Hindawi
Complexity
Volume 2019, Article ID 7125037, 18 pages
https://doi.org/10.1155/2019/7125037

Research Article

Efficient Conical Area Differential Evolution with Biased Decomposition and Dual Populations for Constrained Optimization

Weiqin Ying[1], **Bin Wu**[1], **Yu Wu**[2], **Yali Deng**[1], **Hainan Huang**[1] and **Zhenyu Wang**[1]

[1] *School of Software Engineering, South China University of Technology, Guangzhou 510006, China*
[2] *School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China*

Correspondence should be addressed to Weiqin Ying; yingweiqin@scut.edu.cn and Yu Wu; wuyu@gzhu.edu.cn

Received 13 July 2018; Revised 9 October 2018; Accepted 27 January 2019; Published 20 February 2019

Guest Editor: Wenbo Wang

The constraint-handling methods using multiobjective techniques in evolutionary algorithms have drawn increasing attention from researchers. This paper proposes an efficient conical area differential evolution (CADE) algorithm, which employs biased decomposition and dual populations for constrained optimization by borrowing the idea of cone decomposition for multiobjective optimization. In this approach, a conical subpopulation and a feasible subpopulation are designed to search for the global feasible optimum, along the Pareto front and the feasible segment, respectively, in a cooperative way. In particular, the conical subpopulation aims to efficiently construct and utilize the Pareto front through a biased cone decomposition strategy and conical area indicator. Neighbors in the conical subpopulation are fully exploited to assist each other to find the global feasible optimum. Afterwards, the feasible subpopulation is ranked and updated according to a tolerance-based rule to heighten its diversity in the early stage of evolution. Experimental results on 24 benchmark test cases reveal that CADE is capable of resolving the constrained optimization problems more efficiently as well as producing solutions that are significantly competitive with other popular approaches.

1. Introduction

Most real-world optimization problems are subject to different types of constraints, and these problems are regarded as the constrained optimization problems (COPs) [1–6]. In general, a COP can be expressed as follows:

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\textit{subject to} && g_i(\mathbf{x}) \le 0, \quad i = 1, 2, \ldots, q \\ &&& h_j(\mathbf{x}) = 0, \quad j = q+1, \ldots, l \\ &&& \mathbf{x} = \{x_1, x_2, \ldots, x_n\} \in \Omega \end{aligned} \tag{1}$$

in which Ω is the decision space and x_i is the i-th decision vector with lower bound L_i as well as upper bound U_i. In Eq. (1), $g_i \le 0$ and $h_j = 0$ are, respectively, the i-th inequality constraint and the j-th equality constraint. The feasible region $\Omega_f \subseteq \Omega$ can be expressed as follows:

$$\Omega_f = \{\mathbf{x} \mid g_i(\mathbf{x}) \le 0,\ i = 1, \ldots, q \wedge h_j(\mathbf{x}) = 0,\ j = q+1, \ldots, l\} \tag{2}$$

and the unfeasible region is $\Omega_u = \Omega - \Omega_f$. Given a point $\mathbf{x}' \in \Omega_f$, if $g_i(\mathbf{x}') = 0$ or $h_j(\mathbf{x}') = 0$, the corresponding constraint is regarded to be "active" at the point $\mathbf{x}'$.

In the past few years, because of their outstanding performance, evolutionary algorithms (EAs) [7–12] have been widely used to handle COPs. At the same time, differential evolution (DE) has been employed to produce promising offspring in a population. As unconstrained optimization technique, EAs require additional mechanisms to resolve

constraints. Thus, plenty of constraint-handling methods for EAs have been proposed.

A large number of studies have employed DE as a promising offspring generator to resolve COPs. A hybrid approach using DE was proposed in [13], in which two additional operations are applied to the primary DE. A diversity mechanism has also been incorporated into DE [14] so that the infeasible solutions with promising objective function values are capable of evolving in the next generation. Based on cultural DE operators, a method for constrained optimization proposed by Becerra and Coello [15] maintains both population space and belief space. Dividing the population into several subpopulations to do parallel search, a multipopulated DE algorithm (MPDE) [16] was developed by Tasgetiren and Suganthan. To handle nonlinear constraint functions, Lampinen [17] extended DE to select the better option in constraint space when both trial vector and target vector are infeasible. A generalized DE (GDE) [18] was presented to handle COPs, in which the trial vector takes the place of the target vector when the trial vector dominates it weakly. A DE control parameter [19] was suggested by Brest *et al.* for picking one of three DE mutation strategies based on their previous performance. In [20], a novel mutation operator was applied to DE to combine the best individual and present parent to favor search in promising directions. Moreover, a modified selection procedure is employed to favor feasible over infeasible individuals for constraint-handling in DE [21]. Besides, selecting one out of several available learning strategies according to adaptive probabilities, a self-adaptive DE (SaDE) [22] utilizes gradient information to speed up the convergence of constrained optimization.

In general, the constraint-handling methods for constrained optimization are classified into three categories: (1) methods using penalty functions, (2) methods using the feasibility rule, and (3) methods using multiobjective techniques. The methods using penalty functions aim to make infeasible solutions less likely survive into the next generation than feasible solutions by a penalty related to the constraint violation. The methods using the feasibility rule [23] select the better individual between a pair of given solutions according to the following rule: (1) a feasible individual is preferred over an infeasible one; (2) when both solutions are feasible, the one with the better objective function value is picked; (3) when both solutions are infeasible, the one with a lower degree of constraint violation is chosen. Stochastic ranking (SR) [24] is a method suggested by Runarsson and Yao, which combines the penalty functions and feasibility rule to solve COPs. Many variants of SR have been developed such as stochastic ranking differential evolution algorithm (SRDE) [25], annealing stochastic ranking algorithm (ASR) [26], and differential evolution-based algorithm for constrained global optimization (CDE) [27]. All of these methods are capable of improving the search performance significantly, but they still cannot find a global optimal solution for some complex problems.

The constraint-handling methods using multiobjective techniques transform a constrained optimization problem with single objective into an unconstrained multiobjective optimization problem in order that multiobjective optimization techniques can be applied to solve it. It is beneficial to handle constrained single objective optimization problems by applying multiobjective techniques for the reason that multiobjective techniques help the population maintain better diversity [28, 29]. In [30], methods converting a COP into a multiobjective optimization problem (MOP) are divided into two categories: turning a COP into a biobjective optimization problem (BOP) and transforming a COP into a MOP, in which each constraint violation becomes an objective function. Kalanmoy [31] suggested a method that combines a biobjective evolutionary approach and the penalty function technique in a complementary way. However, this method is inefficient because of the exchange between the biobjective evolutionary method and the classical penalty function technique. The method of combining multiobjective optimization with differential evolution (CMODE) [32] is a successful algorithm with the state-of-the-art performance. An infeasible solution replacement mechanism has been utilized in CMODE for the purpose of improving the quality and feasibility of individuals.

However, the existing constraint-handling methods using multiobjective techniques, such as CMODE, borrow the ideas only from the inefficient dominance-based multiobjective techniques. As a result, in every generation of CMODE, λ offspring are generated once and it is necessary to perform nondominated sorting between the λ offspring and their parents. So far, there has been no efficient method of nondominated sorting, which results in a high computational complexity of CMODE. In addition, CMODE does not have a scheme to construct the Pareto front (PF) systematically for directing the search.

In the past years, more and more decomposition-based multiobjective techniques such as the multiobjective evolutionary algorithm based on decomposition (MOEA/D) [33] have been proposed for the purpose of solving unconstrained MOPs. Decomposition-based multiobjective techniques avoid inefficient nondominated sorting through converting a MOP into a series of scalar objective optimization subproblems and, in general, exhibit great advantages over dominance-based multiobjective techniques. Recently, a conical area EA (CAEA) [34] has achieved higher performance and efficiency than the other popular decomposition-based techniques by dividing an unconstrained BOP into a number of scalar subproblems as well as assigning each subproblem an exclusive subset.

However, as unconstrained multiobjective optimization techniques, decomposition-based multiobjective techniques such as CAEA cannot be utilized to solve COPs directly and require some additional constraint-handling strategies to guide the populations to search towards the optimal feasible solutions. As a consequence, up to now the advantages of decomposition-based multiobjective techniques have not been exploited by any constraint-handling method using multiobjective techniques.

In this paper, a conical area DE (CADE) algorithm is proposed to take advantages of decomposition-based multiobjective techniques to improve both performance and running efficiency of EAs for constraint optimization by

borrowing the idea of an excellent decomposition-based multiobjective technique, CAEA. For the gains of performance, CADE adopts a dual-population scheme in which a feasible subpopulation and a conical subpopulation are designed to search for the optimal feasible solution, respectively, along the feasible segment and the Pareto front (PF). For the improvements of efficiency, due to a cone decomposition strategy, CADE is able to avoid nondominated sorting and update the conical subpopulation in a much more efficient way. In addition, CADE can get a better diversity of population for COPs by building the PF. To produce promising individuals, a self-adaptive parameter for the DE selection operator is also utilized to ensure that CADE can generate a more promising offspring.

This paper is a revised and expanded version of a conference paper by Wu and Ying et al. [35] and reports the motivation and method in more detail as well as a significant amount of additional experimental results and analyses corresponding to a wider range of benchmark test instances for constrained optimization. This paper is organized as follows: Section 2 introduces the advantages and motivation of using decomposition-based multiobjective techniques to handle constraints. In Section 3, the biased cone decomposition and dual populations in CADE are presented in detail. Section 4 describes the CADE framework and procedure. Empirical results for and an analysis of 24 benchmark COPs are provided in Section 5. Finally, Section 6 concludes this paper.

2. Preliminaries

2.1. Decomposition Approaches for MOEAs. Decomposition-based MOEAs, such as MOEA/D, explicitly decompose the task of approximating the PF of a MOP into N subtasks, i.e., scalar objective optimization subproblems. Each individual in the population is in charge of optimizing a different subproblem. Meanwhile, the current solutions to its neighbouring subproblems help each other in a collaborative manner. There are several approaches for constructing scalar subproblems in decomposition-based MOEAs. The most fundamental one for MOEA/Ds is the weighted sum (WS) approach. Given a MOP $\mathbf{f}(\mathbf{x})$ and the k-th weight vector λ^k, $k \in [0..N-1]$, the k-th scalar subproblem $g^{ws}(\mathbf{x} \mid \lambda^k)$ in the WS approach is in the following form:

$$\begin{aligned} &\text{minimize} \quad g^{ws}\left(\mathbf{x} \mid \lambda^k\right) = \sum_{i=1}^{m} \lambda_i^k f_i(\mathbf{x}), \\ &\textit{subject to} \quad \mathbf{x} \in \Omega. \end{aligned} \tag{3}$$

Afterwards, the cone decomposition (CD) approach was utilized in the CAEA for BOPs. It not only divides a BOP into N scalar subproblems, but also associates each scalar subproblem with an exclusive conical subregion and a specially designed conical area indicator. More specifically, given the k-th central observation vector λ^k, $k \in [0..N-1]$, the k-th scalar subproblem $g^{cd}(\mathbf{x} \mid \lambda^k, \mathbf{z}^{ide})$ in the cone decomposition approach is formulated as follows:

$$\begin{aligned} &\text{minimize} \quad g^{cd}\left(\mathbf{x} \mid \lambda^k, \mathbf{z}^{ide}\right) = S_k\left(\mathbf{f}(\mathbf{x}) - \mathbf{z}^{ide}\right), \\ &\textit{subject to} \quad \mathbf{x} \in \Omega \wedge \mathbf{f}(\mathbf{x}) - \mathbf{z}^{ide} \in \mathbf{C}^k, \end{aligned} \tag{4}$$

where $\mathbf{C}^k$ indicates the conical subregion associated with λ^k, $S_k(\cdot)$ denotes the conical area for the input vector in $\mathbf{C}^k$, and $\mathbf{z}^{ide}$ is the current ideal point.

2.2. Advantages of Using Decomposition-Based Multiobjective Techniques to Handle Constraints. In general, the most ordinary penalty function approach for constraint handling defines the constraint violation degree on the i-th constraint of a solution $\mathbf{x}$ as

$$G_i(\mathbf{x}) = \begin{cases} \max\{0, g_i(\mathbf{x})\}, & 1 \le i \le q \\ \max\{0, |h_i(\mathbf{x})| - \delta\}, & q+1 \le i \le l \end{cases} \tag{5}$$

in which δ represents an extremely small positive tolerance value for the equality constraints. Thus, $G(\mathbf{x}) = \sum_{i=1}^{l} G_i(\mathbf{x})$ indicates the overall constraint violation degree of the solution $\mathbf{x}$. Thereafter, the penalty function can be expressed in the following fundamental way:

$$\text{minimize} \quad p(\mathbf{x}, R) = R \cdot G(\mathbf{x}) + f(\mathbf{x}) \tag{6}$$

in which R indicates a penalty factor. However, R is very sensitive for various COPs. Mezura [36] suggests that the parameter value R should be selected in a very proper way so that the penalty function approach could avoid both overpenalization and underpenalization.

Since the desired optimal feasible solution in fact possesses the lowest constraint violation as well as the best objective function value, dominance-based multiobjective approaches, such as CMODE, have been developed to handle various constraints. In most of the multiobjective approaches, the overall constraint violation degree is regarded as the first objective value while the original objective function value is considered as the second one. Consequently, a COP is converted into a BOP without constraints, which can be described as follows:

$$\begin{aligned} &\text{minimize} \quad \mathbf{f}(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x})\right) \\ &\textit{subject to} \quad \mathbf{x} \in \Omega \end{aligned} \tag{7}$$

where $f_1(\mathbf{x}) = G(\mathbf{x})$ and $f_2(\mathbf{x}) = f(\mathbf{x})$. Thereafter the goal of the BOP converted from the COP is to minimize both objective function values.

So far, all the existing multiobjective approaches to constraint handling are based on Pareto dominance. Nevertheless, the advantages of using decomposition-based multiobjective techniques for COPs are discovered in this paper. First of all, there exists the mathematical association between decomposition-based techniques for multiobjective optimization and penalty function methods for constrained optimization. For example, the most fundamental WS aggregation function of the k-th scalar subproblem through decomposition is in the following form:

$$\text{minimize} \quad g^{ws}\left(\mathbf{x} \mid \lambda^k\right) = \lambda_1^k f_1(\mathbf{x}) + \lambda_2^k f_2(\mathbf{x}) \tag{8}$$

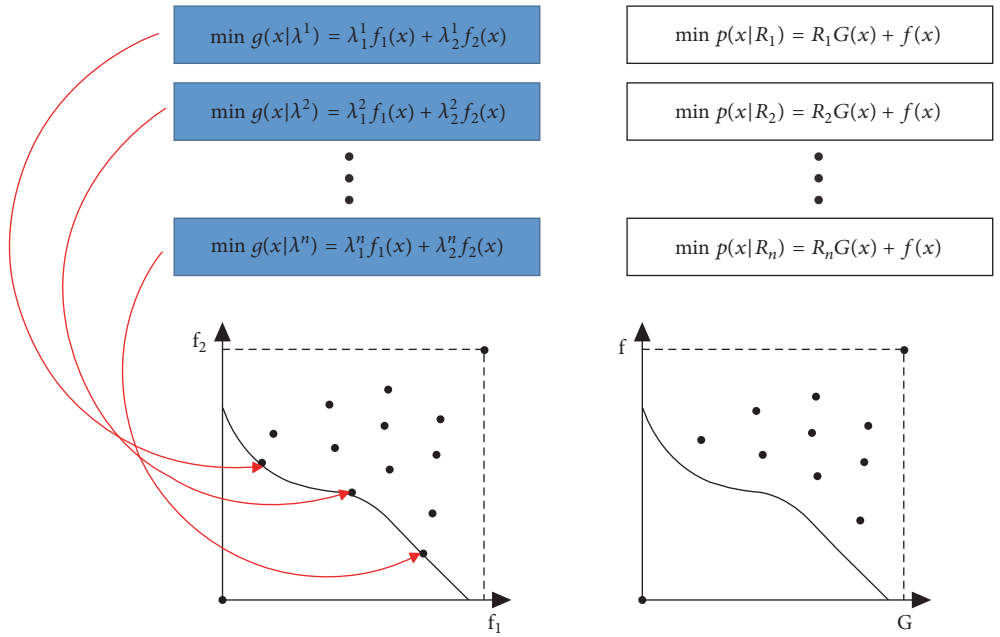

Figure 1: Mathematical association between decomposition approaches for multiobjective optimization and penalty function methods for constrained optimization.

where $\lambda_1^k \geq 0$, $\lambda_1^k \geq 0$, and $\lambda_1^k + \lambda_2^k = 1$. Comparing Eq. (6) with Eq. (8), it can be easily acquired that each different scalar subproblem $g^{ws}(\mathbf{x} \mid \lambda^k)$ is essentially equivalent to a penalty function with a different penalty value $R = \lambda_1^k/\lambda_2^k$. Furthermore, a series of scalar subproblems $g^{ws}(\mathbf{x} \mid \lambda^k)$ in a decomposition-based technique, $k = 1, 2, \ldots, N$, not only take full advantage of the mathematical properties of penalty functions, but also avoid their difficulties of determining a proper penalty value by trying a series of various penalty values from 0 to $+\infty$ in parallel, as shown in Figure 1. Afterwards, it has been proved that the optimal solution of a scalar subproblem in Eq. (8) is a nondominated one within the PF for any convex PF. It suggests that a series of various weight vectors should result in a nondominated solutions' set well distributed along the PF.

Thus decomposition-based techniques can exploit the diverse information about promising nondominated solutions along the PF to guide search from infeasible regions to feasible ones. In addition, due to the elimination of expensive nondominated checking, decomposition-based techniques usually have obviously higher running efficiencies than dominance-based multiobjective ones. In view of the above considerations, decomposition-based multiobjective techniques are very suitable for constraint handling in EAs.

3. Biased Decomposition and Dual Populations

To solve COPs using multiobjective technique, a COP is generally converted into a BOP in which minimizing the constraint violation degree $G(\mathbf{x})$ is the first objective and the primary objective function $f(\mathbf{x})$ is regarded as the second objective. Figure 2 illustrates how the BOP is converted

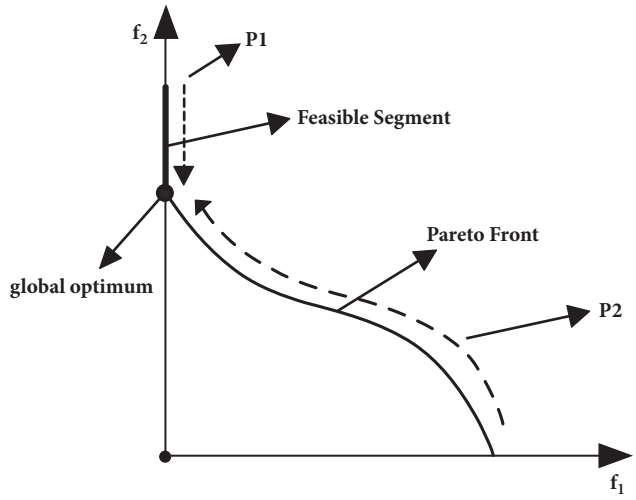

Figure 2: Dual-population scheme for converting a COP into a BOP.

from a COP. Here, all feasible individuals are on the feasible segment while the nondominated individuals lie on the PF. In particular, the intersection between the feasible segment and the PF is just the desired global feasible optimum.

The advantage of constraint-handling methods using multiobjective techniques, such as CMODE, is that they can employ nondominated individuals to guide the population towards the global optimal solution from infeasible regions to feasible regions. How to utilize the nondominated solutions has become the key to the search for the global optimum in multiobjective techniques. Using a multiobjective technique, CMODE is able to create competitive results for solving a COP, but it has to construct nondominated sets by the expensive process of nondominated sorting to update the

population. In order to avoid the high cost of nondominated sorting, a MOP is decomposed into N scalar subproblems using a biased cone decomposition. To discover a local nondominated individual in its associated decision subset, a conical area indicator is employed in the proposed CADE. However, in contrast to CAEA for multiobjective optimization, CADE employs a special dual-population scheme for constrained optimization. Figure 2 also illustrates the dual-population scheme in CADE. This scheme consists of two subpopulations: the feasible one and the conical one, denoted as P1 and P2, respectively, which are designed to search for the global feasible optimum, along the feasible segment and the PF. In particular, the nondominated solutions of P2 are utilized to explore the population on the PF to discover feasible solutions, while P1 exploits the population to discover the local optimal feasible solutions that are able to push the PF forward.

3.1. Conical Subpopulation and Biased Cone Decomposition. Let $A \in \Omega$ be the current set of all solutions searched so far by the CADE algorithm. For the purpose of the division of the objective space and the calculation of the conical area, the current ideal point $\mathbf{z}^{ide}$ and nadir point $\mathbf{z}^{nad}$ for A are first defined as $\mathbf{z}^{ide} = (z_1^{ide}, z_2^{ide})$ and $\mathbf{z}^{nad} = (z_1^{nad}, z_2^{nad})$, where $z_i^{ide} = \min_{\mathbf{x}\in A} f_i(\mathbf{x})$ and $z_i^{nad} = \max_{\mathbf{x}\in A} f_i(\mathbf{x})$, $i = 1, 2$. For the sake of clarity, any objective vector $\mathbf{y}$ in the original coordinate system is transformed through $\overline{\mathbf{y}} = \mathbf{y} - \mathbf{z}^{ide}$ so that the current ideal point becomes the origin $(0, 0)$ in the new coordinate system.

Definition 1 (observation vector). The observation vector for any converted point $\overline{\mathbf{y}} = (\overline{y}_1, \overline{y}_2)$ is $\mathbf{V}(\overline{\mathbf{y}}) = (v_1, v_2)$, where $v_i = \overline{y}_i/(\overline{y}_1 + \overline{y}_2)$, $i = 1, 2$.

It can be easily inferred that an observation vector has the following features: $v_i \geq 0$ and $v_1 + v_2 = 1$. A series of reference observation vectors $\mathbf{V}^k$ in geometric proportion, when a prescribed number of partitions N are provided, can be defined as follows:

$$\mathbf{V}^k = \left(\frac{s\left(1 - q^k\right)}{1 - q}, 1 - \frac{s\left(1 - q^k\right)}{1 - q} \right), \quad k = 0, 1, \ldots, N_2 - 1, \tag{9}$$

where $\mathbf{V}^0 = (0, 1)$ is the first reference observation vector in geometric proportion, where $q > 1$ denotes the proportion. All the reference observation vectors should be on the line $y = 1 - x$, and the last one $\mathbf{V}^{N_2-1} = (1, 0)$ should be $(s(1 - q^{N_2-1})/(1 - q), 1 - s(1 - q^{N_2-1})/(1 - q))$, through which it can be easily obtained that $s = (1 - q)/(1 - q^{N_2})$. Furthermore, the region $\mathbf{C} = \{\mathbf{y} = (y_1, y_2) \mid y_1 \geq 0 \wedge y_2 \geq 0\}$ can be decomposed into N conical subregions where the k-th subregion $\mathbf{C}^k = \{\mathbf{y} \in \mathbf{C} \mid t_{a_k} \leq v_1(\mathbf{y}) < t_{b_k}\}$, $k = 0, 1, \ldots, N_2 - 1$, $t_{a_k} = s(1-q^{k-1}+1-q^k)/2(1-q)$, and $t_{b_k} = s(1-q^k+1-q^{k+1})/2(1-q)$. It can be inferred that the observation vectors of the points in region $\mathbf{C}^k$ are closer to the reference observation vector $\mathbf{V}^k$ than to the other reference observation vectors. It is evident from Figure 2 that the individuals closer to the global optimum on the PF can help the population generate an offspring that is more likely to be near the global optimum than the individuals that are farther away from the global optimum. Hence, CADE employs a biased cone decomposition with geometrical proportions, as shown in Figure 3, rather than the uniform cone decomposition used in CAEA. Similar to CAEA, the conical area is regarded as a significant indicator. Further, CADE compares the conical area of individuals in the same subregion and preserves the one with the smallest conical area. However, because of the biased cone decomposition in geometric proportion, the calculation of the conical area is different from that used in CAEA. The definition of a conical area is as follows.

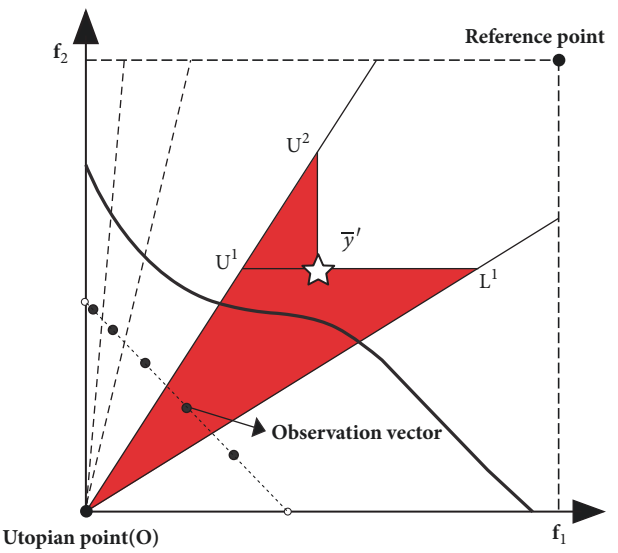

Figure 3: Biased cone decomposition in geometric proportion.

Definition 2 (conical area). Let $\overline{\mathbf{y}'} \in \mathbf{C}^k$, $0 \leq k \leq N - 1$, $\mathbf{y}^r$ represents the reference point, which is set as an approximate infinity point, and all the other individuals dominate this point. Then the conical area for $\overline{\mathbf{y}'}$, referred to as $S(\overline{\mathbf{y}'})$, is the area of the portion $\mathbf{C}(\overline{\mathbf{y}'}) = \{\mathbf{y} \in \mathbf{C}^k \mid \neg(\mathbf{y} \prec \overline{\mathbf{y}'}) \wedge \mathbf{y} \prec \overline{\mathbf{y}^r}\}$.

In CAEA, the region is uniformly divided into N subregions and the m-th reference observation vector is located at the center of the m-th subregion. It is clear from Figure 3 that the section nondominated by $\overline{\mathbf{y}'}$ in conical subregion $\mathbf{C}^k$ $(1 \leq k \leq N - 2)$ with which $\overline{\mathbf{y}'}$ is associated denotes $S(\overline{\mathbf{y}})$. Moreover, the area $S(\overline{\mathbf{y}'})$ can be calculated as follows:

$$\begin{aligned} S\left(\overline{\mathbf{y}'}\right) &= \frac{1}{2} \cdot (b_k - a_k)\left(\overline{y'}_2\right)^2 + \frac{1}{2} \cdot \frac{1}{a_k}\left(\overline{y'}_1 - a\overline{y'}_2\right)^2 \\ &= \frac{1}{2} \cdot \frac{1}{a_k}\overline{y'}_1^{\,2} + \frac{1}{2} \cdot b_k\overline{y'}_2^{\,2} - \overline{y'}_1\overline{y'}_2 \end{aligned} \tag{10}$$

where $1/a_k$ is the slope of the upper boundary, $1/b_k$ denotes the slope of the bottom boundary, and $\overline{\mathbf{y}'} = (\overline{y'}_1, \overline{y'}_2) \in \mathbf{C^k}$. The portion $\mathbf{C^k}$ is concave quadrilateral $OL^1\overline{\mathbf{y}'}U^2$, where $O = (0,0)$, $L^1 = (b_k\overline{y'}_2, \overline{y'}_2)$ is the node between the lower boundary of $\mathbf{C^k}$ and line $f_2 = \overline{y'}_2$, and the intersection point between $f_1 = \overline{y'}_1$ and the upper boundary of $\mathbf{C^k}$ is $U^2 = (\overline{y'}_1, (1/a_k)\overline{y'}_1)$. As a result, the conical area $S(\overline{\mathbf{y}'})$ can be calculated by adding the area of triangle $\triangle OL^1U^1$ to the area of $\triangle yU^2U^1$.

However, because CADE employs a biased cone decomposition strategy that is different from that used in CAEA, a_k and b_k should be calculated in a different way. In CADE, a_k and b_k are calculated, respectively, by $a_k = t_{a_k}/(1 - t_{a_k})$ and $b_k = t_{b_k}/(1 - t_{b_k})$. For the purpose of guiding the population towards discovery of the global optimal solution, if a feasible individual is searched, it is more important to consider the objective function value. Thus, a parameter $\alpha = 0.9$ is employed when calculating the conical area to control the population so that it searches in the direction of the objective function value if solution $\overline{y'} \in \mathbf{C^k}$, where $1 \le k \le N_2 - 2$ and the $\overline{y'}_1 = \alpha \times \overline{y'}_1$.

In addition, reference point $\mathbf{y^r}$ is required to calculate $S(\overline{\mathbf{y}'})$ if $\overline{\mathbf{y}'} \in \mathbf{C^0}$ or $\overline{\mathbf{y}'} \in \mathbf{C^{N_2-1}}$. The current nadir point and ideal point are used to calculate the reference point by $y_i^r = z_i^{nad} - z_i^{ide}$, $i = 1, 2$. If $\overline{\mathbf{y}'} \in \mathbf{C^0}$, the portion $\mathbf{C}(\overline{\mathbf{y}'})$ is a concave pentagon $OL^1\overline{\mathbf{y}'}T^1T^2$, where T^1 is the intersection between $f_1 = \overline{y'}_1$ and $y_2 = \overline{y^r}_2$ and T^2 is the intersection between $f_2 = \overline{y^r}_2$ and $y_1 = 0$. Therefore, $S(\overline{\mathbf{y}'})$ is the sum of the areas of a rectangle $\overline{\mathbf{y}'}T^1T^2U^1$ and a triangle $\triangle OL^1U^1$: $S(\overline{\mathbf{y}'}) = (1/2)\cdot b_0(\overline{y'}_2)^2 + (\overline{y^r}_2 - \overline{y'}_2)\overline{y'}_1$. Similarly, if $\overline{\mathbf{y}'} \in \mathbf{C^{N_2-1}}$, $S(\overline{\mathbf{y}'}) = (1/2)\cdot(1/a_{N_2-1})(\overline{y'}_1)^2 + (\overline{y^r}_2 - \overline{y'}_1)\overline{y'}_2$.

3.2. Feasible Subpopulation and Tolerance-Based Rule. In this paper, we define ω as a constraint tolerance value. Here, tolerance-based sorting is used to sort the feasible subpopulation P1 so that one individual having both a lower objective value and constraint violation degree in range ω precedes others having higher objective values or the violation degree out of the range ω. The tolerance-based dominance relationship, written as $\le_\omega$ between a pair of solutions $\mathbf{x}$ and $\mathbf{x}'$, is defined as follows:

$$\mathbf{x} \le_\omega \mathbf{x}' = \begin{cases} f_2(\mathbf{x}) \le f_2(\mathbf{x}'), & \text{if } f_1(\mathbf{x}) \le \omega \text{ and } f_1(\mathbf{x}') \le \omega \\ f_1(\mathbf{x}) \le \omega \text{ and } f_1(\mathbf{x}') \ge \omega & \\ f_1(\mathbf{x}) \le f_1(\mathbf{x}'), & \text{if } f_1(\mathbf{x}) > \omega \text{ and } f_1(\mathbf{x}') > \omega \end{cases} \tag{11}$$

After the individuals are sorted using the tolerance-based dominance relationship, the feasible subpopulation P1 in CADE is grouped into M levels in sequence. When one new child is used to update P1, it is easy to determine the level at which it lies. If the offspring's objective values are better than that of the last individual based on tolerance-based rule in the k-th level, then this offspring belongs to that level.

Specifically, the constraint tolerance value ω in tolerance-based sorting needs to be controlled over the function evaluations (FES) so that the algorithms can eventually obtain high quality solutions with lower constraint violations. In this paper, ω is managed as follows:

$$\omega(FES) = \frac{1}{2}\cdot\left(1 - \frac{FES}{\theta}\right)\cdot f_1\left(\mathbf{x}^{N_2-1}\right), \quad 0 \le FES \le \theta \tag{12}$$

where $\mathbf{x}^{N_2-1}$ represents the individual associated with the last conical subregion in the current conical subpopulation, $\theta = 0.2Max\_FES$, and Max_FES denotes the maximum number of FES.

4. Proposed Algorithm: CADE

4.1. Main Procedure. The framework of the proposed algorithm, CADE, is presented in Algorithm 1. First, N initial individuals are randomly generated from the decision space Ω. Afterwards, function *AssociateConicalSubpopulation*, described in Algorithm 2, is employed to form the initial conical subpopulation P2 by preserving only one suitable initial individual for every conical subregion since the objective space is divided to N_2 conical subregions according to the biased cone decomposition in CADE. Specifically, *AssociateConicalSubpopulation* calculates the index of the subregion where each individual lies. Among the individuals within the same subregion, only the one with the smallest conical area is preserved for it. Then, every subregion without any individual is associated with the individual closest to the central observation vector of this subregion. The rest of the individuals are used to form the initial feasible subpopulation P1. Notice that function $d(V', V'')$ in Algorithm 2 returns the Euclidean distance between vectors V' and V''.

Subsequently, in every generation, function *GenerateChild*, presented in Algorithm 3 and explained in the next subsection in detail, is called by CADE to generate the offspring by adaptive selection of DE operators. Thereafter, we calculate the index of the offspring and decide whether to update the ideal point $\mathbf{z}^{ide}$. Afterwards, the conical subpopulation P2 and the feasible subpopulation P1 are, respectively, updated by functions *ConeUpdatePopTwo* and *UpdatePopOne*, which are presented in Algorithms 4 and 5 and are explained in Section 4.3 in detail. When FES mod $N = 0$, the ω and *Ada_rate* are updated. Finally, the best optimal solution with lowest constraint violation is outputted.

Input: N_1: the size of feasible sub-population P1; N_2: the size of conical sub-population P2;
Max_FES: the maximum number of function evaluations;
Ada_rate: the adaptive parameter to choose the operation to generate a child;
c_{P2}: the parameter control the individuals of P2 to generate an offspring.
Output: $\mathbf{x}^*$:the best solution in the final population.

$\theta \longleftarrow 0.2Max\_FES$; $Ada\_rate \longleftarrow 0.5$;
$N = N_1 + N_2$; $FES \longleftarrow N$;
Create N initial solutions $P' \longleftarrow \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^{N-1}\}$ by uniformly randomly sampling from the decision space Ω;
$\mathbf{z}^{ide} \longleftarrow (z_1^{ide}, z_2^{ide})$ where $z_j^{ide} = \min_{\mathbf{y} \in P'} f_j(\mathbf{y})$, $j = 1, 2$;
P2 $\longleftarrow$ $AssociateConicalSubpopulation(P', N_2, \mathbf{z}^{ide})$;
Rank the rest individuals through the tolerance-based sorting to form P1;
Group P1 into M levels in sequence;
while $FES \leq Max\_FES$ **do**
    $\mathbf{y} \longleftarrow GenerateChild(\text{P1}, \text{P2}, Ada\_rate)$;
    Update $\mathbf{z}^{ide}$;
    if $\mathbf{z}^{ide}$ *is successfully updated and* $FES \leq \theta$ **then**
        Group the individuals in P1 through the tolerance-based sorting;
    end
    P2 $\longleftarrow$ $ConeUpdatePopTwo(\text{P2}, \mathbf{y}, s, q)$;
    P1 $\longleftarrow$ $UpdatePopOne(\text{P1}, \mathbf{y}, \theta)$;
    if $FES \bmod N = 0$ **then**
        Update Ada_rate and ω;
        if $FES \geq \theta$ **then**
            Update c_{P2};
        end
    end
end
return $\mathbf{x}^*$

Algorithm 1: The framework of CADE.

4.2. Selection and Reproduction. In CADE, the offspring is created using function *GenerateChild* of Algorithm 3. For stability, CADE employs two DE operators, DE/rand/exp and DE/current-to-rand/exp, to generate offspring. Both of them have demonstrated outstanding abilities to solve COPs. They are presented, respectively, as follows:

(1) DE/rand/exp

$$v_{i,t} = x_{p1,t} + F \cdot \left(x_{p2,t} - x_{p1,t}\right) \tag{13}$$

(2) DE/Current-to-rand/exp

$$v_{i,t} = x_{p1,t} + F \cdot \left(x_{p2,t} - x_{p1,t}\right) + F \cdot \left(x_{p3,t} - x_{p4,t}\right) \tag{14}$$

where $x_{pj,t}$ represents the t-th variable of solution $\mathbf{x}_{pj}$, $j = 1, 2, 3, 4$. Because of the bias cone decomposition technique utilized in P2, in different conical subregions, a solution closer to the Ω_f should have better conical area than the one further away from Ω_f. Moreover, the solution with the smallest conical area has local optimal property, compared with others in the same subregions. Therefore, a conical area-based tournament is employed to pick the first parent from the whole population. Here, the solution with the better conical area is chosen between two individuals selected randomly from the whole population. The first parent employed in both DE operators, referred to as $\mathbf{x}_{p1}$, is chosen with a probability of 0.75 according to the conical area-based tournament and is chosen randomly from the whole population with a probability of 0.25. In multiobjective optimization, the neighborhood of one solution is beneficial to its local search. In the proposed CADE, the individual in the first conical subregion of P2 is attached to the last individual of P1, which is the best one based on the ω comparison rule. Thus, P1 and P2 are united as one and every individual is indexed. The neighborhood of one solution consists of the first T solutions closest to it, where T is the neighbor size. The rest of the required parent individuals, referred to as either $\mathbf{x}_{p2}$ and $\mathbf{x}_{p3}$ or $\mathbf{x}_{p2}$, $\mathbf{x}_{p3}$, and $\mathbf{x}_{p4}$, are picked from the neighborhood of the first parent with a probability of 0.5 and are chosen randomly from the entire population with a probability of 0.5, each of which should be distinct from each other. Moreover, CADE uses an adaptive selection parameter Ada_rate to control the probability in which the first DE operator is selected. In every generation, Ada_rate is updated as follows:

$$Ada\_rate = 0.5 \times Ada\_rate + 0.5 \times \frac{m_1}{(m_1 + m_2)} \tag{15}$$

Input: P': the initial population; N_2:the size of P2; $\mathbf{z}^{ide}$: the ideal point.
Output: P2: the conical subpopulation associated with individuals.
Create a subpopulation P2 $\longleftarrow \{\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^{N_2-1}\}$ where $\mathbf{x}^0 = \mathbf{x}^1 = \cdots = \mathbf{x}^{N_2-1} = null$;
for $k = 0 \longrightarrow N_2 - 1$ **do**
  if $P' \cap \mathbf{C}^k \neq \emptyset$ **then**
    $\mathbf{x}^k \longleftarrow \text{argmin}_{\mathbf{y} \in P' \cap \mathbf{C}^k} S(\mathbf{f}(\mathbf{y}) - \mathbf{z}^{ide})$;
    $P' \longleftarrow P' \setminus \{\mathbf{x}^k\}$;
  end
end
for $k = 0 \longrightarrow N_2 - 1$ **do**
  if $\mathbf{x}^k = null$ **then**
    $\mathbf{x}^k \longleftarrow \text{argmin}_{\mathbf{y} \in P'} d(\mathbf{V}(\mathbf{f}(\mathbf{y}) - \mathbf{z}^{ide}), \mathbf{V}^k)$;
    $P' \longleftarrow P' \setminus \{\mathbf{x}^k\}$;
  end
end
return *P2*

Algorithm 2: *AssociateConicalSubpopulation*.

Input: P1: the current feasible subpopulation; P2: the current conical subpopulation;
Ada_rate: the adaptive rate to choose the operator to generate a child.
Output: **y**: an offspring generated by the adaptive DE operators
$\mathbf{x}_{p1} \longleftarrow$ pick a individual according to the conical area-based tournament in probability 0.75 and randomly in probability 0.25;
Generate a uniformly distributed random number *rand* between 0 and 1;
if *rand* $\leq$ *Ada_rate* **then**
  Randomly choose two different solutions, $\mathbf{x}_{p2}$ and $\mathbf{x}_{p3}$, from the neighborhood of $\mathbf{x}_{p1}$ in probability 0.5 and from the entire population in probability 0.5;
  Generate an offspring **y** by DE/rand/exp from $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ and $\mathbf{x}_{p3}$;
else
  Randomly select three different solutions, $\mathbf{x}_{p2}$, $\mathbf{x}_{p3}$, and $\mathbf{x}_{p4}$, from the neighborhood of $\mathbf{x}_{p1}$ in probability 0.5 and from the entire population in probability 0.5;
  Generate an offspring **y** by DE/current-to-rand/exp from $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$, $\mathbf{x}_{p3}$ and $\mathbf{x}_{p4}$;
end
return **y**

Algorithm 3: *GenerateChild*.

where m_1 denotes the count of using DE/rand/exp to update the individuals successfully while m_2 represents the count of using DE/current-to-rand/exp. To avoid the situation in which only one operator is used, *Ada_rate* is set to 0.95 when *Ada_rate* $>$ 0.95 and to 0.05 when *Ada_rate* $<$ 0.05.

In order to find the global feasible optimum finally, only the individuals near the feasible region in P2 are considered during the selection when FES $\geq \theta$. That means that CADE does not take the conical subregions far away from the feasible region into account. Specifically, when FES $\geq \theta$, only the individuals with indexes less than $c_{P2} \times N_2$ in P2 participate in the selection, where

$$c_{P2} = 1 - 3 \times \frac{FES}{Max\_FES}. \tag{16}$$

In addition, c_{P2} is set to 0.1 when $c_{P2} < 0.1$.

4.3. Update of the Subpopulations. When an offspring is generated, both P1 and P2 need to be updated. When updating P2, the subregion where the offspring lies is first located according to the biased cone decomposition. In addition, index k_1 of the corresponding conical subregion is calculated, as described in function *ConeUpdatePopTwo* of Algorithm 4. If $k_1 = 0$, the offspring **y** is used to update the solution in $\mathbf{C^0}$ according to the feasibility rule in order to make the solution in $\mathbf{C^0}$ satisfied with the constraints. If not, the index k_2 of the current solution $\mathbf{x}^{k_1}$ associated with this conical subregion in P2 is calculated. If $k_1 \neq k_2$, then the offspring is saved and associated with this conical subregion. Otherwise, if $k_1 = k_2$, the conical areas of the offspring **y** and the solution $\mathbf{x}^{k_1}$ are compared and the one with the smaller conical area is saved.

In contrast, the procedure for updating P1 is different. As shown in function *UpdatePopOne* of Algorithm 5, when FES $\leq \theta$, offspring **y** is used to update the first individual $\mathbf{y}'$ in the level where **y** lies according to the tolerance-based dominance

Input: **y**: an offspring for update; P2: the current conical subpopulation.
Output: P2: the updated conical subpopulation.

$$k_1 \longleftarrow \left\lfloor \log_q \left(\frac{2}{1+1/q} - \frac{(1-q)\left(f_1(\mathbf{y}) - z_1^{ide}\right)}{s(1+1/q)\left(f_1(\mathbf{y}) - z_1^{ide} + f_2(\mathbf{y}) - z_2^{ide}\right)} \right) \right\rfloor;$$

if $k_1 = 0$ **then**
  Use **y** to update the individual in $\mathbf{C}^0$ based on the feasibility rule;
else
  $$tmp \longleftarrow \frac{2}{1+1/q} - \frac{(1-q)\left(f_1\left(\mathbf{x}^{k_1}\right) - z_1^{ide}\right)}{s(1+1/q)\left(f_1\left(\mathbf{x}^{k_1}\right) - z_1^{ide} + f_2\left(\mathbf{x}^{k_1}\right) - z_2^{ide}\right)};$$
  $k_2 \longleftarrow \lfloor \log_q tmp \rfloor$;
  if $k_1 \neq k_2 \vee S(\mathbf{f}(\mathbf{y}) - \mathbf{z}^{ide}) < S(\mathbf{f}(\mathbf{x}^{k_1}) - \mathbf{z}^{ide})$ **then**
    $\mathbf{x}^{k_1} \longleftarrow \mathbf{y}$;
  end
end
return $P2$

Algorithm 4: *ConeUpdatePopTwo*.

Input: **y**: an offspring for update; P1: the current feasible subpopulation; ω: the constraint tolerance value.
Output: P1: the updated feasible subpopulation.

if $FES \leq \theta$ **then**
  Update the first individual $\mathbf{y}'$ worse than **y** in the level of P1 where **y** belongs to based on the ω-comparison rule.
else
  Randomly select one individual $\mathbf{y}'$ in P1, apply the feasibility-based rule to update $\mathbf{y}'$ with **y**.
end
return $P1$

Algorithm 5: *UpdatePopOne*.

rule. This is because it makes P1 maintain diversity in the constraint violation range ω. In the later stage, i.e., FES $> \theta$, CADE picks an individual randomly from P1 and uses the feasibility-based rule to update it. In such a situation, the feasibility-based rule helps P1 converge to the global optimum more quickly.

4.4. Computational Complexity. The main operation of major computational cost in CADE is updating the P1 and P2. In the early stage, CADE needs to find the proper individual to update $P1$ and it is easy to know that the average search depth is $N_1/2M$, while the costs to update P2 in the whole progress and P1 in the late stage are 1. Since the early stage is set to $0.2 \times Max\_FES$, the computational cost of CADE is $0.2 \times N_1/2M + 0.8 + 1 = N_1/10M + 1.8$. Thus, the computational time complexity of CADE could be regarded as $O(N_1/10M)$.

4.5. Differences between CMODE and CADE. Regarded as a successful constraint-handling method using multiobjective techniques, CMODE performs very well compared with other approaches. Both CADE and CMODE handle COPs using multiobjective techniques. However, there are still the following differences between CMODE and CADE.

(1) In CMODE, the nondominated individuals are preserved for evolution, and the PF is not constructed systematically. In contrast, CADE employs the biased cone decomposition to construct the PF.

(2) CMODE does not have a mechanism to maintain the diversity of nondominated solutions, which does not ensure a wide distribution diversity of nondominated individuals. Moreover, there is no neighborhood structure in CMODE. Because of the bias cone partition strategy, CADE guarantees a proportional sampling of PF, which provides it with a neighborhood structure to accelerate searching.

(3) In order to find nondominated individuals, CMODE has to perform nondominated sorting among the λ offspring and their parents with a complexity of $O(4\lambda)$ for each offspring. On the contrary, CADE utilizes the decomposition-based multiobjective technique to perfectly avoid the inefficient nondominated sorting and has a computational complexity of $O(1)$, which implies that the efficiency of CADE is obviously higher than that of CMODE.

5. Empirical Results and Discussion

In this section, the general performance of CADE is firstly validated on 24 widely used benchmark COPs containing different types reported in [37]. Then, the performance of

Table 1: 24 benchmark test instances.

Prob.	n	type of objective function	LI	NI	LE	NE	a	$f(\mathbf{x}^*)$
$g01$	13	Quadratic	9	0	0	0	6	-15.0000000000
$g02$	20	Nonlinear	0	2	0	0	1	-0.8036191041
$g03$	10	Polynomial	0	0	0	0	1	-1.0005001000
$g04$	5	Quadratic	0	6	0	0	2	-30665.5386717833
$g05$	4	Cubic	2	0	0	3	3	5126.4967140071
$g06$	2	Cubic	0	2	0	0	6	-6961.8138755802
$g07$	10	Quadratic	3	5	0	0	6	24.3062090682
$g08$	2	Nonlinear	0	2	0	0	0	-0.0958250414
$g09$	7	Polynomial	0	4	0	0	2	680.6300573744
$g10$	8	Linear	3	3	0	0	0	70492480205287
$g11$	2	Quadratic	0	0	0	1	1	0.7499000000
$g12$	3	Quadratic	0	1	0	0	0	–1.0000000000
$g13$	5	Nonlinear	0	0	0	3	3	0.0539415140
$g14$	10	Nonlinear	0	0	3	0	3	-47.7648884595
$g15$	3	Quadratic	0	0	1	1	2	961.7150222900
$g16$	5	Nonlinear	4	34	0	0	4	-1.9051552585
$g17$	6	Nonlinear	0	0	0	4	4	8853.5338748065
$g18$	9	Quadratic	0	13	0	0	0	-0.8660254038
$g19$	15	Nonlinear	0	5	0	0	0	32.6555929502
$g20$	24	Linear	0	6	2	12	16	0.2049794002
$g21$	7	Linear	0	1	0	5	6	193.7245100697
$g22$	22	Linear	0	1	8	11	19	236.4309755040
$g23$	9	Linear	0	2	3	1	6	-400.0551000000
$g24$	2	Linear	0	2	0	0	2	-5.5080132716

CADE on these test instances is compared against with those of four popular existing algorithms, namely, SaDE [22], MPDE [16], GDE [18], and CMODE [32]. Table 1 lists some features of 24 test instances, in which the number of decision variables is n, the number of constraints active at the global optimum is a, and the objective function value of the best known solution is $f(\mathbf{x}^*)$. Moreover, the four kinds of constraints, which are linear equality constraints, nonlinear equality constraints, linear inequality constraints, and nonlinear inequality constraints, are represented as LE, NE, LI, and NI, respectively. Specially, for the reason that no feasible solution for g20 has been found so far, the $f(\mathbf{x}^*)$ for g20 in Table 1 which comes from [32], the optimal solution is a little infeasible.

In our experiments, the sizes of feasible subpopulation P1 and conical subpopulation P2 are set to 120 and 60, respectively, giving a total population size of 180. In addition, the size of neighborhood T is set to 20. In CADE, the number M of levels for P1 and proportion q for P2 are, respectively, set to 2 and 1.1. In addition, scaling parameter F is randomly picked, respectively, from [0.5, 0.6] and crossover control factor C_r is from [0.8, 0.85] for DE/current-to-rand/exp while [0.9, 0.95] for DE/rand/exp in the adaptive hybrid DE operators. The termination criterion is satisfied when FES gets to 5×10^5 for every algorithm on every test instance. The other parameters for SaDE, MPDE, GDE, and CMODE used the corresponding recommended values provided, respectively, by [16, 18, 22, 32]. All the five algorithms are implemented in C++ and executed on an Intel Core i5-4278U 2.60 GHz PC with 8GB RAM. To evaluate the performances of these five methods, 25 statistically independent runs of five approaches are executed for each test case.

5.1. General Performance of CADE. As suggested by Liang *et al.* [37], if a feasible individual x has the objective function value $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0.0001$, where $\mathbf{x}^*$ represents the global optimal feasible solution, solution $\mathbf{x}$ can be considered as an individual that meets the requirement for success. Consequently, the difference between $f(\mathbf{x})$ and $f(\mathbf{x}^*)$ is referred to as the function error value for the individual $\mathbf{x}$ in our experiments. The experimental results of CADE are presented in the manner proposed by Liang *et al.* [37]. Tables 2–5 record the best, median, worst, mean, and standard deviation of $f(\mathbf{x}^\#) - f(\mathbf{x}^*)$ in which $\mathbf{x}^\#$ is the best-so-far individual when FES is, respectively, at 5×10^3, 5×10^4, and 5×10^5. Here, the mean value of the violations of the overall constraints at the median solution is expressed by v, and sequence of c represents the number of violations (including inequality and equality constraints) by $G(\mathbf{x}) > 1.0$, $0.01 < G(\mathbf{x}) \leq 1.0$, $0.0001 < G(\mathbf{x}) \leq 0.01$. Finally, the numbers in the parentheses after the error values (for the best, median, and worst solutions) indicate the number of unsatisfied constraints.

Tables 2–5 show that, in each run, at least one feasible solution could be discovered for 10 out of 24 test instances, i.e.,

Table 2: General performance in terms of function error values achieved by CADE for test functions g01-g06.

FES	Item	g01	g02	g03	g04	g05	g06
	Best	1.0019e+01(0)	4.1819e-01(0)	9.9606e-01(0)	1.3850e+02(0)	2.9616e+02(3)	8.1950e+01(0)
	Median	1.2516e+01(0)	4.6526e-01(0)	9.9606e-01(0)	1.8195e+02(0)	-2.0757e+01(3)	2.1331e+02(0)
	Worst	8.2039e+00(1)	5.1730e-01(0)	9.9064e-01(1)	2.4131e+02(0)	-4.2073e+01(3)	5.5703e+02(0)
5×10^3	c	0,0,0	0,0,0	0,0,0	0,0,0	3,0,0	0,0,0
	v	7.1886e-02	0.0000e+00	4.5259e-04	0.0000e+00	5.6316e+00	0.0000e+00
	Mean	1.0462e+01	4.7199e-01	9.7483e-01	1.7785e+02	8.8416e+01	2.2864e+02
	Std	2.2425e+00	3.4614e-02	3.3375e-02	3.7263e+01	1.1190e+02	1.6962e+02
	Best	3.4502e+00(0)	1.2228e-01(0)	1.2077e-01(0)	2.5507e+00(0)	1.8690e+00(3)	2.4484e-01(0)
	Median	4.5006e+00(0)	1.2228e-01(0)	4.8250e-01(0)	3.7399e+00(0)	1.7565e+01(3)	2.0633e+00(0)
	Worst	5.9370e+00(0)	2.2918e-01(0)	9.8971e-01(0)	6.8874e+00(0)	7.6184e+01(4)	2.5392e+01(0)
5×10^4	c	0,0,0	0,0,0	0,0,0	0,0,0	1,2,0	0,0,0
	v	0.0000e+00	0.0000e+00	1.5020e-04	0.0000e+00	2.8626e-01	0.0000e+00
	Mean	4.6377e+00	1.6855e-01	5.8456e-01	3.6371e+00	1.2538e+01	4.3982e+00
	Std	7.4639e-01	3.7580e-02	2.5971e-01	1.1370e+00	2.3045e+01	7.0974e+00
	Best	0.0000e+00(0)	8.2885e-12(0)	-1.0003e-11(0)	-2.9104e-11(0)	-1.8190e-12(0)	3.3651e-11(0)
	Median	0.0000e+00(0)	6.9388e-09(0)	-1.0002e-11(0)	-2.9104e-11(0)	-1.8190e-12(0)	3.3651e-11(0)
	Worst	0.0000e+00(0)	2.1762e-08(0)	-1.0002e-11(0)	-2.5466e-11(0)	-1.8190e-12(0)	3.3651e-11(0)
5×10^5	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
	Mean	0.0000e+00	8.4495e-09	-1.0002e-11	-2.7649e-11	-1.8190e-12	3.3651e-11
	Std	0.0000e+00	4.0419e-09	2.9606e-16	1.7822e-12	0.0000e+00	0.0000e+00

Table 3: General performance in terms of function error values achieved by CADE for test functions g07-g12.

FES	Item	g07	g08	g09	g10	g11	g12
	Best	3.8889e+01(0)	5.0449e-04(0)	3.7054e+01(0)	1.6100e+04(0)	2.4117e-04(0)	1.4196e-04(0)
	Median	4.9337e+01(0)	3.8840e-03(0)	5.7153e+01(0)	1.4906e+04(1)	7.6689e-03(0)	3.4115e-04(0)
	Worst	7.9337e+01(0)	8.3131e-03(0)	2.5886e+02(0)	1.6175e+03(2)	1.8138e-01(1)	7.9159e-03(0)
5×10^3	c	0,0,0	0,0,0	0,0,0	0,1,0	0,0,0	0,0,0
	v	0.0000e+00	0.0000e+00	0.0000e+00	1.1763e-01	3.0374e-05	0.0000e+00
	Mean	6.7318e+01	2.8911e-03	8.5154e+01	8.6601e+03	2.9141e-02	1.1298e-03
	Std	1.4768e+01	2.6133e-03	6.3287e+01	5.9987e+03	5.3485e-02	2.2659e-03
	Best	1.1906e+00(0)	6.6944e-06(0)	3.2957e+00(0)	7.2014e+02(0)	2.0799e-06(0)	4.5076e-08(0)
	Median	2.2255e+00(0)	1.7618e-05(0)	4.2117e+00(0)	1.0794e+02(1)	3.4848e-05(0)	4.5530e-07(0)
	Worst	2.4811e+00(0)	2.9758e-04(0)	1.4487e+01(0)	1.0794e+02(1)	1.2432e-04(0)	4.2346e-05(0)
5×10^4	c	0,0,0	0,0,0	0,0,0	0,1,0	0,0,0	0,0,0
	v	0.0000e+00	0.0000e+00	0.0000e+00	8.4681e-03	0.0000e+00	0.0000e+00
	Mean	1.7385e+00	8.1565e-05	7.6103e+00	1.5275e+03	4.1823e-05	6.7478e-06
	Std	3.9932e-01	9.3552e-05	3.3171e+00	2.0254e+03	3.4067e-05	1.2597e-05
	Best	-2.0229e-11(0)	1.8036e-11(0)	1.7053e-12(0)	-3.8199e-11(0)	-1.1102e-16(0)	0.0000e+00(0)
	Median	-2.0229e-11(0)	1.8036e-11(0)	1.7053e-12(0)	-3.7289e-11(0)	-1.1102e-16(0)	0.0000e+00(0)
	Worst	-2.0222e-11(0)	1.8036e-11(0)	1.8190e-12(0)	-3.7289e-11(0)	-1.1102e-16(0)	0.0000e+00(0)
5×10^5	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
	Mean	-2.0228e-11	1.8036e-11	1.7280e-12	-3.7471e-11	-1.1102e-16	0.0000e+00
	Std	2.8422e-15	0.0000e+00	4.5475e-14	3.6380e-13	0.0000e+00	0.0000e+00

g02, g04, g06, g07, g08, g09, g12, g16, g19, and g24, at 5×10^3 FES. For all test cases, the feasible solutions could be found at 5×10^5 FES apart from for g20 and g22. These feasible solutions satisfy the success condition, which reveals that feasible subpopulation cooperate with conical subpopulation in an effective way to search the global optimum along with feasible segment and front, respectively. Moreover, Tables 2–5 show that CADE has the ability to discover a successful

Table 4: General performance in terms of function error values achieved by CADE for test functions g13-g18.

FES	Item	g13	g14	g15	g16	g17	g18
	Best	5.2884e-01(3)	-1.9559e+01(3)	1.5862e-01(2)	9.5457e-02(0)	4.0663e+02(4)	-4.1993e-03(7)
	Median	5.4448e-01(3)	-5.1944e+01(3)	3.9348e+00(2)	1.2002e-01(0)	2.5818e+02(4)	1.0617e+00(8)
	Worst	1.5346e+02(3)	-1.0743e+02(3)	-8.7590e-01(2)	3.7129e-01(0)	-2.3880e+02(4)	1.2335e-01(9)
5×10^3	c	0,3,0	2,1,0	0,2,0	0,0,0	3,1,0	8,0,0
	v	4.8577e-01	2.0852e+00	3.1505e-01	0.0000e+00	6.0726e+00	1.8294e+00
	Mean	1.5894e+01	-6.7887e+01	8.1095e-01	2.2827e-01	4.6156e+01	5.3222e-01
	Std	4.5857e+01	2.3997e+01	1.3060e+00	9.2966e-02	3.4684e+02	1.4233e+00
	Best	2.5676e-04(3)	2.0077e+00(3)	9.0520e-02(2)	2.1206e-02(0)	4.2563e+01(4)	4.7040e-01(0)
	Median	2.5676e-04(3)	7.1756e+00(3)	9.0520e-02(2)	2.7016e-03(1)	8.3273e+01(4)	8.4874e-01(2)
	Worst	-6.5759e-04(3)	-1.2915e+00(3)	-5.8096e-04(2)	-1.1755e-03(1)	5.9447e+01(4)	7.1524e-01(3)
5×10^4	c	0,0,3	2,1,0	0,2,0	0,1,0	0,4,0	0,1,1
	v	6.2552e-03	3.9795e-02	2.2536e-02	4.8299e-05	1.7290e-01	2.2566e-03
	Mean	1.2573e-04	3.8283e+00	1.8120e-01	2.2574e-02	5.0181e+01	6.5964e-01
	Std	3.2821e-04	2.6577e+00	2.5239e-01	2.0683e-02	3.6762e+01	1.2267e-01
	Best	4.1897e-11(0)	8.5123e-12(0)	-3.9108e-11(0)	-3.4786e-11(0)	-1.8190e-11(0)	1.5561e-11(0)
	Median	4.1897e-11(0)	8.5123e-12(0)	-3.9108e-11(0)	-3.4786e-11(0)	-1.8190e-11(0)	1.5561e-11(0)
	Worst	4.1897e-11(0)	8.5123e-12(0)	-3.9108e-11(0)	-3.4786e-11(0)	-1.8190e-11(0)	1.5561e-11(0)
5×10^5	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
	Mean	4.1898e-11	8.5123e-12	-3.9108e-11	-3.4786e-11	-1.8190e-11	1.5561e-11
	Std	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00

Table 5: General performance in terms of function error values achieved by CADE for test functions g19-g24.

FES	Item	g19	g20	g21	g22	g23	g24
	Best	1.5969e+02(0)	5.3337e+00(20)	3.0006e+02(5)	5.0661e+03(19)	-7.7365e+02(4)	1.1476e-02(0)
	Median	2.1692e+02(0)	8.5284e+00(20)	4.0330e+02(5)	5.3464e+02(19)	-8.3952e+02(5)	4.1902e-02(0)
	Worst	3.8249e+02(0)	7.6293e+00(20)	-4.7567e+01(5)	3.3353e+03(19)	1.9358e+02(5)	7.5907e-02(0)
5×10^3	c	0,0,0	2,13,5	2,3,0	18,1,0	4,1,0	0,0,0
	v	0.0000e+00	3.6135e+00	1.3235e+00	5.7146e+06	1.1465e+00	0.0000e+00
	Mean	2.6193e+02	6.7814e+00	2.1449e+02	2.3106e+03	-6.5119e+02	4.1903e-02
	Std	7.6441e+01	1.7735e+00	2.0049e+02	2.1771e+03	4.7236e+02	1.9796e-02
	Best	1.9030e+00(0)	4.8246e-01(19)	3.7369e+00(4)	-2.2863e+02(19)	2.7238e+02(4)	1.1897e-04(0)
	Median	4.2276e+00(0)	8.0830e-01(20)	-1.8928e+02(5)	-2.2897e+02(19)	-9.5580e+02(5)	4.4583e-04(0)
	Worst	6.3510e+00(0)	1.9845e+00(20)	-1.8928e+02(5)	-1.1532e+01(19)	-1.6631e+03(6)	1.8623e-03(0)
5×10^4	c	0,0,0	2,11,7	0,5,0	17,2,0	1,4,0	0,0,0
	v	0.0000e+00	7.7876e-01	5.7351e-02	8.7773e+04	2.8224e-01	0.0000e+00
	Mean	4.0062e+00	1.1667e+00	7.7144e+00	-1.7509e+02	-5.6276e+02	6.6728e-04
	Std	1.3232e+00	5.6944e-01	1.6594e+02	8.5440e+01	6.4914e+02	5.1798e-04
	Best	4.6327e-11(0)	-1.1489e-01(1)	-7.7904e-11(0)	4.5220e+02(19)	-2.8422e-13(0)	4.6736e-12(0)
	Median	4.6327e-11(0)	-1.1489e-01(1)	-2.5807e-11(0)	7.5375e+02(18)	-2.8422e-13(0)	4.6736e-12(0)
	Worst	4.6327e-11(0)	-1.1489e-01(1)	1.3098e+02(0)	-2.3643e+02(19)	4.5844e-10(0)	4.6736e-12(0)
5×10^5	c	0,0,0	0,0,1	0,0,0	2,8,8	0,0,0	0,0,0
	v	0.0000e+00	2.0543e-04	0.0000e+00	4.3339e+01	0.0000e+00	0.0000e+00
	Mean	4.6327e-11	-1.1489e-01	1.0478e+01	5.1261e+00	4.5668e-11	4.6736e-12
	Std	0.0000e+00	0.0000e+00	2.5666e+01	3.7603e+02	1.3759e-10	0.0000e+00

solution for g08, g11, and g12 when the FES is 5×10^4. In addition, the global optimum for each of g01, g03, g04, g05, g07, g10, g11, g12, g15, g16, g17, and g18 can be discovered by CADE at 5×10^5 FES in every run. Particularly, for g20, the function objective values of the optimal solutions found by CADE in each run are better than 0.2049794002, and only one constraint, which has a value between 0.0001 and 0.001, is not satisfied. Moreover, both function objective values and constraint violation values are better than those reported in [32]. Thus, in this paper, the success condition remains

Table 6: Comparison of CADE with SADE, MPDE, GDE, and CMODE in terms of feasible rate.

Prob.	SaDE	MPDE	GDE	CMODE	CADE
g02	100%	100%	100%	100%	100%
g03	100%	100%	96%	100%	100%
g05	100%	100%	96%	100%	100%
g11	100%	100%	100%	100%	100%
g13	100%	88%	88%	100%	100%
g14	100%	100%	100%	100%	100%
g15	100%	100%	100%	100%	100%
g17	100%	96%	76%	100%	100%
g18	100%	100%	84%	100%	100%
g19	100%	100%	100%	100%	100%
g20	0%	0%	0%	0%	0%
g21	100%	100%	88%	100%	100%
g22	100%	0%	0%	0%	0%
g23	100%	100%	88%	100%	100%
mean	**95.83**%	91%	88.17%	91.67%	91.67%

Table 7: Comparison of CADE with SADE, MPDE, GDE, and CMODE in terms of success rate.

Prob.	SaDE	MPDE	GDE	CMODE	CADE
g02	84%	92%	72%	100%	100%
g03	96%	84%	4%	100%	100%
g05	100%	100%	92%	100%	100%
g11	100%	96%	100%	100%	100%
g13	100%	48%	40%	100%	100%
g14	80%	100%	96%	100%	100%
g15	100%	100%	96%	100%	100%
g17	4%	28%	16%	100%	100%
g18	92%	100%	76%	100%	100%
g19	100%	100%	88%	100%	100%
g20	0%	0%	0%	0%	0%
g21	60%	68%	60%	80%	92%
g22	0%	0%	0%	0%	0%
g23	88%	100%	40%	100%	100%
mean	83.5%	84%	74.17%	90.83%	**91.33**%

$f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0.0001$, where $\mathbf{x}$ is feasible for g20. What is more, the best solution found by CADE for g22 is still quite far from the feasible region, which indicates that g22 is very challenging for CADE to resolve.

The convergence graphs over the FES of the function error values (in log scale) obtained by CADE in the median runs for the 24 benchmark test instances except g20 and g22 are displayed in Figure 4. Note that the points whose function error values are less than or equal to 0 are not plotted. Further, because CADE is not able to resolve g20 and g22, Figure 4 does not plot the convergence graphs obtained by CADE for these two test instances. Figure 4 shows that CADE is able to find the global optimal solution for g04, g06, g08, g11, g12, g16, and g24 by only using about 1.5×10^5 FES. In addition, Figure 4 also clearly indicates that CADE requires about 3×10^5 FES for g01, g05, g09, g13, g14, g15, g17, and g21. Furthermore, it can be inferred from Figure 4 that, for 13 out of 22 test instances, CADE can eventually find the best known optimal solutions at 5×10^5 FES, from which it can be inferred that the dual-population scheme is able to make sure the population search the global optimal solution in an outstanding manner. Moreover, CADE obtains the solutions with the function error values of at most 10^{-10} in the median runs for all the 22 test functions except g02, which demonstrates that the feasibility-based rule of updating P1 in the later stage enables CADE to search a global optimum quickly.

5.2. Comparison with Popular DE-Based Approaches. In order to clarify the advantages of CADE for constrained optimization, CADE is further compared with four popular DE-based approaches, SaDE, MPDE, GDE, and CMODE. As suggested in [37], three performance metrics, feasible rate, success rate, and function error value, are used to assess performance of these algorithms. Tables 6 and 7, respectively, record the mean feasible rates and success rates obtained by these five DE-based algorithms for these test instances. It is probably worth pointing out that Tables 6 and 7 do not

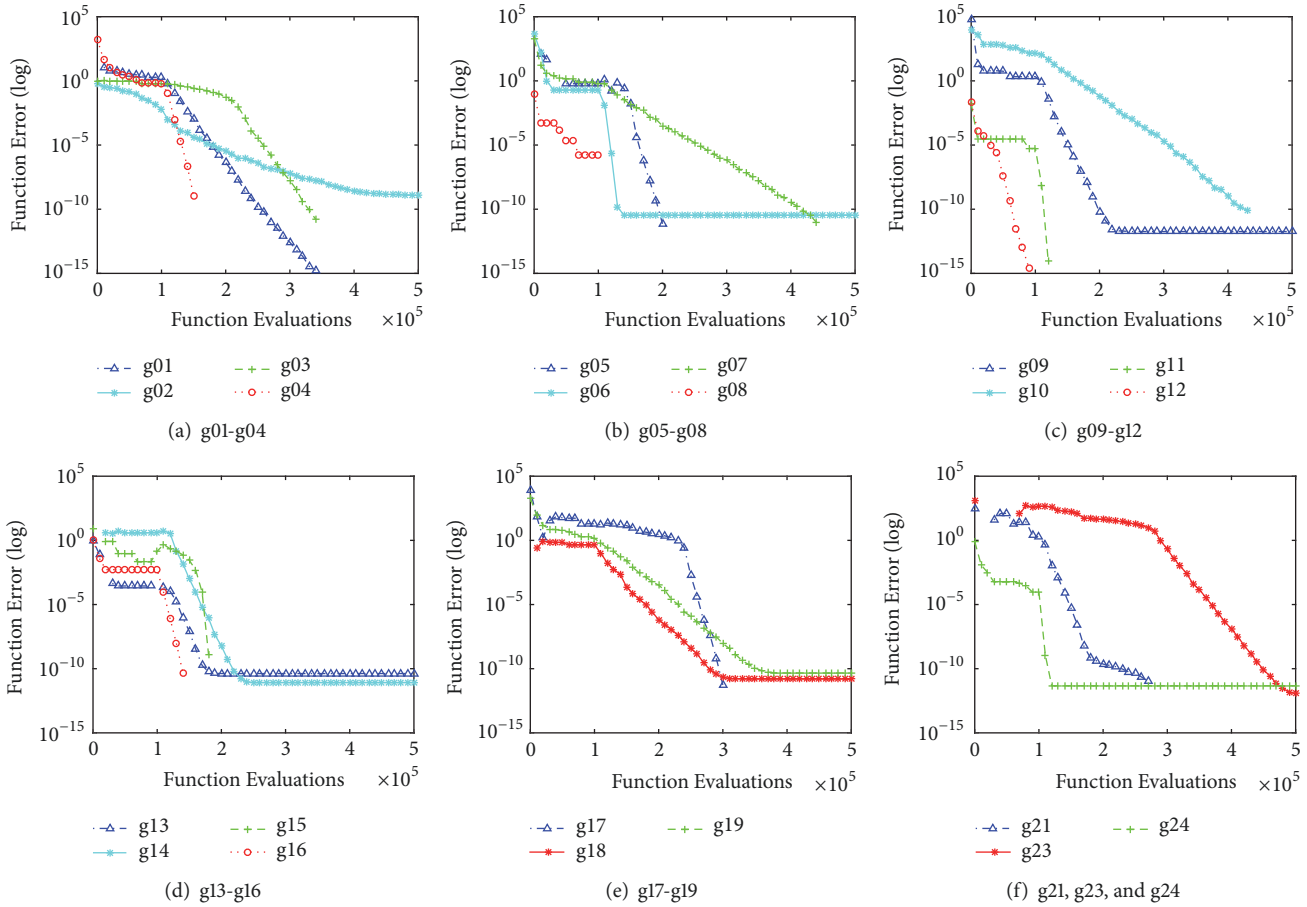

FIGURE 4: Convergence curves for general performance in terms of function error values obtained by CADE for the benchmark test instances except g20 and g22.

explicitly list the results of these algorithms for g01, g04, g06, g07, g08, g09, g10, g12, g16, and g24 for the reason that both their mean feasible rates and success rates are 100% for these ten test cases. With respect to the mean feasible rate, the performance of SaDE is better than that of the others, especially for g22. Meanwhile, the feasible rate acquired by CADE is equal to that by CMODE for each test instance. However, although SaDE is able to obtain the highest feasible rate for g22, it needs extra gradient information to find a feasible optimal solution. Table 7 indicates that CADE exhibits the substantially better success rates. Once a feasible solution had been found, it is more important for algorithms to optimize the objective value. Therefore, despite the fact that SaDE obtains the better performance in terms of mean feasible rate, the overall performances of CADE and CMODE are much better than those of SaDE.

Specifically, CADE obtains the best success rate, followed by CMODE and MPDE, while SaDE and GDE achieve the worst rates. In particular, CADE is capable of acquiring the success rate of 100% on all 24 test functions except g20, g21, and g22 and 92% on g21 which is the best value of the five approaches. Tables 6 and 7 imply that CADE can not only find a feasible solution on most test cases, but also guide the population search toward a lower objective value by constructing the PF comprehensively using the bias cone decomposition strategy.

Table 8 further presents the best, worst, and mean of the function error values, respectively, returned by these five algorithms at 5×10^5 FES for 17 test instances. Note that if an infeasible solution is finally acquired in each of the 25 independent runs of one algorithm for one test case, the mean function error value for this algorithm is not listed because it has no comparative significance. In addition, the numbers of unsatisfied constraints for the best and worst solutions are displayed as numbers in the parentheses after the error values. Because these algorithms obtain the nearly same exact optimal solution for g01, g06, g11, g12, and g24, Table 8 does not show the results for these seven test instances. Moreover, g20 and g22 are difficult for these four approaches to resolve, and none of them can find at least one successful solution. Hence, the results of these two test functions are not reported. In Table 8, the best results of the five compared approaches for each test problem are specially highlighted in boldface. It is evident from Table 8 that CADE performs the best, followed by CMODE and MPDE, while SaDE behaves the worst. Specifically, for 14 out of the 17 test cases, CADE obtains obviously lower error values than do the other four algorithms. Moreover, for g21, CADE can obtain

Table 8: Comparison of function error values achieved by five DE-based approaches when FES = 5×10^5 for test instances g02-g05, g07-g10, g13-g19, g21, and g23.

Prob.	Item	SaDE	MPDE	GDE	CMODE	CADE
	Best	8.0719e-10(0)	2.3414e-06(0)	5.0812e-08(0)	4.1726e-09(0)	**8.2885e-12(0)**
*g*02	Worst	1.8353e-02(0)	1.1014e-02(0)	2.2776e-02(0)	1.1836e-07(0)	**2.1762e-08(0)**
	Mean	2.0560e-03	8.0224e-04	4.3170e-03	2.0387e-08	**4.8107e-09**
	Best	1.3749e-10(0)	**-1.0003e-11**(0)	-9.8170e-12(0)	2.3964e-10(0)	**-1.0003e-11(0)**
*g*03	Worst	1.3389e-04(0)	4.0082e-01(0)	9.9739e-01(1)	2.5794e-09(0)	**-1.0002e-11(0)**
	Mean	1.3532e-05	2.5237e-02	-	1.1665e-09	**-1.0002e-11**
	Best	2.1667e-07(0)	3.6380e-12(0)	8.0036e-11(0)	7.6398e-11(0)	**-2.9104e-11(0)**
*g*04	Worst	2.1667e-07(0)	3.6380e-12(0)	8.0036e-11(0)	7.6398e-11(0)	**-2.5466e-11(0)**
	Mean	2.1667e-07	3.6380e-12	8.0036e-11	7.6398e-11	**-2.7649e-11**
	Best	0.0000e+00(0)	0.0000e+00(0)	0.0000e+00(0)	**-1.8190e-12(0)**	**-1.8190e-12(0)**
*g*05	Worst	0.0000e+00(0)	1.8190e-12(0)	8.2996e+02(2)	**-1.8190e-12(0)**	**-1.8190e-12(0)**
	Mean	0.0000e+00	7.2760e-14	-	**-1.8190e-12**	**-1.8190e-12**
	Best	6.8180e-08(0)	-2.0190e-11(0)	7.9822e-11(0)	7.9783e-11(0)	**-2.0229e-11(0)**
*g*07	Worst	6.3431e-05(0)	-2.0179e-11(0)	1.6532e-06(0)	7.9783e-11(0)	**-2.0229e-11(0)**
	Mean	4.7432e-06	-2.0186e-11	1.2966e-07	7.9783e-11	**-2.0229e-11**
	Best	8.1964e-11(0)	4.1633e-17(0)	8.1964e-11(0)	8.1964e-11(0)	**-1.8036e-11(0)**
*g*08	Worst	8.1964e-11(0)	4.1633e-17(0)	8.1964e-11(0)	8.1964e-11(0)	**-1.8036e-11(0)**
	Mean	8.1964e-11	4.1633e-17	8.1964e-11	8.1964e-11	**-1.8036e-11**
	Best	3.7440e-07(0)	1.1369e-13(0)	-9.7884e-11(0)	**-9.8225e-11(0)**	1.7053e-12(0)
*g*09	Worst	3.7440e-07(0)	1.1369e-13(0)	-9.7771e-11 (0)	**-9.8111e-11(0)**	1.7053e-12(0)
	Mean	3.7440e-07	1.1369e-13	-9.7884e-11	**-9.8198e-11**	1.7053e-12
	Best	6.6393e-11(0)	-1.8190e-12(0)	6.9122e-11(0)	6.2755e-11(0)	**-3.8199e-11(0)**
*g*10	Worst	7.8300e-06(0)	-9.0949e-13(0)	6.9122e-11(0)	6.3664e-11(0)	**-3.7289e-11(0)**
	Mean	1.6907e-06	-1.0186e-12	6.9122e-11	6.2827e-11	**-3.7471e-11**
	Best	4.1898e-11(0)	-9.7145e-17(0)	4.1898e-11(0)	**4.1897e-11(0)**	**4.1897e-11(0)**
*g*13	Worst	1.0696e-06(0)	9.2964e-01(0)	1.4403e+00(3)	**4.1897e-11(0)**	**4.1897e-11(0)**
	Mean	8.0263e-08	2.5364e-01	-	**4.1897e-11**	**4.1897e-11**
	Best	2.9050e-06(0)	**8.5123e-12**(0)	9.1518e-12(0)	**8.5123e-12**(0)	**8.5123e-12(0)**
*g*14	Worst	2.5233e-04(0)	**8.5123e-12**(0)	1.9281e-02(0)	8.5194e-12(0)	**8.5123e-12(0)**
	Mean	4.6979e-05	**8.5123e-12**	7.7373e-04	8.5194e-12	**8.5123e-12**
	Best	6.0822e-11(0)	0.0000e+00(0)	6.0936e-11(0)	6.0822e-11(0)	**-3.9108e-11(0)**
*g*15	Worst	6.0822e-11(0)	2.1514e-05(0)	4.8047e+00(0)	6.0822e-11(0)	**-3.9108e-11(0)**
	Mean	6.0822e-11	8.6055e-07	1.9219e-01	6.0822e-11	**-3.9108e-11**
	Best	6.5214e-11(0)	5.3291e-15(0)	6.5216e-11(0)	6.5213e-11(0)	**-3.4786e-11(0)**
*g*16	Worst	6.5214e-11(0)	5.3291e-15(0)	6.5217e-11(0)	6.5213e-11(0)	**-3.4786e-11(0)**
	Mean	6.5214e-11	5.3291e-15	6.5216e-11	6.5213e-11	**-3.4786e-11**
	Best	8.1858e-11(0)	3.6380e-12(0)	1.8189e-12(0)	1.8189e-12(0)	**-1.8190e-11(0)**
*g*17	Worst	7.4058e+01(0)	7.4058e+01(0)	2.8637e+02(4)	1.8189e-12(0)	**-1.8190e-11(0)**
	Mean	6.9670e+01	5.1353e+01	-	1.8189e-12	**-1.8190e-11**
	Best	2.7581e-011(0)	3.3307e-16(0)	1.5561e-11(0)	**1.5561e-11(0)**	**1.5561e-11(0)**
*g*18	Worst	1.9163e-001(0)	4.4409e-16(0)	-3.1844e+00(6)	**1.5561e-11(0)**	**1.5561e-11(0)**
	Mean	1.5312e-002	4.2633e-16	-	**1.5561e-11**	**1.5561e-11**
	Best	5.4456e-11(0)	4.6327e-11(0)	4.6448e-11(0)	1.1027e-10(0)	**4.6327e-11(0)**
*g*19	Worst	3.1141e-09(0)	4.6526e-11(0)	1.8287e-04(0)	5.4446e-10(0)	**4.6327e-11(0)**
	Mean	4.2052e-10	4.6348e-11	1.7894e-05	2.4644e-10	**4.6327e-11**
	Best	0.0000e+00(0)	3.3498e-11(0)	3.4987e-11(0)	**-3.1237e-10(0)**	-7.7904e-11(0)
*g*21	Worst	6.0712e-03(0)	1.3098e+02(0)	2.3612e+02(5)	**1.3097e+02(0)**	1.3098e+02(0)
	Mean	7.6753e-04	4.1913e+01	-	2.6195e+01	**1.0478e+01**
	Best	3.9790e-13(0)	3.9790e-13(0)	1.3112e-08(0)	1.8758e-12(0)	**-2.8422e-13(0)**
*g*23	Worst	1.3788e-03(0)	**6.2527e-13(0)**	6.4487e+02(4)	2.8063e-10(0)	4.5844e-10(0)
	Mean	1.2061e-04	**3.9790e-13**	-	4.4772E-11	4.5668e-11

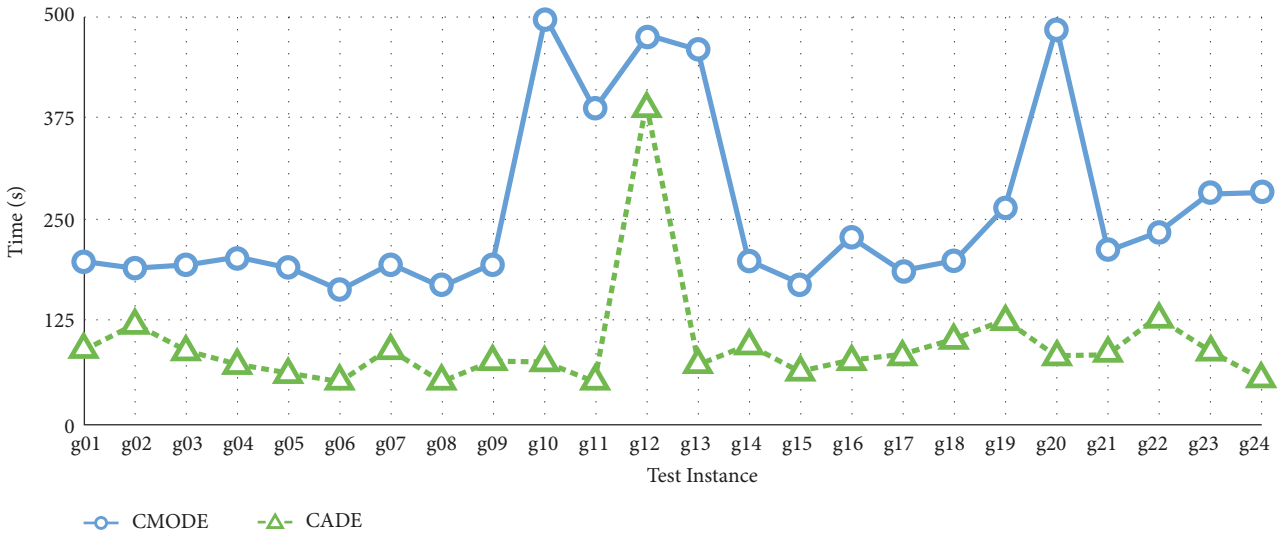

Figure 5: Comparison of CPU time spent by CMODE and CADE.

a more stable performance than the other four algorithms because, although each of them can find a successful solution, CADE gains a better mean value than the other methods. In particular, for g02 and g03, only CADE and CMODE can discover solutions satisfying the success condition in each run, and CADE performs better than CMODE.

In addition, the line charts of the average CPU time (in seconds) over 25 runs spent by two algorithms using multiobjective constraint-handling techniques, CMODE and CADE, for g01-g24 are plotted in Figure 5, which clearly indicates that, compared with CMODE, CADE spends much less CPU time on constrained optimization of all 24 test problems. It can be easily inferred that, on average, CADE only needs about 50% of the time CMODE requires. In conclusion, CADE not only produces competitive results due to the dual-population scheme, but also offers an obviously higher efficiency than CMODE does as a result of the biased cone decomposition for the conical subpopulation.

6. Conclusion

In this paper, a constraint-handling method using decomposition-based multiobjective techniques, CADE, is proposed to solve COPs. CADE employs a dual population scheme so that the information in the PF can be utilized to find the global optimum in a more efficient way. Specifically, more promising individuals can be preserved in the conical subregion closer to the global optimum by the biased cone decomposition. Moreover, the conical area indicator is used to help the conical subpopulation approximate the PF. In addition, the tolerance-based sorting for the feasible subpopulation in the early stage keeps a good diversity of population. The experimental results demonstrate that CADE, by employing the dual population scheme and biased cone decomposition, achieves competitive results for constrained optimization.

Our ongoing research focuses on applications of CADE for various engineering problems such as pressure vessel design problems. In the future, we also intend to study theoretically whether CADE possesses the global convergence property for constrained optimization. Specifically, we plan to model the evolution of CADE as a finite Markov chain, which is also widely utilized to prove the global convergence property of evolutionary algorithms for unconstrained optimization problems. Then, since the conical and feasible subpopulations of CADE are designed, respectively, for global and local searches, we expect to apply a finite Markov chain analysis to prove that the conical subpopulation of CADE, P2, finds and captures the global feasible optimum, $\mathbf{x}^*$, with probability 1 as the number of FES approaches infinity.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported partially by the Natural Science Foundation of Guangdong Province, China, under Grants 2015A030313204, 2017A030310013, and 2018A030313389, in part by the Pearl River S&T Nova Program of Guangzhou under Grant 2014J2200052, in part by the National Natural Science Foundation of China under Grants 61203310 and 61503087, in part by the Fundamental Research Funds for the Central Universities, SCUT, under Grant 2017MS043, in part by the Major Research and Development Program for Industrial Technology of Guangzhou City under Grant

201802010025, and in part by the Platform Development Program for Innovation and Entrepreneurship at Colleges in Guangzhou under Grant 2019PT103. The authors thank Kim Moravec, PhD, from Liwen Bianji, Edanz Editing China (www.liwenbianji.cn/ac), for editing the English text of a draft of this manuscript.

References

[1] C. Yu, K. L. Teo, L. Zhang, and B. Yanqin, "A new exact penalty function method for continuous inequality constrained optimization problems," *Journal of Industrial and Management Optimization*, vol. 6, no. 4, pp. 895–910, 2017.

[2] P. Zhang, J. Li, X. Hu, and J. Hu, "Crossover-based artificial bee colony algorithm for constrained optimization problems," *Neural Computing & Applications*, vol. 26, no. 7, pp. 1587–1601, 2017.

[3] S. Gao and W. Chen, "A partition-based random search for stochastic constrained optimization via simulation," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 62, no. 2, pp. 740–752, 2017.

[4] F. Prieto-Castrillo, A. S. Gazafroudi, J. Prieto, and J. M. Corchado, "An ising spin-based model to explore efficient flexibility in distributed power systems," *Complexity*, vol. 2018, Article ID 5905932, 16 pages, 2018.

[5] S. Chen, J. Yang, Y. Li, and J. Yang, "Multiconstrained network intensive vehicle routing adaptive ant colony algorithm in the context of neural network analysis," *Complexity*, vol. 2017, Article ID 8594792, 9 pages, 2017.

[6] R. A. Rahman, G. Kendall, R. Ramli, Z. Jamari, and K. R. Ku-Mahamud, "Shrimp feed formulation via evolutionary algorithm with power heuristics for handling constraints," *Complexity*, vol. 2017, Article ID 7053710, 12 pages, 2017.

[7] C. A. Coello 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.

[8] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

[9] P. Caamaño, G. Varela, and R. J. Duro, "An evolutionary approach to constrained sampling optimization problems," *Applied Soft Computing*, vol. 51, pp. 266–279, 2017.

[10] S. B. Hamida, *Extension of Evolutionary Algorithms to Constrained Optimization*, Springer International Publishing, Cham, Switzerland, 2016.

[11] F. Snyman and M. Helbig, "Solving Constrained Multi-objective Optimization Problems with Evolutionary Algorithms," in *Advances in Swarm Intelligence*, pp. 57–66, Springer International Publishing, Cham, Switzerland, 2017.

[12] L. Fan, Y. Liang, X. Liu, and X. Gu, "A hybrid evolutionary algorithm based on dominated clustering for constrained bi-objective optimization problems," in *Proceedings of the 12th International Conference on Computational Intelligence and Security, CIS '17*, pp. 543–546, 2017.

[13] J.-P. Chiou and F.-S. Wang, "A hybrid method of differential evolution with application to optimal control problems of a bioprocess system," in *Proceedings of the IEEE International Conference on Evolutionary Computation, ICEC '98*, pp. 627–632, 1998.

[14] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation ACM '05*, pp. 225–232, New York, NY, USA, 2005.

[15] R. L. Becerra and C. A. C. Coello, "Culturizing differential evolution for constrained optimization," in *Proceedings of the Mexican International Conference in Computer Science*, pp. 304–311, 2004.

[16] M. F. Tasgetiren and P. N. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 33–40, 2006.

[17] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," in *Proceedings of the Congress on Evolutionary Computation, CEC '02*, pp. 1468–1473, 2002.

[18] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC '06*, pp. 207–214, 2006.

[19] J. Brest, B. Bošković, and V. Žumer, "An improved self-adaptive differential evolution algorithm in single objective constrained real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[20] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "Modified Differential Evolution for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC '06*, pp. 25–32, 2006.

[21] K. Zielinski and R. Laur, "Constrained single-objective optimization using differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 223–230, 2006.

[22] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 17–24, 2006.

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

[24] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2002.

[25] J. Liu, Z. Fan, and E. Goodman, "SRDE: an improved differential evolution based on stochastic ranking," in *ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 345–352, ACM, New York, NY, USA, 2009.

[26] W.-Q. Ying, D.-X. Peng, Y.-H. Xie, and Y. Wu, "An annealing stochastic ranking mechanism for constrained evolutionary optimization," in *Proceedings of the International Conference on Information System and Artificial Intelligence, ISAI '17*, pp. 576–580, 2017.

[27] H. Wazir, M. A. Jan, W. K. Mashwani, and T. T. Shah, "A penalty function based differential evolution algorithm for constrained optimization," *Nucleus*, vol. 53, no. 1, pp. 155–166, 2016.

[28] C. Segura, C. A. C. Coello, E. Segredo, G. Miranda, and C. León, "Improving the diversity preservation of multi-objective approaches used for single-objective optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 3198–3205, 2013.

[29] H. A. Abbass and K. Deb, "Searching under multi-evolutionary pressures," in *Evolutionary Multi-Criterion Optimization*, vol.

2632 of *Lecture Notes in Computer Science*, pp. 391–404, Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2003.

[30] E. Mezura-Montes and C. A. C. Coello, *Constrained Optimization via Multiobjective Evolutionary Algorithms*, Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2008.

[31] K. Deb and R. Datta, "A bi-objective constrained optimization algorithm using a hybrid evolutionary and penalty function approach," *Engineering Optimization*, vol. 45, no. 5, pp. 503–527, 2013.

[32] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[33] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[34] W. Ying, X. Xu, Y. Feng, and Y. Wu, "An efficient conical area evolutionary algorithm for bi-objective optimization," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E95-A, no. 8, pp. 1420–1425, 2012.

[35] B. Wu, W. Ying, Y. Wu, Y. Xie, and Z. Wang, "A conical area differential evolution with dual populations for constrained optimization," in *Computational Intelligence and Intelligent Systems*, vol. 874 of *Communications in Computer and Information Science*, pp. 52–64, Springer Singapore, Singapore, Asia, 2018.

[36] 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.

[37] J. J. Liang, T. P. Runarsson, E. Mezura-Montes et al., *Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization*, vol. 41, Nangyang Technological University, Singapore, Asia, 2006.