Information Sciences 222 (2013) 302–322

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An improved $(\mu + \lambda)$-constrained differential evolution for constrained optimization

Guanbo Jia [a], Yong Wang [a,b,c,*], Zixing Cai [a,c], Yaochu Jin [d]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, PR China
[b] Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon 8525, Hong Kong, PR China
[c] Hunan Engineering Laboratory for Advanced Control and Intelligent Automation, Changsha 410083, PR China
[d] Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, UK

ARTICLE INFO

Article history:
Received 3 March 2011
Received in revised form 15 November 2011
Accepted 7 January 2012
Available online 15 January 2012

Keywords:
Constrained optimization
Differential evolution
Constraint-handling mechanism
Adaptive tradeoff model
Individual archiving technique

ABSTRACT

To overcome the main drawbacks of $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) [45], this paper proposes an improved version of $(\mu + \lambda)$-CDE, named ICDE, to solve constrained optimization problems (COPs). ICDE mainly consists of an improved $(\mu + \lambda)$-differential evolution (IDE) and a novel archiving-based adaptive tradeoff model (ArATM). Therein, IDE employs several mutation strategies and the binomial crossover of differential evolution (DE) to generate the offspring population. Moreover, a new mutation strategy named "current-to-rand/best/1" is proposed by making use of the current generation number in IDE. Since the population may undergo three situations during the evolution (i.e., the infeasible situation, the semi-feasible situation, and the feasible situation), like $(\mu + \lambda)$-CDE, ArATM designs one constraint-handling mechanism for each situation. However, unlike $(\mu + \lambda)$-CDE, in the constraint-handling mechanism of the infeasible situation, the hierarchical nondominated individual selection scheme is utilized, and an individual archiving technique is proposed to maintain the diversity of the population. Furthermore, in the constraint-handling mechanism of the semi-infeasible situation, the feasibility proportion of the combined population consisting of the parent population and the offspring population is used to convert the objective function of each individual. It is noteworthy that ICDE adopts a fixed tolerance value for the equality constraints. In addition, in this paper two criteria are used to compute the degree of constraint violation of each individual in the population, according to the difference among the violations of different constraints. By combining IDE with ArATM, ICDE has the capability to maintain a good balance between the diversity and the convergence of the population during the evolution. The performance of ICDE has been tested on 24 well-known benchmark test functions collected for the special session on constrained real-parameter optimization of the 2006 IEEE Congress on Evolutionary Computation (IEEE CEC2006). The experimental results demonstrate that ICDE not only overcomes the main drawbacks of $(\mu + \lambda)$-CDE but also obtains very competitive performance compared with other state-of-the-art methods for constrained optimization in the community of constrained evolutionary optimization.

1. Introduction

In the fields of science and engineering, there exist a large number of constrained optimization problems (COPs). In general, COPs with single-objective in minimization sense can be formulated as follows:

* Corresponding author at: School of Information Science and Engineering, Central South University, Changsha 410083, PR China.
E-mail address: ywang@csu.edu.cn (Y. Wang).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.01.017

$$\begin{aligned} &\text{minimize} \quad f(\vec{x}) \\ &\text{subject to} \quad g_j(\vec{x}) \leqslant 0,\ j = 1, \ldots, l \\ &\qquad\qquad h_j(\vec{x}) = 0,\ j = l+1, \ldots, p \end{aligned}$$

where $\vec{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subseteq S$ is an n-dimensional decision vector, $f(\vec{x})$ is the objective function, $g_j(\vec{x})$ is the jth inequality constraint, $h_j(\vec{x})$ is the jth equality constraint, Ω is the feasible region defined by the l inequality constraints and the $(p–l)$ equality constraints, and S is an n-dimensional rectangular search space in $\Re^n$ defined by the boundary constraints:

$$L_i \leqslant x_i \leqslant U_i, \quad 1 \leqslant i \leqslant n \tag{1}$$

where L_i and U_i are the lower and upper bounds of the ith decision variable x_i, respectively.

For COPs, if $\vec{x} \in \Omega$, $\vec{x}$ is called a feasible solution; otherwise, $\vec{x}$ is called an infeasible solution. An inequality constraint $g_j(\vec{x})$ $(j \in \{1, \ldots, l\})$ is considered *active* at a solution $\vec{x} \in \Omega$ if $g_j(\vec{x}) = 0$ at $\vec{x}$. All equality constraints $h_j(\vec{x})$ $(j = l+1, \ldots, p)$ are considered *active* at all solutions of Ω.

When solving COPs, the equality constraints are usually transformed into the following inequality constraints:

$$|h_j(\vec{x})| - \delta \leqslant 0 \tag{2}$$

where $j = l + 1, \ldots, p$ and δ is the tolerance value for the equality constraints.

In addition, the degree of constraint violation of a solution $\vec{x}$ on the jth constraint is calculated as follows:

$$G_j(\vec{x}) = \begin{cases} \max\{0, g_j(\vec{x})\}, & 1 \leqslant j \leqslant l \\ \max\{0, |h_j(\vec{x})| - \delta\}, & l+1 \leqslant j \leqslant p \end{cases} \tag{3}$$

Due to the presence of different kinds of constraints, multi-modal objective function, and concave feasible region, COPs are usually very difficult to be solved. Traditional optimization methods may either get stuck in a local optimum easily or need expensive computational cost when solving COPs. Evolutionary algorithms (EAs), inspired by nature, have turned out to be a very powerful optimization tool that requires little specific domain knowledge and is easy to implement. Because of the above advantages, EAs have gained much attention and have been broadly applied to solve COPs during the past decade. Meanwhile, various constraint-handling techniques have also been developed. By combining the constraint-handling techniques with EAs, a lot of constrained optimization evolutionary algorithms (COEAs) have been proposed.

As pointed out by Wang et al. [46], COEAs need to achieve two goals. The first goal is to approach or enter the feasible region promptly, and the second goal is to find the feasible optimal solution. Thus, in order to obtain competitive performance, a COEA should contain effective constraint-handling technique and powerful search algorithm. The existing constraint-handling techniques can be mainly classified into three groups: methods based on penalty functions [10,13,43], methods based on preference of feasible solutions over infeasible solutions [8,16,25,34], and methods based on multiobjective optimization techniques [1,5–7,44,46,47,51]. In addition, the current popular search algorithms involve evolution strategy (ES) [50], differential evolution (DE) [11,12,31,49], particle swarm optimization (PSO) [38,39,47,48], etc.

Recently, a $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) has been proposed by Wang and Cai [45] to solve COPs. $(\mu + \lambda)$-CDE mainly consists of a $(\mu + \lambda)$-differential evolution ($(\mu + \lambda)$-DE) and an improved adaptive tradeoff model (IATM). According to the experimental results on 24 well-known benchmark test functions collected for the special session on constrained real-parameter optimization of IEEE CEC2006 [22], $(\mu + \lambda)$-CDE is an effective method for solving COPs. However, the tolerance value δ for the equality constraints in $(\mu + \lambda)$-CDE has to be changed dynamically according to the following equation:

$$\delta_{t+1} = \begin{cases} \frac{\delta_t}{\delta'} & \text{if } \delta_t > 0.0001 \\ 0.0001 & \text{otherwise} \end{cases} \tag{4}$$

where t is the generation number, the initial value of δ (i.e., δ_0) is set to $n \log 10(\max_{i=1,\ldots,n}(U_i - L_i))$, and the change rate of δ (i.e., δ') is equal to 1.015.

Although Eq. (4) is adopted by Wang and Cai [45] to deal with different COPs, based on our experiments, both the initial value δ_0 and the change rate δ' of the tolerance value δ in Eq. (4) are problem-dependent. That is, when $(\mu + \lambda)$-CDE is used to solve a new COP rather than the 24 test functions in Liang et al. [22], Eq. (4) might not be an effective way for setting the tolerance value δ. For example, we revise the upper bound of the first variable (i.e., x_1) of test function g21 in Liang et al. [22], and obtain the following test function (called g25) to further investigate the effectiveness of Eq. (4). It is necessary to emphasize that other parts of test functions g21 are not changed and the global optimum of test function g25 is the same as that of test function g21.

$$\begin{aligned} &\text{minimize} \quad f(\vec{x}) \\ &\text{subject to} \quad g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leqslant 0 \end{aligned}$$

$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$
$$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$
$$h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$$
$$h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$$
$$h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$$

where $n = 7$, $0 \leqslant x_1 \leqslant 245$, $0 \leqslant x_2, x_3 \leqslant 40$, $100 \leqslant x_4 \leqslant 300$, $6.3 \leqslant x_5 \leqslant 6.7$, $5.9 \leqslant x_6 \leqslant 6.4$, and $4.5 \leqslant x_7 \leqslant 6.25$. The global optimum is at $\vec{x}^* = (193.724510070034967, 5.56944131553368433e\text{-}27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$ where $f(\vec{x}^*) = 193.724510070035$.

According to the experimental results over 25 independent runs, both the feasible rate and the success rate of $(\mu + \lambda)$-CDE are only 68% for test function g25, compared with 100% for test function g21.

In order to overcome the drawbacks of the dynamic setting for the tolerance value δ, a fixed value (e.g., 0.0001) is usually suggested for δ [22]. However, we find that if δ is fixed to 0.0001 in $(\mu + \lambda)$-CDE, both the feasible rate and the success rate will significantly decrease when solving the 24 benchmark test functions.

Motivated by the above considerations, this paper proposes an improved version of $(\mu + \lambda)$-CDE, termed ICDE. ICDE employs an improved $(\mu + \lambda)$-differential evolution (IDE) as the search engine and an archiving-based adaptive tradeoff model (ArATM) as the constraint-handling technique to solve COPs. Instead of dynamically changing the tolerance value δ, ICDE assigns a constant (i.e., 0.0001) suggested by Liang et al. [22] for the tolerance value δ. In IDE, several mutation strategies and the binomial crossover of DE are adopted to generate the offspring population. Moreover, a novel mutation strategy named "current-to-rand/best/1" is proposed in IDE. Subsequently, the parent population and the offspring population are combined to obtain a combined population, and ArATM is used to select some potential individuals from the combined population for the next generation. Since during the evolution a population may inevitably experience three situations (i.e., the infeasible situation, the semi-feasible situation, and the feasible situation), ArATM designs three different constraint-handling mechanisms for the three situations respectively. In the infeasible situation, ArATM adopts the hierarchical nondominated individual selection scheme developed in ATMES [47]. Furthermore, an individual archiving technique is proposed to maintain the diversity of the population. In the semi-feasible situation, ArATM employs the feasibility proportion of the combined population, rather than the feasibility proportion of the last population used by $(\mu + \lambda)$-CDE [45], to adjust the tradeoff between the objective function and the degree of constraint violations. By combining IDE with ArATM, ICDE shows strong search ability and can handle various kinds of constraints effectively. This paper utilizes the 24 well-known benchmark test functions collected for the special session on constrained real-parameter optimization of IEEE CEC2006 to measure the performance of ICDE. The experimental results show that ICDE overcomes the main drawbacks of $(\mu + \lambda)$-CDE. Furthermore, ICDE is very competitive with or superior to other state-of-the-art COEAs.

The remainder of this paper is organized as follows. Sections 2 and 3 introduce some basic ideas of multiobjective optimization and DE, respectively. In Section 4, the related work is briefly reviewed. Section 5 presents a detailed description of ICDE. In Section 6, ICDE is tested on the 24 well-known benchmark test functions, and the performance of ICDE is compared with that of several state-of-the-art COEAs. Section 7 discusses the effectiveness of some mechanisms and the rationality of the parameter settings in ICDE. Finally, Section 8 concludes this paper.

2. Basics of multiobjective optimization

Since the constraint-handling technique developed in this paper belongs to the methods based on multiobjective optimization techniques, firstly the multiobjective optimization problems (MOPs) are described in this section, and then some basic definitions about multiobjective optimization are introduced briefly.

A MOP can be formulated as follows:

$$\text{minimize } \vec{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_m(\vec{x}))$$

where m is the number of objective functions, $\vec{x} = (x_1, \ldots, x_n) \in S \subset \Re^n$ is the decision vector which contains n decision variables, and S is the search space.

Definition 1. (*Pareto dominance*): Consider two decision vectors $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_n)$, $\vec{a}$ is said to Pareto dominate $\vec{b}$ (denoted as $\vec{a} \prec \vec{b}$), if and only if

$$\forall i \in \{1, \ldots, m\}, \quad f_i(\vec{a}) \leqslant f_i(\vec{b}) \text{ and } \exists j \in \{1, \ldots, m\},\ f_j(\vec{a}) < f_j(\vec{b}) \tag{5}$$

In this case, $\vec{b}$ is said to be Pareto dominated by $\vec{a}$. If Pareto dominance does not hold between $\vec{a}$ and $\vec{b}$, they are considered *nondominated* with each other.

Definition 2. (*Pareto optimality*): $\vec{a} \in S$ is said to be Pareto optimal in S, if and only if $\neg\exists \vec{b} \in S$ satisfies $\vec{b} \prec \vec{a}$.

Definition 3. (*Pareto optimal set*): The Pareto optimal set (denoted as P^*) is the set of all the Pareto optimal solutions:

$$P^* = \{\vec{a} \in S | \neg \exists \vec{b} \in S, \vec{b} \prec \vec{a}\} \tag{6}$$

Definition 4. (*Pareto front*): The Pareto front (denoted as PF^*) is defined as:

$$PF^* = \{\vec{f}(\vec{a}) | \vec{a} \in P^*\} \tag{7}$$

3. Basics of differential evolution

Differential evolution (DE) is proposed by Storn and Price in 1995 [37,38]. DE is a simple yet efficient EA, which has been widely applied to solve continuous optimization problems [32].

DE starts the search with an initial population containing *NP* individuals, which are randomly sampled from the search space. Then, one individual called the target vector in the population is used to generate a mutant vector by the mutation operation. So far, more than six mutation strategies have been proposed [26,30], and four of them employed in ICDE are introduced as follows:

- "rand/1":

$$\vec{v}_i = \vec{x}_{r1} + F \times (\vec{x}_{r2} - \vec{x}_{r3}) \tag{8}$$

- "rand/2":

$$\vec{v}_i = \vec{x}_{r1} + F \times (\vec{x}_{r2} - \vec{x}_{r3}) + F \times (\vec{x}_{r4} - \vec{x}_{r5}) \tag{9}$$

- "current-to-rand/1":

$$\vec{v}_i = \vec{x}_i + F \times (\vec{x}_{r1} - \vec{x}_i) + F \times (\vec{x}_{r2} - \vec{x}_{r3}) \tag{10}$$

- "current-to-best/1":

$$\vec{v}_i = \vec{x}_i + F \times (\vec{x}_{best} - \vec{x}_i) + F \times (\vec{x}_{r1} - \vec{x}_{r2}) \tag{11}$$

where r_1, r_2, r_3, r_4, and r_5 are integers randomly selected from {1, 2, ..., *NP*} and satisfy $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$, the scaling factor F is usually a real number between 0 and 1, $\vec{x}_i$ is the current individual in the population, $\vec{x}_{best}$ is the best individual in the population, and $\vec{v}_i$ is the mutant vector.

After mutation, all the components of the mutant vector are checked whether they violate the boundary constraints. If the jth component $v_{i,j}$ of the mutant vector $\vec{v}_i$ violates the boundary constraint, $v_{i,j}$ is reflected back from the violated boundary constraint as follows [17]:

$$v_{i,j} = \begin{cases} 2L_j - v_{i,j} & \text{if } v_{i,j} < L_j \\ 2U_j - v_{i,j} & \text{if } v_{i,j} > U_j \\ v_{i,j} & \text{otherwise} \end{cases} \tag{12}$$

Subsequently, the crossover operation is implemented on the mutant vector and the target vector to generate a trial vector $\vec{u}_i$. Two commonly used crossover operations are the binomial crossover and the exponential crossover. In ICDE, the binomial crossover is employed and executed as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand \leqslant CR \quad \text{or} \quad j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases} \tag{13}$$

where $i \in \{1,2,\ldots,NP\}$, $j \in \{1,2,\ldots,n\}$, *rand* is a uniformly distributed random number between 0 and 1, j_{rand} is a randomly selected integer from 1 to n, *CR* is the crossover control parameter, and $u_{i,j}$ is the jth component of the trial vector $\vec{u}_i$.

Finally, the target vector $\vec{x}_i$ is compared with the trial vector $\vec{u}_i$ in terms of the objective function value and the better one survives into the next generation:

$$\vec{x}_i = \begin{cases} \vec{u}_i & \text{if } f(\vec{u}_i) \leqslant f(\vec{x}_i) \\ \vec{x}_i & \text{otherwise} \end{cases} \tag{14}$$

4. The related work

During the past decade, DE has been widely applied to solve COPs and a lot of DE-based methods have been proposed. Storn [36] proposed a DE-based constraint adaptation named CADE. This method first relaxes all original constraints to make

all individuals in the population feasible, and then reaches the original constraints by tightening the constraints step-by-step. Lampinen and Zelinka [19–21] combined DE with a penalty technique to solve mixed discrete-integer-continuous optimization problems. Lin et al. [23] developed a hybrid DE including a multiplier updating. Moreover, an adaptive scheme for penalty parameters is incorporated. Lampinen [18] presented an extended DE for handling nonlinear constraint functions, which uses a new replacement criterion to deal with the constraint functions. Based on ES and DE, Runarsson and Yao [34] proposed an improved stochastic ranking. The differential variation proposed in this method is a variant of the mutation operator of DE. Mezura-Montes et al. [28] proposed a DE-based method. In this method, a diversity mechanism is introduced which allows the infeasible individuals with promising objective function values to survive into the next population. In addition, each parent is used to generate more than one offspring. Qin and Suganthan [33] proposed a self-adaptive DE called SaDE, in which the learning strategies and the parameters settings are self-adapted during the evolution and a local search operator is added to speed up the convergence. For solving COPs, Huang et al. [14] generalized SaDE by incorporating the constraint-handling technique proposed by Deb [8]. Takahama and Sakai [40] applied the ε constrained method to a DE and presented εDE, in which gradient-based mutation and feasible elite preserving strategy are proposed. Tasgetiren and Suganthan [42] presented a multi-populated DE called MDE. In MDE, the population is divided into several subpopulations which conduct the search in parallel. Moreover, MDE introduces a regrouping schedule periodically for information exchange among the subpopulations. Kukkonen and Lampinen [17] introduced the generalized DE (GDE). In GDE, the trial vector replaces its target vector if it weakly dominates its target vector. Brest et al. [4] proposed jDE-2, in which three DE mutation strategies are used and two control parameters (i.e., F and CR) are self-adapted. Mezura-Montes et al. [29] allowed each individual to generate more than one offspring, and designed a new mutation strategy by combining information of both the best individual and the current parent individual to find new search directions. Becerra and Coello Coello [2] proposed a DE-based culture algorithm, in which different knowledge sources are used to influence the variation operator of DE. Huang et al. [15] presented a coevolution DE approach named CDE, which incorporates a coevolution model into DE and uses two kinds of populations to evolve solutions and penalty factors, respectively. Mezura-Montes [24] summarized the most recent advances in nature-inspired methods for solving COPs. Takahama and Sakai [41] proposed an improved εDE. To solve problems with many equality constraints and find feasible solutions stably, the improved εDE uses faster reduction of the relaxation of equality constraints and high gradient-based mutation rate. Brest [3] presented a self-adaptive DE named ε–j DE, in which the ε constrained method is used to handle constraints. Moreover, this method employs several mutation strategies and reduces the population size during the evolution. Additionally, two control parameters (i.e., F and CR) are self-adapted. Mezura-Montes and Palomeque-Ortiz [27] proposed an adaptive diversity DE named A-DDE, which self-adapts three parameters and controls the fourth one by a decreasing function.

5. Proposed algorithm

In this paper, an improved version of $(\mu + \lambda)$-CDE (referred as ICDE) is proposed to solve COPs. ICDE mainly consists of an improved $(\mu + \lambda)$-differential evolution (IDE) and an archiving-based adaptive tradeoff model (ArATM). Therein, IDE is used as the search engine and ArATM is employed to handle constraints. Next, ICDE is described in detail.

5.1. Algorithmic framework

ICDE works as follows:

Step (1) Set $t = 0$ and $A = \Phi$ where t denotes the generation number and A denotes the archive.

Step (2) Uniformly and randomly sample μ points from the search space S to form the initial population $P_0 = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_\mu\}$.

Step (3) Compute the constraint violations of all the individuals in P_0. According to the difference among the constraint violations, determine the criterion to compute the degree of constraint violation of each individual during the evolution (see Section 5.2 for details).

Step (4) Compute the objective function value $f(\vec{x}_i)$ and the degree of constraint violation $G(\vec{x}_i)$ of each individual in P_0.

Step (5) Generate λ offspring by executing IDE on all the individuals in P_t. These λ offspring form the offspring population Q_t (see Section 5.3 for details).

Step (6) Compute the objective function value and the degree of constraint violation of each individual in Q_t.

Step (7) Combine P_t with Q_t to obtain a combined population H_t (i.e., $H_t = P_t \cup Q_t$).

Step (8) Select μ potential individuals from H_t to constitute the next population P_{t+1} by ArATM (see Section 5.4 for details).

Step (9) Set $t = t + 1$.

Step (10) If the termination criterion is not satisfied, go to Step (5); otherwise, stop and output the best individual $\vec{x}_{best}$ in P_t.

In the following, the detailed features of ICDE are discussed.

5.2. Methods to compute the degree of constraint violation

Since the violations of different constraints may have different scales for COPs, ICDE uses two criteria to compute the degree of constraint violation of each individual in the population as $(\mu + \lambda)$-CDE, according to the difference among the violations of different constraints.

In the initial population P_0, the constraint violations of all the individuals are calculated at first. Then, the difference (denoted as *diff*) among the violations of all the constraints is measured by Eq. (15):

$$diff = \max_{j=1,\ldots,p}(\max_{i=1,\ldots,\mu} G_j(\vec{x}_i)) - \min_{j=1,\ldots,p}(\max_{i=1,\ldots,\mu} G_j(\vec{x}_i)) \tag{15}$$

If *diff* is less than a user-defined parameter η which means that the difference among the violations of all the constraints is not significant, the first criterion described by Eq. (16) is employed to compute the degree of constraint violation of each individual, with the aim of emphasizing the difference among the violations of different constraints:

$$G(\vec{x}_i) = \sum_{j=1}^{p} G_j(\vec{x}_i), \quad i = 1, \ldots, \mu \tag{16}$$

On the contrary, if *diff* is bigger than η which means that the difference among the violations of all the constraints is significant, the second criterion is used to allot equal importance to all constraints as follows. Firstly, the constraint violations of all the individuals are normalized by Eq. (17) for each constraint:

$$G'_j(\vec{x}_i) = \frac{G_j(\vec{x}_i)}{\max_{k=1,\ldots,\mu} G_j(\vec{x}_k)}, \quad i = 1, \ldots, \mu, \quad j = 1, \ldots, p \tag{17}$$

Thereafter, the mean of the normalized constraint violations of each individual is calculated for all constraints by Eq. (18) and considered as the degree of constraint violation of each individual:

$$G(\vec{x}_i) = \frac{\sum_{j=1}^{p} G'_j(\vec{x}_i)}{p}, \quad i = 1, \ldots, \mu \tag{18}$$

In the above process, *diff* is firstly computed using the individuals in the initial population P_0. Subsequently, the criterion, which is employed to compute the degree of constraint violation of each individual in the population, will be determined by comparing *diff* with η. If $diff < \eta$, the first criterion will be used; otherwise, the second criterion will be used. Note that once the criterion is determined, it will be fixed during the evolution.

5.3. Improved $(\mu + \lambda)$-differential evolution (IDE)

IDE serves as the search engine in ICDE, and is designed by improving $(\mu + \lambda)$-DE used in $(\mu + \lambda)$-CDE. Note that $(\mu + \lambda)$-DE is similar to $(\mu + \lambda)$-ES [35]. Compared with the classic DE, IDE also includes the mutation and crossover operations yet the selection operation is not applied. In IDE, the population P_t (i.e., the parent population) with μ individuals generates a new population Q_t (i.e., the offspring population) with λ individuals by performing the following procedure:

Step (1) Set $Q_t = \Phi$;
Step (2) **For** each individual $\vec{x}_i (i = 1, \ldots, \mu)$ in P_t;
Step (3) Generate the first offspring $\vec{y}_1$ by implementing the "rand/1" strategy and the binomial crossover of DE;
Step (4) Generate the second offspring $\vec{y}_2$ by implementing the "rand/2" strategy and the binomial crossover of DE;
Step (5) Generate the third offspring $\vec{y}_3$ by implementing a new mutation strategy (named "current-to-rand/best/1") proposed in this paper and the improved breeder genetic algorithm (BGA) mutation [46];
Step (6) $Q_t = Q_t \cup \vec{y}_1 \cup \vec{y}_2 \cup \vec{y}_3$;
Step (7) **End**

In Step (5), the "current-to-rand/best/1" strategy proposed in this paper is constructed by combining the "current-to-rand/1" strategy with the "current-to-best/1" strategy, according to the current generation number. The "current-to-rand/best/1" strategy is implemented as follows. At first, the current generation number (denoted as *current_gen*) is compared with a threshold generation number *threshold_gen*, which is equal to the total generation number *total_gen* multiplied by a factor k (i.e., $threshold\_gen = k \times total\_gen$). If $current\_gen > threshold\_gen$, the "current-to-best/1" strategy is used to generate the mutant vector $\vec{v}_i$ for $\vec{x}_i$. Afterward, the improved BGA mutation [46] described by Eq. (19) is applied to the mutant vector $\vec{v}_i$ with a probability p_m for producing the offspring $\vec{y}_3 = \{y_{3,1}, y_{3,2}, \ldots, y_{3,n}\}$, with the aim of increasing the diversity of the population:

$$y_{3,j} = \begin{cases} v_{i,j} \pm rang_i \times \sum_{s=0}^{15} \alpha_s 2^{-s} & rand < 1/n \\ v_{i,j} & \text{otherwise} \end{cases}, j = 1, \ldots, n \tag{19}$$

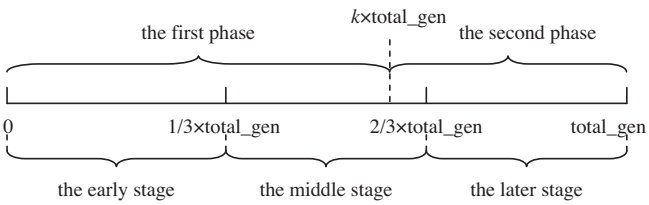

Fig. 1. The schematic diagram shows three stages and two phases of the whole evolutionary process during the implementation of the "current-to-rand/best/1" strategy.

where *rand* is a uniformly distributed random number between 0 and 1, $rang_i$ defines the mutation range and is set to $(U_i - L_i) \times (1 - current\_gen/total\_gen)^6$, the sign + or − is chosen with a probability of 0.5, and $\alpha_s \in \{0,1\}$ is randomly generated with the probability $p_r(\alpha_s = 1) = 1/16$. In addition, if $current\_gen \leqslant threshold\_gen$, the "current-to-rand/1" strategy is used to generate the mutant vector $\vec{v}_i$ and the improved BGA mutation is not implemented (i.e., in this case $\vec{y}_3 = \vec{v}_i$).

In order to further explain the implementation of the "current-to-rand/best/1" strategy, we draw a schematic diagram in Fig. 1. As shown in Fig. 1, the whole evolutionary process is divided into two phases. Generally speaking, if we briefly divide the evolutionary process into three stages (i.e., the early stage, the middle stage, and the later stage), we expect that the first phase includes the early stage and some part of the middle stage of evolution, while the second phase includes the remaining part of the middle stage and the later stage of evolution. Next, the motivation to design the "current-to-rand/best/1" strategy is explained.

In the first phase, in order to prevent the population from getting stuck in a local optimum, it is very important to enhance the global search ability of the population. Since in the "current-to-rand/1" strategy the individuals learn the information from other individuals randomly chosen from the population, this strategy is able to enhance the global search ability of the population. Thus, the "current-to-rand/1" strategy is used in the first phase. Note that to further improve the global search ability of the population in the first phase, the first scaling factor of the "current-to-rand/1" strategy is randomly chosen between 0 and 1. In addition, it is necessary to accelerate the convergence of the population and guide the population toward the global optimum in the second phase. Since the "current-to-best/1" strategy exploits the information of the best individual in the current population, the convergence velocity of the population is very fast when implementing this strategy. Therefore, this strategy is well suited for the second phase.

Based on the above analysis, the "current-to-rand/best/1" strategy has the capability to maintain a good balance between the diversity and convergence of the population to some extent, by combining the "current-to-rand/1" strategy with the "current-to-best/1" strategy.

5.4. Archiving-based adaptive tradeoff model (ArATM)

By combining the parent population P_t with the offspring population Q_t, a combined population H_t is obtained. Note that for constrained optimization, the combined population H_t may inevitably experience three situations (i.e., the infeasible situation, the semi-feasible situation, and the feasible situation). In the infeasible situation, all the individuals in H_t are infeasible; in the semi-feasible situation, H_t consists of both feasible and infeasible individuals; and in the feasible situation, all the individuals in H_t are feasible.

As pointed out by Wang et al. [46], a good COEA should not only handle constraints effectively but also optimize the objective function successfully. Consequently, it is critical for a COEA to make a suitable tradeoff between the objective function and the degree of constraint violation. Based on the above consideration, a novel archiving-based adaptive tradeoff model (ArATM) is proposed in ICDE, and in ArATM three different constraint-handling mechanisms are designed for the above three situations, respectively.

5.4.1. The infeasible situation

In the infeasible situation, the combined population H_t contains infeasible individuals only. At the early stage of evolution, H_t is usually in the infeasible situation, especially when solving the highly COPs (i.e., the COPs have very small feasible region). Thus, the population should firstly enter the feasible region promptly. Meanwhile, the diversity of the population should be maintained.

To achieve the above purpose, a constraint-handling mechanism is designed in ArATM for the infeasible situation. This mechanism treats the objective function $f(\vec{x})$ and the degree of constraint violation $G(\vec{x})$ as two objectives, and deals with them simultaneously. As a result, the original COP is transformed into a bi-objective optimization problem.

In the infeasible situation, ArATM is executed as follows:

Step (1) If archive A is not empty, *randsize* individuals are randomly selected from A and put into the combined population H_t, where *randsize* is an integer randomly generated between 0 and $|A|$;

Step (2) Set $A = \Phi$;

Step (3) Set $P_{t+1} = \Phi$;
Step (4) **While** $|P_{t+1}| < \mu$ /* the hierarchical nondominated individual selection scheme [47]*/;
Step (5) The nondominated individuals in H_t are identified based on Pareto dominance;
Step (6) The nondominated individuals are sorted in ascending order based on their degree of constraint violations;
Step (7) The first half of the nondominated individuals are selected and stored into the population P_{t+1};
Step (8) These selected nondominated individuals are removed from H_t;
Step (9) **End**
Step (10) If $|P_{t+1}| > \mu$, delete the last $(|P_{t+1}| - \mu)$ individuals in P_{t+1} and store them into H_t;
Step (11) All the individuals in H_t are stored into A.

In Step (1) and Step (4), $|A|$ and $|P_{t+1}|$ denote the number of individuals in A and P_{t+1}, respectively. In the above procedure, the hierarchical nondominated individual selection scheme is used to motivate the population toward the feasible region and to maintain a good diversity of the population simultaneously. However, due to the limitation of the population size, in this selection scheme some good individuals (i.e., individuals with some important information) cannot win in the competition and, as a result, cannot enter the next population P_{t+1}, which results in the loss of the diversity of the population. To further enhance the diversity of the population, in Step (11) an individual archiving technique is proposed. In the individual archiving technique, the individuals, which cannot survive into the next population, are stored into the archive A and offered an opportunity to compete with individuals in the next combined population H_{t+1} again (Step (1)). Thus, some good individuals in A might win in the next competition, which plays a very important role in enhancing the diversity of the population throughout the evolution. By combining the hierarchical nondominated individual selection scheme with the individual archiving technique, the constraint-handling mechanism is very effective for the infeasible situation.

5.4.2. The semi-feasible situation

In the semi-feasible situation, the combined population H_t contains both infeasible and feasible individuals. For constrained optimization, how to deal with the infeasible individuals in the population is a very important issue. Since some information carried by certain infeasible individuals might be crucial to find the optimal solution, it is not reasonable to eliminate all infeasible individuals in the semi-feasible situation. Based on the above consideration, an adaptive fitness transformation scheme is designed in the semi-feasible situation, which selects not only some feasible individuals with small objective function values but also some infeasible individuals with both small degree of constraint violations and small objective function values for the next generation. The adaptive fitness transformation scheme is executed as follows [45].

At first, the feasible individuals and the infeasible individuals in H_t are identified and their subscripts are recorded into two sets Z_1 and Z_2, respectively. Here, assume that the size of H_t is NP (i.e., $NP = \mu + \lambda$).

$$Z_1 = \{i | G(\vec{x}_i) = 0, i \in \{1, \ldots, NP\}\} \tag{20}$$

$$Z_2 = \{i | G(\vec{x}_i) > 0, i \in \{1, \ldots, NP\}\} \tag{21}$$

Then, the best feasible individual $\vec{x}_{best}$ and the worst feasible individual $\vec{x}_{worst}$ are found by Eqs. (22) and (23), respectively:

$$f(\vec{x}_{best}) = \min_{i \in Z_1} f(\vec{x}_i) \tag{22}$$

$$f(\vec{x}_{worst}) = \max_{i \in Z_1} f(\vec{x}_i) \tag{23}$$

Next, the objective function $f(\vec{x}_i)$ is converted by equation (24) and, as a result, a converted objective function $f'(\vec{x}_i)$ $(i = 1, \ldots, NP)$ is obtained:

$$f'(\vec{x}_i) = \begin{cases} f(\vec{x}_i), & i \in Z_1 \\ \max\{\varphi \times f(\vec{x}_{best}) + (1 - \varphi) \times f(\vec{x}_{worst}), f(\vec{x}_i)\}, & i \in Z_2 \end{cases} \tag{24}$$

where φ is the feasibility proportion of the combined population H_t.

Subsequently, a normalized objective function $f_{nor}(\vec{x}_i)$ $(i = 1, \ldots, NP)$ is obtained by normalizing the converted objective function $f'(\vec{x}_i)$ by Eq. (25):

$$f_{nor}(\vec{x}_i) = \frac{f'(\vec{x}_i) - \min_{j \in Z_1 \cup Z_2} f'(\vec{x}_j)}{\max_{j \in Z_1 \cup Z_2} f'(\vec{x}_j) - \min_{j \in Z_1 \cup Z_2} f'(\vec{x}_j)}, \quad i = 1, \ldots, NP \tag{25}$$

Meanwhile, the degree of constraint violation of each individual $\vec{x}_i$ $(i = 1, \ldots, NP)$ is normalized by Eq. (26). Note that, if the degree of constraint violation of each individual is computed using the second criterion, it has been a normalized value and does not need to be normalized again. However, if the degree of constraint violation of each individual is computed using the first criterion, it needs to be normalized.

$$G_{nor}(\vec{x}_i) = \begin{cases} 0, & i \in Z_1 \\ G(\vec{x}_i), & i \in Z_2 \text{ and } criterion = 2 \\ \frac{G(\vec{x}_i) - \min_{j \in Z_2} G(\vec{x}_j)}{\max_{j \in Z_2} G(\vec{x}_j) - \min_{j \in Z_2} G(\vec{x}_j)}, & i \in Z_2 \text{ and } criterion = 1 \end{cases} \tag{26}$$

where "*criterion* = 1" denotes $G(\vec{x}_i)$ is computed using the first criterion while "*criterion* = 2" denotes $G(\vec{x}_i)$ is computed using the second criterion.

Finally, a final fitness function $f_{final}(\vec{x}_i)$ is obtained by adding the normalized objective function value $f_{nor}(\vec{x}_i)$ and the normalized degree of constraint violation $G_{nor}(\vec{x}_i)$ of each individual $\vec{x}_i$ $(i = 1, \ldots, NP)$ together:

$$f_{final}(\vec{x}_i) = f_{nor}(\vec{x}_i) + G_{nor}(\vec{x}_i), \quad i = 1, \ldots, NP \tag{27}$$

In the semi-feasible situation, μ individuals with the smallest $f_{final}(\vec{x}_i)$ are selected to constitute the population P_{t+1} for the next generation.

5.4.3. The feasible situation

In the feasible situation, all the individuals in the combined population H_t are feasible. In this case, COPs become unconstrained optimization problems. Therefore, all the individuals are compared with each other based only on their objective function values, and μ individuals with the least objective function values in H_t are selected to constitute the population P_{t+1} for the next generation.

5.5. Differences between ICDE and $(\mu + \lambda)$-CDE

Since ICDE is an improved version of $(\mu + \lambda)$-CDE, both $(\mu + \lambda)$-CDE and ICDE mainly consist of a search engine (i.e., $(\mu + \lambda)$-DE and IDE, respectively) and a constraint-handling technique (i.e., IATM and ArATM, respectively). However, some differences between $(\mu + \lambda)$-CDE and ICDE still exist and are summarized as follows:

- In both $(\mu + \lambda)$-DE and IDE, one parent generates three offspring by using three mutation strategies. Moreover, the first offspring and the second offspring are generated by the "rand/1" strategy and the "rand/2" strategy, respectively. In $(\mu + \lambda)$-DE, the third offspring is generated by an improved "current-to-best/1" strategy based on the feasibility proportion of the last population. However, IDE designs a new mutation strategy named "current-to-rand/best/1" to generate the third offspring according to the current generation number.
- For the infeasible situation, IATM and ArATM design different constraint-handling mechanisms to handle constraints. Compared with IATM, ArATM randomly selects *randsize* individuals from a predefined archive A where *randsize* is an integer randomly generated between 0 and the size of A, and then puts them into the combined population H_t to enhance the diversity of the population. After that, ArATM employs the hierarchical nondominated individual selection scheme developed in Ref. [47] to select the potential individuals for the next generation. Moreover, the archive A is used to store the individuals of H_t which cannot survive into the next population.
- For the semi-feasible situation, IATM and ArATM use different ways to convert the objective function. In IATM, the objective function is converted based on the feasibility proportion of the last population. However, the conversion of the objective function in ArATM is based on the feasibility proportion of the combined population H_t, the aim of which is to achieve a good balance between the diversity and the convergence of the population.
- In $(\mu + \lambda)$-CDE, the tolerance value δ for the equality constraints should be changed dynamically. Moreover, the initial value and the change rate of the tolerance value δ are problem-dependent in $(\mu + \lambda)$-CDE. However, in ICDE a fixed tolerance value δ (i.e., 0.0001) is set for the equality constraints during the evolution, which eliminates some extra parameters for ICDE.

6. Experimental study

6.1. Test functions

In this section, the performance of the proposed algorithm (i.e., ICDE) is tested on 24 well-known benchmark test functions collected for the special session on constrained real-parameter optimization of IEEE CEC2006 [22]. These test functions involve various kinds (linear, nonlinear, polynomial, quadratic, and cubic) of objective functions with different numbers of decision variables and different kinds (linear inequalities, linear equalities, nonlinear inequalities, and nonlinear equalities) and numbers of constraints. Table 1 exhibits the details of these 24 test functions. In this table, $\rho = |\Omega|/|S|$ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints, LE is the number of linear equality constraints, NE is the number of nonlinear equality constraints, a is the number of constraints active at the optimal solution, and $f(\vec{x}^*)$ is the objective function value of the best known solution $\vec{x}^*$. Note that 11 test functions (i.e., g03, g05, g11, g13, g14, g15, g17, g20, g21, g22, and g23) contain equality constraints. Because Wang and Cai [45] have found a better solution for test function g17: $\vec{x}^* = (201.784462493550, 99.999999999999, 383.071034852773, 419.999999999999, -10.907682614506, 0.073148231208)$ with $f(\vec{x}^*) = 8853.53387480648$, the $f(\vec{x}^*)$ of g17 in Table 1 is different from that in Liang et al. [22].

Table 1
Details of 24 benchmark test functions.

Prob.	n	Type of objective function	ρ (%)	LI	NI	LE	NE	a	$f(\vec{x}^*)$
g01	13	Quadratic	0.0111	9	0	0	0	6	−15.0000000000
g02	20	Nonlinear	99.9971	0	2	0	0	1	−0.8036191042
g03	10	Polynomial	0.0000	0	0	0	1	1	−1.0005001000
g04	5	Quadratic	51.1230	0	6	0	0	2	−30665.5386717834
g05	4	Cubic	0.0000	2	0	0	3	3	5126.4967140071
g06	2	Cubic	0.0066	0	2	0	0	2	−6961.8138755802
g07	10	Quadratic	0.0003	3	5	0	0	6	24.3062090681
g08	2	Nonlinear	0.8560	0	2	0	0	0	−0.0958250415
g09	7	Polynomial	0.5121	0	4	0	0	2	680.6300573745
g10	8	Linear	0.0010	3	3	0	0	6	7049.2480205286
g11	2	Quadratic	0.0000	0	0	0	1	1	0.7499000000
g12	3	Quadratic	4.7713	0	1	0	0	0	−1.0000000000
g13	5	Nonlinear	0.0000	0	0	0	3	3	0.0539415140
g14	10	Nonlinear	0.0000	0	0	3	0	3	−47.7648884595
g15	3	Quadratic	0.0000	0	0	1	1	2	961.7150222899
g16	5	Nonlinear	0.0204	4	34	0	0	4	−1.9051552586
g17	6	Nonlinear	0.0000	0	0	0	4	4	8853.5338748065
g18	9	Quadratic	0.0000	0	13	0	0	6	−0.8660254038
g19	15	Nonlinear	33.4761	0	5	0	0	0	32.6555929502
g20	24	Linear	0.0000	0	6	2	12	16	0.2049794002
g21	7	Linear	0.0000	0	1	0	5	6	193.7245100700
g22	22	Linear	0.0000	0	1	8	11	19	236.4309755040
g23	9	Linear	0.0000	0	2	3	1	6	−400.0551000000
g24	2	Linear	79.6556	0	2	0	0	2	−5.5080132716

6.2. Experimental setup

All the experiments are executed on Windows XP SP2 with Pentium Dual-Core 2.5 GHz processor and 2.0 GB RAM. The parameters in ICDE are set as follows: $\mu = 70$, $\lambda = 210$, $F = 0.8$, $CR = 0.9$, $p_m = 0.05$, $\eta = 200$, $k = 0.6$, and $\delta = 0.0001$. For each test function, ICDE runs 25 times independently and is stopped when a maximum of 5×10^5 fitness evaluations (FES) suggested by Liang et al. [22] is reached in each run.

6.3. Experimental results

The experimental results of ICDE are summarized in Tables 2–5. As suggested by Liang et al. [22], the best, median, worst, mean, and standard deviation (Std) of the function error values ($f(\vec{x}) - f(\vec{x}^*)$) of the achieved best solution $\vec{x}$ after 5×10^3, 5×10^4, and 5×10^5 FES in each run are included in these tables. Therein, "c" is a sequence of numbers indicating the number of constraints that are violated at the median solution by more than 1.0, between 0.01 and 1.0, and between 0.0001 and 0.01,

Table 2
Function error values achieved when FES = 5×10^3, FES = 5×10^4, and FES = 5×10^5 for test functions g01-g06.

FES		Prob.					
		g01	g02	g03	g04	g05	g06
5×10^3	Best	5.89E+00(0)	4.45E−01(0)	8.01E−01(0)	5.61E+01(0)	2.46E+00(3)	1.46E+00(0)
	Median	7.04E+00(0)	5.12E−01(0)	9.99E−01(1)	1.16E+02(0)	2.34E+02(3)	8.79E+00(0)
	Worst	8.12E+00(0)	5.51E−01(0)	8.55E−01(1)	1.89E+02(0)	3.22E+01(3)	2.84E+01(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 2, 0	0, 0, 0
	Mean	7.03E+00	5.11E−01	9.57E−01	1.16E+02	3.60E+01	1.06E+01
	Std	6.05E−01	2.54E−02	5.36E−02	3.09E+01	6.12E+01	6.97E+00
5×10^4	Best	2.56E−02(0)	2.62E−01(0)	1.85E−01(0)	3.06E−08(0)	7.00E−11(0)	3.37E−11(0)
	Median	4.91E−02(0)	3.02E−01(0)	4.41E−01(0)	2.75E−07(0)	1.38E−09(0)	3.37E−11(0)
	Worst	8.69E−02(0)	3.52E−01(0)	8.57E−01(0)	2.59E−06(0)	1.66E−07(0)	3.37E−11(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	5.28E−02	3.02E−01	4.60E−01	4.36E−07	1.11E−08	3.37E−11
	Std	1.58E−02	2.68E−02	1.72E−01	4.95E−07	3.33E−08	1.98E−26
5×10^5	Best	0.00E+00(0)	1.28E−08(0)	−1.00E−11(0)	7.64E−11(0)	−1.82E−12(0)	3.37E−11(0)
	Median	0.00E+00(0)	1.39E−07(0)	−1.00E−11(0)	7.64E−11(0)	−1.82E−12(0)	3.37E−11(0)
	Worst	0.00E+00(0)	7.43E−07(0)	−1.00E−11(0)	7.64E−11(0)	−1.82E−12(0)	3.37E−11(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	0.00E+00	2.28E−07	−1.00E−11	7.64E−11	−1.82E−12	3.37E−11
	Std	0.00E+00	2.06E−07	1.63E−16	2.64E−26	1.24E−27	1.98E−26

Table 3
Function error values achieved when FES = 5×10^3, FES = 5×10^4, and FES = 5×10^5 for test functions g07-g12.

FES		Prob.					
		g07	g08	g09	g10	g11	g12
5×10^3	Best	5.50E+01(0)	8.20E−11(0)	1.22E+01(0)	2.24E+03(0)	2.38E−07(0)	3.73E−05(0)
	Median	1.09E+02(0)	8.56E−11(0)	4.52E+01(0)	3.69E+03(0)	3.97E−06(0)	3.45E−04(0)
	Worst	2.02E+02(0)	1.44E−10(0)	9.51E+01(0)	5.72E+03(0)	1.34E−02(0)	1.18E−02(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	1.10E+02	9.25E−11	4.78E+01	3.86E+03	5.66E−04	1.78E−03
	Std	3.64E+01	1.61E−11	2.08E+01	8.19E+02	2.68E−03	3.24E−03
5×10^4	Best	8.30E−02(0)	8.20E−11(0)	2.87E−07(0)	3.78E+01(0)	0.00E+00(0)	0.00E+00(0)
	Median	1.72E−01(0)	8.20E−11(0)	1.09E−06(0)	9.06E+01(0)	0.00E+00(0)	0.00E+00(0)
	Worst	2.60E−01(0)	8.20E−11(0)	3.18E−06(0)	1.40E+02(0)	0.00E+00(0)	0.00E+00(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	1.75E−01	8.20E−11	1.23E−06	8.89E+01	0.00E+00	0.00E+00
	Std	4.05E−02	5.19E−18	6.54E−07	2.68E+01	0.00E+00	0.00E+00
5×10^5	Best	7.98E−11(0)	8.20E−11(0)	−9.82E−11(0)	6.18E−11(0)	0.00E+00(0)	0.00E+00(0)
	Median	7.98E−11(0)	8.20E−11(0)	−9.81E−11(0)	6.28E−11(0)	0.00E+00(0)	0.00E+00(0)
	Worst	7.98E−11(0)	8.20E−11(0)	−9.81E−11(0)	6.28E−11(0)	0.00E+00(0)	0.00E+00(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	7.98E−11	8.20E−11	−9.82E−11	6.27E−11	0.00E+00	0.00E+00
	Std	5.26E−15	2.78E−18	5.76E−14	2.52E−13	0.00E+00	0.00E+00

Table 4
Function error values achieved when FES = 5×10^3, FES = 5×10^4, and FES = 5×10^5 for test functions g13-g18.

FES		Prob.					
		g13	g14	g15	g16	g17	g18
5×10^3	Best	4.86E−01(3)	−2.80E+01(3)	4.85E−04(2)	6.14E−02(0)	9.19E+01(4)	1.69E−01(2)
	Median	3.16E+00(3)	−5.43E+01(3)	6.08E−02(2)	1.01E−01(0)	−2.21E+01(4)	−1.14E+00(6)
	Worst	5.93E−01(3)	−8.79E+01(3)	1.61E−01(2)	1.71E−01(0)	−2.83E+01(4)	−2.05E+00(7)
	c	0, 3, 0	2, 1, 0	0, 1, 1	0, 0, 0	1, 3, 0	5, 1, 0
	Mean	1.48E+00	−5.72E+01	1.03E−01	1.01E−01	1.11E+02	−5.39E−01
	Std	2.99E+00	1.84E+01	1.63E−01	2.48E−02	1.35E+02	9.64E−01
5×10^4	Best	3.68E−10(0)	1.38E−02(0)	6.08E−11(0)	1.48E−08(0)	1.00E+02(0)	7.55E−03(0)
	Median	2.16E−08(0)	4.03E−02(1)	6.08E−11(0)	3.24E−08(0)	3.95E−01(2)	1.44E−02(0)
	Worst	4.62E−05(1)	3.24E−02(3)	6.08E−11(0)	1.33E−07(0)	2.18E−01(4)	2.25E−02(0)
	c	0, 0, 0	0, 0, 1	0, 0, 0	0, 0, 0	0, 0, 2	0, 0, 0
	Mean	2.37E−06	5.96E−02	1.32E−26	4.49E−08	2.43E+01	1.49E−02
	Std	9.24E−06	6.45E−02	2.64E−27	2.99E−08	4.35E+01	3.95E−03
5×10^5	Best	4.19E−11(0)	8.51E−12(0)	6.08E−11(0)	6.52E−11(0)	−1.82E−11(0)	1.56E−11(0)
	Median	4.19E−11(0)	8.51E−12(0)	6.08E−11(0)	6.52E−11(0)	−1.82E−11(0)	1.56E−11(0)
	Worst	4.19E−11(0)	8.52E−12(0)	6.08E−11(0)	6.52E−11(0)	−1.46E−11(0)	1.56E−11(0)
	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	4.19E−11	8.51E−12	6.08E−11	6.52E−11	−1.78E−11	1.56E−11
	Std	1.13E−17	2.36E−15	1.32E−26	1.32E−26	9.51E−13	6.60E−27

respectively. In addition, the numbers in the parentheses after the function error values of the best, median, and worst solutions are the numbers of constraints unsatisfying the feasible conditions at the best, median, and worst solutions, respectively.

As shown in Tables 2–5, ICDE can find feasible solutions in all runs by using 5×10^3 FES for 13 test functions (i.e., g01, g02, g04, g06, g07, g08, g09, g10, g11, g12, g16, g19, and g24). Moreover, for five other test functions (i.e., g03, g05, g15, g18, and g21), feasible solutions are reached consistently by ICDE in all runs by using 5×10^4 FES. For five other test functions (i.e., g13, g14, g17, g22, and g23), ICDE is able to obtain feasible solutions in all runs by using 5×10^5 FES. Note that, ICDE fails to find feasible solutions only for g20. This test function is a highly constrained problem since it contains 14 equality constraints. To the best of our knowledge, no feasible solutions have been found so far for this test function.

By further analyzing the results in Tables 2–5, ICDE is capable of reaching the best known solution in Table 1 for test function g08 in each run by using 5×10^3 FES, while 5×10^4 FES are needed in each run to find a good feasible approximation for 10 other test functions (i.e., g04, g05, g06, g09, g11, g12, g13, g15, g16, and g24). Moreover, for 12 other test functions (i.e., g01, g02, g03, g07, g10, g14, g17, g18, g19, g20, g21, and g23), solutions obtained by ICDE are very close or equal to the best known solutions in each run by using 5×10^5 FES. For g22, ICDE cannot find any solution that is very close or equal to the best known solution in any run. Since g22 contains 22 decision variables and 19 equality constraints, it is very difficult for

Table 5
Function error values achieved when FES = 5×10^3, FES = 5×10^4, and FES = 5×10^5 for test functions g19-g24.

FES		Prob.					
		g19	g20	g21	g22	g23	g24
5×10^3	Best	2.70E+02(0)	7.01E+00(20)	7.45E+02(5)	4.38E+03(19)	5.80E+02(4)	4.68E−05(0)
	Median	3.55E+02(0)	9.06E+00(20)	5.35E+02(4)	1.39E+04(19)	6.53E+02(4)	1.53E−04(0)
	Worst	4.81E+02(0)	9.38E+00(20)	1.65E+02(5)	1.77E+04(19)	6.79E+02(4)	5.93E−04(0)
	c	0, 0, 0	2, 16, 2	2, 0, 2	14, 5, 0	2, 1, 1	0, 0, 0
	Mean	3.62E+02	8.19E+00	3.26E+02	1.22E+04	5.01E+02	2.12E−04
	Std	5.17E+01	1.64E+00	1.91E+02	4.91E+03	2.07E+02	1.51E−04
5×10^4	Best	2.83E+00(0)	3.93E−01(20)	2.49E−02(0)	4.54E+03(19)	1.60E+02(4)	4.67E−12(0)
	Median	5.18E+00(0)	6.83E−01(20)	4.37E−02(0)	4.01E+03(19)	4.70E+01(4)	4.67E−12(0)
	Worst	8.12E+00(0)	8.87E−01(20)	1.60E−01(0)	7.98E+03(19)	2.69E+02(4)	4.67E−12(0)
	c	0, 0, 0	1, 17, 2	0, 0, 0	13, 5, 1	0, 2, 2	0, 0, 0
	Mean	5.63E+00	5.79E−01	5.32E−02	6.54E+03	1.39E+02	4.67E−12
	Std	1.32E+00	1.74E−01	2.91E−02	3.69E+03	9.09E+01	0.00E+00
5×10^5	Best	4.63E−11(0)	−1.72E−06(20)	−3.05E−10(0)	2.81E+00(0)	−1.71E−13(0)	4.67E−12(0)
	Median	4.63E−11(0)	−6.14E−06(20)	−2.58E−10(0)	2.04E+01(0)	5.68E−14(0)	4.67E−12(0)
	Worst	4.63E−11(0)	3.46E−05(20)	−1.68E−10(0)	6.05E+01(0)	1.08E−12(0)	4.67E−12(0)
	c	0, 0, 0	2, 12, 1	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	Mean	4.63E−11	−4.67E−06	−2.53E−10	2.29E+01	1.02E−13	4.67E−12
	Std	5.06E−15	1.18E−05	2.80E−11	1.32E+01	2.92E−13	0.00E+00

EAs to enter the feasible region and find the optimal solution. Moreover, no EAs have found any solution that is very close or equal to the best known solution for g22 so far. For the sake of clarity, a histogram is drawn in Fig. 2 to show the distribution of the objective function values obtained by ICDE in 25 runs for g22. As shown in Fig. 2, all the 25 solutions provided by ICDE are not very far from the best known solution. The objective function values are distributed between 236.43 and 270 in 21 out of 25 trials; meanwhile, the objective function values of the remaining four trials are distributed between 270 and 297. To the best of our knowledge, for g22 ICDE obtains the minimum mean function error value (i.e., 22.9) among the existing methods by using 5×10^5 FES.

Table 6 records the FES required in each run for obtaining a solution satisfying the following success condition suggested by Liang et al. [22]: $f(\vec{x}) - f(\vec{x}^*) \leqslant 0.0001$ and $\vec{x}$ is a feasible solution. For test function g20, the objective function values of individuals in the population are smaller than that of the best known solution when the population is very close to the feasible region. Therefore, a changed success condition (i.e.,$|f(\vec{x}) - f(\vec{x}^*)| \leqslant 0.0001$) is used in this paper. Moreover, the feasible rate (rate of runs where at least one feasible solution is found), the success rate (rate of runs where at least one solution satisfying the success condition is found), and the success performance (the mean FES for successful runs multiplied by the number of total runs and divided by the number of successful runs) are also presented in Table 6.

As shown in Table 6, ICDE obtains 100% feasible rate for all the 24 test functions except for g20. On the other hand, ICDE obtains 100% success rate for all the 24 test functions except for g22. As far as the success performance is concerned, to satisfy the success condition, ICDE requires less than 5×10^3 FES for three test functions, less than 5×10^4 FES for 11 test functions, less than 2×10^5 FES for 16 test functions, less than 3×10^5 FES for 18 test functions, and less than 4×10^5 FES for all the test functions except for g22.

Figs. 3–8 show the convergence graphs of $\log(f(\vec{x}) - f(\vec{x}^*))$ over FES at the median run for each test function with the termination of 5×10^5 FES, where $\vec{x}$ is the best solution found. Note that points which satisfy $f(\vec{x}) - f(\vec{x}^*) \leqslant 0$ are not plotted in these figures. Since ICDE cannot find feasible solutions for g20 and all solutions obtained by ICDE for g22 cannot achieve the success condition, the convergence graphs of these two test functions are not included in Figs. 3–8.

From Figs. 3–8, it can be seen that a fast convergence (less than 2×10^5 FES) can be achieved for all the test functions except for six test functions (i.e., g02, g03, g10, g19, g21, and g23). For g03, about 2.5×10^5 FES are needed. Moreover, for four other test functions (i.e., g02, g10, g19, and g21), the target accuracy level can be reached by using about 3×10^5 FES. In addition, by using about 4×10^5 FES, the convergence is achieved for g23. From the above analysis, it can be concluded that ICDE has a quite fast convergence speed for most of these 22 test functions.

6.4. Comparison with some other state-of-the-art algorithms in constrained evolutionary optimization

To further verify the performance of ICDE, this subsection compares ICDE with five state-of-the-art methods: MPDE [42], GDE [17], jDE-2 [4], MDE [29], and $(\mu + \lambda)$-CDE [45] in terms of two performance metrics (i.e., the feasible rate and the success rate). Note that, DE is used by all these six methods. Moreover, all these methods run 25 times independently for all the 24 test functions.

Table 7 summarizes the experimental results for the above six methods. In this table, results for MPDE, GDE, jDE-2, MDE, and $(\mu + \lambda)$-CDE are directly taken from the original references respectively. Due to space limitation, the results of those test functions for which all the six methods can achieve both 100% feasible rate and 100% success rate are omitted in Table 7.

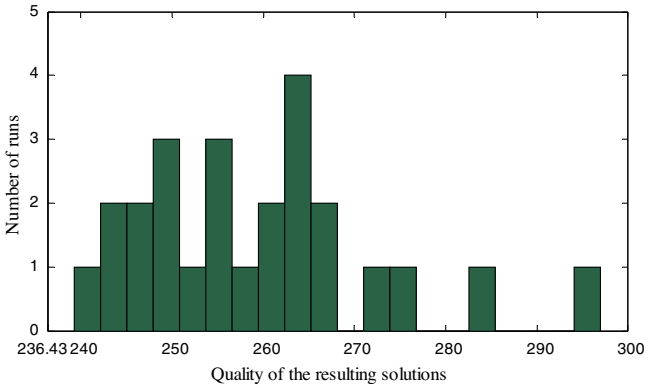

Fig. 2. Plot shows the number of runs versus the quality of the resulting solutions derived from 25 independent runs on test function g22.

Table 6
Number of FES to achieve the fixed accuracy level ($(f(\vec{x}) - f(\vec{x}^*)) \leqslant 0.0001$), feasible rate, success rate, and success performance.

Prob.	Best	Median	Worst	Mean	Std	Feasible rate (%)	Success rate (%)	Success performance
g01	92260	106540	115570	105776	5618.8	100	100	105776
g02	242830	281470	324100	283528	24408.2	100	100	283528
g03	175630	210490	245980	212657	19952.7	100	100	212657
g04	34300	36820	41020	36770	1791.5	100	100	36770
g05	24430	28420	31990	27933	1921.5	100	100	27933
g06	11200	12880	15400	13040	1013.1	100	100	13040
g07	121450	135730	146020	134789	5785.5	100	100	134789
g08	1120	1960	2380	1943	320.3	100	100	1943
g09	35770	37870	40180	37929	1152.6	100	100	37929
g10	315280	325570	336490	325007	5827.2	100	100	325007
g11	2380	3850	12880	4404	1955.8	100	100	4404
g12	3010	6580	8260	6488	1289.0	100	100	6488
g13	23590	34720	51940	34325	6822.1	100	100	34325
g14	79240	85120	97300	85758	4535.9	100	100	85758
g15	8890	9940	11620	10074	682.8	100	100	10074
g16	21280	25060	27370	25001	1558.0	100	100	25001
g17	80290	106540	138040	103230	14050.4	100	100	103230
g18	129010	134680	159460	138998	9921.3	100	100	138998
g19	268870	297640	308140	296145	8866.4	100	100	296145
g20	67690	383110	444010	357918	92127.7	0	100	357918
g21	306250	317170	332500	317447	6987.5	100	100	317447
g22	–	–	–	–	–	100	0	–
g23	336700	366730	386050	364806	11948.7	100	100	364806
g24	4270	5740	6790	5740	627.1	100	100	5740

As shown in Table 7, the mean feasible rate of ICDE (i.e., 95.83%) is superior to that of the other five methods. For MPDE, GDE, jDE-2, MDE, $(\mu + \lambda)$-CDE, and ICDE, the corresponding number of test functions for which the feasible solutions cannot be found in all runs is 4, 9, 2, 2, 2, and 1, respectively. For ICDE, g20 is only the test function with less than 100% feasible rate. Note that, although g22 is a highly constrained problem, ICDE can achieve 100% feasible rate, compared with 16% provided by $(\mu + \lambda)$-CDE and 0% provided by the other four methods. On the other hand, in terms of the mean success rate, the performance of ICDE also surpasses that of the five competitors. For MPDE, GDE, jDE-2, MDE, $(\mu + \lambda)$-CDE, and ICDE, the corresponding number of test functions which cannot be successfully solved in all runs is 8, 13, 11, 4, 3, and 1, respectively. For ICDE, g22 is only the test function that cannot be successfully solved in any run. All the other five methods also fail to solve this test function. It is necessary to emphasize that for g20, $(\mu + \lambda)$-CDE and ICDE are capable of obtaining 100% success rate.

In addition, the Friedman test [9] is also used to compare these six methods. Firstly, these six methods are ranked in terms of their feasible rates and success rates on each test function, respectively, in which the best method is ranked 1 and the worst one is ranked 6. If some methods have the same feasible rate or success rate, the average rank is assigned. Afterward, the average rank of each method on all the test functions is computed. Table 7 shows the average ranks of these six methods. It can be seen that ICDE has the smallest average ranks (i.e., 3.2917 and 2.8021, respectively) on all the 24 test functions among the six methods compared, in terms of both the feasible rate and the success rate.

Note that MPDE, GDE, jDE-2, MDE, and ICDE assign a constant value (i.e., 0.0001) to the tolerance values δ. However, $(\mu + \lambda)$-CDE changes the tolerance value δ dynamically during the evolution. Moreover, the settings of both the initial value

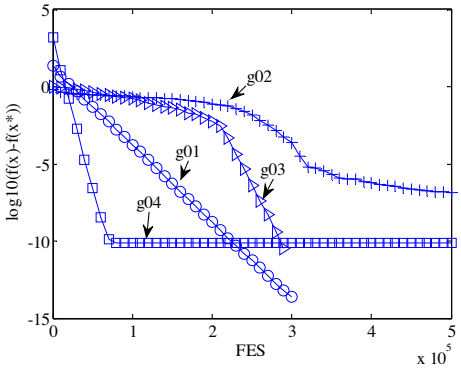

Fig. 3. Convergence graph for g01–g04.

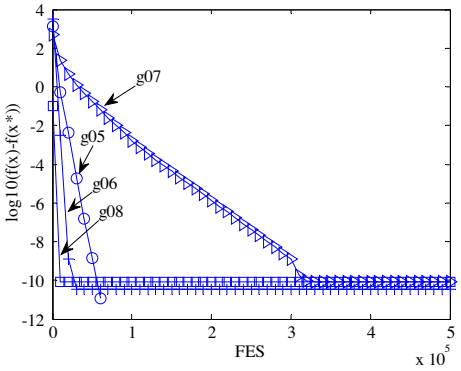

Fig. 4. Convergence graph for g05–g08.

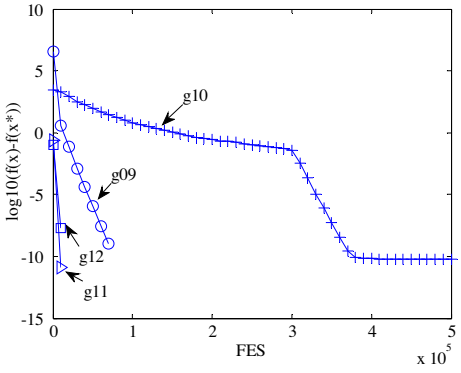

Fig. 5. Convergence graph for g09–g12.

and the change rate of the tolerance value δ are problem-dependent. The readers may be interested in the performance of $(\mu + \lambda)$-CDE with $\delta = 0.0001$ (denoted as $(\mu + \lambda)$-CDE-1) on the 24 test functions. Table 8 summarizes the experimental results of ICDE and $(\mu + \lambda)$-CDE-1 on eight test functions (i.e., g02, g05, g13, g15, g17, g21, g22, and g23). Note that, in Table 8 we omit the results of those test functions for which ICDE and $(\mu + \lambda)$-CDE-1 achieve the same feasible and success rates. According to the best, median, worse, mean, and standard deviation of the objective function values of the resulting solutions in Table 8, ICDE performs significantly better than $(\mu + \lambda)$-CDE-1. Moreover, both the feasible and success rates of ICDE are 95.83% and much better than those of $(\mu + \lambda)$-CDE-1 (91.67% and 81.83%, respectively). In addition, ICDE also performs better than $(\mu + \lambda)$-CDE. The above discussion verifies our motivation to propose ICDE.

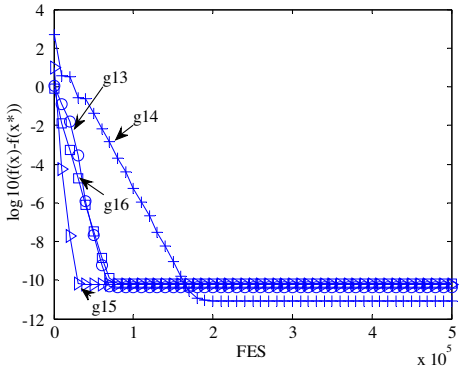

Fig. 6. Convergence graph for g13–g16.

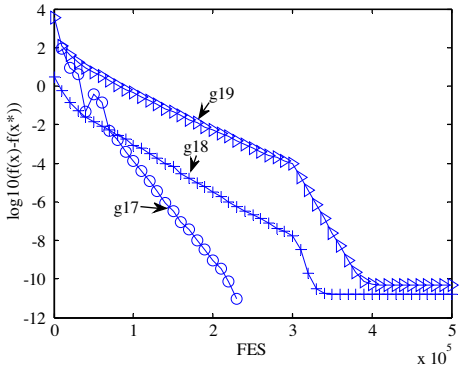

Fig. 7. Convergence graph for g17–g19.

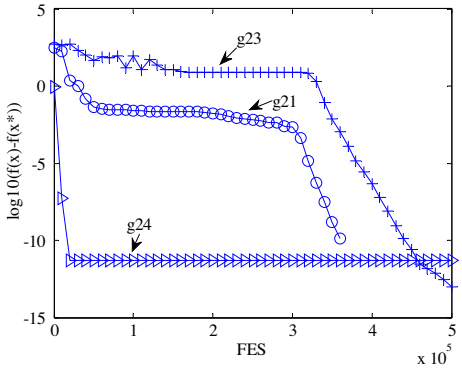

Fig. 8. Convergence graph for g21, g23 and g24.

Based on the above discussion, we can conclude that ICDE not only overcomes the drawbacks of $(\mu + \lambda)$-CDE but also performs significantly better than the five state-of-the-art algorithms.

7. Discussion

In this section, a series of experiments using the 24 benchmark test functions are carried out to demonstrate the effectiveness of some mechanisms and the rationality of some parameter settings in ICDE. Therein, the mean and standard deviation of the function error values $(f(\vec{x}) - f(\vec{x}^*))$ for the achieved best solution $\vec{x}$ in each run are calculated as the performance

Table 7
Comparison of ICDE with respect to MPDE [42], GDE [17], jDE-2 [4], MDE [29], and $(\mu + \lambda)$-CDE [45] in terms of feasible rate and success rate.

Prob.	Feasible rate						Success rate					
	MPDE	GDE	jDE-2	MDE	$(\mu + \lambda)$-CDE	ICDE	MPDE	GDE	jDE-2	MDE	$(\mu + \lambda)$-CDE	ICDE
g02	100%	100%	100%	100%	100%	100%	92%	72%	92%	16%	96%	100%
g03	100%	96%	100%	100%	100%	100%	84%	4%	0%	100%	100%	100%
g05	100%	96%	100%	100%	100%	100%	100%	92%	68%	100%	100%	100%
g11	100%	100%	100%	100%	100%	100%	96%	100%	96%	100%	100%	100%
g13	88%	88%	100%	100%	100%	100%	48%	40%	0%	100%	100%	100%
g14	100%	100%	100%	100%	100%	100%	100%	96%	100%	100%	100%	100%
g15	100%	100%	100%	100%	100%	100%	100%	96%	96%	100%	100%	100%
g17	96%	76%	100%	100%	100%	100%	28%	16%	4%	100%	100%	100%
g18	100%	84%	100%	100%	100%	100%	100%	76%	100%	100%	100%	100%
g19	100%	100%	100%	100%	100%	100%	100%	88%	100%	0%	100%	100%
g20	0%	0%	4%	0%	0%	0%	0%	0%	0%	0%	100%	100%
g21	100%	88%	100%	100%	100%	100%	68%	60%	92%	100%	92%	100%
g22	0%	0%	0%	0%	16%	100%	0%	0%	0%	0%	0%	0%
g23	100%	88%	100%	100%	100%	100%	100%	40%	92%	100%	100%	100%
Mean	91.00%	88.17%	91.83%	91.67%	92.33%	95.83%	84.00%	74.17%	76.67%	84.00%	95.33%	95.83%
Average rank	3.5417	4.0833	3.3125	3.4375	3.3333	3.2917	3.5521	4.3750	4.1042	3.2813	2.9271	2.8021

Table 8
Comparison of ICDE with respect to $(\mu + \lambda)$-CDE-1 on test functions g02, g05, g13, g15, g17, g21, g22, and g23 in terms of the best, median, worse, mean, and standard deviation of the objective function values of the resulting solutions, the feasible rate, and the success rate over 25 independent runs.

Prob.	Method	Best	Median	Worse	Mean	Std Dev	Feasible rate (%)	Success rate (%)
g02	ICDE	−0.803619	−0.803619	−0.803618	−0.803619	2.057E−07	100	100
	$(\mu + \lambda)$-CDE-1	−0.803619	−0.803619	−0.793084	−0.803198	2.107E−03	100	96
g05	ICDE	5126.496714	5126.496714	5126.496714	5126.496714	2.785E−12	100	100
	$(\mu + \lambda)$-CDE-1	5126.496714	5126.790268	5403.775786	5161.495574	7.615E+01	100	48
g13	ICDE	0.053942	0.053942	0.053942	0.053942	2.284E−17	100	100
	$(\mu + \lambda)$-CDE-1	0.053942	0.438803	0.497341	0.271810	1.974E−01	100	40
g15	ICDE	961.715022	961.715022	961.715022	961.715022	5.802E−13	100	100
	$(\mu + \lambda)$-CDE-1	961.715022	962.121714	965.269158	962.572689	1.128E+00	100	32
g17	ICDE	8853.533875	8853.533875	8853.533875	8853.533875	1.050E−12	100	100
	$(\mu + \lambda)$-CDE-1	8853.533875	8927.591747	9139.061438	8915.611557	5.753E+01	100	20
g21	ICDE	193.724510	193.724510	193.724510	193.724510	2.804E−11	100	100
	$(\mu + \lambda)$-CDE-1	193.724510	193.724510	324.702842	219.920176	5.347E+01	100	80
g22	ICDE	239.245930	256.806260	296.969718	259.305154	1.318E+01	100	0
	$(\mu + \lambda)$-CDE-1	432.618427	10195.980817	19759.703742	9565.607084	5.910E+03	0	0
g23	ICDE	−400.055100	−400.055100	−400.055100	−400.055100	2.999E−13	100	100
	$(\mu + \lambda)$-CDE-1	−400.055100	−400.054966	−58.894855	−362.332489	8.202E+01	100	48
					Mean	ICDE	95.83	95.83
						$(\mu + \lambda)$-CDE-1	91.67	81.83

indicators. According to these performance indicators, the Mann–Whitney test at a 0.05 significance level will be performed among the algorithms in next experiments. Note that in this section, (1) the feasible rate of one test function is provided to replace the average and standard deviation of the function error values, if for this test function feasible solutions cannot be consistently found by an algorithm in all runs, except for test function g20 for which no feasible solutions have been found so far; (2) to maintain a fair comparison, in this section all the experiments use the same parameter settings recommended in Section 6 unless new parameter settings are mentioned, and 25 independent runs are executed for all the 24 test functions; and (3) only the experimental results of those test functions for which the algorithms in comparison have significance differences are recorded in order to save space.

7.1. Effectiveness of the individual archiving technique

In ICDE, an individual archiving technique is proposed in the constraint-handling mechanism of the infeasible situation. To verify the effectiveness of this technique, this subsection performs another experiment for ICDE without the individual archiving technique (denoted as ICDE-1). In ICDE-1, the archive A is not used during the evolution. The experimental results of ICDE and ICDE-1 for test functions g13 and g22 are summarized in Table 9.

Table 9
Comparison of ICDE with respect to ICDE-1 on test functions g13 and g22 over 25 independent runs. "Mean Value" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Result in parentheses denotes the success rate. Result in square brackets denotes the feasible rate. The Mann–Whitney test at a 0.05 significance level is performed between ICDE and ICDE-1.

Prob.	Alg.	
	ICDE Mean Value ± Std Dev	ICDE-1 Mean Value ± Std Dev
g13	4.19E−11 ± 1.13E−17 (100%)	9.24E−02 ± 1.68E−01[a] (76%)
g22	2.29E+01 ± 1.32E+01 (0%)	[64%] (0%)

[a] ICDE performs significantly better than ICDE-1.

From Table 9, it can be seen that ICDE performs significantly better than ICDE-1 on g13 and g22, according to the performance indicators and the Mann–Whitney test.

Test function g13 is a highly constrained problem since it contains three nonlinear equality constraints. Even if these nonlinear equality constraints have been relaxed by the tolerance value 0.0001, the solution of g13 is still very difficult. It is very important for an algorithm to maintain a good diversity of the population throughout the evolution when solving g13. As shown in Table 9, the success rate of ICDE-1 is only 76%. However, 100% success rate is achieved by ICDE. This is because by the individual archiving technique, ICDE is able to maintain a better diversity of the population than ICDE-1.

Among the 24 test functions, g22 is one of the most highly constrained problems. It contains 20 constraints (1 nonlinear inequality constraint, 8 linear equality constraints, and 11 nonlinear equality constraints). So far, the optimal solution of g22 has not yet been found by the pure EAs. From Table 9, it can be seen that although the success rates of both ICDE and ICDE-1 are 0%, the feasible rate of ICDE is 100% and the feasible rate of ICDE-1 is only 64%. The above results verify the effectiveness of the individual archiving technique again. Note that ICDE is the first pure EA that is able to solve g22 with 100% feasible rate, according to the best of our knowledge.

From the above discussion, it can be concluded that for ICDE the individual archiving technique plays a very important role in maintaining the diversity of the population, especially when solving test functions g13 and g22.

7.2. Effectiveness of the "current-to-rand/best/1" strategy

In ICDE, a new mutation strategy (named the "current-to-rand/best/1" strategy) is proposed to generate the offspring. To confirm the effectiveness of this mutation strategy, another algorithm (denoted as ICDE-2) is constructed by replacing this mutation strategy with the improved "current-to-best/1" strategy [45] in ICDE.

Table 10 summarizes the experimental results of ICDE and ICDE-2 for six test functions (i.e., g10, g13, g17, g21, g22, and g23). From Table 10, it can be seen that ICDE is significantly better than ICDE-2 on the above six test functions according to the performance indicators and the Mann–Whitney test.

Test function g10 contains eight decision variables and six inequality constraints. Moreover, the estimated feasibility ratio of g10 in Table 1 is only 0.001%. From Table 10, it can be seen that ICDE-2 provides only 4% success rate for g10. According to our observation to the evolutionary process of ICDE-2, in the improved "current-to-best/1" strategy, the "current-to-best/1" strategy is frequently used and the "current-to-rand/1" strategy is rarely used in the infeasible situation. However, in the semi-infeasible and feasible situations, the phenomenon of using these two mutation strategies is just opposite. Actually,

Table 10
Comparison of ICDE with respect to ICDE-2 on test functions g10, g13, g17, g21, g22, and g23 over 25 independent runs. "Mean Value" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Result in parentheses denotes the success rate. Result in square brackets denotes the feasible rate. The Mann–Whitney test at a 0.05 significance level is performed between ICDE and ICDE-2.

Prob.	Alg.	
	ICDE Mean Value ± Std Dev	ICDE-2 Mean Value ± Std Dev
g10	6.27E−11 ± 2.52E−13 (100%)	9.08E−04 ± 9.05E−04[a] (4%)
g13	4.19E−11 ± 1.13E−17 (100%)	1.08E−01 ± 1.76E−01[a] (72%)
g17	−1.78E−11 ± 9.51E−13 (100%)	1.19e+01 ± 2.77e+01[a] (84%)
g21	−2.53E−10 ± 2.80E−11 (100%)	3.26E+01 ± 5.49E+01[a] (76%)
g22	2.29E+01 ± 1.32E+01 (0%)	[44%] (0%)
g23	1.02E−13 ± 2.92E−13 (100%)	1.73E+01 ± 1.37E+01[a] (0%)

[a] ICDE performs significantly better than ICDE-2.

the frequency of using the "current-to-rand/1" strategy is about 99 times of that of using the "current-to-best/1" strategy. Since the use of the "current-to-rand/1" strategy will slow down the convergence velocity, in ICDE-2 the population converges too slowly to the optimal solution in 24 out 25 runs for g10. In contrast, ICDE achieves 100% success rate. This is because ICDE firstly uses the "current-to-rand/1" strategy, and then switches to the "current-to-best/1" strategy according to the current generation number. After doing this, ICDE keeps a good balance between the diversity and convergence of the population.

Test function g13 contains three nonlinear equality constraints. To find the optimal solution of g13, it is vital to maintain a good diversity of the population, especially in the early stage of evolution. As shown in Table 10, ICDE-2 provides only 72% success rate. This is because the frequent use of the "current-to-best/1" strategy in the early stage results in the loss of the diversity of the population. Thus, the population converges to the current best individual promptly and ICDE-2 fails to find the optimal solution in 7 out of 25 runs. In contrast, ICDE puts emphasis on maintaining the diversity of the population by consistently using the "current-to-rand/1" strategy in the early stage of evolution and, consequently, ICDE obtains 100% success rate.

Both test functions g17 and g21 are highly constrained problems. Test function g17 contains four nonlinear equality constraints and test function g21 contains five nonlinear equality constraints. To solve these two test functions, the population should have a good diversity in the early stage while a fast convergence is required in the later stage of evolution. Due to the improved "current-to-best/1" strategy, ICDE-2 cannot maintain a good diversity of the population in the early stage and the population converges very slowly in the later stage of evolution. As a result, ICDE-2 provides only 84% and 76% success rates for g17 and g21, respectively. However, by making use of the "current-to-rand/best/1" strategy, ICDE obtains a very good diversity and a fast convergence of the population in the early and later stages of evolution, respectively. Therefore, ICDE achieves 100% success rate. The above analysis can be further verified by the results of g22 in Table 10. Although g22 is also a highly constrained problem, the feasible rate of ICDE is 100%. However, ICDE-2 can only attain 44% feasible rate.

Test function g23 contains six constraints (two nonlinear inequality constraints, three linear equality constraints, and one nonlinear equality constraint). Since ICDE-2 frequently uses the "current-to-rand/1" strategy in the later stage of evolution, the population converges very slowly and finally fails to converge to the optimal solution in all runs. ICDE uses the "current-to-best/1" strategy in the later stage of evolution, so the population can converge to the optimal solution promptly. As a result, 100% success rate is obtained by ICDE.

From the above discussion, it can be seen that the "current-to-rand/best/1" strategy has the capability to maintain a good diversity of the population in the early stage of evolution and accelerate the convergence of the population in the later stage of evolution. In summary, ICDE is able to effectively balance the diversity and the convergence of the population during the evolution.

7.3. Effectiveness of the feasibility proportion of the combined population H_t

In Eq. (24), instead of using the feasible proportion of the last population as in $(\mu + \lambda)$-CDE, ICDE uses the feasibility proportion of the combined population H_t to convert the objective function of each individual in the combined population H_t. To verify the effectiveness of the above improvement, this subsection tests a new algorithm (denoted as ICDE-3) by replacing the feasibility proportion of the combined population H_t with the feasibility proportion of the last population in ICDE. The experimental results of ICDE and ICDE-3 for test function g23 are summarized in Table 11.

From Table 11, it can be seen that ICDE is statistically better than ICDE-3 on test function g23 according to the Mann–Whitney test. Moreover, the success rate of ICDE (i.e., 100%) is higher than that of ICDE-3 (i.e., 24%).

Eq. (24) attempts to make a good tradeoff between the feasible and infeasible solutions adaptively, according to the information obtained during the evolution. To exploit the feedback information of the population accurately, using the feasibility proportion of the combined population H_t is more reasonable in Eq. (24). In contrast, using the feasibility proportion of the last population might mislead the algorithm to make a wrong choice among the feasible and infeasible individuals, and thus, break the balance between the diversity and convergence of the population in the end.

The experimental results for g23 in Table 11 verify the above analysis. According to our observation, for this test function, the feasibility proportion of the last population is much larger than that of the combined population H_t. After using the feasibility proportion of the last population, ICDE-3 makes a wrong choice, that is, more infeasible individuals are selected

Table 11
Comparison of ICDE with respect to ICDE-3 on test function g23 over 25 independent runs. "Mean Value" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Result in parentheses denotes the success rate. The Mann–Whitney test at a 0.05 significance level is performed between ICDE and ICDE-3.

Prob.	Alg.	
	ICDE Mean Value ± Std Dev	ICDE-3 Mean Value ± Std Dev
g23	1.02E−13 ± 2.92E−13 (100%)	5.05E+00 ± 5.98E+00[a] (24%)

[a] ICDE performs significantly better than ICDE-3.

Table 12
Experimental results of ICDE with varying k on test functions g02, g20, and g22 over 25 independent runs. "Mean Value" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs, respectively. Result in parentheses denotes the success rate. Result in square brackets denotes the feasible rate. The Mann–Whitney test at a 0.05 significance level is performed between ICDE and other variants of ICDE.

Prob.	k				
	0.4 Mean Value ± Std Dev	0.5 Mean Value ± Std Dev	0.6 Mean Value ± Std Dev	0.7 Mean Value ± Std Dev	0.8 Mean Value ± Std Dev
g02	4.05E−03 ± 5.72E−03[a] (64%)	1.24E−03 ± 3.44E−03[a] (88%)	2.28E−07 ± 2.06E−07 (100%)	2.21E−07 ± 2.94E−07 (100%)	2.04E−07 ± 6.70E−07 (100%)
g20	−4.49E−06 ± 1.37E−06 (100%)	−4.58E−06 ± 5.32E−06 (100%)	−4.67E−06 ± 1.18E−05 (100%)	−3.87E−05 ± 1.42E−04 (92%)	−7.98E−05 ± 1.40E−04[a] (60%)
g22	[84%] (0%)	[92%] (0%)	2.29E+01 ± 1.32E+01 (0%)	[96%] (0%)	[92%] (0%)

[a] means that the corresponding result is significantly worse than that of ICDE with $k = 0.6$.

while some good feasible individuals are eliminated. As a result, the convergence of the population is decelerated and ICDE-3 fails to converge to the optimal solution in 19 out of 25 runs. In contrast, ICDE can succeed in solving g23 in all runs.

The above comparison demonstrates that it is more appropriate and effective for ICDE to adopt the feasibility proportion of the combined population H_t in Eq. (24). It is because by using the feasibility proportion of the combined population H_t, ICDE has a good capability to balance the feasible and infeasible solutions, especially when solving test function g23.

7.4. The rationality of k

In ICDE, the factor k plays a very important role in keeping a good balance between the diversity and convergence of the population throughout the evolution. If k is too small, the "current-to-best/1" strategy will be used very frequently. As a result, the convergence of the population may be very fast while the diversity of the population may become poor. Thus, premature convergence might occur. However, if k is too big, the "current-to-rand/1" strategy will be used very frequently. As a result, the diversity of the population may be very good while the convergence of the population may become slow. Ultimately, incomplete convergence might occur. Therefore, a proper value of k is vital for the performance of ICDE.

In this subsection, the factor k is tested with five different values (i.e., 0.4, 0.5, 0.6, 0.7, and 0.8) and the experimental results for three test functions (i.e., g02, g20, and g22) are summarized in Table 12.

Test function g02 includes 20 decision variables and a nonlinear objective function. Moreover, the estimated feasibility ratio of g02 in Table 1 is very large (i.e., 99.9971%). Therefore, to find the global optimum, the population should have strong global search ability. From Table 12, it can be seen that the results become better with the increase of k from 0.4 to 0.8. This is because a bigger value of k enhances the diversity of the population, and thus, ICDE can obtain better results for g02. The above results also suggest that a suitable value of k is between 0.6 and 0.8 for g02.

Test function g20 is one of the most difficult constrained problems among these 24 test functions, which contains 20 constraints (six nonlinear inequality constraints, two linear equality constraints, and 12 nonlinear equality constraints). So far, no feasible solution has been found by the pure EAs for g20. As shown in Table 12, the performance of the algorithms gradually decreases with the increase of k from 0.4 to 0.8. This is because with the increase of k, the "current-to-best/1" strategy is used less and less, and thus, the convergence of the population becomes slower and slower. As a result, incomplete convergence frequently occurs. The success rates of ICDE with $k = 0.7$ and 0.8 are 92% and 60%, respectively, while all the other three algorithms (i.e., ICDE with $k = 0.4$, 0.5, and 0.6) obtain 100% success rate. Additionally, the results of ICDE with $k = 0.6$ are statistically better than that of ICDE with $k = 0.8$, according to the Mann–Whitney test. Therefore, a value between 0.4 and 0.6 is suitable for k especially when solving g20.

Test function g22 is also a very highly constrained problem. So far, no EAs have found the optimal solution for this test function. Moreover, finding feasible solutions for g22 is also considerably difficult. As indicated in Table 12, ICDE with $k = 0.6$ obtains the best feasible rate (i.e., 100%) among the five algorithms. Moreover, with the decrease of k from 0.6 to 0.4, the feasible rates also gradually decrease. The reason is that in these cases the convergence velocity becomes faster while the diversity of the population becomes worse. As a result, no feasible solution can be found in some trials. On the other hand, the feasible rates gradually decrease with the increase of k from 0.6 to 0.8. This is because with the increase of k, the diversity of the population becomes better however the convergence of the population becomes slower. As a result, the convergence velocity of the population might be too slow to enter the feasible region in some trials. Therefore, ICDE with $k = 0.6$ can keep a good balance between the diversity and the convergence of the population, especially for g22.

From the above discussion, it is obvious that ICDE with $k = 0.6$ can keep the best balance between the diversity and the convergence of the population and obtain the best overall performance on the 24 test functions. Therefore, $k = 0.6$ is recommended for ICDE.

8. Conclusion and future work

Very recently, Wang and Cai [45] proposed $(\mu + \lambda)$-constrained differential evolution ($(\mu + \lambda)$-CDE) for constrained optimization problems (COPs). However, $(\mu + \lambda)$-CDE has to dynamically change the tolerance value for the equality constraints.

Moreover, both the initial value and the change rate of the tolerance value are problem-dependent. Recognizing the above drawbacks of $(\mu + \lambda)$-CDE, we propose an improved version of $(\mu + \lambda)$-CDE, named ICDE.

ICDE is mainly composed of an improved $(\mu + \lambda)$-differential evolution (IDE) and an archiving-based adaptive tradeoff model (ArATM). Note that the former serves as the search engine and the latter is used to handle the constraints. Moreover, ICDE uses two criteria to compute the degree of constraint violation of each individual in the population, according to the difference among the violations of different constraints. Compared with $(\mu + \lambda)$-CDE, ICDE has four main differences: (1) a new mutation strategy named "current-to-rand/best/1" is designed in ICDE to replace the improved "current-to-best/1" strategy in $(\mu + \lambda)$-CDE for producing the offspring; (2) ICDE uses the hierarchical nondominated individual selection scheme [47] and proposes an individual archiving technique in the constraint-handling mechanism of the infeasible situation; (3) in the constraint-handling mechanism of the semi-infeasible situation, instead of using the feasibility proportion of the last population as in $(\mu + \lambda)$-CDE, the feasibility proportion of the combined population H_t is employed by ICDE to convert the objective function of each individual; and 4) instead of dynamically changing the tolerance value for the equality constraint as in $(\mu + \lambda)$-CDE, a fixed tolerance value for the equality constraints is adopted in ICDE.

The performance of ICDE is tested on 24 benchmark test functions collected for the special session on constrained real parameter optimization of IEEE CEC2006. The experimental results show that ICDE not only overcomes the main drawbacks of $(\mu + \lambda)$-CDE but also obtains very competitive performance compared with other state-of-the-art methods for COPs in the community of constrained evolutionary optimization. ICDE can achieve both 100% feasible rate and 100% success rate for all these 24 test functions except for two highly constrained test functions (i.e., g20 and g22). For test function g20, ICDE obtains 100% success rate. Moreover, the feasible rate of ICDE is 100% for g22. Note that ICDE is the first pure EA that provides 100% feasible rate for g22, to the best of our knowledge. Both the mean feasible and success rates derived from ICDE are 95.83%. In addition, the effectiveness of some mechanisms proposed in ICDE is verified empirically. Furthermore, the rationality of the setting for the factor k is also studied by experiments.

In future studies, we will introduce some other search algorithms and constraint-handling techniques to further improve the performance of ICDE. Moreover, we are considering the possibility of extending ICDE to deal with constrained combinational optimization problems and constrained multiobjective optimization problems. Finally, the application of ICDE to solve real-world optimization problems is also our future work.

The source code of ICDE is available upon request from the corresponding author.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61273314, 60805027, 61175064, and 90820302), in part by the Hong Kong Scholars Program, and in part by the Research Fund for the Doctoral Program of Higher Education of China (Grant No. 200805330005). The authors appreciate the anonymous reviewers for their very constructive comments and suggestions, which greatly improve the quality of this paper.

References

[1] H. Aguirre, S.B. Rionda, C.A. Coello Coello, G.L. Lizárraga, E. Mezura-Montes, Handling constraints using multiobjective optimization concepts, International Journal for Numerical Methods in Engineering 59 (15) (2004) 1989–2017.
[2] R.L. Becerra, C.A. Coello Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering 195 (33–36) (2006) 4303–4322.
[3] J. Brest, Constrained real-parameter optimization with ε-self-adaptive differential evolution, in: Efrén Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Studies in Computational Intelligence, vol. 198, Springer-Verlag, Berlin, 2009, pp. 73–93. chapter 3.
[4] J. Brest, V. Zumer, M.S. Maucec, Self-adaptive differential evolution algorithm in constrained real-parameter optimization, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 215–222.
[5] Z. Cai, Y. Wang, A multiobjective optimization-based evolutionary algorithm for constrained optimization, IEEE Transactions on Evolutionary Computation 10 (6) (2006) 658–675.
[6] C.A. Coello Coello, Constraint-handling using an evolutionary multiobjective optimization technique, Civil Engineering and Environmental Systems 17 (4) (2000) 319–346.
[7] C.A. Coello Coello, E. Mezura-Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics 16 (3) (2002) 193–203.
[8] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2–4) (2000) 311–338.
[9] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[10] R. Farmani, J.A. Wright, Self-adaptive fitness formulation for constrained optimization, IEEE Transactions on Evolutionary Computation 7 (5) (2003) 445–455.
[11] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Information Sciences 181 (18) (2011) 3749–3765.
[12] W. Gong, A. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Information Sciences 181 (24) (2011) 5364–5386.
[13] S. Hamida, M. Schoenauer, ASCHEA: new results using adaptive segregational constraint handling, in: Proceedings of the Congress on Evolutionary Computation (CEC'2002), IEEE Press, Piscataway, New Jersey, USA, 2002, pp. 884–889.
[14] V.L. Huang, A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for constrained real-parameter optimization, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 17–24.
[15] F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics Computation 186 (1) (2007) 340–356.
[16] F. Jiménez, J. Vrdegay, Evolutionary techniques for constrained optimization problems, in: Proceedings of 7th European Congress on Intelligence Techniques and Soft Computing (EUFIT'99), Springer-Verlag, Berlin, Germany, 1999.

[17] S. Kukkonen, J. Lampinen, Constrained real-parameter optimization with generalized differential evolution, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 207–214.
[18] J. Lampinen, A constraint handling approach for the differential evolution algorithm, Proceedings of the Congress on Evolutionary Computation (CEC'2002), vol. 2, IEEE Service Center, Piscataway, New Jersey, USA, 2002, pp. 1468–1473.
[19] J. Lampinen, I. Zelinka, Mechanical engineering design optimization by differential evolution, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, UK, 1999, pp. 127–146.
[20] J. Lampinen, I. Zelinka, Mixed integer-discrete-continuous optimization by differential evolution, Part 1: the optimization method, in: Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing, Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, 1999, pp. 71–76.
[21] J. Lampinen, I. Zelinka, Mixed integer-discrete-continuous optimization by differential evolution, Part 2: a practical example, in: Proceedings of MENDEL'99, 5th International Mendel Conference on Soft Computing, Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, 1999, pp. 77–81.
[22] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2006.
[23] Y.C. Lin, K.S. Hwang, F.S. Wang, Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization problems, Proceedings of the Congress on Evolutionary Computation (CEC'2002), vol. 1, Piscataway, New Jersey, USA, 2002, pp. 872–877.
[24] E. Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Springer, Berlin, Germany, 2009. ISBN: 978-3-642-00618-0.
[25] E. Mezura-Montes, C.A. Coello Coello, A simple multimembered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 1–17.
[26] E. Mezura-Montes, M. Miranda-Varela, R. Gómez-Ramón, Differential evolution in constrained numerical optimization: an empirical study, Information Sciences 180 (22) (2010) 4223–4262.
[27] E. Mezura-Montes, A.G. Palomeque-Ortiz, Self-adaptive and deterministic parameter control in differential evolution for constrained optimization, in: Efrén Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Studies in Computational Intelligence, vol. 198, Springer-Verlag, Berlin, 2009, pp. 95–120. chapter 3.
[28] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello Coello, Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005), vol. 1, ACM Press, Washington, DC, USA, 2005, pp. 225–232.
[29] E. Mezura-Montes, J. Velázquez-Reyes, C.A. Coello Coello, Modified differential evolution for constrained optimization, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 25–32.
[30] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artificial Intelligence Review 33 (1–2) (2010) 61–106.
[31] Q. Pan, L. Wang, L. Gao, W. Li, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, Information Sciences 181 (3) (2011) 668–685.
[32] K. Price, R. Storn, J.A. Lampinen, Differential evolution: a practical approach to global optimization (Natural Computing Series), 1st ed., Springer, New York, 2005. ISBN: 3540209506.
[33] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, Proceedings of the Congress on Evolutionary Computation (CEC'2005), vol. 2, IEE Service Center, Edinburgh, Scotland, 2005, pp. 1785–1791.
[34] T. Runarsson, X. Yao, Search biases in constrained evolutionary optimization, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Systems and Human 35 (2) (2005) 233–243.
[35] H.P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981.
[36] R. Storn, System design by constraint adaptation and differential evolution, IEEE Transactions on Evolutionary Computation 3 (1) (1999) 22–34.
[37] R. Storn, K. Price, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, Berkeley, CA, Technical Report, TR-95-012, 1995.
[38] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[39] C. Sun, J. Zeng, J. Pan, An improved vector particle swarm optimization for constrained optimization problems, Information Sciences 181 (6) (2011) 1153–1163.
[40] T. Takahama, S. Sakai, Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 1–8.
[41] T. Takahama, S. Sakai, Solving difficult constrained optimization problems by the ε-constrained differential evolution with gradient-based mutation, in: Efrén Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Studies in Computational Intelligence, vol. 198, Springer-Verlag, Berlin, 2009, pp. 51–72. chapter 3.
[42] M.F. Tasgetiren, P.N. Suganthan, A multi-populated differential evolution algorithm for solving constrained optimization problem, in: Proceedings of the Congress on Evolutionary Computation (CEC'2006), IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, 2006, pp. 33–40.
[43] B. Tessema, G.G. Yen, An adaptive penalty formulation for constrained evolutionary optimization, IEEE Transactions on System, Man, and Cybernetics – Part A: Systems and Humans 39 (3) (2009) 565–578.
[44] S. Venkatraman, G.G. Yen, A generic framework for constrained optimization using genetic algorithms, IEEE Transactions on Evolutionary Computation 9 (4) (2005) 424–435.
[45] Y. Wang, Z. Cai, Constrained evolutionary optimization by means of $(\mu+\lambda)$-differential evolution and improved adaptive trade-off model, Evolutionary Computation 19 (2) (2011) 249–285.
[46] Y. Wang, Z. Cai, G. Guo, Y. Zhou, Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 37 (3) (2007) 560–575.
[47] Y. Wang, Z. Cai, Y. Zhou, W. Zeng, An adaptive tradeoff model for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 80–92.
[48] X. Wang, Y. He, L. Dong, H. Zhao, Particle swarm optimization for determining fuzzy measures from data, Information Sciences 181 (19) (2011) 4230–4252.
[49] M. Weber, F. Neri, V. Tirronen, A study on scale factor in distributed differential evolution, Information Sciences 181 (12) (2011) 2488–2511.
[50] L. Wei, Z. Chen, J. Li, Evolution strategies based adaptive Lp LS-SVM, Information Sciences 181 (14) (2011) 3000–3016.
[51] Y. Zhou, Y. Li, J. He, L. Kang, Multi-objective and MGG evolutionary algorithm for constrained optimization, Proceedings of the Congress on Evolutionary Computation (CEC'2003), vol. 1, IEEE Press, Piscataway, New Jersey, USA, 2003, pp. 1–5.