutions) so that they are "similar" (at a genotypic level) to the antigens.

3. Related Works

3.1. Classical ABC Algorithm. In a natural bee swarm, there are three kinds of honey bees to search foods generally, which include the employed bees, the onlookers, and the scouts (both the onlookers and the scouts are also called unemployed bees). The employed bees search the food around the food source in their memory, meanwhile they deliver their food information to the onlookers. The onlookers tend to select good food sources from those founded by the employed bees, then further search the foods around the selected food source. The scouts are translated from a few employed bees, which abandon their food sources and search new ones. In a word, the food search of bees is collectively performed by the employed bees, the onlookers, and the scouts [29].

By simulating the foraging behaviors of honey bee swarm, Karaboga recently invented ABC algorithm for numerical function optimization [6]. The pseudo code for the ABC algorithm is listed in Table 2 and the detail descriptions are given below.

In initialization phase, the algorithm generates a group of food sources corresponding to the solutions in the search space [30]. The food sources are produced randomly within the range of the boundaries of the variables.

$$x_{i,j} = x_j^{\min} + \text{rand}\,(0,1)\left(x_j^{\max} - x_j^{\min}\right), \tag{5}$$

where $i = 1, 2, \ldots, \text{SN}$, $j = 1, 2, \ldots, D$. SN is the number of food sources and equals to half of the colony size. D is the dimension of the problem, representing the number of parameters to be optimized. $x_j^{\min}$ and $x_j^{\max}$ are lower and upper bounds of the jth parameter. The fitness of food sources will be evaluated. Additionally, counters which store the number of trials of each bee are set to 0 in this phase.

In the employed bees' phase, each employed bee is sent to the food source in its memory and finds a neighboring food source. The neighboring food source is produced according to Equation (6) as follows.

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{k,j}\right), \tag{6}$$

where k is a randomly selected food source different from i, j is a randomly selected dimension. ϕ is a random number which is uniformly distributed in range $[-1, 1]$. The new food source v is determined by changing one dimension on x. If the value in this dimension produced by this operation exceeds its predetermined boundaries, it will set to be the boundaries.

The new food source is then evaluated. A greedy selection is applied on the original food source and the new one. The better one will be kept in the memory. The trials counter of this food will be reset to zero if the food source is improved, otherwise, its value will be incremented by one.

In the onlooker bees' phase, the onlookers receive the information of the food sources shared by employed bees. Then they will each choose a food source to exploit, depending on a probability related to the nectar amount of the food source (fitness values of the solution). That is to say, there may be more than one onlooker bee choosing the same food source if the source has a higher fitness. The probability is calculated according to Equation (7) as followed.

$$P_i = \frac{\text{fitness}_i}{\sum_{j=1}^{\text{SN}} \text{fitness}_j}. \tag{7}$$

Table 2: Original ABC algorithm.

```
1   Initialize the food sources and evaluate the nectar amount (fitness) of food sources
    Send the employed bees to the current food source
    Iteration = 0
2   Do while (the termination conditions are not met)
    2.1  /*Employed Bees' Phase*/
         for (each employed bee)
              Find a new food source in its neighborhood following the Equation (6)
              Evaluate the fitness of the new food source, apply greedy selection
         end for
    2.2  Calculate the probability P for each food source according to the Equation (7)
    2.3  /*Onlooker Bees' Phase*/
         for (each onlooker bee)
              Send onlooker bees to food sources depending on P
              Find a new food source in its neighborhood following the Equation (6)
              Evaluate the fitness of the new food source, apply greedy selection
         end for
    2.4  /*Scout Bees' Phase*/
         if (any employed bee becomes scout bee)
              Send the scout bee to a randomly produced food source
         end if
    2.5  Memorize the best solution achieved so far
         Iteration = Iteration +1
    end while
3   Output the best solution achieved
```

After food sources have been chosen, each onlooker bee finds a new food source in its neighborhood following Equation (6), just like the employed bee does. A greedy selection is applied on the new and original food sources, too.

In scout bees' phase, if a food source hasn't been improved for a predetermined cycle, which is a control parameter called "limit", the food source is abandoned and the bee becomes a scout bee. A new food source will be produced randomly in the search space using Equation (5), as in the case of initialization phase.

The employed, onlooker and scout bees' phase will recycle until the termination condition is met. The best food source which presents the best solution is then outputted.

3.2. Constraint Consensus Strategy. CC strategy, using a variety of projection algorithms to solve the feasibility problems with nonlinear and nonconvex constraints. The key idea is to help a currently infeasible solution to quickly move to the feasible region or move close to the feasible region by constructing a consensus among the currently violated constraints [7, 31, 32].

Before elaborating the content of CC strategy, it is more significant to demonstrate some concepts. The feasibility vector for an individual constraint is the vector extending from an infeasible point to its orthogonal projection on the constraint [33]. The measure, of how close an infeasible point to feasibility, is the minimum Euclidean distance between the point and the feasible region, referred to here as the feasibility distance [33].

The first step is to find the feasibility vector for each constraint that is violated at the current point x. The feasibility vector is calculated according to the following formula:

$$fv = \frac{v d \nabla c(x)}{\|\nabla c(x)\|^2}, \tag{8}$$

where $\nabla c(x)$ is the gradient of the constraint, and $\|\nabla c(x)\|$ is its length, v is the constraint violation $\nabla c(x) - b$, $v = 0$ for satisfied constraints, and b_i is the right side of the constraint, d is +1 if it is necessary to increase $c(x)$ to satisfy the constraint, and −1 if it is necessary to decrease $c(x)$ to satisfy the constraint.

The consensus vector is constructed by component-wise averaging of the feasibility vectors for the violated constraints. Let n_j represent the number of violated constraints that have variable x_j as a component, f_{ij} represents the component for variable x_j in the feasibility vector for the ith constraint, and s_j represents the sum of the f_{ij} for variable x_j over the feasibility vectors for all of the violated constraints. The component of the consensus vector for variable x_j is then given by s_j/n_j. If this vector is too short, then the iterations are halted with an unsuccessful outcome [7].

The constraint consensus algorithm is summarized in Table 3. The algorithm halts when every constraint has a constraint violation of zero or a feasibility distance is less than α, i.e., NINF is zero.

4. Constraint Consensus Update-Based ABC Algorithm

In this section, the overall flow chart of the proposed algorithm is introduced and discussed first.

Table 3: Constraint consensus algorithm.

Inputs:	a set of constraints
	an initial point x
	a feasibility distance tolerance α
	a movement tolerance β
1	NINF = 0, for all j: $n_j = 0$, $s_j = 0$
2	**for** every constraint c_i
	2.1 **if** c_i is violated
	2.1.1 Find the feasibility vector and the feasibility distance
	2.1.2 **if** the feasibility distance is greater than α
	NINF = NINF+1
	for every variable x_j in c_i
	$n_j \leftarrow n_j + 1$, $s_j \leftarrow s_j + f_{ij}$
	end for
	end if
	end if
	end for
3	**if** NINF = 0, then exist successfully
4	**for** every variable x_j
	$t_j = s_j / n_j$
	end for
5	**if** $\|t\| <= \beta$ then exit unsuccessfully
6	$x \leftarrow x + t$
7	**if** necessary, reset x to respect any violated variable bounds
8	Go to step1

4.1. Overall Flow Chart. The overall process of the proposed algorithm is illustrated in Figure 1. In Figure 1, a population with size *NP* is initialized stochastically. Suppose that there are *IF* infeasible individuals in the initialized population. New offspring of the *P* infeasible individuals, but not all *IF* infeasible individuals, are generated based on the CC strategy introduced in Section 3.2. The classical generation strategy of ABC algorithm is applied to remaining infeasible individuals and all feasible individuals with size *NP-P*. The aim is to save computing time, as well as to maintain the diversity of the population. For each newly generated individual, if it is better than its corresponding parent, it survives in the next generation.

The pseud-code of the algorithm is shown in Table 4 and will be explained below.

4.2. Detailed Steps of the ABCCC. In this algorithm, a new update operator that combines the concept of the CC method with the traditional mutation operator is proposed. This proposal will drive infeasible points to a better search space rather than random movements. The effect of this approach is to quickly reduce the total violation. Note that the proposed update operator will only be beneficial for infeasible individuals, and the update operator based on ABC method will be used for remaining individuals (including feasible and infeasible individuals).

First, the population is initialized by the following formula:

$$x_{i,d} = L_d + \text{rand}\,(0,1)(U_d - L_d), \tag{9}$$

where U_d and L_d are the upper and lower bound for decision variable $x_{i,d}$ and d is the dimension of the variables.

Then, for some of the infeasible solutions with size *P*, new offspring are generated as following:

$$u_{i,d} = x_{cc,d} + F^*(x_{cc,d} - x_{i,d}), \tag{10}$$

where x_{cc} is the value obtained by the constraint consensus strategy and F is a random factor in the range of [0.4, 0.9].

For the remaining *NP-P* individuals (some infeasible and all feasible ones), the update method based on ABC algorithm is adopted.

For each newly generated individual, if it is better than its corresponding parent, it survives in the next generation, and the whole population is sorted based on the fitness values or constraint violations. In this paper, individuals are selected based on the following criteria: (1) between two feasible solutions, the smaller fitness value is (for the minimization problem), the fitter individual is; (2) a feasible solution is always better than an infeasible one; and (3) between two infeasible solutions, the one having the smaller sum of constraint violation is selected.

Equality constraints are converted to inequalities by the following form, where ε is a small tolerance value, i.e., 10^{-4}.

$$\left|h_j(x)\right| - \varepsilon \le 0. \tag{11}$$

5. Results and Analysis

In this section, we present and analyze the performance of the proposed ABCCC algorithm.

5.1. Test Function. The algorithm was tested by solving the set of benchmark problems that were introduced in the CEC 2017 competition on Constrained Optimization [34].

These problems have different mathematical characteristics, such as the objective function or the constraints are either linear or nonlinear. The constraints are either equality or inequality type. The objective function is either unimodal or multimodal. And the feasible space may be very tiny compared to the search space.

The 10 test functions are listed below, where *D* is the number of decision variables, and both *o* and *M* are constants.

(1) C01

$$\begin{aligned} &\min\ f(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 \quad z = x - o, \\ &g(x) = \sum_{i=1}^{D}\left[z_i^2 - 5000\cos\left(0.1\pi z_i\right) - 4000\right] \le 0, \\ &x \in [-100, 100]^D. \end{aligned} \tag{12}$$

(2) C02

$$\begin{aligned} &\min\ f(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 \quad z = x - o,\ y = M^* z, \\ &g(x) = \sum_{i=1}^{D}\left[y_i^2 - 5000\cos\left(0.1\pi y_i\right) - 4000\right] \le 0, \\ &x \in [-100, 100]^D. \end{aligned} \tag{13}$$

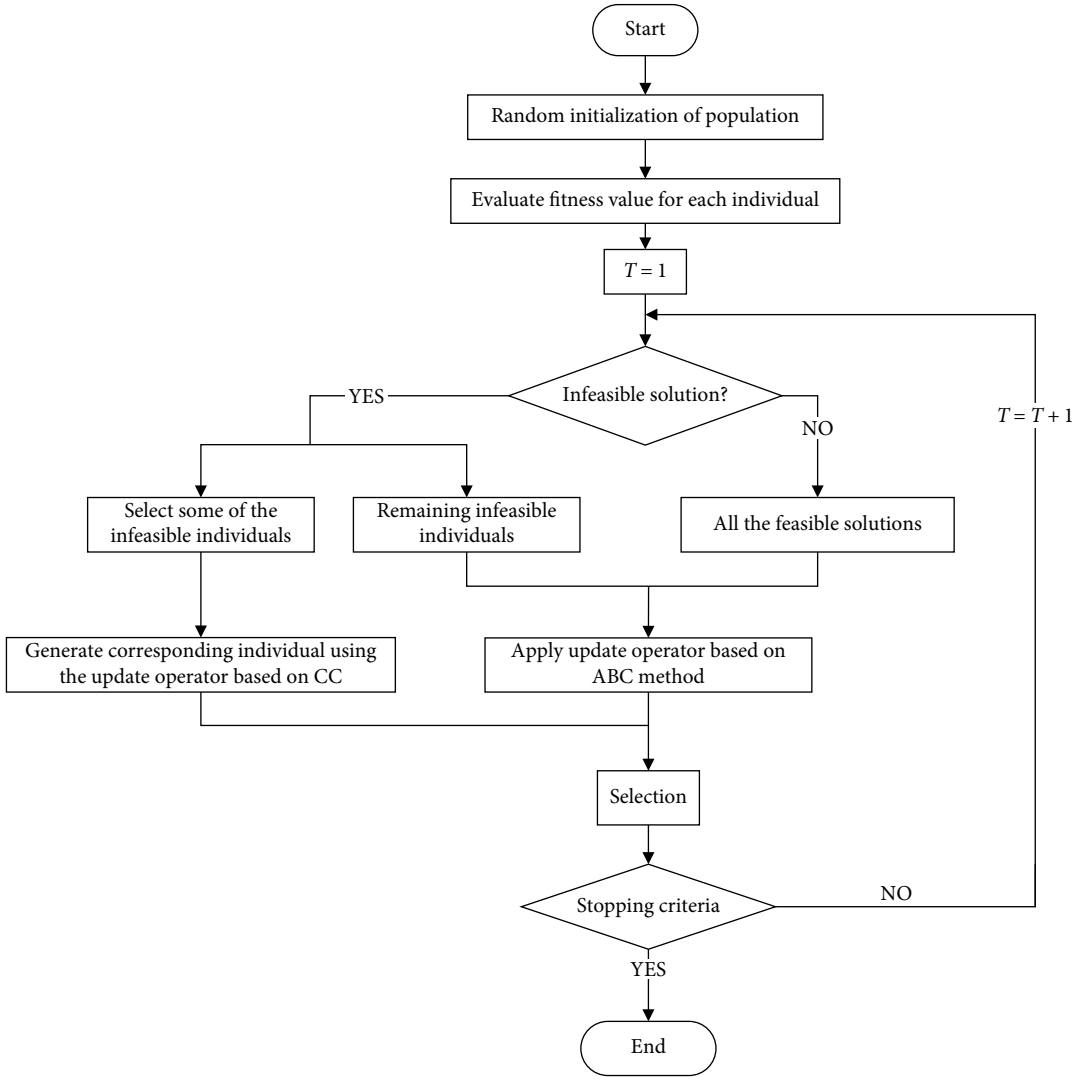


Figure 1: Overall flow chart of the proposed algorithm.

Table 4: Algorithm of the ABCCC.

1	Randomly initialize a population of size NP
	The variables in each individual are generated using equation (9)
	$t = 0$
2	**if** there are infeasible solutions
	2.1 Select P of the infeasible solutions, and generate new offspring using (10)
	2.2 **for** each individual in the remaining ($NP - P$) individuals
	New offspring are generated using mutation strategy based on ABC
	end for
	2.3 Sort the entire population based on the selection mechanism
	$t = t + 1$
	end if
3	Go to step 2

(3) C03

$$\begin{aligned}
&\min\ f(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 \quad z = x - o,\\
&g(x) = \sum_{i=1}^{D}\left[z_i^2 - 5000\cos\left(0.1\pi z_i\right) - 4000\right] \le 0,\\
&h(x) = -\sum_{i=1}^{D} z_i \sin\left(0.1\pi z_i\right) = 0,\\
&x \in [-100, 100]^D.
\end{aligned} \tag{14}$$

(4) C04

$$\begin{aligned}
&\min\ f(x) = \sum_{i=1}^{D}\left[z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right] \quad z = x - o,\\
&g_1(x) = -\sum_{i=1}^{D} z_i \sin\left(2z_i\right) \le 0,\\
&g_2(x) = \sum_{i=1}^{D} z_i \sin\left(z_i\right) \le 0,\\
&x \in [-10, 10]^D.
\end{aligned} \tag{15}$$

(5) C05

$$\begin{aligned}
&\min\ f(x) = \sum_{i=1}^{D-1}\left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right),\\
&z = x - o,\ y = M_1 * \mathrm{z},\ w = M_2 * z,\\
&g_1(x) = \sum_{i=1}^{D}\left[y_i^2 - 50\cos\left(2\pi y_i\right) - 40\right] \le 0,\\
&g_2(x) = \sum_{i=1}^{D}\left[w_i^2 - 50\cos\left(2\pi w_i\right) - 40\right] \le 0,\\
&x \in [-10, 10]^D.
\end{aligned} \tag{16}$$

(6) C06

$$\begin{aligned}
&\min\ f(x) = \sum_{i=1}^{D}\left[z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right] \quad z = x - o,\\
&h_1(x) = -\sum_{i=1}^{D} z_i \sin\left(z_i\right) = 0,\\
&h_2(x) = \sum_{i=1}^{D} z_i \sin\left(\pi z_i\right) = 0,\\
&h_3(x) = -\sum_{i=1}^{D} z_i \cos\left(z_i\right) = 0,\\
&h_4(x) = \sum_{i=1}^{D} z_i \cos\left(\pi z_i\right) = 0,\\
&h_5(x) = \sum_{i=1}^{D}\left(z_i \sin\left(2 * \sqrt{\left|z_i\right|}\right)\right) = 0,\\
&h_6(x) = -\sum_{i=1}^{D}\left(z_i \sin\left(2 * \sqrt{\left|z_i\right|}\right)\right) = 0,\\
&x \in [-20, 20]^D.
\end{aligned} \tag{17}$$

(7) C07

$$\begin{aligned}
&\min\ f(x) = \sum_{i=1}^{D}\left(z_i \sin\left(z_i\right)\right) \quad z = x - o,\\
&h_1(x) = \sum_{i=1}^{D}\left(z_i - 100\cos\left(0.5 z_i\right) + 100\right) = 0,\\
&h_2(x) = -\sum_{i=1}^{D}\left(z_i - 100\cos\left(0.5 z_i\right) + 100\right) = 0,\\
&x \in [-50, 50]^D.
\end{aligned} \tag{18}$$

(8) C08

$$\begin{aligned}
&\min\ f(x) = \max(z),\\
&z = x - o,\ y_l = z_{(2l-1)},\ w_l = z_{(2l)} \text{ where } l = 1, \ldots, \tfrac{D}{2},\\
&h_1(x) = \sum_{i=1}^{D/2}\left(\sum_{j=1}^{i} y_j\right)^2 = 0,\\
&h_2(x) = \sum_{i=1}^{D/2}\left(\sum_{j=1}^{i} w_j\right)^2 = 0,\\
&x \in [-100, 100]^D.
\end{aligned} \tag{19}$$

(9) C09

$$\begin{aligned}
&\min\ f(x) = \max(z),\\
&z = x - o, y_l = z_{(2l-1)}, w_l = z_{(2l)} \text{ where } l = 1, \ldots, \tfrac{D}{2},\\
&g(x) = \prod_{i=1}^{D/2} w_i \le 0,\\
&h(x) = \sum_{i=1}^{D/2-1}\left(y_i^2 - y_{i+1}\right)^2 = 0,\\
&x \in [-10, 10]^D.
\end{aligned} \tag{20}$$

(10) C10

$$\begin{aligned}
&\min\ f(x) = \max(z) \quad z = x - o,\\
&h_1(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 = 0,\\
&h_2(x) = \sum_{i=1}^{D-1}\left(z_i - z_{i+1}\right)^2 = 0,\\
&x \in [-100, 100]^D.
\end{aligned} \tag{21}$$

5.2. Experiments Settings. Parameters setting of the proposed ABCCC algorithm include population size *NP*, which was 100 and the percentage of the selected infeasible individuals, which was 50%. The random factor F was 0.5. The total number of iterations was set to 100 and the number of independent runs for each problem was set to 10. Parameters setting of the DECC and PSOCC algorithm were similar to those of ABCCC, specifically, the scaling factor F of DE was set to 0.5, and the crossover rate CR was 0.4. For PSO, the learning factors C_1 and C_2 were usually equal, and value was 1.4. The inertia weight w was set to 0.8. The parameters of CC strategy were set as follows: tolerance value of feasibility distance α was set to 10^{-6}; movement tolerance β was 0.0001 and a preset number of iterations μ was 10. For equality constraints the tolerance value ε was set to 10^{-4}.

According to the [33] , there are many variations of the original CC method which have been proposed, such as: (1) feasibility-distance-based (Fdnear/Fdfar) algorithms, (2) average direction-based (Dbavg) algorithm, (3) maximum direction-based (Dbmax) algorithm, (4) direction-based and bound-based (DBbnd) algorithm. They differ by the way they construct the consensus vector. In the basic CC method, all feasibility vectors are treated equally and the movement is created by averaging nonzero components of the feasibility vector. The feasibility-distance-based algorithms use the length of the feasibility vector associated with each violated constraint to set the consensus vector. In the "near" mode, the consensus vector is set equal to the shortest feasibility vector. In the "far" mode, the opposite is true. The Dbavg method decides the direction of movement in a dimension

Table 5: Comparison results of the proposed algorithm with other two methods.

Problem	Algorithm	10D			30D		
		Best	Mean	Std	Best	Mean	Std
C01	ABCCC	**4.3422e−01**	**1.0276e+00**	**7.7749e−01**	6.2693e+03	8.2442e+03	1.6027e+03
	DECC	2.0552e+01	5.7294e+01	2.6456e+01	8.7930e+03	1.3306e+04	2.3850e+03
	PSOCC	1.9394e+01	3.0953e+01	7.5116e+00	**1.2742e+03**	**1.6774e+03**	**2.2386e+02**
C02	ABCCC	**1.1917e−01**	**5.6705e−01**	**3.6207e−01**	1.4588e+03	3.4858e+03	1.0644e+03
	DECC	2.2081e+01	3.8531e+01	1.2447e+01	5.0072e+03	5.9380e+03	4.8469e+02
	PSOCC	2.8930e+01	3.6837e+01	4.6696e+00	**8.9951e+02**	**1.5818e+03**	**3.6121e+02**
C03	ABCCC	**2.6169e−01**	**7.2534e−01**	**4.5649e−01**	6.2305e+03	8.9104e+03	1.8025e+03
	DECC	1.9347e+01	7.9392e+01	2.6244e+01	1.0745e+04	1.3418e+04	1.9380e+03
	PSOCC	2.5698e+01	3.2883e+01	3.7946e+00	**1.4090e+03**	**2.1856e+03**	**7.4170e+02**
C04	ABCCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**5.9930e−09**	**4.4008e−08**	**5.5059e−08**
	DECC	1.7852e+01	3.2618e+01	7.5740e+00	2.3784e+02	2.5582e+02	1.1614e+01
	PSOCC	4.4996e+01	6.7187e+01	1.4799e+01	3.3885e+02	3.6265e+02	1.4894e+01
C05	ABCCC	**3.8795e−04**	**5.4388e−03**	**9.4572e−03**	**1.6887e−02**	**2.3190e−01**	**2.7175e−01**
	DECC	7.2160e+00	8.1310e+00	6.5960e−01	1.0310e+02	1.3773e+02	2.6676e+01
	PSOCC	2.0620e+01	4.6958e+01	3.0318e+01	3.0447e+02	4.4667e+02	1.3757e+02
C06	ABCCC	**0.0000e+00**	**1.9903e−01**	**6.2940e−01**	**7.8153e+02**	3.1383e+03	1.7131e+03
	DECC	1.1025e+02	1.5862e+02	5.6512e+01	1.4014e+03	1.8484e+03	**2.9160e+02**
	PSOCC	6.9656e+01	1.3079e+02	5.3494e+01	1.0215e+03	**1.5199e+03**	3.8129e+02
C07	ABCCC	**−5.2474e+02**	**−5.2474e+02**	**6.7115e-06**	**−8.9397e+02**	**−6.0124e+02**	1.8827e+02
	DECC	−3.1552e+02	−3.0261e+02	8.6554e+00	−3.4917e+02	−2.3050e+02	5.7684e+01
	PSOCC	−2.3716e+02	−2.0193e+02	2.3230e+01	−4.6299e+02	−3.4168e+02	**5.7643e+01**
C08	ABCCC	1.2499e+00	2.6492e+00	1.0700e+00	7.8818e+00	1.3153e+01	5.0152e+00
	DECC	**7.6596e−01**	**1.4086e+00**	**4.8360e−01**	1.1897e+01	1.5149e+01	2.4929e+00
	PSOCC	1.2276e+00	2.2839e+00	5.8413e−01	**7.3701e+00**	**9.6496e+00**	**1.4993e+00**
C09	ABCCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	**−3.0939e−01**	8.6376e−01
	DECC	1.8540e−01	3.2352e−01	1.2396e−01	2.9752e+00	3.4867e+00	**3.8672e−01**
	PSOCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	6.6104e−01	6.6047e−01
C10	ABCCC	1.0175e+01	1.5753e+01	5.1411e+00	4.4011e+01	5.1414e+01	4.9977e+00
	DECC	**4.5172e−01**	**1.1297e+00**	6.1977e−01	2.4098e+01	2.8066e+01	**2.0066e+00**
	PSOCC	1.3172e+00	2.3567e+00	**4.3120e−01**	**8.1363e+00**	**1.1921e+01**	2.4818e+00

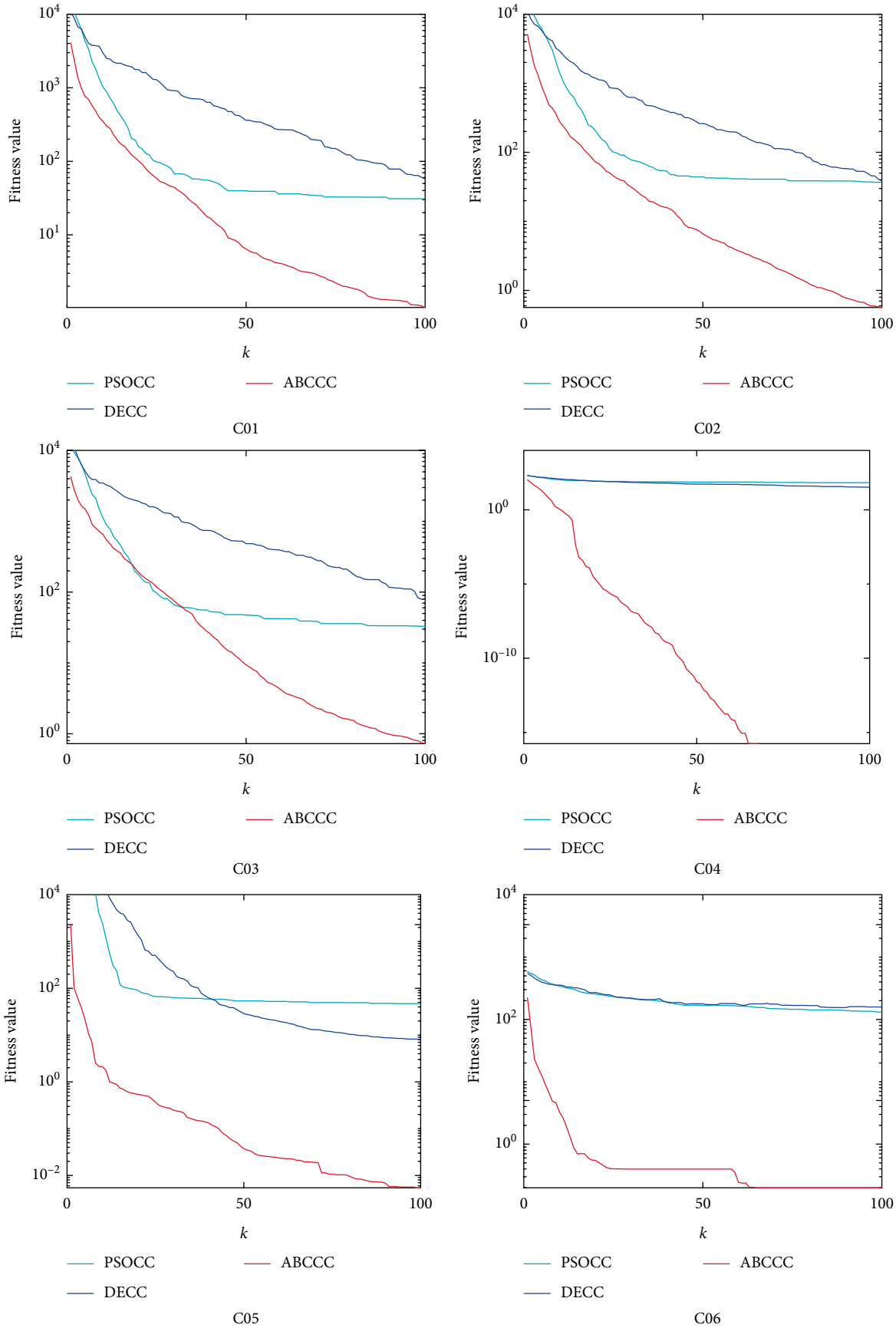
by a simple count of the number of votes for positive or negative movement, and the magnitude of the movement is decided by averaging the projections in the winning direction. Dbmax decides the direction of the movement based on the most common sign among the components of the feasibility vectors, whether positive or negative. Then, the largest proposed movement in the winning direction is set as the consensus vector. In the DBbnd method, the size of the movement in each component depends on the types of constraints that include that variable. Movements in the selected direction suggested by equality constraints are totaled, for inequalities only the largest movement in the selected direction is added [33].

Therefore, for some of the infeasible individuals with size P, the above methods are used respectively. In addition, the newly proposed algorithm (ABCCC) is compared with the original algorithm (DECC). Moreover, the DE optimizer of the original algorithm is also replaced with PSO. That is, all the remaining individuals (including feasible and infeasible) evolved with PSO. The results of the comparison are shown in the next section.

5.3. Results. In this section, ABCCC, PSOCC, and DECC are first tested. Then ABCCC is compared with different CC variants. Next, the effect of parameter P in ABCCC is analyzed. And next, ABCCC is compared with other advanced algorithms against CEC2017. Finally, the comparison results of ABCCC and ABC's variants are displayed.

5.3.1. ABCCC and Other EACC against CEC2017. The numerical comparison results for both 10-D and 30-D test problems are presented in Table 5. In order to observe the performance of the algorithm more intuitively, the convergence curves for each test function are shown in Figure 2. Box plot results are shown in Figure 3.

In the 10-D case, from Table 5, clearly, the proposed ABCCC algorithm could obtain more suitable solutions with smaller fitness values than PSOCC and DECC, especially for the best



(a)

Figure 2: Continued.

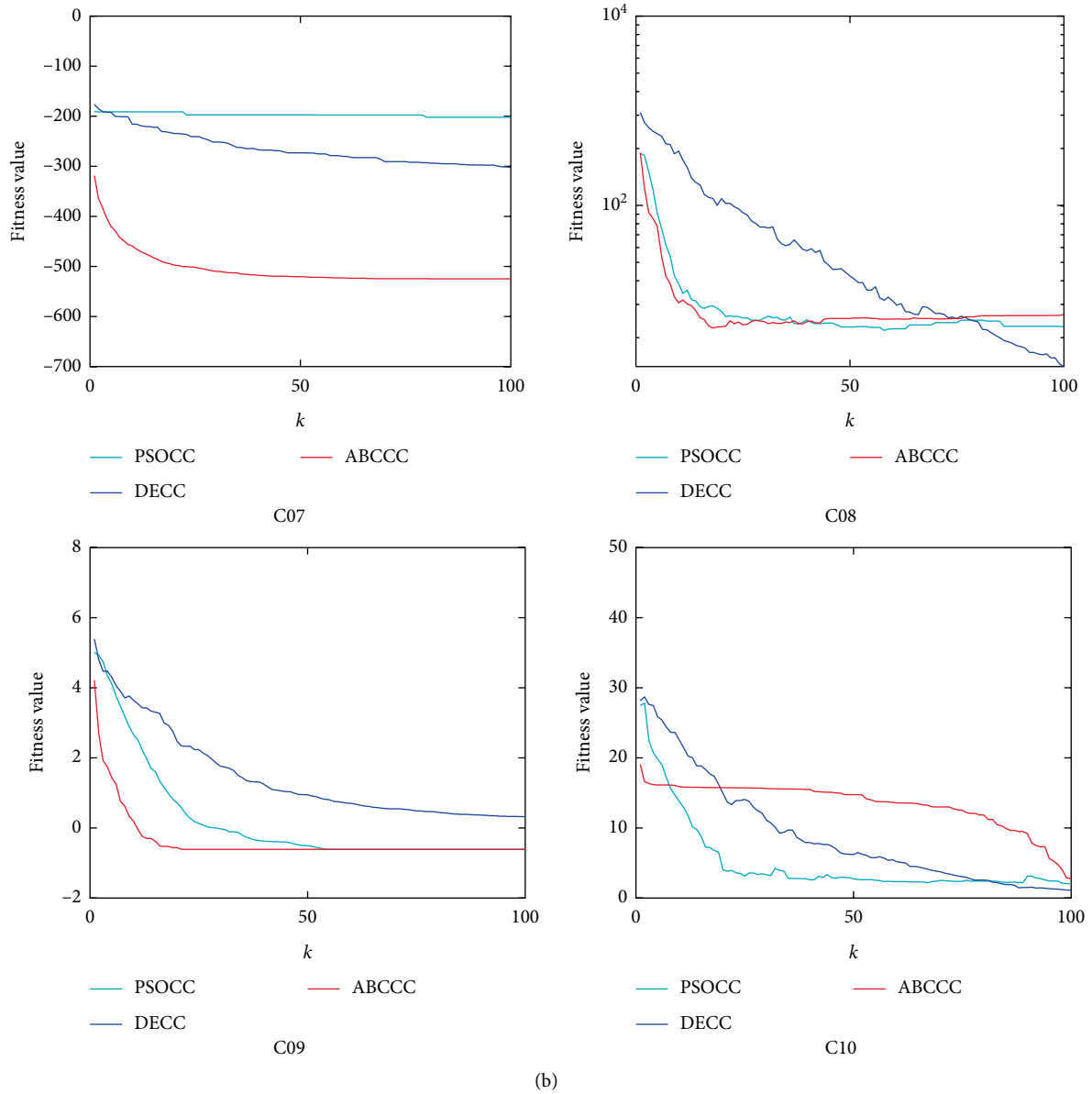


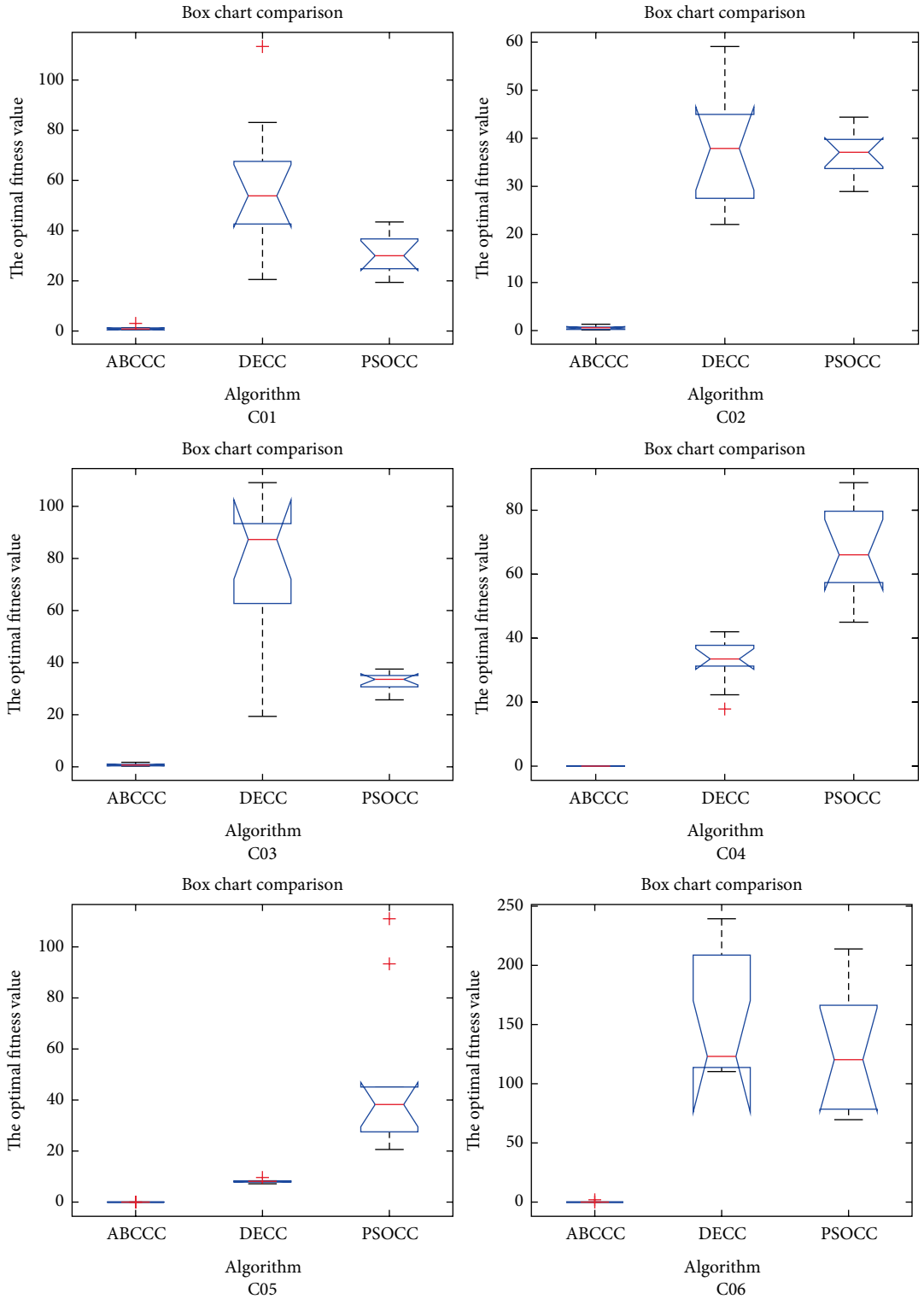
Figure 2: Average convergence of the algorithms (10-D).

and average results in most cases (C01, C02, C03, C04, C05, C06, C07, and C09). From the curves shown in Figure 2, it was proved that ABCCC owned a faster convergence rate than others and obtained the more ideal global solutions. Nevertheless, there were exceptions in a few test functions (C08 and C10). For these two functions, the optimal solutions were obtained by DECC.

With the increase of dimension, it was more difficult and time-consuming to find the minimum value by evolutionary algorithms. It was foreseeable that the results of 30-D were worse than that of 10-D. Table 5 demonstrated that, for the 30-D results, the proposed ABCCC algorithm displayed its instability. From Table 5, for C01, C02, and C03, the best solutions obtained by PSOCC were smaller than ABCCC and DECC. Then ABCCC outperformed DECC. For the remaining test functions (C04, C05, C06, C08, C09, and C10), ABCCC got the best solutions, but the solutions were distinctly larger than the ones in the case of 10 dimensions. For the special case C07 function, both the best value and the average value of 30-D was smaller than the ones of 10-D.

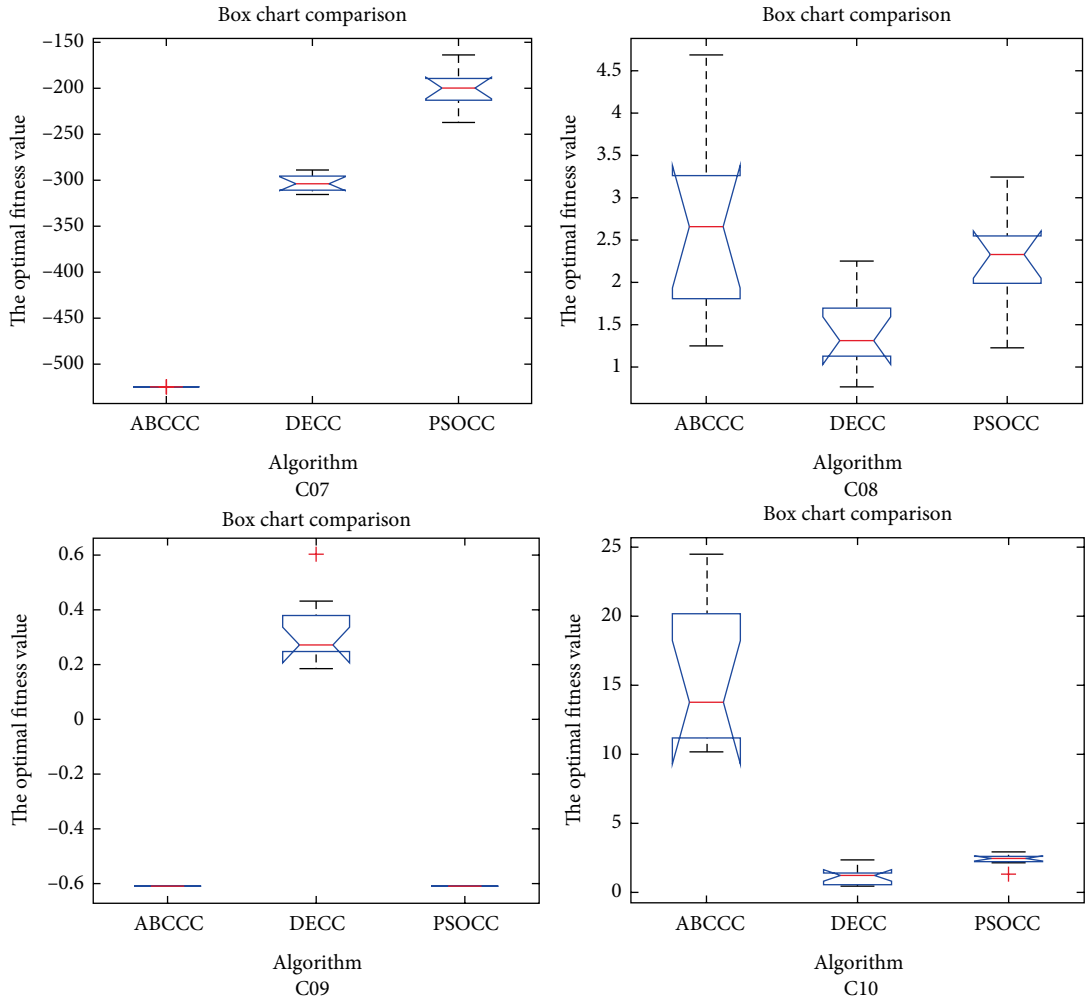
Figure 3 shows the box plots of the algorithms. For each box, the central mark indicates the median, and the bottom and top edges of the box indicates the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the "+" symbol. From Figure 3, ABCCC succeeded on most test functions (C01, C02, C03, C04, C05, C06, C07, and C09).

5.3.2. ABCCC with Different CC Variants Against CEC2017. The comparison results for ABCCC with different



(a)

Figure 3: Continued.



(b)

Figure 3: Box plot comparison results of the three algorithms (10-D).

CC variants for both 10-D and 30-D test problems are shown in Table 6. Figure 4 is the convergence curves for each test function. Figure 5 shows the box plot results of the algorithm.

From Table 6, there is no rule to follow for the occurrence of optimal solution. Different algorithms get similar results. Moreover, it was obvious that the optimum value of 30-D was larger than the value of 10-D except C07 for ABCCC and its variants. There was also not much difference between CC and its variants in 30-D results. From Figure 5, it was obviously that there were no regular patterns to judge which algorithm performed better within ABCCC and its variants.

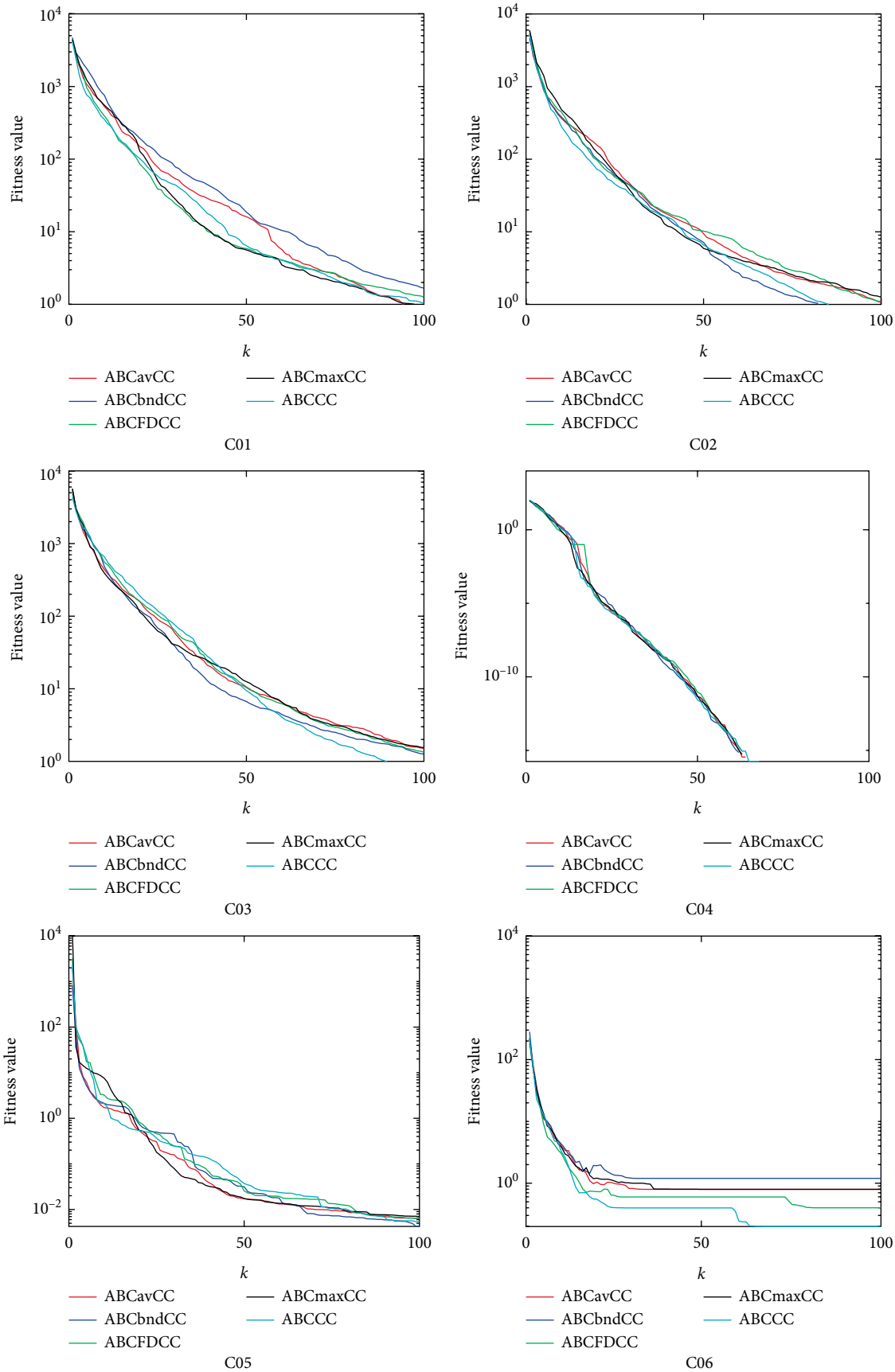
From Figure 4, intuitively, there was no obvious difference among the experimental results of ABCCC and its variants. In particular, for the test functions C04 and C09, the outcomes obtained by different variants were nearly the same because the global optimal solutions had been successfully found out. From Table 6, for C06, although all of ABCCC and its variants might probably find out the global optimal solutions (the same values of best), the mean values and the standard deviation were entirely different, which meant their robustness was various. For C07, it was a little different from C06. The optimal solutions (the same value of best and mean) might be obtained by ABCCC and its variants, but the standard deviation was distinct. In summary, the results indicated that, in most cases, ABCCC and its variants had similar effects.

5.3.3. Performance of ABCCC with Different Values of P against CEC2017. In this section, ABCCC was run with different percentages of infeasible individuals (P) in the whole individuals, where P is set as 10%, 30%, 50%, 70%, and 90% to evaluate their influence on results. The detailed results are shown in Table 7. It is expected that the search cycle will increase with higher values of P.

When comparing the search cycle, the ABCCC algorithm with $P = 10\%$ would be better as it costs minimum time to get results. Considering the quality of solutions, the performance of ABCCC with $P = 50\%$ is slightly better than most other cases based on the average fitness values in Table 7. According to the best fitness values, ABCCC with $P = 70\%$ is superior to ABCCCs with other values of P.

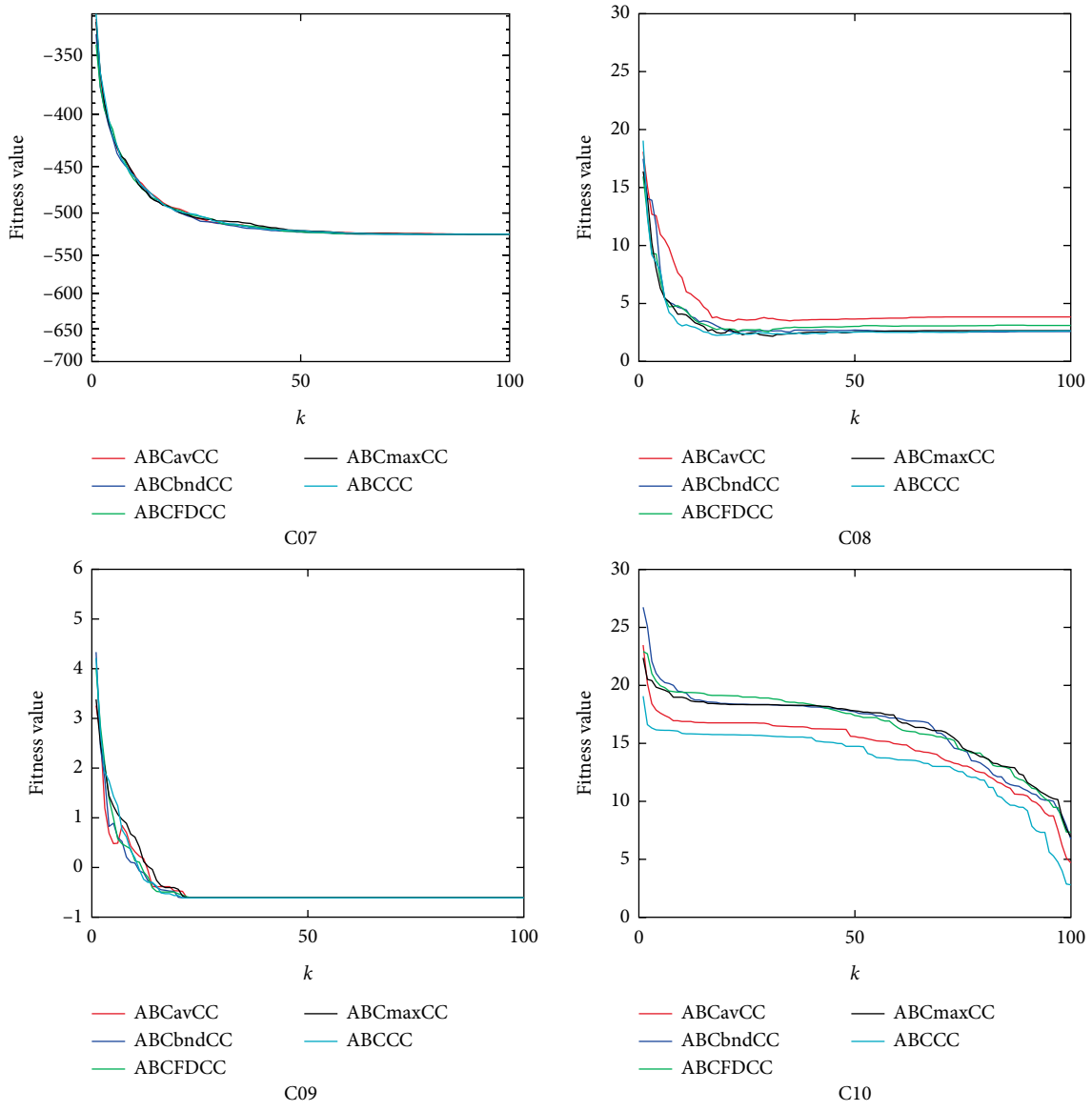
Table 6: Comparison results of the proposed algorithm with other variants.

Problem	Algorithm	10D			30D		
		Best	Mean	Std	Best	Mean	Std
C01	ABCCC	4.3422e−01	1.0276e+00	7.7749e−01	6.2693e+03	**8.2442e+03**	1.6027e+03
	ABCavCC	2.6666e−01	**8.1525e−01**	6.3591e−01	**5.3333e+03**	8.6063e+03	1.5452e+03
	ABCmaxCC	2.8739e−01	9.1758e−01	**5.7175e−01**	5.7557e+03	8.8971e+03	2.2734e+03
	ABCbndCC	**2.6214e−01**	1.6712e+00	1.6987e+00	5.7557e+03	8.8971e+03	2.2734e+03
	ABCFDCC	5.7396e−01	1.2627e+00	5.7950e−01	7.0198e+03	8.7193e+03	**1.2469e+03**
C02	ABCCC	1.1917e−01	5.6705e−01	3.6207e−01	1.4588e+03	**3.4858e+03**	1.0644e+03
	ABCavCC	1.5624e−01	1.0622e+00	8.9996e−01	1.6507e+03	3.6622e+03	1.3132e+03
	ABCmaxCC	8.9291e−02	1.2594e+00	1.8626e+00	2.8866e+03	4.2196e+03	**1.0230e+03**
	ABCbndCC	**7.6185e−02**	**4.6963e−01**	**3.1958e−01**	2.3144e+03	4.1829e+03	1.1693e+03
	ABCFDCC	1.7417e−01	1.0920e+00	9.4148e−01	**1.0826e+03**	4.3641e+03	1.9628e+03
C03	ABCCC	2.6169e−01	**7.2534e−01**	**4.5649e−01**	6.2305e+03	8.9104e+03	**1.8025e+03**
	ABCavCC	2.8291e−01	1.5084e+00	8.4892e−01	6.3652e+03	9.5822e+03	2.2970e+03
	ABCmaxCC	**1.8380e−01**	1.5424e+00	1.2224e+00	6.0213e+03	8.8483e+03	2.4095e+03
	ABCbndCC	1.8797e−01	1.2587e+00	1.2260e+00	**4.9709e+03**	7.7572e+03	1.9659e+03
	ABCFDCC	2.8988e−01	1.3396e+00	5.5769e−01	5.1170e+03	**7.2360e+03**	2.3147e+03
C04	ABCCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	5.9930e−09	4.4008e−08	5.5059e−08
	ABCavCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	8.0700e−09	**2.1109e−08**	1.4693e−08
	ABCmaxCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	2.5637e−09	5.0740e−08	9.2429e−08
	ABCbndCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	7.5989e−09	2.3105e−08	2.2866e−08
	ABCFDCC	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**2.3212e−09**	2.1475e−08	**1.1726e−08**
C05	ABCCC	3.8795e−04	5.4388e−03	9.4572e−03	1.6887e−02	2.3190e−01	2.7175e−01
	ABCavCC	**2.6890e−04**	6.3786e−03	1.2543e−02	7.1646e−03	2.9009e−01	4.2186e−01
	ABCmaxCC	4.5148e−04	7.0324e−03	1.4091e−02	5.4169e−03	6.5210e−01	1.2583e+00
	ABCbndCC	2.7664e−04	**4.2588e−03**	**3.2153e−03**	**3.8594e−03**	7.3446e−01	1.3437e+00
	ABCFDCC	4.3109e−04	6.3481e−03	5.0212e−03	6.0190e−03	**2.1450e−01**	**2.0751e−01**
C06	ABCCC	**0.0000e+00**	**1.9903e−01**	**6.2940e−01**	7.8153e+02	3.1383e+03	1.7131e+03
	ABCavCC	**0.0000e+00**	7.9625e−01	1.0280e+00	3.6692e+02	1.8784e+03	1.7233e+03
	ABCmaxCC	**0.0000e+00**	7.9609e−01	1.0278e+00	2.0786e+02	1.4075e+03	1.2753e+03
	ABCbndCC	**0.0000e+00**	1.1942e+00	1.0278e+00	1.9917e+02	**1.4675e+03**	**1.3062e+03**
	ABCFDCC	**0.0000e+00**	3.9994e−01	8.4317e−01	**1.6313e+02**	2.3210e+03	2.2130e+03
C07	ABCCC	**−5.2474e+02**	**−5.2474e+02**	6.7115e−06	−8.9397e+02	−6.0124e+02	1.8827e+02
	ABCavCC	**−5.2474e+02**	**−5.2474e+02**	1.0406e−02	−1.4919e+03	**−1.4823e+03**	**6.1156e+00**
	ABCmaxCC	**−5.2474e+02**	**−5.2474e+02**	3.0676e−08	**−1.4950e+03**	−1.4822e+03	1.0069e+01
	ABCbndCC	**−5.2474e+02**	**−5.2474e+02**	**1.3458e−11**	−9.0179e+02	−6.8494e+02	1.9621e+02
	ABCFDCC	**−5.2474e+02**	**−5.2474e+02**	1.3291e−08	**−1.4950e+03**	−1.4822e+03	1.0069e+01
C08	ABCCC	2.5150e+00	5.0474e+00	1.2945e+00	**7.8818e+00**	**1.3153e+01**	**5.0152e+00**
	ABCavCC	4.3560e+00	5.4574e+00	**1.0707e+00**	1.7922e+01	3.9656e+01	1.4675e+01
	ABCmaxCC	**9.5318e−01**	4.8754e+00	2.6297e+00	1.7232e+01	3.8949e+01	1.1115e+01
	ABCbndCC	1.4529e+00	4.5194e+00	2.1072e+00	1.1273e+01	1.9942e+01	6.4448e+00
	ABCFDCC	2.2534e+00	**3.9948e+00**	1.1559e+00	2.4065e+01	3.4540e+01	6.6039e+00
C09	ABCCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	**3.0939e−01**	8.6376e−01
	ABCavCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	4.7143e−01	6.8766e−01
	ABCmaxCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	3.2199e−01	8.5846e−01
	ABCbndCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	9.1733e−01	6.4206e−01
	ABCFDCC	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	**−6.0917e−01**	4.3180e−01	**6.1748e−01**
C10	ABCCC	1.0175e+01	**1.5753e+01**	5.1411e+00	4.4011e+01	5.1414e+01	4.9977e+00
	ABCavCC	**6.1631e+00**	1.6773e+01	5.8186e+00	4.2706e+01	5.0985e+01	5.8012e+00
	ABCmaxCC	1.3028e+01	1.8441e+01	**3.4762e+00**	4.9827e+01	6.1087e+01	**2.7835e+00**
	ABCbndCC	1.0416e+01	1.8453e+01	7.1632e+00	4.0410e+01	5.0126e+01	7.1920e+00
	ABCFDCC	9.1202e+00	1.6989e+01	4.8894e+00	**3.9870e+01**	**4.7673e+01**	3.7433e+00



(a)

Figure 4: Continued.



(b)

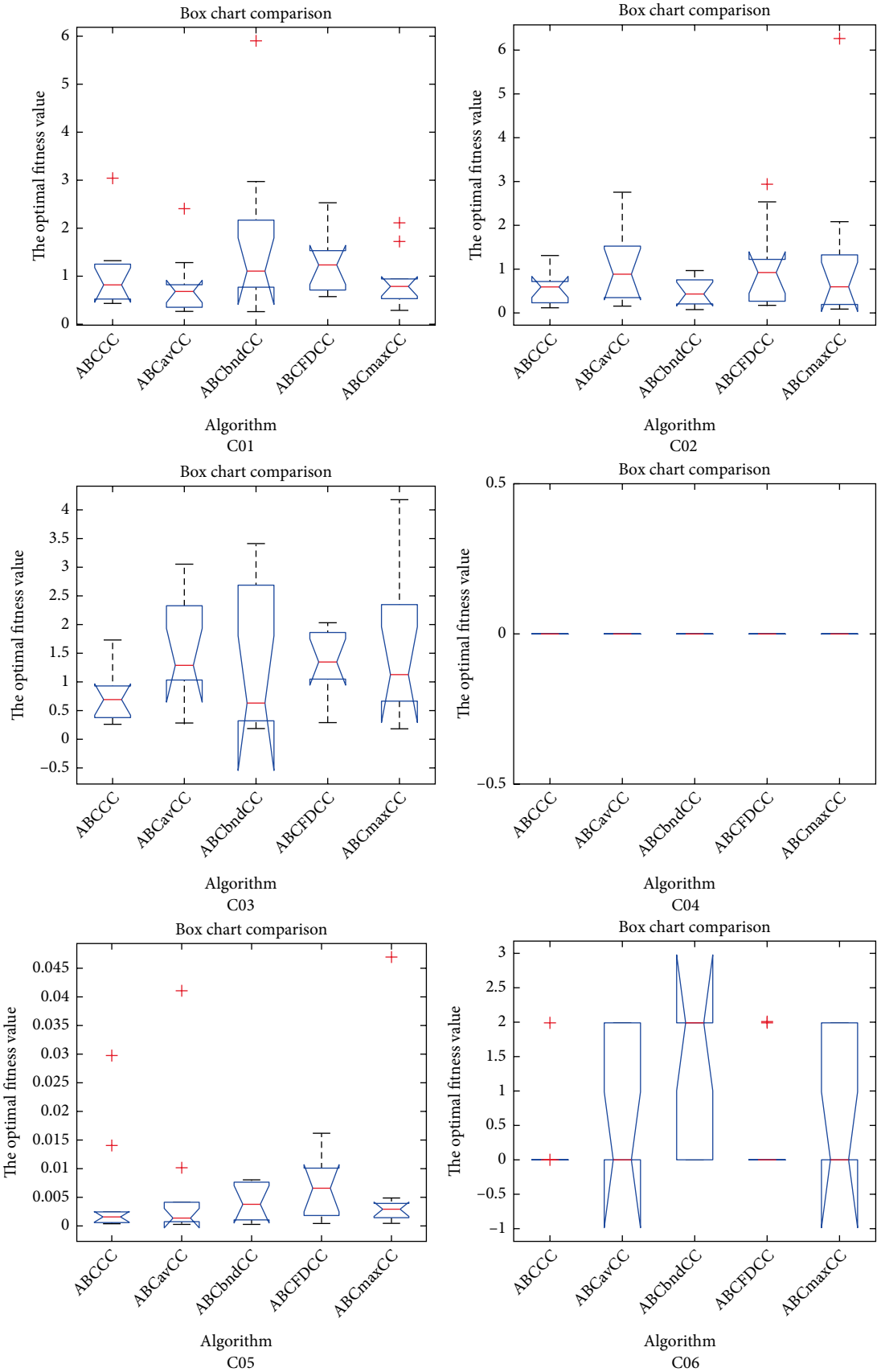
Figure 4: Average convergence of the five methods (10-D).

Figure 6 shows the convergence curves of C02 function, it is clear that the ABCCC with $P = 50\%$ convergences faster and obtains better optimal solution.

5.3.4. ABCCC Compared with CLSHADE and UDE against CEC2017. From the above mentioned analysis, ABCCC with $P = 50\%$ is regarded as the best algorithm. So we compare it with some famous constraint algorithms, such as CLSHADE [8] and UDE [9]. The detailed results are shown in Tables 8 and 9. From these tables, based on the fitness values for 10-D and 30-D test problems, ABCCC is able to obtain better optimal solutions for most test functions. But for 10-D or 30-D C01 and C02, CALSHADE gets the optimal solutions. For C08 and C10, UDE performs best with 10-D case, but CALSHADE performs better with 30-D case for C10.

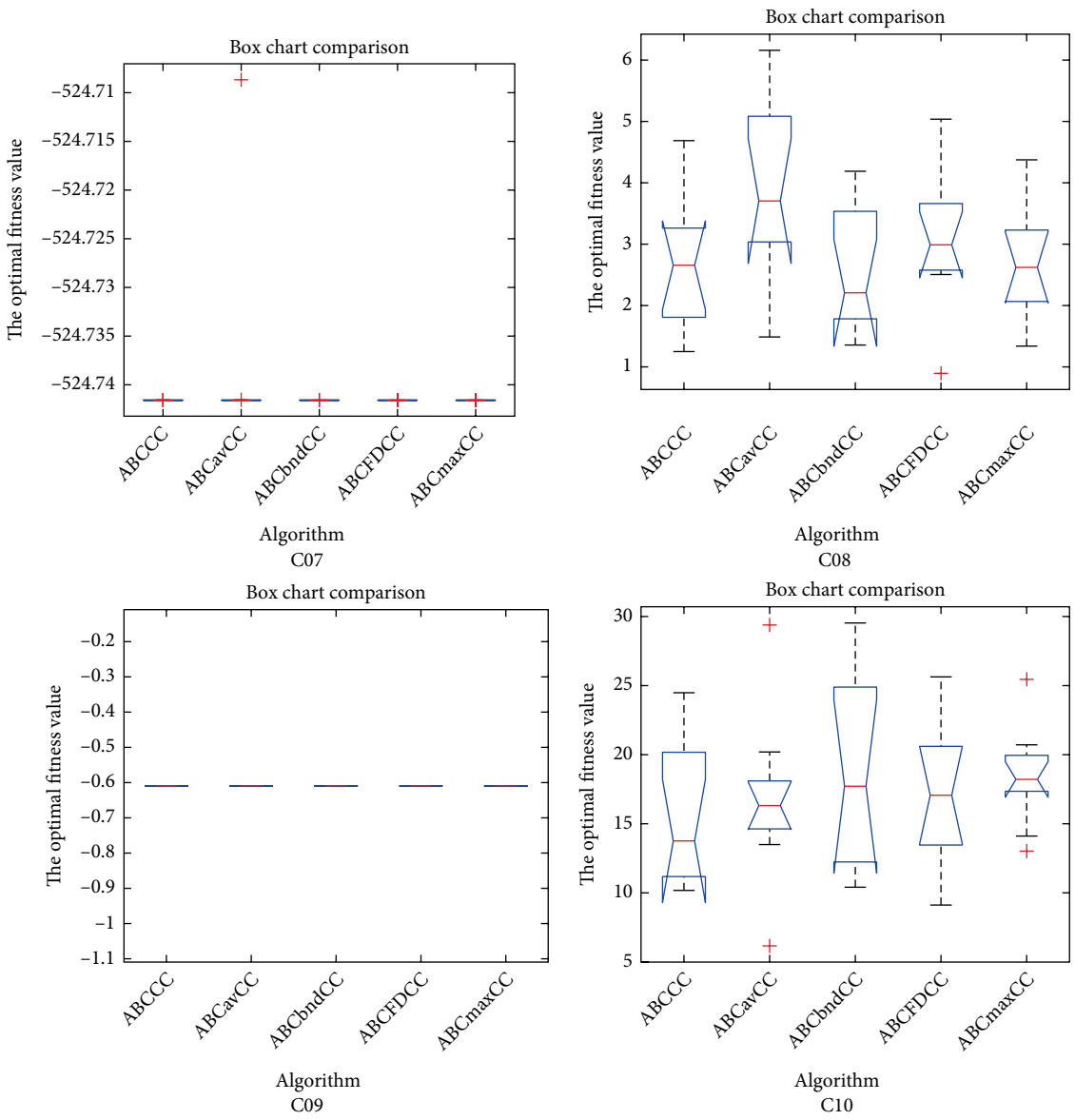
The results of our proposed algorithm are compared with ones of CLSHADE and UDE, which shows that the ABCCC algorithm outperforms the other algorithms.

5.3.5. ABCCC Compared with ABC's Variants with CC Strategy against CEC2017. Many researchers have proposed many other variants of ABC algorithm, such as CABC [10] and GABC [11]. The proposed ABCCC algorithm is also compared with ABC's variants with CC strategy against CEC2017. The results are shown in Tables 10 and 11. Vividly, ABCCC performs best for the most test functions except 10-D C01,



(a)

Figure 5: Continued.



(b)

Figure 5: Box plot comparison results of the algorithm and its variants (10-D).

C08, and C10 cases. The GABCCC gets the optimum for C01 and C08. For C10, CABCCC finds the final solution. When the dimension is 30, based on the average fitness values, ABCCC performs better than others.

GABCCC utilizes the global optimal value to mutate the individual in the search cycle, but lacks of diversity. Although CABCCC randomly selects dimensions and carries out mutation operations on individuals of each dimension, which also declines its convergence rate. During the test, the search cycle becomes longer and the results are not accurate enough.

6. Conclusion

During the last few decades, many evolutionary algorithms have been introduced to solve constrained optimization problems. However, it is the lack of effective constraint handling technologies in evolutionary optimization. In this research, a new updating strategy for the ABC algorithm has been introduced, which is inspired by the usefulness of the concepts constrained consensus (CC) approach. It is inevitable to make some considerable improvements of utilizing CC in evolutionary algorithms. To minimize the computational time and

Table 7: Comparison results of the proposed algorithm with different values of P (10-D).

	P-value	Best	Aver	Std		P-value	Best	Aver	Std
	$P = 0.1$	3.6224e−01	2.1745e+00	3.7883e+00		$P = 0.1$	**9.6252e−02**	7.5394e−01	5.1756e−01
	$P = 0.3$	6.6545e−01	1.0406e+00	**3.4463e−01**		$P = 0.3$	1.2425e−01	1.9530e+00	2.8007e+00
C01	$P = 0.5$	4.3422e−01	**1.0276e+00**	7.7749e−01	C02	$P = 0.5$	1.1917e−01	**5.6705e−01**	**3.6207e−01**
	$P = 0.7$	8.0072e−01	1.4671e+00	7.0296e−01		$P = 0.7$	2.6261e−01	9.3691e−01	1.2533e+00
	$P = 0.9$	**2.7358e−01**	1.2008e+00	6.8819e−01		$P = 0.9$	2.5149e−01	9.4132e−01	1.2486e+00
	$P = 0.1$	3.3249e−01	1.2147e+00	5.6815e−01		$P = 0.1$	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
	$P = 0.3$	5.2694e−01	1.3587e+00	7.7649e−01		$P = 0.3$	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
C03	$P = 0.5$	2.6169e−01	**7.2534e−01**	**4.5649e−01**	C04	$P = 0.5$	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
	$P = 0.7$	**1.7941e−01**	2.2968e+00	3.1205e+00		$P = 0.7$	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
	$P = 0.9$	3.1033e−01	3.1911e+00	3.8990e+00		$P = 0.9$	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
	$P = 0.1$	3.1579e−04	**3.0156e−03**	**3.4689e−03**		$P = 0.1$	**0.0000e+00**	2.0000e−01	6.3244e−01
	$P = 0.3$	1.3199e−03	1.2395e−02	1.7242e−02		$P = 0.3$	**0.0000e+00**	7.9602e−01	1.0277e+00
C05	$P = 0.5$	3.8795e−04	5.4388e−03	9.4572e−03	C06	$P = 0.5$	**0.0000e+00**	**1.9903e−01**	**6.2940e−01**
	$P = 0.7$	**1.4399e−04**	5.3629e−03	6.7180e−03		$P = 0.7$	**0.0000e+00**	1.9921e+00	1.3279e+00
	$P = 0.9$	6.0703e−04	1.2685e−02	2.9036e−02		$P = 0.9$	3.9851e+00	1.1887e+02	1.4430e+02
	$P = 0.1$	**−5.2474e+02**	−5.2474e+02	**2.1047e−07**		$P = 0.1$	5.1001e+00	8.7588e+00	2.7110e+00
	$P = 0.3$	**−5.2474e+02**	−5.2474e+02	2.6055e−07		$P = 0.3$	1.1888e+00	2.8243e+00	**9.0287e−01**
C07	$P = 0.5$	**−5.2474e+02**	−5.2474e+02	6.7115e−06	C08	$P = 0.5$	1.2499e+00	**2.6492e+00**	1.0700e+00
	$P = 0.7$	**−5.2474e+02**	**−5.2453e+02**	6.7970e−01		$P = 0.7$	2.5761e+00	4.3356e+00	1.0971e+00
	$P = 0.9$	**−5.2474e+02**	−5.2474e+02	9.0928e−06		$P = 0.9$	**6.5659e−01**	4.0359e+00	1.9499e+00
	$P = 0.1$	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**		$P = 0.1$	1.4054e+01	2.0605e+01	5.9842e+00
	$P = 0.3$	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**		$P = 0.3$	8.9854e+00	1.6984e+01	5.7525e+00
C09	$P = 0.5$	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	C10	$P = 0.5$	1.0175e+01	**1.5753e+01**	5.1411e+00
	$P = 0.7$	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**		$P = 0.7$	**8.3204e+00**	1.7478e+01	5.2565e+00
	$P = 0.9$	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**		$P = 0.9$	8.9586e+00	1.6326e+01	**4.7826e+00**

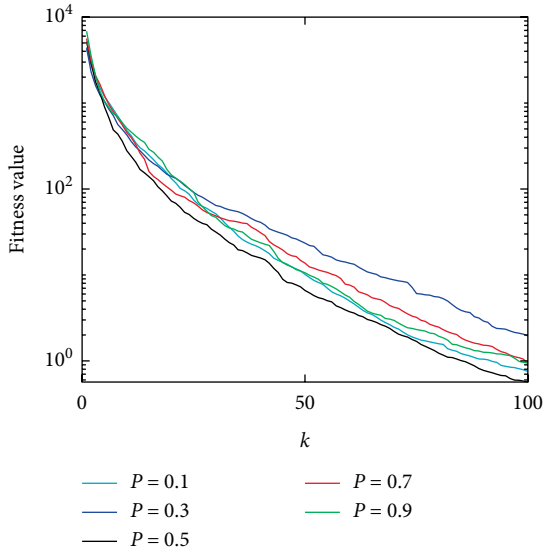


Figure 6: Convergence curves of ABCCC with different values of P (C02).

maintain a good diversity within the population, the new updating strategy based on CC method was applied to some of the infeasible individuals in each generation. The remaining individuals (including feasible and infeasible) were evolved by ABC method.

The new method was tested on a set of constrained problems. Experiments were compared with different variants of the approach. It also compared the performance of ABCCC with DECC and PSOCC. In most cases, the results of the algorithm show that ABCCC could obtain better results and converged faster than the other two methods. However, the CC method and some of its variants perform similarly in the test functions.

The proposed ABCCC algorithm was also compared with some famous constraint algorithms, such as CLSHADE [8] and UDE [9]. The experimental results showed that ABCCC algorithm performed well and could get the optimal solution. Furthermore, when ABCCC algorithm was compared with ABC's variants using CC strategy, the results of our algorithm were superior to those of the other algorithms.

Table 8: Comparison results of the proposed algorithm with CLSHADE and UDE (10-*D*).

	ABCCC			CALSHADE			UDE		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
C01	4.3422e−01	1.0276e+00	7.7749e−01	**1.6437e−02**	**1.7105e−01**	**2.6825e−01**	3.1160e+03	5.2019e+03	1.6872e+03
C02	1.1917e−01	5.6705e−01	3.6207e−01	**2.4550e−03**	**1.3606e−01**	**2.7692e−01**	2.4394e+03	4.3835e+03	1.6716e+03
C03	**2.6169e−01**	**7.2534e−01**	**4.5649e−01**	1.7797e+04	2.2553e+05	3.2910e+05	2.0650e+03	4.1730e+03	1.7840e+03
C04	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	1.8905e+01	3.4916e+01	1.0797e+01	1.0086e+02	1.3747e+02	1.9821e+01
C05	**3.8795e−04**	**5.4388e−03**	**9.4572e−03**	2.9752e+00	4.0586e+00	7.6361e−01	1.3716e+04	3.6528e+04	1.9615e+04
C06	**0.0000e+00**	**1.9903e−01**	**6.2940e−01**	1.6082e+02	3.7367e+02	2.1675e+02	2.1652e+02	3.0200e+02	5.8823e+01
C07	**−5.2474e+02**	**−5.2474e+02**	**6.7115e−06**	−2.1798e+02	−5.9272e+00	1.0154e+02	−4.0431e+02	−3.5367e+02	2.6542e+01
C08	1.2499e+00	2.6492e+00	1.0700e+00	1.5861e−02	8.6148e−02	7.8003e−02	**−9.0367e+01**	**−9.0367e+01**	**0.0000e+00**
C09	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	−4.9730e−03	1.8271e−01	3.5761e−01	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**
C10	1.0175e+01	1.5753e+01	5.1411e+00	8.7000e−05	4.6556e−03	3.1575e−03	**−5.9754e+01**	**−5.9754e+01**	**0.0000e+00**

Table 9: Comparison results of the proposed algorithm with CLSHADE and UDE (30-*D*).

	ABCCC			CALSHADE			UDE		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
C01	6.2693e+03	8.2442e+03	1.6027e+03	**1.2011e+03**	**2.5536e+03**	**9.5781e+02**	5.2924e+04	7.3012e+04	1.2711e+04
C02	1.4588e+03	3.4858e+03	1.0644e+03	**5.9671e+02**	**2.3085e+03**	**4.3565e+03**	4.3910e+04	6.1857e+04	1.3371e+04
C03	**6.2305e+03**	**8.9104e+03**	**1.8025e+03**	2.1618e+05	1.2755e+06	9.7969e+05	5.8671e+04	7.3478e+04	1.1760e+04
C04	**5.9930e−09**	**4.4008e−08**	**5.5059e−08**	1.5181e+02	1.9628e+02	3.3406e+01	5.9023e+02	7.7886e+02	8.2539e+01
C05	**1.6887e−02**	**2.3190e−01**	**2.7175e−01**	5.6912e+01	1.4006e+02	5.5703e+01	1.1546e+06	1.6625e+06	4.9473e+05
C06	**7.8153e+02**	3.1383e+03	1.7131e+03	2.0262e+03	4.1866e+03	1.0709e+03	1.8111e+03	**2.2144e+03**	**3.1697e+02**
C07	**−8.9397e+02**	−6.0124e+02	1.8827e+02	−1.7379e+02	−4.2553e+01	9.6391e+01	−8.4446e+02	**−7.0215e+02**	**1.1019e+02**
C08	7.8818e+00	**1.3153e+01**	**5.0152e+00**	1.6420e+01	2.9903e+01	9.7259e+00	**−9.0367e+01**	2.3797e+01	4.1907e+01
C09	**−6.0917e−01**	**−3.0939e−01**	**8.6376e−01**	3.0878e+00	9.9707e+00	4.4863e+00	4.3277e+00	5.8267e+00	1.2888e+00
C10	4.4011e+01	5.1414e+01	4.9977e+00	**5.4461e+00**	**9.9359e+00**	**3.9284e+00**	2.0576e+01	3.5824e+01	1.4403e+01

Table 10: Comparison results of the proposed algorithm with CABCCC and GABCCC (10-*D*).

	ABCCC			CABCCC			GABCCC		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
C01	4.3422e−01	1.0276e+00	7.7749e−01	4.0138e+02	1.1968e+03	4.4830e+02	**2.7349e−01**	**8.8519e−01**	**9.0434e−01**
C02	**1.1917e−01**	**5.6705e−01**	**3.6207e−01**	1.9751e+02	8.9848e+02	5.8191e+02	1.7921e−01	1.7533e+00	3.1740e+00
C03	**2.6169e−01**	**7.2534e−01**	**4.5649e−01**	3.8785e+02	7.8629e+02	2.6196e+02	2.8725e−01	1.2546e+00	6.3387e−01
C04	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	6.3515e+01	7.9414e+01	1.0983e+01	0.0000e+00	1.9362e−14	2.1747e−14
C05	**3.8795e−04**	**5.4388e−03**	**9.4572e−03**	7.5505e+02	9.1375e+03	7.7903e+03	4.9405e−04	3.8794e−02	6.2302e−02
C06	**0.0000e+00**	**1.9903e−01**	**6.2940e−01**	1.0994e+02	1.7150e+02	4.6959e+01	0.0000e+00	4.5917e+00	1.1764e+01
C07	**−5.2474e+02**	**−5.2474e+02**	**6.7115e−06**	−4.2280e+02	−3.8719e+02	2.4310e+01	−5.2446e+02	−5.2152e+02	2.8083e+00
C08	1.2499e+00	2.6492e+00	1.0700e+00	5.3728e+00	1.2295e+01	6.8359e+00	**1.1618e+00**	**2.3740e+00**	**1.3252e+00**
C09	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**	−6.0917e−01	1.8960e−01	7.5705e−01	**−6.0917e−01**	**−6.0917e−01**	**0.0000e+00**
C10	1.0175e+01	1.5753e+01	5.1411e+00	**7.7372e+00**	**1.5545e+01**	**7.2551e+00**	9.8156e+00	1.6622e+01	4.4697e+00

Table 11: Comparison results of the proposed algorithm with CABCCC and GABCCC (30-*D*).

	ABCCC			CABCCC			GABCCC		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
C01	6.2693e+03	8.2442e+03	1.6027e+03	1.2110e+04	1.7983e+04	4.2949e+03	**4.8615e+03**	**7.1251e+03**	**1.4398e+03**
C02	1.4588e+03	**3.4858e+03**	**1.0644e+03**	1.4360e+04	2.3061e+04	6.0050e+03	**1.0432e+03**	4.2680e+03	2.1291e+03
C03	6.2305e+03	**8.9104e+03**	1.8025e+03	2.4989e+04	4.0926e+04	1.0697e+04	**6.0663e+03**	9.1298e+03	**1.6006e+03**
C04	**5.9930e−09**	**4.4008e−08**	**5.5059e−08**	3.7553e+02	4.1474e+02	1.9197e+01	3.0818e−07	1.4143e−06	1.4172e−06
C05	**1.6887e−02**	**2.3190e−01**	**2.7175e−01**	1.8892e+05	3.6283e+05	9.8900e+04	3.2886e−02	3.9283e+00	4.4008e+00
C06	**7.8153e+02**	**3.1383e+03**	**1.7131e+03**	8.0127e+04	9.1573e+04	7.7743e+03	9.1011e+02	4.1485e+03	2.6910e+03
C07	**−8.9397e+02**	−6.0124e+02	1.8827e+02	−3.8285e+02	−3.0648e+02	**5.4701e+01**	−8.5701e+02	**−7.2916e+02**	9.0257e+01
C08	7.8818e+00	1.3153e+01	**5.0152e+00**	9.9695e+00	1.1728e+01	9.0648e+00	**5.0169e+00**	**6.7117e+00**	6.0091e+00
C09	**−6.0917e−01**	**−3.0939e−01**	8.6376e−01	1.9915e+00	3.0483e+00	**6.4821e−01**	**−6.0917e−01**	2.7884e−01	8.0281e−01
C10	4.4011e+01	5.1414e+01	**4.9977e+00**	**9.0911e+00**	**1.2839e+01**	5.0827e+00	3.8029e+01	4.2848e+01	3.3428e+01

For future work, we will analyze the effect of different parameter values on the performance of the algorithm and try to improve the proposed algorithm further.

Data Availability

The matlab data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the manuscript.

Acknowledgments

This work is supported by National key Research and Development Program of China under Grants nos. 2017YFB1103604, National Natural Science Foundation of China under Grants nos. 61602343, 61806143, and 41772123, Tianjin Province Science and Technology Projects under Grants nos. 17JCQNJC04500 and 17JCYBJC15100, and Basic Scientific Research Business Funded Projects of Tianjin (2017KJ093, 2017KJ094).

References

[1] Y. Wang, B. Xu, G. Sun, and S. Yang, "A two-phase differential evolution for uniform designs in constrained experimental domains," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 665–680, 2017.

[2] T. Tušar, P. Korošec, G. Papa, B. Filipič, and J. Šilc, "A comparative study of stochastic optimization methods in electric motor design," *Applied Intelligence*, vol. 27, no. 2, pp. 101–111, 2007.

[3] E. Cuevas, M. A. D. Cortés, and D. A. O. Navarro, "Social-spider algorithm for constrained optimization," *Advances of Evolutionary Computation: Methods and Operators*, Springer, pp. 175–202, 2016.

[4] G. Bai, Z. Xie, and L. Wang, "Practical constrained optimization of auction mechanisms in E-commerce sponsored search advertising," *Computer Science and Game Theory*, 2018, https://arxiv.org/abs/1807.11790.

[5] A. Antoniou and L. Wu-Sheng, *Practical Optimization Algorithms and Engineering Applications*, Springer, Boston, USA, pp. 231–263, 2007.

[6] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[7] J. W. Chinneck, "The constraint consensus method for finding approximately feasible points in nonlinear programs," *INFORMS Journal on Computing*, vol. 16, no. 3, pp. 255–265, 2004.

[8] A. Zamuda, "Adaptive constraint handling and success history differential evolution for CEC 2017 constrained real-parameter optimization," Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia, 2017.

[9] T. Anupam, S. Krishnendu, V. Pranjal, and S. Dipti, "Unified differential evolution algorithm for constrained optimization problems," National University, Singapore, 2017.

[10] W. F. Gao, S. Y. Liu, and L. L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1011–1024, 2013.

[11] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[12] Alice E. Smith and David W. Coit, "Constraint handling techniques-penalty functions chapter C 5.2," *Handbook of Evolutionary Computation*, Oxford University Press and Institute of Physics Publishing, 1997.

[13] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11–12, pp. 1245–1287, 2002.

[14] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–253, 1994.

[15] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, IEEE, pp. 579–584, Orlando, FL, USA, 1994.

[16] A. B. Hadj-Alouane and J. C. Bean, "A Genetic algorithm for the multiple-choice integer program," *Operations Research*, vol. 45, no. 1, pp. 92–101, 1997.

[17] J. C. Bean and A. B. Hadj-Alouane, "A dual genetic algorithm for bounded integer programs," Department of Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI, USA, 1992, TR 92–53.

[18] A. M. A. C. Rocha and R. Vilaca, "A computational study on different penalty functions with DIRECT algorithm," *ICCSA 2013: Computational Science and its Applications – ICCSA*, Springer, Berlin, Heidelberg, pp. 318–332, 2013.

[19] 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.

[20] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with gradient-based mutation and feasible elites," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1–8, IEEE, Vancouver, BC, Canada, 2006.

[21] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[22] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[23] C. K. Monson and K. D. Seppi, "Linear equality constraints and homomorphous mappings in PSO," in *2005 IEEE Congress on Evolutionary Computation*, IEEE, Edinburgh, Scotland, UK, 2005.

[24] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[25] D. Powell and M. Skolnick, "Using genetic algorithms in engineering design optimization with nonlinear constraints,"

in *Proceedings of the 5th International Conference of Genetic Algorithms*, pp. 424–431, Morgan Kaufmann, San Mateo, CA, 1993.

[26] Y. Wang, Z. Cai, G. Guo, and Y. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 3, pp. 560–575, 2007.

[27] P. Chen, J. Huang, and X. Zhang, "A primal-dual fixed point algorithm for minimization of the sum of three convex separable functions," *Fixed Point Theory and Applications*, vol. 2016, no. 1, Springer, 2016.

[28] H. S. Bernardino, H. J. C. Barbosa, A. C. C. Lemonge, and L. G. Fonseca, "A new hybrid AIS-GA for constrained optimization problems in mechanical engineering," in *IEEE Congress on Evolutionary Computation*, IEEE, Hong Kong, China, 2008.

[29] M. EI-Abd, "A cooperative approach to the artificial bee colony algorithm," in *IEEE Congress on Evolutionary Computation*, IEEEBarcelona, Spain, 2010.

[30] L. Ma, H. Kunyuan, Y. Zhu, and H. Chen, "Cooperative artificial bee colony algorithm for multi-objective RFID network planning," *Journal of Network and Computer Applications*, vol. 42, pp. 143–162, 2014.

[31] N. M. Hamza, D. L. Essam, and R. A. Sarker, "Constraint consensus mutation-based differential evolution for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 447–459, 2016.

[32] N. M. Hamza, S. M. Elsayed, D. L. Essam, and R. A. Sarker, "Differential evolution combined with constraint consensus for constrained Optimization," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 865–872, IEEE, New Orleans, LA, USA, 2011.

[33] W. Ibrahim and J. W. Chinneck, "Improving solver success in reaching feasibility for sets of nonlinear constraints," *Computers & Operations Research*, vol. 35, no. 5, pp. 1394–1411, 2008.

[34] W. Guohua, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, 2010.