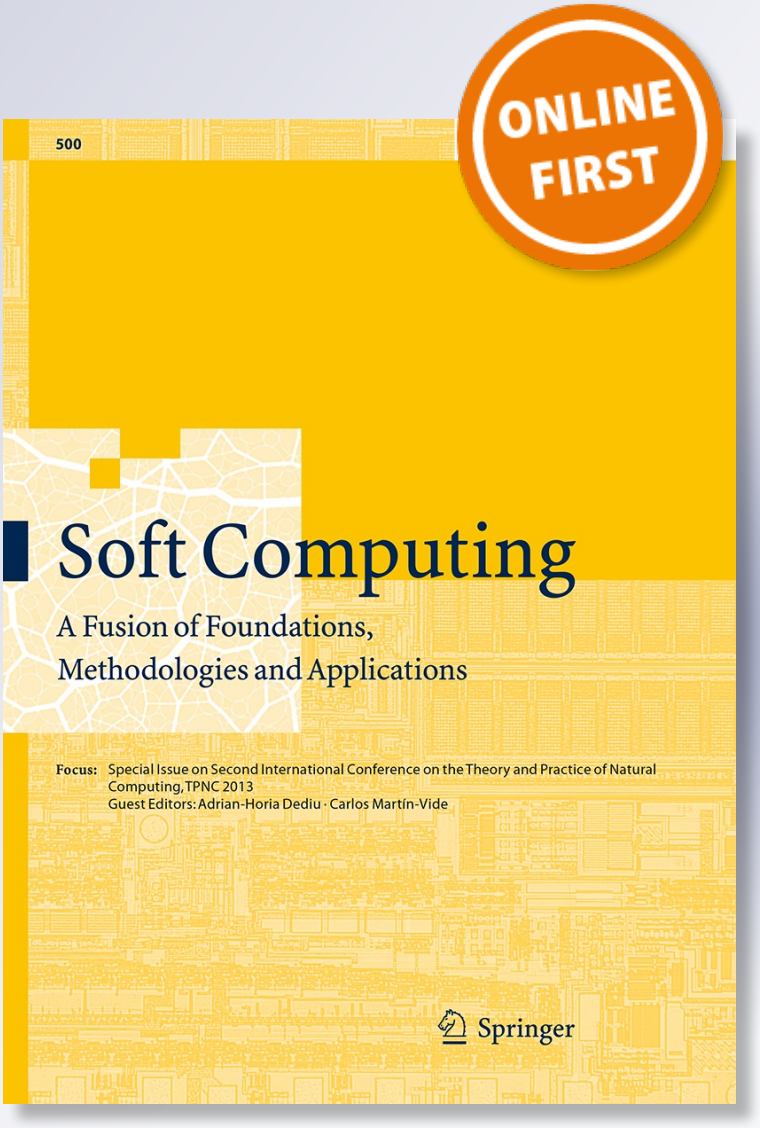
Constrained differential evolution using generalized opposition-based learning

Wenhong Wei, Jianlong Zhou, Fang Chen & Huaqiang Yuan

Soft Computing
A Fusion of Foundations, Methodologies and Applications

ISSN 1432-7643

Soft Comput
DOI 10.1007/s00500-015-2001-1



Springer

Soft Comput
DOI 10.1007/s00500-015-2001-1

FOUNDATIONS

Constrained differential evolution using generalized opposition-based learning

Wenhong Wei[1,2] · **Jianlong Zhou**[2] · **Fang Chen**[2] · **Huaqiang Yuan**[1]

Abstract Differential evolution (DE) is a well-known optimization approach to deal with nonlinear and complex optimization problems. However, many real-world optimization problems are constrained problems that involve equality and inequality constraints. DE with constraint handling techniques, named constrained differential evolution (CDE), can be used to solve constrained optimization problems. In this paper, we propose a new CDE framework that uses generalized opposition-based learning (GOBL), named GOBL-CDE. In GOBL-CDE, firstly, the transformed population is generated using general opposition-based learning in the population initialization. Secondly, the transformed population and the initial population are merged and only half of the best individuals are selected to compose the new initial population to proceed mutation, crossover, and selection. Lastly, based on a jumping probability, the transformed population is calculated again after generating new populations, and the fittest individuals are selected to compose new population from the union of the current population and the transformed population. The GOBL-CDE framework can be applied to most CDE variants. As examples, in this study, the framework is applied to two popular representative CDE variants, i.e., rank-iMDDE and εDEag. Experiment results on 24 benchmark functions from CEC'2006 and 18 benchmark functions from CEC'2010 show that the proposed framework is an effective approach to enhance the performance of CDE algorithms.

Communicated by A. Di Nola.

✉ Wenhong Wei
weiwh@dgut.edu.cn

[1] School of Computer, Dongguan University of Technology, Dongguan 523808, China

[2] National ICT Australia, Eveleigh, NSW 2015, Australia

Keywords Constrained optimization · Differential evolution · Machine learning · General opposition-based learning

1 Introduction

Real-world optimization problems are often complex and involve equality and inequality constraints. The optimization problems with some constraints are named as constrained optimization problems (COPs) (Goldberg and Samtani 1986). There exist many studies on solving COPs using evolutionary algorithms (EAs) (Michalewicz 1995; Coello 2002). Differential evolution (DE), which was first proposed by Storn and Price (1997), is one of the most powerful EAs for global numerical optimization. DE, coupled with constraint handling techniques, has been used to solve COPs and has been dubbed constrained differential evolution (CDE).

Machine learning is concerned with the development of computer algorithms and techniques that are able to learn, i.e., to improve automatically through experience (Alpaydin 2004). The concept of opposition-based learning (OBL) introduced by Tizhoosh (2005) is a new machine intelligence algorithm inspired by the phenomena of sudden and radial changes in social revolution. Many machine intelligence algorithms consider finding the solution of a given problem as function approximation, so OBL attempts to approximate the optimal solution from two opposite sides. OBL has thus far been applied to many EAs including DE and PSO (Al-Qunaieer et al. 2010; Xu et al. 2014; Wang 2012). The basic idea behind OBL is that whenever we seek the solution in a direction, it is beneficial to consider the opposite direction as well.

The GOBL is a generalized OBL introduced in Wang et al. (2009). The main idea behind GOBL is to transform

Published online: 11 January 2016

solutions in the current search space to a new search space. By simultaneously considering the solutions in the current search space and the transformed search space, GOBL can provide a higher chance of finding solutions which are closer to the global optimum.

As we well know, CDE has obtained good performance on many COPs. However, it may easily become trapped at local optimization. To enhance the performance of CDE on complex problems, this paper presents a novel CDE framework called GOBL-CDE using GOBL. The GOBL-CDE uses similar procedure of opposition-based differential evolution DE (ODE) when also using transformed population initialization and dynamic generation jumping with GOBL (Rahnamayan et al. 2008).

The major advantages of the GOBL-CDE framework are its simplicity and generality, which makes it easily combine with most of the CDE variants. To evaluate the effectiveness of GOBL-CDE, the proposed framework is applied to two representative CDE variants, rank-iMDDE (ranking-based mutation operator and improved dynamic diversity mechanism DE) (Gong et al. 2014) and εDEag (ε constrained DE with an archive and gradient-based mutation) (Takahama and Sakai 2009). We chose 24 benchmark functions presented in CEC'2006 (Liang et al. 2006) and 18 benchmark functions presented in CEC'2010 (Mallipeddi and Suganthan 2010) as the test suite. Experimental results verify our expectation that the GOBL-CDE framework is able to accelerate these CDE variants in the majority of test cases, and GOBL-CDE is also able to enhance the performance for CDE variants in solving constrained optimization problems.

The main contributions of this study are as follows:

1. In CDE, the GOBL is employed to compare the fitness of an individual to its transformed individual and retain the fitter one in the population to accelerate the CDE without making premature convergence.
2. The proposed approach is simple, because it keeps the simple structure of the CDE algorithm, and no additional parameter is introduced. The proposed approach can be easily applied to other advanced CDE variants (i.e., rank-iMDDE, εDEag) and thus cooperate with different kinds of modifications in advanced DE variants to further improve the performance. This provides a promising approach for solving benchmarks.

The rest of this paper is organized as follows. We firstly introduce the previous work in Sect. 2. The proposed framework (GOBL-CDE) is then presented in detail in Sect. 3. Section 4 reports experimental results. Finally, conclusions are drawn in Sect. 5.

2 Related work

In this section, we first introduce the COPs used in this work. Then, the original DE algorithm and GOBL are described.

2.1 Constrained optimization problems

Constrained optimization problems can be divided into many different categories based on their characteristics and mathematical properties. They may contain different types of variables, such as real, integer and discrete, and may have equality and/or inequality constraints. The objective and constraint functions could be either linear or nonlinear. The functions may be either continuous or discontinuous, and either unimodal or multimodal. The feasible region of such problems could be either a tiny or a significant portion of the search space. Again, the feasible region could be either one single bounded region or a collection of multiple disjointed regions. The definition of COPs is as follows:

$$\text{minimize } f(x) \tag{1}$$

subject to

$$g_j(x) \le 0, \quad j = 1, \ldots, q \tag{2}$$

$$h_j(x) = 0, \quad j = q+1, \ldots, m \tag{3}$$

$$\text{low}_i \le x_i \le \text{up}_i, \quad i = 1, \ldots, n \tag{4}$$

where $x = (x_1, x_2, \ldots, x_n)$ is an n dimensional vector, $f(x)$ is an objective function, $g_j(x) \le 0$ and $h_j(x) = 0$ are q inequality constraints and m–q equality constraints, respectively. Functions f, g_j and h_j are linear or nonlinear real-valued functions. Values up_i and low_i are the upper bound and the lower bound of x_i, respectively. Also, let the feasible space in which every point satisfies all constraints be denoted by U and the search space in which every point satisfies the upper and lower bound constraints be denoted by S ($\subset U$).

In evolutionary constrained optimization, equality constraints are always converted into inequality constraints

$$|h_j(x)| - \delta \le 0 \tag{5}$$

where $j \in \{q+1, \ldots, m\}$ and δ is a positive tolerance value. The distance of a solution x from the jth constraint can be constructed as:

$$G_j(x) = \begin{cases} \max\{0, g_j(x)\}, & 1 \le j \le q \\ \max\{0, |h_j(x)| - \delta\}, & q+1 \le j \le m \end{cases} \tag{6}$$

Then, the distance of the solution x from the boundaries of the feasible set, which also reflects the degree of its constraint violation, can be denoted as:

$$G(x) = \sum_{j=1}^{m} G_j(x) \tag{7}$$

2.2 Differential evolution

DE is a population-based and directed search method. Like other EAs, it starts with an initial population vector, which is randomly generated when no preliminary knowledge about the solution space is available. The initial population has NP candidate individuals (solutions) with n-dimensional parameter vectors; each individual is denoted as $x_{i,t} = (x_{1i,t}, x_{2i,t}, \ldots, x_{ni,t})$, where $(i = 1, 2, \ldots, NP)$ NP is the population size, and t is the current generation. It has three main operators: mutation, crossover and selection. The main procedure of DE is described in the following subsections.

Mutation This operator generates a mutant vector $v_{i,t}$ with respect to each individual $v_{i,t}$ (named a target vector). The originally proposed and most frequently referred to mutation strategies in the literature are Storn and Price (1997) and Price et al. (2005):

$$\text{rand}/1: \; v_{i,t} = x_{r1,t} + F \cdot (x_{r2,t} - x_{r3,t}) \tag{8}$$

$$\text{best}/1: \; v_{i,t} = x_{\text{best},t} + F \cdot (x_{r1,t} - x_{r2,t}) \tag{9}$$

$$\text{current-to-best}/1: \; v_{i,t} = x_{i,t} + F \cdot (x_{\text{best},t} - x_{i,t}) + F \cdot (x_{r1,t} - x_{r2,t}) \tag{10}$$

$$\text{best}/2: \; v_{i,t} = x_{\text{best},t} + F \cdot (x_{r1,t} - x_{r2,t}) + F \cdot (x_{r3,t} - x_{r4,t}) \tag{11}$$

$$\text{rand}/2: \; v_{i,t} = x_{r\_1,t} + F \cdot (x_{r2,t} - x_{r3,t}) + F \cdot (x_{r4,t} - x_{r5,t}) \tag{12}$$

where indices $r1, r2, r3, r4$, and $r5$ are distinct integers randomly selected from the set $\{1, 2, \ldots, \text{NP}\}\backslash\{i\}$ and $x_{\text{best},t}$ is the best individual in the population at generation t. The scaling factor F is a real number and is usually chosen between 0 and 1.

Crossover After mutation, a binomial crossover operation is applied to create the trial vector $u_{i,t}$ as follows:

$$u_{ji,t} = \begin{cases} v_{ji,t}, & \text{if } \text{rand}_j(0,1) \le \text{CR} \text{ or } j = j_{\text{rand}} \\ x_{ji,t}, & \text{otherwise} \end{cases} \tag{13}$$

where $i = 1, 2, \ldots, \text{NP}$; $j = 1, 2, \ldots, n$; j_{rand} is an integer randomly chosen from 1 to n; and $\text{rand}_j(0, 1)$ is a uniformly distributed random number on the interval [0,1] which is newly generated for each j. The parameter j_{rand} is adopted to ensure that the trial vector $u_{i,t}$ differs from its target vector $x_{i,t}$ by at least one parameter. The crossover control parameter CR usually ranges between 0 and 1.

Selection In the selection operation, the target vector $x_{i,t}$ is compared against the trial vector $u_{i,t}$ in terms of the objective function value, and the better one will be added to the next population:

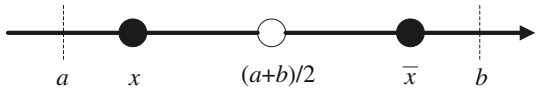


Fig. 1 An example for opposite number

$$x_{i,t+1} = \begin{cases} u_{i,t}, & \text{if } f(u_{i,t}) \le f(x_{i,t}) \\ x_{i,t}, & \text{otherwise} \end{cases} \tag{14}$$

2.3 Opposition-based learning

Before concentrating on OBL, we need to define the concept of opposite numbers. An opposition-based number can be defined as follows (Tizhoosh 2005).

Definition 1 Let $x \in [a, b]$ be a real number. The opposite number $\overline{x}$ is defined by

$$\overline{x} = a + b - x, \quad \overline{x} \in [a, b] \tag{15}$$

Figure 1 shows an example for opposite number.

Similarly, the opposite point in n-dimensional space can be defined as follows.

Definition 2 Let $P = (x_1, x_2, \ldots, x_n)$ be a point in n-dimensional space, where $x_1, x_2, \ldots, x_n \in R$ and $x_j \in [a_j, b_j]$, $\forall j \in [1, 2, \ldots, n]$. The opposite point $\overline{P} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$ is completely defined by its coordinates

$$\overline{x_j} = a_j + b_j - x_j \tag{16}$$

Now, by employing the definition of opposite point, the opposition-based optimization can be defined as follows.

Definition 3 Let $P = (x_1, x_2, \ldots, x_n)$ be a point in n-dimensional space (i.e., candidate solution), assume $f(x)$ is a fitness function which is used to measure the candidate's fitness. According to the definition of the opposite point, $\overline{P} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$ is the opposite of P. Now if $f(\overline{P}) < f(P)$, i.e., $\overline{P}$ has a better fitness than P, then point P can be replaced with $\overline{P}$; otherwise, we continue with P. Hence, the point and its opposite point are evaluated simultaneously to continue with the fitter one.

2.4 Generalized opposition-based learning

In Wang et al. (2009), a generalized OBL, called GOBL, is proposed by transforming candidates in current search space to a new search space. By simultaneously evaluating the candidates in the current search space and transformed search space, it can provide more chance of finding candidate solutions closer to the global optimum.

Definition 4 Let $x \in [a, b]$ be a solution in the current search space S. The transformed solution $\overline{x}$ in the transformed space $\overline{S}$ is defined by

$$\overline{x} = k(a + b) - x \tag{17}$$

where k is a random number in [0, 1], and $\overline{x} \in [k(a + b) - b, k(a + b) - a]$.

For a given problem, it is possible that $\overline{x}$ may jump out of the box-constraint $[a, b]$. When this happens, the GOBL will be invalid, because the transformed candidate is infeasible. To avoid this case, the transformed candidate is assigned to a random value as follows:

$$\overline{x} = \text{rand}(a, b), \quad \text{if} \quad \overline{x} < a || \overline{x} > b \tag{18}$$

where rand(a, b) is a random number in $[a, b]$, and $[a, b]$ is the interval boundaries of current population.

Definition 5 Let $P = (x_1, x_2, \ldots, x_n)$ be a point in n-dimensional search space, where $x_1, x_2, \ldots, x_n \in R$ and $x_j \in [a_j, b_j]$, $\forall j \in [1, 2, \ldots, n]$. The point $\overline{P} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$ in transformed space is completely defined by its coordinates

$$\overline{x_j} = k(a_j + b_j) - x_j \tag{19}$$

where k = rand(0, 1).

$$\overline{x} = \text{rand}(a_j, b_j), \quad \text{if } \overline{x} < a_j || \overline{x} > b_j \tag{20}$$

with regard to GOBL-based optimization, the method in the Definition 3 is adopted to select best fitness candidates in P and $\overline{P}$.

2.5 Constrained differential evolution

Differential evolutions combined with constraint handling techniques have been proposed and successfully used for solving COPs (Wang et al. 2009). Storn (1999) proposed constraint adaptation with DE (CADE) to solve COPs. In CADE, all the constraints of the problems are relaxed at the beginning so that all the individuals in the initial population are feasible and the constraints are gradually tightened until they arrive at the original constraints. Takahama and Sakai (2006) proposed a method called εDE, which applies the ε constrained method to DE. And then they proposed an improved εDE in which faster reduction of the relaxation of equality constraints and a higher rate of gradient-based mutation are adopted (Takahama and Sakai 2009). Brest (2009) proposed an ε-self-adaptive DE which employs the ε constrained method similar to Takahama and Sakai (2006) to handle constraints. In this method, several mutation strategies are used and a self-adaptive mechanism is applied to the control parameters F and CR. Note that this method reduces the population size during the evolution. Mezura-Montes et al. (2005, 2007) proposed a multi-member diversity based DE (MDDE) for the COPs. Gong et al. (2014) proposed a rank-iMDDE which is an improved constrained differential evolution. In rank-iMDDE, the ranking-based mutation operator and improved dynamic diversity mechanism are presented to maintain either infeasible or feasible solutions in the population. Wang and Cai (2012) presented a dynamic hybrid framework which is called DyHF for solving constrained optimization problems. This framework consists of two major steps: global search model and local search model. Elsayed et al. (2011) proposed SAMODE which is a multiple search operator based DE, where different operators are selected adaptively. ISAMODE-CMA proposed by Hansen et al. (2003) is an improved version of SAMODE. In ISAMODE-CMA, both mixed mutation operators and CMA-ES-based local search are implemented. Tasgetiren et al. (2015) proposed a differential evolution algorithm based on a variable neighborhood search algorithm (DE_VNS) to solve the constrained real-parameter optimization problems. Guo et al. (2015) proposed theorems to prove that a min–max algorithm can be used to solve a max–min problem without any algorithmic changes, and based on the theorems, a constraint-activated differential evolution to solve constrained min–max problems was proposed. Wang et al. (2015) presented a new method to address the issue of how to balance constraints and objective function. In their method, after generating an offspring for each parent in the population by making use of differential evolution, the well-known feasibility rule is used to compare the offspring and its parent. Gong et al. (2015) proposed an adaptive ranking mutation operator (ARMOR) for DE when solving the COPs. Gao et al. (2015) proposed a dual-population differential evolution (DPDE) with coevolution for constrained optimization problems. Wang and Cai (2011) proposed a $(\mu + \lambda)$-CDE for the COPs. In $(\mu + \lambda)$-CDE, three different DE mutation strategies are used to generate three offspring for each target parent; additionally, the IATM is proposed to handle constraints. To overcome the main drawbacks of $(\mu+\lambda)$-CDE, Jia et al. (2013) proposed an improved version of $(\mu + \lambda)$-CDE which consists of an improved $(\mu + \lambda)$-differential evolution (IDE) and a novel archiving-based adaptive tradeoff model (ArATM). Mallipeddi and Suganthan (2010) proposed an ensemble of constraint handling techniques (ECHT) to solve COPs. In this method, each constraint handling technique has its own population. As an instantiation of this method, DE is combined with four constraint handling techniques. Wang and Cai (2012) proposed an improved version of the CW method, called CMODE, which combines multiobjective optimization with differential evolution to deal with constrained optimization problems. Liu et al. (2010) proposed a novel hybrid

algorithm named PSO-DE, which integrates particle swarm optimization (PSO) with differential evolution (DE) to solve constrained numerical and engineering optimization problems.

2.6 OBL-based and GOBL-based optimization

Opposition-based learning-based optimization uses the concept of OBL to compare the fitness of an individual to its opposite and retaining the fitter one in the population to accelerate the EAs. The OBL was initially applied to accelerate reinforcement learning (Tizhoosh 2005, 2006; Ahandani and Alavi-Rad 2012) and back propagation learning in neural networks (Ventresca and Tizhoosh 2006). Rahnamayan et al. (2006) has proved that opposite numbers are more likely to be closer to the optimal solution than a purely random one. The OBL was recently applied to different EAs, especially DE. Rahnamayan et al. (2008) for the first time proposed opposition-based DE (ODE) employed the OBL for population initialization and also for generation jumping. Subudhi and Jena (2009) presented a new DE approach based on OBL applied for training neural network used for non-linear system identification. Balamurugan and Subramanian (2009) presented an opposition-based self-adaptive DE; in the algorithm, a multi-objective function was formulated by assigning the relative weight to each of the objectives and then optimized by opposition-based self adaptive DE. Bošković et al. (2011) presented a DE with opposition-based optimization approach to chess evaluation function tuning. The approach was employed and upgraded with a history mechanism to improve the evaluation of individuals and the tuning process.

There are many varieties of opposition-based learning. Rahnamayan et al. (2008) enhanced the previous methods by replacing opposite points with quasi-opposite points. The authors proved that a quasi-opposite point has a higher probability of being closer to the solution than an opposite point. Xu et al. (2011) proposed opposition-based learning using the current optimum (COOBL), and COOBL combined with differential evolution for function optimization. Wang et al. (2009) generalized OBL using search space and transformed search space, and then they proposed GODE based on the random GOBL model (Wang et al. 2009). The GODE uses the GOBL method to initialize population and produces new candidates in evolutionary generations. The original DE is chosen as a parent algorithm and the proposed GOBL model is embedded in DE to improve its performance. With regard to CDE using OBL, Omran and Salman (2009) proposed a new population based meta-heuristic optimization algorithm called CODEQ to solve COPs. CODEQ is a combination of concepts from chaotic search, opposition-based learning, DE, and quantum mechanics. However, the performance of CODEQ was tested only in five constrained problems which are selected elaborately; it is obvious that CODEQ cannot get better results in other constrained problems. So this study merges generalized opposition-based learning technique with constraint handling techniques and proposes GOBL-CDE to solve COPs. Similar to GODE, GOBL-CDE uses the GOBL to initialize population and jump generation. In addition, the performance of GOBL-CDE was tested in 24 benchmark functions in CEC'2006 and 18 benchmark functions in CEC'2010.

Some differences between the GODE and the GOBL-CDE are as follows:

1. The GODE is an algorithm; however, the GOBL-CDE is a framework. The GOBL-CDE framework can be applied to most of the CDE variants.
2. The GODE only can solve unconstrained optimization problems. However, the GOBL-CDE combined with GOBL, transformed fitness measures technique and ranking techniques, can solve constrained optimization problems.
3. In the GODE, given a probability p_0, if rand$(0, 1) \leq p_0$, then execute the GOBL; otherwise, execute the classical DE. That is, in the GODE, GOBL and classical DE are executed alternately. But in the GOBL-CDE, the generalized opposition occurs with a probability, and the DE combined with GOBL and constraint handing techniques is executed in every generation to improve the exploitation performance of DE.

3 GOBL-CDE framework

3.1 Motivations

In a given population, purely random selection of solutions has the chance of visiting or even revisiting unproductive regions of the search space. It is well known that the chance of this occurring is lower for opposite numbers than it is for purely random ones. In fact, a mathematical proof has been proposed to show that, in general, opposite numbers are more likely to be closer to the optimal solution than purely random ones. The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e., guess and opposite guess) to achieve a better approximation for the current candidate solution.

The OBL can be used in two stages of DE: firstly, in the initialization stage to achieve fitter starting candidate solutions, while a priori knowledge about the initial members does not exist; secondly, carrying out the DE to force the current population to jump into some new candidate solutions, which ideally are fitter than the current ones. As we well known, the OBL has been proved to be efficient for DE, especially

it can accelerate the convergence rate of DE (Rahnamayan et al. 2008).

The GOBL, a generalized OBL, also includes two stages. These two stages are called generalized opposition-based population initialization and generalized opposition-based generation jumping, respectively. Similar to OBL, combining with the constraint handling techniques, GOBL is also able to be used in the two stages of CDE. Therefore, based on the above motivations, in this study, we propose a CDE framework merging GOBL with transformed fitness measures technique and ranking techniques to solve COPs to enhance the performance for CDE variants.

3.2 GOBL-based population initialization

The algorithm of utilizing GOBL for population initialization is presented as the following.

Algorithm 1: GOBL_population_initialization

Initialize the population P_0 with a size of NP randomly;
for ($i = 0$; $i < NP$; i ++)
  for ($j = 0$; $j < n$; j ++)
    $GOP_{ji,0} = k * (a_j + b_j) - P_{ji,0}$; // k is rand(0, 1).
    if $GOP_{ji,0} < a_j \,||\, GOP_{ji,0} > b_j$
      $GOP_{ji,0} = rand(a_j, b_j)$
    endif
  endfor
endfor
Select NP the fittest individuals from $\{P_0 \cup GOP_0\}$ as the initial population P_0;
return(P_0);

Using GOBL in population initialization, the fitter starting candidate solutions can be obtained. Besides the population initialization, the transformed population GOP is calculated and the NP fittest individuals are selected from the union of the current population P and GOP.

3.3 GOBL-based generation jumping

In this stage, based on a jumping rate J_r (i.e., jumping probability), after generating new population by CDE operators, the transformed population is calculated and the NP fittest individuals are selected from the union of the current population and the transformed population. Unlike GOBL-based initialization, generation jumping calculates the opposite population dynamically. Instead of using variables' predefined interval boundaries ($[a_j, b_j]$), generation jumping calculates the transformation of each variable based on the minimum (MIN_j) and maximum (MAX_j) values of that variable in the current population. The algorithm of the GOBL-based generation jumping is described as the following.

Algorithm 2: GOBL_generation_jumping (P_t)

if (rand(0, 1) < J_r) //J_r is a jumping rate.
  for ($i = 0$; $i < NP$; i ++)
    for ($j = 0$; $j < n$; j ++)
      $GOP_{ji,t} = k * (MIN(P_{ji,t}) + MAX(P_{ji,t})) - P_{ji,t}$; // k is rand(0, 1).
      if $GOP_{ji,t} < MIN(P_{ji,t}) \,||\, GOP_{ji,t} > MAX(P_{ji,t})$
        $GOP_{ji,t} = rand(MIN(P_{ji,t}), MIN(P_{ji,t}))$
      endif
    endfor
  endfor
end
Select NP the fittest individuals from $\{P_t \cup GOP_t\}$ as the new population P_t;
return(P_t);

By staying within variables' interval static boundaries, GOBL can jump outside of the already shrunken search space and the knowledge of the current reduced space (converged population) would be lost. Hence, the transformed points are calculated using variables' current interval in the population ($[\mathrm{MIN}_j, \mathrm{MAX}_j]$), which is, as the search progresses, increasingly smaller than the corresponding initial range $[a_j, b_j]$.

3.4 Transformed fitness measures

Generally, the fitness value is equal to objective function value in unconstrained DE. However, in CDE, the fitness value is not equal to objective function value simply. In Gong et al. (2014), the adaptive fitness transformation (AFT) is adopted to classify populations into three situations to calculate the final transformed fitness value of each individual, that is, infeasible, semi-feasible, and feasible situations.

(1) Infeasible situation In the infeasible situation, the population contains only infeasible solutions. The main task of the optimization technique is to find the feasible solutions. Therefore, in this situation, the fitness value is equal to the constraint violation (e.g., $G(x)$ in Eq. (7)) of each individual. The objective function values are not considered at all.

$$f_{\text{fitness}}(x_i) = G(x_i) \tag{21}$$

(2) Semi-feasible situation As suggested in Wang and Cai (2011), in the semi-feasible situation, it is expected that some important feasible individuals (feasible individuals with small objective function values) and infeasible individ-

uals (infeasible individuals with slight constraint violations and small objective function values) should remain in the next population since such individuals are very promising for searching for the optimal solution. Therefore, to balance the influence of the objective function value and constraint violation, fitness transformation techniques could be a good choice.

In the semi-feasible situation, the population is divided into feasible group (Z_1) and the infeasible group (Z_2) based on the feasibility of each solution. Thereafter, the objective function value $f(x_i)$ of the solution x_i is converted into

$$f'(x_i) = \begin{cases} f(x_i), & i \in Z_1 \\ \max\{\varphi \cdot f(x_{\text{best}}) \\ +(1-\varphi) \cdot f(x_{\text{worst}}), f(x_i)\}, & i \in Z_2 \end{cases} \tag{22}$$

where φ is the feasibility ratio of the last population, and x_{best} and x_{worst} are the best and worst solutions in the feasible group Z_1, respectively. After obtaining the converted objective function value of each solution, it is then normalized as:

$$f_{\text{nor}}(x_i) = \frac{f'(x_i) - \min_{j \in Z_1 \cup Z_2} f'(x_j)}{\max_{j \in Z_1 \cup Z_2} f'(x_j) - \min_{j \in Z_1 \cup Z_2} f'(x_j)} \tag{23}$$

If we use Eq. (7) to calculate the constraint violation of each solution, the normalized constraint violation can be evaluated as:

$$G_{\text{nor}}(x_i) = \begin{cases} 0, & i \in Z_1 \\ \frac{G(x_i) - \min_{j \in Z_2} G(x_j)}{\max_{j \in Z_2} G(x_j) - \min_{j \in Z_2} G(x_j)}, & i \in Z_2 \end{cases} \tag{24}$$

Finally, the final fitness function is obtained as follows:

$$f_{\text{fitness}}(x_i) = f_{\text{nor}}(x_i) + G_{\text{nor}}(x_i) \tag{25}$$

(3) Feasible situation In this situation, all individuals in the population are feasible, and the COPs can be viewed as unconstrained optimization problems. Thus, the fitness value is equal to the objective function value $f(x_i)$ of each individual.

$$f_{\text{fitness}}(x_i) = f(x_i) \tag{26}$$

In GOBL-CDE, we adopt AFT techniques to select NP fittest individuals from the population and the transformed population. The algorithm of selecting NP fittest individuals is given as the following.

Algorithm 3: Select_Fittest_Individual (P, GOP)

Compute the objective function value and the degree of constraint violation for each individual in the population P;

$num\_1 = |\, \text{G\_nor}(x_i) = 0 \,|$ // num_1 denotes that the number of feasible solutions are found in the population P, and $0 \le num\_1 \le NP$, $i = 1, 2, ..., NP$.

$\varphi_1 = \frac{num\_1}{NP}$;

Determinate the current situation of the population P according to the value of φ_1 and compute Fit_P using AFT;

Compute the objective function value and the degree of constraint violation for each individual in the population GOP;

$num\_2 = |\, \text{G\_nor}(x_i) = 0 \,|$ // num_2 denotes that the number of feasible solutions are found in the population GOP, and $0 \le num\_2 \le NP$, $i = 1, 2, ..., NP$.

$\varphi_2 = \frac{num\_2}{NP}$;

Determinate the current situation of the population GOP according to the value of φ_2 and compute Fit_{GOP} using AFT;

if ($Fit_{GOP} < Fit_P$)

  $Q = GOP$;

else

  $Q = P$;

end

return Q;

In the Algorithm 3, the function Select_Fittest_Individual (P, GOP) is used to select NP fittest individuals from P and GOP. In addition, $\text{G\_nor}(x_i) = 0$ denotes that it finds feasible solutions in the population, $i = 1, 2, \ldots, \text{NP}$. φ_1 and φ_2 are the feasibility ratio of the population P and GOP, respectively, $0 \le \varphi_1, \varphi_2 \le 1$. For example, while $\varphi_1 = 1$, the population P is feasible situation. While $\varphi_1 = 0$, the population P is infeasible situation, and while $0 < \varphi_1 < 1$, the population P is semi-feasible situation.

3.5 Ranking techniques

To utilize the information of good vectors in the CDE population, in this study, we assign a ranking for each vector according to its fitness. First, the population is sorted in ascending order (i.e., from the best to the worst) based on the fitness of each vector. Then, the ranking of a vector is assigned. Thus, the ranking of ith individual (vector) in the sorted population is assigned as follows:

$$R_i = \text{NP} + 1 - i, \quad i = 1, 2, \ldots, \text{NP} \tag{27}$$

where NP is the population size, and i is the index of the ith individual in the sorted population. According to Eq. (27), better individual in the sorted population will obtain better ranking.

After obtaining the ranking of each individual, we then calculate the selection probability p_i for each individual x_i. The selection probability is calculated according to the situation of the current population for the COPs. In different situations, different methods are used to calculate the selection

probability. For example, the probabilities can be calculated as follows:

In the infeasible situation:

$$p_i = 0.5 \cdot \left(1 - \cos\left(\frac{R_i}{\mathrm{NP}} \cdot \pi\right)\right) \tag{28}$$

In the semi-feasible situation:

$$p_i = 0.5 \cdot \left(1 - \cos\left(\frac{R_i}{\mathrm{NP}} \cdot \pi\right)\right) \tag{29}$$

In the feasible situation:

$$p_i = \frac{\arccos\left(1 - 2 \cdot \frac{R_i}{\mathrm{NP}}\right)}{\pi} \tag{30}$$

where $i = 1, 2, \ldots, \mathrm{NP}$.

The ranking techniques were integrated into the mutation operator in some DE algorithms, and experimental results indicated that the mutation operator using ranking techniques is able to enhance the performance of the DE algorithms. The algorithm of the mutation operator using ranking techniques is described as the following.

Algorithm 4: Ranking-based Mutation Operation (P) // DE/rand/1

Sort P according to the fitness;
Calculate the probability p_i for each individual based on Eq.(28); // $i = 1, 2, \ldots, NP$.
Randomly select $r1 \in \{1, NP\}$; // base vector index
while rand $> p_{r1}$ or $r1 = i$
    Randomly select $r2 \in \{1, 2, ..., NP\}$;
end
Randomly select $r2 \in \{1, 2, ..., NP\}$; // terminal vector index
while rand $> p_{r2}$ or $r2 = r1$ or $r2 = i$
    Randomly select $r2 \in \{1, 2, ..., NP\}$;
end
Randomly select $r3 \in \{1, 2, ..., NP\}$; // starting vector index
while $r3 = r2$ or $r2 = r1$ or $r3 = i$
    Randomly select $r3 \in \{1, 2, ..., NP\}$;
end

In the mutation operation of GOBL, the parents are proportionally selected according to their rankings in the current population. The higher ranking a parent obtains, the more opportunity it will be selected.

3.6 GOBL-CDE algorithm

In this section, we propose a framework of the GOBL-CDE algorithm, which utilize the GOBL techniques, i.e., transformed population initialization as well as transformed generation jumping, AFT techniques and ranking techniques. In GOBL-CDE, we adopt AFT techniques to select NP fittest individuals from the population and the transformed population. In addition, the new population, which is made from the NP fittest individuals, is sorted using ranking techniques to utilize the information of good vectors in mutation operation.

At last, the GOBL-CDE framework is described in Algorithm 5. From the Algorithm 5, it can be seen that if we invoke Algorithm 1, Algorithm 2 and Algorithm 4 corresponding to their position in other CDE variants, the GOBL-CDE will be applied to those CDE variants simply.

Algorithm 5: Pseudo code of GOBL-CDE framework

t=0; // t denotes the generation number.
Generate $P_0 = (x_1, ..., x_{NP})$;
Invoke Algorithm 1;
Compute the objective function value and the degree of constraint violation for each individual in the population P_0;
repeat
    $t = t + 1$;
    $P_t = P_{t-1}$;
    for each individual in the population P_t do
        Invoke Algorithm 4 generate trial vectors;
        Apply crossover and selection operators to get P_t';
    end
    $P_t = P_t'$
    Invoke Algorithm 2;
    Compute the objective function value and the degree of constraint violation for each individual in the population P_t;
until the stopping criterion is met;
output: the best individual of the population P_t

Complexity analysis Compared with the original DE, the GOBL-CDE has an additional computational burden in the calculation of generalized opposition population and population sorting. During one generation, the computation complexity of calculation of generalized opposition population is $O(G_{\max} \cdot \mathrm{NP} \cdot n)$, where $G_{\max}$ is the maximal number of generation. In addition, the complexity of the population sorting is $O(G_{\max} \cdot \mathrm{NP} \cdot \log(\mathrm{NP}))$. Since the complexity of the original DE algorithm is also $O(G_{\max} \cdot \mathrm{NP} \cdot n)$, the overall complexity of the GOBL-CDE is $O(2 \cdot G_{\max} \cdot \mathrm{NP} \cdot n + G_{\max} \cdot \mathrm{NP} \cdot \log(\mathrm{NP}))$, that is, $O(G_{\max} \cdot \mathrm{NP} \cdot (2n + \log(\mathrm{NP})) = O(G_{\max} \cdot \mathrm{NP} \cdot n)$. Hence, the GOBL-CDE does not significantly increase the overall complexity compared with the original DE.

The major advantages of the GOBL-CDE framework are its simplicity and generality, which make it easily combine with most of the CDE variants. In addition, the GOBL-CDE framework does not improve the complexity of the algorithms when it is applied into other CDE variants.

4 Experimental results and analysis

In this section, comprehensive experiments are conducted to evaluate the performance of the GOBL-CDE framework. Firstly, the GOBL-CDE framework is applied to rank-iMDDE as GOBL-rank-iMDDE, and GOBL-rank-iMDDE is used to optimize the functions presented in CEC'2006 (Liang et al. 2006) and CEC'2010 (Mallipeddi and Suganthan 2010) to verify the enhanced performance of the GOBL-CDE framework. Secondly, the GOBL-CDE framework is also applied to εDEag as GOBL-εDEag; GOBL-εDEag is tested on 24 benchmark functions from CEC'2006 and 18 benchmark functions from CEC'2010 to verify the enhanced performance of the GOBL-CDE framework. Finally, to further evaluate the effectiveness of the GOBL-CDE framework, GOBL-rank-iMDDE and GOBL-εDEag are compared with other state-of-the-art EAs for the COPs and to achieve enhanced performance, the GOBL-CDE framework is also applied to other CDE algorithms to improve those performance.

4.1 Experimental setup

Note that the proposed GOBL-CDE versions do not introduce any additional parameters; thus, the parameters in GOBL-rank-iMDDE are set to be the same as rank-iMDDE (Gong et al. 2014) for fair comparison; the details are given as follows:

1. population size: NP = 70 (Gong et al. 2014);
2. crossover rate: CR = 0.9 (Gong et al. 2014);
3. scaling factor: F = rand(0.3, 0.9) (Gong et al. 2014);
4. number of offspring generated by each target individual: $n_0 = 5$ (Gong et al. 2014);
5. tolerance of equality: δ = 1e−4 (Gong et al. 2014);
6. initial value of the selection ratio: $S_{r_0} = 0.7$ (Gong et al. 2014);
7. two coefficients in the ranking-based mutation: $k_1 = 2.0$ and $k_2 = 0.50$ (Gong et al. 2014).

For the same reason, the parameters in GOBL-εDEag are set to be the same as εDEag (Takahama and Sakai 2009).

1. population size: NP = $4n$ (Takahama and Sakai 2009);
2. crossover rate: CR = 0.9 (Takahama and Sakai 2009);
3. scaling factor: $F = 0.5$ (Takahama and Sakai 2009);
4. others are archive size: $M = 100n$ (Takahama and Sakai 2009);
5. control generations: $T_c = 1000$ (Takahama and Sakai 2009);
6. initial ε level: $\theta = 0.9$ (Takahama and Sakai 2009);
7. gradient-based mutation rate: $P_g = 0.2$ (Takahama and Sakai 2009);
8. number of repeating the mutation: $R_g = 3$ (Takahama and Sakai 2009);

According to "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization" (Liang et al. 2006), the two approaches are performed on 50 independent runs for each CEC'2006 benchmark function with the Max_FEs = 5×10^5. According to "Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization" (Mallipeddi and Suganthan 2010), each variant has been run 50 times for each CEC'2010 test problem, where the stopping criteria are to run up to 2×10^5 fitness evaluations (FEs) for 10D instances, and 6×10^5 FEs for 30D, respectively.

The following three evaluation criteria are adopted to measure the final performance of each algorithm.

1. NFEs: It is used to record the number of function evaluations in each run for finding a solution satisfying $f(x) - f(x^*) \leq$ 1e−4 and x is feasible, where x^* is the known-optimal solution of a specific problem.
2. Convergence graphs: The convergence graphs for some typical functions are plotted to illustrate the mean error performance of the best solution over the total run in the respective experiments.
3. Statistics by Wilcoxon and Friedman tests (Derrac et al. 2011; Alcal-Fdez et al. 2008; Corder and Foreman 2009): to identify differences between pairs of algorithms on all problems, the multi-problem Wilcoxon signed-rank test is carried out. The Friedman test is used to obtain the rankings of multiple algorithms (more than two algorithms) on all problems.

4.2 Comparison of GOBL-rank-iMDDE with rank-iMDDE

With respect to the benchmark functions in CEC'2006, the results of the final solutions of GOBL-rank-iMDDE and rank-iMDDE are shown in Table 1, where the boldface means that the algorithm obtains the best known solution or better result in the specific function. In addition, several representative convergence graphs of the selected functions are given in Fig. 2. From the results shown in Table 1, it can be seen as the following. Firstly, GOBL-rank-iMDDE consistently obtains the best known optimal solutions over all 50 runs in 24 functions. Secondly, rank-iMDDE only obtains the best known optimal solutions in 22 functions, and the two functions (g02 and g22) are not the best known optimal.

The NFEs in each run are recorded in Table 2. It can be seen from Table 2 that GOBL-rank-iMDDE gets fewer NFEs than rank-iMDDE in 17 out of 24 functions, and only in

Table 1 Function values achieved by GOBL-rank-iMDDE and rank-iMDDE for all benchmark functions of CEC'2006

Prob	GOBL-rank-iMDDE					Rank-iMDDE				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
g01	**−15**	**−15**	**−15**	**−15**	0.0000E+00	**−15**	**−15**	**−15**	**−15**	0.0000E+00
g02	**−0.784939221**	**−0.80361905**	**−0.80361902**	**−0.80361219**	1.30598E−02	−0.771749392	**−0.80361905**	−0.80361882	−0.80201189	1.11335E−02
g03	**−1.0005001**	**−1.0005001**	**−1.0005001**	**−1.0005001**	0.00000E+00	**−1.0005001**	**−1.0005001**	**−1.0005001**	**−1.0005001**	0.0000E+00
g04	**−30,665.53867**	**−30,665.5387**	**−30,665.53867**	**−30,665.53867**	7.31261E−12	**−30,665.5387**	**−30,665.5387**	**−30,665.5387**	**−30,665.5387**	7.31261E−12
g05	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	7.31261E−12	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	7.31261E−12
g06	**−6961.81388**	**−6961.81388**	**−6961.81388**	**−6961.81388**	1.37111E−11	**−6961.81388**	**−6961.81388**	**−6961.81388**	**−6961.81388**	1.37111E−11
g07	**24.3062091**	**24.3062091**	**24.3062091**	**24.3062091**	1.85838E−07	**24.3062091**	**24.3062091**	**24.3062091**	**24.3062091**	7.74285E−07
g08	**−0.09582504**	**−0.09582504**	**−0.09582504**	**−0.09582504**	8.36862E−17	**−0.09582504**	**−0.09582504**	**−0.09582504**	**−0.09582504**	8.36862E−17
g09	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	2.28519E−13	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	2.28519E−13
g10	**7049.248021**	**7049.248021**	**7049.248021**	**7049.248021**	3.97851E−06	**7049.248021**	**7049.248021**	**7049.248021**	**7049.248021**	1.50134E−05
g11	**0.7499**	**0.7499**	**0.7499**	**0.7499**	1.78531E−15	**0.7499**	**0.7499**	**0.7499**	**0.7499**	1.78531E−15
g12	**−1**	**−1**	**−1**	**−1**	0.00000E+00	**−1**	**−1**	**−1**	**−1**	0.00000E+00
g13	**0.053941514**	**0.053941514**	**0.053941514**	**0.053941514**	1.10644E−07	**0.053941514**	**0.053941514**	**0.053941514**	**0.053941514**	3.03523E−07
g14	**−47.7648885**	**−47.7648885**	**−47.7648885**	**−47.7648885**	2.18295E−05	**−47.7648885**	**−47.7648885**	**−47.7648885**	**−47.7648885**	2.23259E−05
g15	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	7.99817E−13	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	7.99817E−13
g16	**−1.905155259**	**−1.90515526**	**−1.905155259**	**−1.905155259**	1.56214E−15	**−1.90515526**	**−1.90515526**	**−1.90515526**	**−1.90515526**	1.56214E−15
g17	**8853.539675**	**8853.539675**	**8853.539675**	**8853.539675**	1.02352E+00	**8853.539675**	**8853.539675**	**8853.539675**	**8853.539675**	1.31206E+00
g18	**−0.8660254**	**−0.8660254**	**−0.8660254**	**−0.8660254**	8.48424E−08	**−0.8660254**	**−0.8660254**	**−0.8660254**	**−0.8660254**	8.62135E−08
g19	**32.65562397**	**32.65559313**	**32.65559552**	**32.65561099**	2.20457E−12	32.6564019	**32.65559313**	**32.65559552**	**32.65561099**	1.70016E−03
g20	**0.230238661**	**0.156974392**	**0.181091379**	**0.181248278**	1.38456E−02	0.231788761	0.15859795	**0.187195512**	**0.18552127**	1.17475E−02
g21	**193.7245101**	**193.7245101**	**193.7245101**	**193.7245101**	1.45452E−10	**193.7245101**	**193.7245101**	**193.7245101**	**193.7245101**	1.68174E−10
g22	**1999.99994**	**617.7734457**	**1596.603082**	**1570.663954**	2.36313E+02	1999.999961	754.8448291	1696.486899	1635.507815	2.94817E+02
g23	**−400.0551**	**−400.0551**	**−400.0551**	**−400.0551**	7.88043E−05	−378.0715141	**−400.0551**	**−400.0551**	**−378.180867**	8.20199E−05
g24	**−5.508013272**	**−5.50801327**	**−5.508013272**	**−5.508013272**	1.78531E−15	**−5.50801327**	**−5.50801327**	**−5.50801327**	**−5.50801327**	1.78531E−15

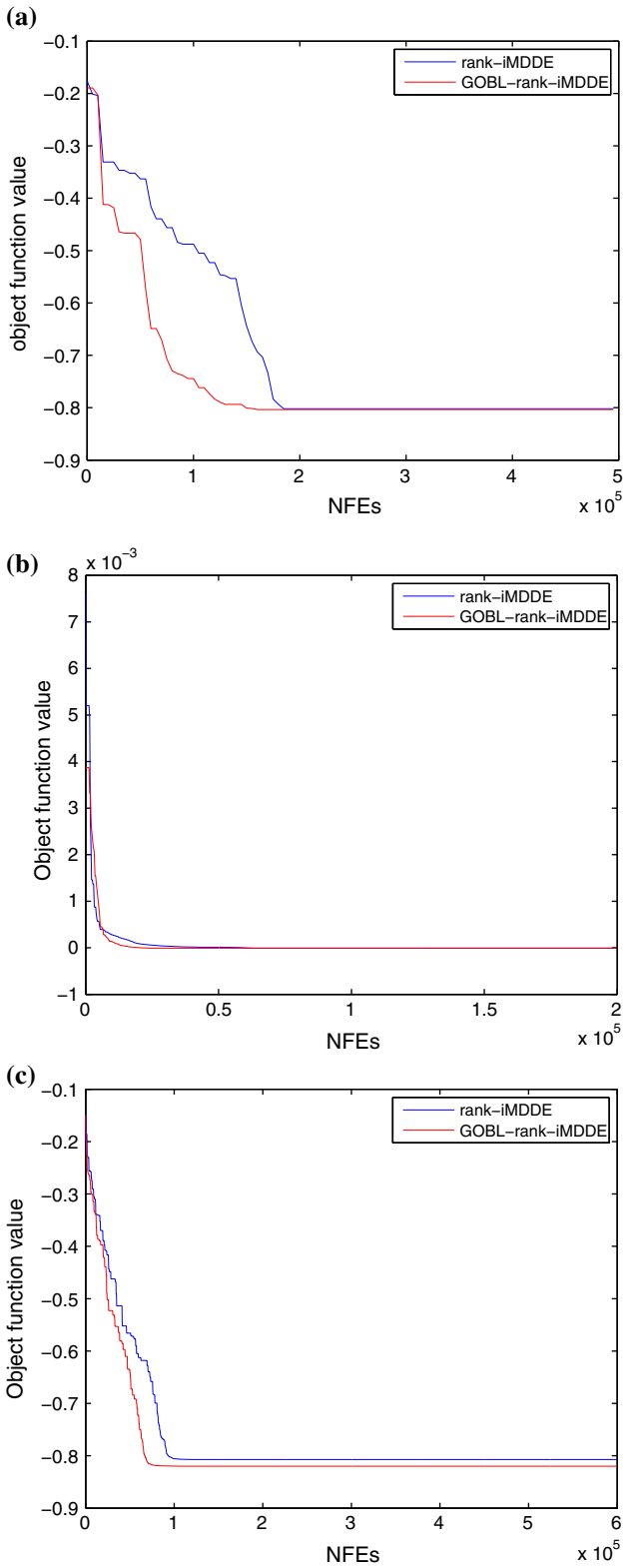


Fig. 2 Convergence graphs of GOBL-rank-iMDDE and rank-iMDDE for the selected functions

four functions rank-iMDDE obtains fewer NFEs than GOBL-rank-iMDDE. Furthermore, both GOBL-rank-iMDDE and rank-iMDDE obtain the same NFEs in three functions.

With respect to 36 test instances (18 test problems each with 10 and 30 dimensions) as proposed in CEC'2010, the results of the final solutions are shown in Tables 3 and 4, where the boldface means that the algorithm obtains the best known solution or better result in the specific function.

Table 3 shows that GOBL-rank-iMDDE obtains better mean values than rank-iMDDE in 12 out of 18 functions, and only in 6 functions rank-iMDDE obtains better mean values than GOBL-rank-iMDDE.

It can be seen from Table 4 that GOBL-rank-iMDDE obtains better mean values than rank-iMDDE in 14 out of 18 functions, and only in 3 functions does rank-iMDDE obtain better mean values than GOBL-rank-iMDDE. In addition, both GOBL-rank-iMDDE and rank-iMDDE have the same values in three functions.

To sum up, the results confirm our expectation that the proposed GOBL-rank-iMDDE improves the performance of rank-iMDDE in the majority of benchmark functions in CEC'2006 and CEC'2010. Besides we also find that the higher dimensions functions have in CEC'2010, the better performance GOBL-rank-iMDDE can obtain.

4.3 Comparison of GOBL-εDEag with εDEag

The results of the final solutions of GOBL-εDEag and εDEag on the benchmark functions presented in CEC'2006 are shown in Table 5, where the boldface means that the algorithm obtains the best known solution or better result in the specific function. In addition, several representative convergence graphs of the selected functions are given in Fig. 3.

According to Table 5, it can be seen as the following. Firstly, both GOBL-εDEag and εDEag get the best known optimal solutions in 23 functions. Secondly, GOBL-εDEag gets better values than εDEag in function g22.

With regard to NFEs, it can be seen from Table 6 that GOBL-εDEag gets fewer NFEs than εDEag in 16 out of 24 functions, and only in six functions εDEag obtains fewer NFEs than GOBL-εDEag. Furthermore, both GOBL-εDEag and εDEag obtain the same NFEs in two functions.

For the benchmark functions in CEC'2010 for 10 and 30 dimensions, the results of the final solutions from GOBL-εDEag and εDEag are shown in Tables 7 and 8, where the boldface means that the algorithm obtains the best known solution or better result in the specific function.

In Table 7, it can be seen that in comparison with the mean values for the 10D function instances, GOBL-εDEag is better for seven functions, while εDEag is better for only five functions; in addition, both GOBL-εDEag and εDEag get the same value in six functions.

From Table 8, with respect to mean values for 30D function instances, GOBL-εDEag gets better values than εDEag in 11 out of 18 functions, and only in seven functions does εDEag obtain better values than GOBL-εDEag.

Table 2 Number of FEs to achieve the success condition by GOBL-rank-iMDDE and rank-iMDDE for all benchmark functions of CEC'2006

Prob	GOBL-rank-iMDDE					Rank-iMDDE				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
g01	**60,050**	**52,050**	**57,050**	**56,877.5**	1.7308E+03	60,300	53,550	**57,050**	56,887.5	1.4023E+03
g02	**231,800**	99,800	**166,800**	**160,633.3**	2.8067E+04	247,300	**99,050**	172,800	182,183.3	3.5801E+04
g03	**26,550**	**21,050**	**22,800**	**23,478.6**	2.0398E+03	28,550	28,550	28,550	28,550.0	0.0000E+00
g04	**25,300**	**20,050**	22,925	**22,450.0**	9.8281E+02	25,550	21,050	**22,800**	22,832.5	9.1471E+02
g05	**24,050**	**20,300**	**22,300**	**22,240.0**	8.6100E+02	25,800	**20,300**	22,425	22,445.0	9.7454E+02
g06	10,050	**8050**	**8800**	**8890.0**	3.3993E+02	**9550**	**8050**	**8800**	8895.0	3.3856E+02
g07	**181,300**	**47,550**	**84,550**	**91,791.9**	2.9416E+04	**18,9050**	48,800	84,925	91,996.4	3.0857E+04
g08	191,300	2050	86,925	88,315.0	3.6974E+04	2800	1550	2300	**2335.0**	2.6116E+02
g09	**24,050**	**16,800**	**19,050**	**19,502.5**	1.6811E+03	**24,050**	17,300	19,300	19,525.0	1.2510E+03
g10	**361,050**	**69,800**	**98,050**	**119,040.0**	6.3536E+04	430,550	72,550	118,550	182,550.0	1.2103E+05
g11	**8050**	**3800**	5425	**5577.5**	9.0096E+02	8300	4300	**5300**	5605.0	9.0981E+02
g12	**4800**	**2550**	**3800**	**3667.5**	4.0179E+02	**4800**	**2550**	**3800**	3710.0	4.2271E+02
g13	**119,800**	17,300	**22,175**	**24,902.8**	1.2392E+04	162,050	**17,050**	22,800	27,168.3	2.0246E+04
g14	402,300	**106,300**	**200,800**	**222,780.0**	7.6298E+04	353,050	136,050	229,050	229,780.0	4.9927E+04
g15	**15,550**	11,800	13,550	13,510.0	6.5397E+02	**15,550**	**11,050**	**13,300**	**13,330.0**	6.9837E+02
g16	**16,550**	**13,050**	**15,050**	**15,085.0**	7.1864E+02	17,050	**13,050**	15,300	15,247.5	7.5653E+02
g17	**26,050**	21,300	**23,050**	**23,250**	1.3130E+03	26,800	**19,300**	23,300	23,280.0	1.5149E+03
g18	64,365	**47,510**	52,210	51,654.5	3.9103E+03	**63,142**	48,541	**51,312**	**51,299.1**	3.6103E+03
g19	230,184	159,870	190,857	180,748.0	2.14193E+04	230,184	159,870	190,857	180,748.0	2.14193E+04
g20	5,000,000	5,000,000	5,000,000	5,000,000	0.0000E+00	5,000,000	5,000,000	5,000,000	5,000,000	0.0000E+00
g21	**167,550**	**53,500**	**61,550**	**66,856.5**	2.3079E+04	175,800	53,550	62,300	68,310.6	2.2458E+04
g22	5,000,000	5,000,000	5,000,000	5,000,000	0.0000E+00	5,000,000	5,000,000	5,000,000	5,000,000	0.0000E+00
g23	**319,550**	130,550	203,675	213,831.3	5.6563E+04	383,800	**114,550**	**199,300**	**208,580.0**	6.8289E+04
g24	**5250**	**3750**	**4550**	**4592.5**	2.9571E+02	5300	3800	4800	4685.0	3.2648E+02

Table 3 Function values achieved by GOBL-rank-iMDDE and rank-iMDDE for all benchmark functions of CEC'2010 for 10D

Prob	GOBL-rank-iMDDE					Rank-iMDDE				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
g01	**−7.25879E−01**	**−7.47310E−01**	**−7.47310E−01**	**−7.45222E−01**	4.92047E−03	−7.00642E−01	**−7.47310E−01**	**−7.47310E−01**	−7.42078E−01	1.04466E−02
g02	**−1.37987E+00**	**−2.2776E+00**	**−1.95502E+00**	**−1.90494E+00**	1.72846E−01	−6.88889E−01	−2.23230E+00	−1.94728E+00	−1.90312E+00	2.59773E−01
g03	**8.87555E+00**	**0.00000E+00**	**8.87555E+00**	**5.68035E+00**	4.30352E+00	**8.87555E+00**	**0.00000E+00**	**8.87555E+00**	6.56791E+00	3.93265E+00
g04	**−9.97381E−06**	**−1.00000E−05**	**−1.00000E−05**	**−9.99948E−06**	3.70387E−09	−7.61566E−06	**−1.00000E−05**	**−1.00000E−05**	−9.95231E−06	3.37196E−07
g05	5.88830E+02	−2.42912E+02	**1.66570E+02**	1.86259E+02	1.91476E+02	**5.14309E+02**	**−2.48197E+02**	1.79854E+02	**1.70494E+02**	1.71073E+02
g06	**−3.22378E+02**	**−5.56233E+02**	**−5.02245E+02**	−4.80048E+02	6.09162E+01	−2.92450E+02	−5.34117E+02	−4.96562E+02	**−4.92099E+02**	4.08609E+01
g07	**4.74188E−11**	**0.00000E+00**	**0.00000E+00**	**9.48390E−13**	6.70603E−12	3.98658E+00	**0.00000E+00**	**0.00000E+00**	7.97343E−02	5.63787E−01
g08	**4.08758E+01**	**0.00000E+00**	1.05729E+01	7.21683E+00	7.97551E+00	**4.08758E+01**	**0.00000E+00**	**8.82095E+00**	**6.95602E+00**	7.56333E+00
g09	**8.47793E+05**	**4.96464E−10**	2.27141E+03	**3.08595E+04**	1.28581E+05	1.27430E+06	8.53458E+01	**1.78872E+03**	4.31117E+04	1.88148E+05
g10	8.93413E+03	1.73175E+01	5.57635E+02	**1.27768E+03**	1.75656E+03	**2.99048E+03**	**1.60442E+01**	**3.21447E+02**	5.93654E+02	7.07639E+02
g11	**−1.52271E−03**	**−8.73415E−02**	**−8.73415E−02**	**−5.64468E−02**	4.16113E−02	**−1.52271E−03**	**−8.73415E−02**	**−8.73415E−02**	−4.78649E−02	4.32061E−02
g12	9.97405E+01	**−5.70090E+02**	−9.39783E+01	−2.29670E+02	2.54731E+02	**6.88334E+01**	**−5.70090E+02**	**−1.17346E+02**	**−2.45570E+02**	2.60671E+02
g13	**−6.22764E+01**	**−6.84294E+01**	**−6.84294E+01**	**−6.62693E+01**	2.37856E+00	−6.16487E+01	**−6.84294E+01**	−6.55785E+01	−6.58627E+01	2.55136E+00
g14	**2.71764E−01**	**0.00000E+00**	**0.00000E+00**	**5.43528E−03**	3.84332E−02	4.36910E−01	**0.00000E+00**	**0.00000E+00**	1.18374E−02	6.51367E−02
g15	**4.49744E+00**	**0.00000E+00**	**3.67324E+00**	**1.96497E+00**	1.87586E+00	**4.49744E+00**	**0.00000E+00**	**3.67324E+00**	2.47379E+00	1.80408E+00
g16	**3.10474E−02**	**0.00000E+00**	**0.00000E+00**	4.27489E−03	1.03050E−02	**3.10474E−02**	**0.00000E+00**	**0.00000E+00**	**1.53341E−03**	6.36056E−03
g17	**9.92690E−02**	**0.00000E+00**	2.25737E−08	**2.01087E−02**	3.48319E−02	1.31508E+00	**0.00000E+00**	**1.19590E−13**	5.57327E−02	1.96429E−01
g18	6.12672E−01	**0.00000E+00**	**3.70549E−33**	1.64706E−02	9.10453E−02	**3.65963E−01**	**0.00000E+00**	6.16298E−33	**7.48489E−03**	5.17443E−02

Table 4 Function values achieved by GOBL-rank-iMDDE and rank-iMDDE for all benchmark functions of CEC'2010 for 30D

Prob	GOBL-rank-iMDDE					Rank-iMDDE				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
c01	**−7.70788E−01**	**−8.21884E−01**	**−8.21057E−01**	**−8.17133E−01**	1.24907E−02	−7.67648E−01	**−8.21884E−01**	−8.14291E−01	−8.09802E−01	1.15080E−02
c02	**2.97405E+00**	**−8.62191E−01**	**8.90998E−01**	**9.50465E−01**	9.09600E−01	4.24369E+00	−6.44299E−01	9.67861E−01	1.18946E+00	1.14242E+00
c03	**5.40685E+10**	**1.14325E−18**	**5.84011E+06**	**2.26562E+09**	9.76655E+09	7.74784E+11	2.86893E+01	2.06426E+08	2.06291E+10	1.09558E+11
c04	**2.44096E−01**	**9.64777E−04**	**3.49707E−01**	**5.43417E−02**	5.48875E+03	**1.00017E+00**	**9.64777E−04**	4.58590E−02	1.71154E−01	2.89627E−01
c05	5.81863E+02	1.13680E+02	3.96676E+02	3.69226E+02	1.32522E+02	**5.62929E+02**	**6.55588E+01**	**3.77913E+02**	**3.67828E+02**	1.38080E+02
c06	**5.77244E+02**	**−3.03284E+02**	2.96743E+02	**2.77532E+02**	2.36470E+02	5.99754E+02	−2.77563E+02	**2.95258E+02**	2.88686E+02	2.11835E+02
c07	**3.98662E+00**	**0.00000E+00**	**2.63279E−26**	**1.18930E−01**	1.09252E+00	**3.98662E+00**	**0.00000E+00**	2.82401E−26	1.59465E−01	7.89147E−01
c08	**1.71557E+03**	**0.00000E+00**	**1.13166E−20**	**1.68278E+02**	4.00407E+02	3.22050E+03	**0.00000E+00**	3.98662E+00	2.10622E+02	5.67824E+02
c09	5.00626E+13	**1.48858E+10**	**1.10301E+12**	**5.66632E+12**	9.78648E+12	4.35349E+13	1.98777E+10	4.44353E+12	9.48079E+12	1.14574E+13
c10	3.45227E+13	**2.47774E+09**	**2.69705E+11**	**3.32094E+12**	7.01045E+12	**2.03774E+13**	8.94870E+09	8.94538E+11	4.74699E+12	6.60359E+12
c11	**−5.90090E+64**	**−5.90090E+64**	**−5.90090E+64**	**−5.90090E+64**	8.26750E+49	**−5.90090E+64**	**−5.90090E+64**	**−5.90090E+64**	**−5.90090E+64**	8.26750E+49
c12	−1.87335E−01	**−3.63307E+00**	**−1.99263E−01**	**−2.67701E−01**	4.85651E−01	**−1.99263E−01**	−1.99263E−01	**−1.99263E−01**	−1.99263E−01	8.35762E−08
c13	**−6.11662E+01**	**−6.77094E+01**	**−6.49084E+01**	**−6.47740E+01**	1.74205E+00	−6.07695E+01	−6.76537E+01	**−6.49084E+01**	−6.47245E+01	1.38393E+00
c14	3.71268E+06	**0.00000E+00**	3.22478E−26	**1.23446E+04**	5.72027E+05	**1.22708E+06**	**0.00000E+00**	**2.89094E−26**	2.62677E+04	1.73514E+05
c15	1.03007E+11	4.21506E+00	2.28692E+01	5.81735E+09	1.83684E+10	**1.52204E+10**	**2.83681E−16**	**2.16039E+01**	**9.43942E+08**	2.93311E+09
c16	4.91616E−02	**0.00000E+00**	9.83935E−04	**3.76882E−03**	7.90023E−03	**4.16939E−02**	**0.00000E+00**	**4.94764E−04**	4.06937E−03	7.62020E−03
c17	**9.36638E+02**	7.56745E+00	**4.49304E+01**	**1.06132E+02**	1.95390E+02	1.30802E+03	**4.44186E+00**	4.62546E+01	1.23850E+02	2.73935E+02
c18	2.44096E+04	**4.13959E+01**	**3.49707E+03**	5.43417E+03	5.48875E+03	**1.71903E+04**	2.49341E+02	3.59706E+03	**4.42845E+03**	4.03008E+03

Table 5 Function values achieved by GOBL-εDEag and εDEag for all benchmark functions of CEC'2006

Prob	GOBL-εDEag					εDEag				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
g01	**−15**	**−15**	**−15**	**−15**	0.00000E+00	**−15**	**−15**	**−15**	**−15**	0.00000E+00
g02	**−0.803609744**	**−0.803619049**	**−0.803617744**	**−0.803618318**	5.33507E−07	−0.803605318	**−0.803619049**	−0.803617317	**−0.803618318**	1.75230E−06
g03	**−1.00050003**	**−1.000500094**	**−1.000500044**	**−1.00050004**	1.97112E−07	**−1.00050003**	**−1.00050003**	**−1.00050003**	**−1.00050003**	2.95820E−11
g04	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	7.42599E−12	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	1.56900E−06
g05	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	1.85650E−12	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	3.12541E−10
g06	**−6961.813876**	**−6961.813876**	**−6961.813876**	**−6961.813876**	9.18988E−09	**−6961.813876**	**−6961.813876**	**−6961.813876**	**−6961.813876**	9.54329E−09
g07	**24.3062091**	**24.3062091**	**24.3062091**	**24.3062091**	3.33088E−10	**24.3062091**	**24.3062091**	**24.3062091**	**24.3062091**	2.18310E−15
g08	**−0.095825041**	**−0.095825041**	**−0.095825041**	**−0.095825041**	5.60864E−17	**−0.095825041**	**−0.095825041**	**−0.095825041**	**−0.095825041**	6.97987E−17
g09	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	2.78144E−13	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	3.12387E−08
g10	**7049.248021**	**7049.248021**	**7049.248021**	**7049.248021**	1.86024E−10	**7049.248021**	**7049.248021**	**7049.248021**	**7049.248021**	0.00000E+00
g11	**0.7499**	**0.7499**	**0.7499**	**0.7499**	0.00000E+00	**0.7499**	**0.7499**	**0.7499**	**0.7499**	0.00000E+00
g12	**−1**	**−1**	**−1**	**−1**	0.00000E+00	**−1**	**−1**	**−1**	**−1**	0.00000E+00
g13	**0.053941514**	**0.053941514**	**0.053941514**	**0.053941514**	7.60109E−12	**0.053941514**	**0.053941514**	**0.053941514**	**0.053941514**	5.78733E−12
g14	**−47.76488846**	**−47.76488846**	**−47.76488846**	**−47.76488846**	1.48030E−13	**−47.76488846**	**−47.76488846**	**−47.76488846**	**−47.76488846**	3.87630E−14
g15	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	2.64655E+00	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	2.43301E−06
g16	**−1.905155259**	**−1.905155259**	**−1.905155259**	**−1.905155259**	3.35831E−15	**−1.905155259**	**−1.905155259**	**−1.905155259**	**−1.905155259**	4.89713E−15
g17	**8853.539675**	**8853.539675**	**8853.539675**	**8853.539675**	5.56949E−12	**8853.539675**	**8853.539675**	**8853.539675**	**8853.539675**	4.13251E−12
g18	**−0.866025399**	**−0.866025404**	**−0.8660254**	**−0.866025358**	6.77677E−08	**−0.8660254**	**−0.8660254**	**−0.8660254**	**−0.8660254**	7.43308E−08
g19	**32.65559295**	**32.65559295**	**32.65559295**	**32.65559295**	2.41682E−07	**32.65559295**	**32.65559295**	**32.65559295**	**32.65559295**	2.46070E−06
g20	0.272731135	**0.16566318**	**0.196735952**	**0.202435629**	2.75063E−08	**0.206323642**	0.206323642	0.206323642	0.206323642	4.23620E−02
g21	**193.7245101**	**193.7245101**	**193.7245101**	**193.7245101**	6.99575E−08	**193.7245101**	**193.7245101**	**193.7245101**	**193.7245101**	3.34170E−14
g22	19,230.79485	**1441.945001**	7552.522438	**3565.694744**	5.71595E+03	**14,038.49409**	2131.365535	**4241.444535**	4131.365535	2.52615E+04
g23	**−400.0551**	**−400.0551**	**−400.0551**	**−400.0551**	2.71353E−08	**−400.0551**	**−400.0551**	**−400.0551**	**−400.0551**	1.11390E−14
g24	**−5.508013272**	**−5.508013272**	**−5.508013272**	**−5.508013272**	1.81299E−15	**−5.508013272**	**−5.508013272**	**−5.508013272**	**−5.508013272**	2.52440E−29

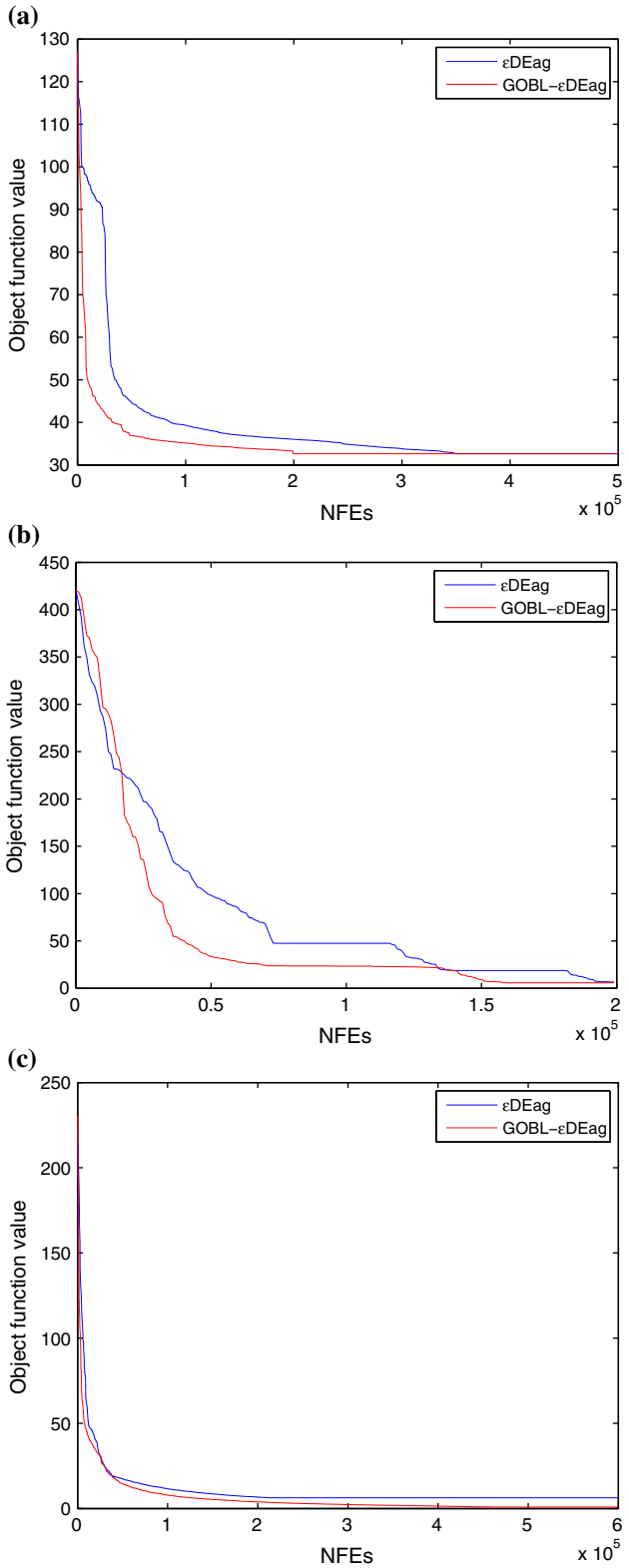


Fig. 3 Convergence graphs of GOBL-εDEag and εDEa for the selected functions

Above all, although εDEag was the winner algorithm in the CEC'2010 competition, the results show that the proposed GOBL-εDEag still improves the performance of εDEag in the majority of benchmark functions in CEC'2006 and CEC'2010, and the higher dimensions functions have in CEC'2010, the better performance GOBL-εDEag can obtain.

4.4 Comparison with other state-of-the-art EAs

For further evaluation of the proposed framework, firstly, GOBL-rank-iMDDE and GOBL-εDEag are compared with other state-of-the-art EAs for COPs. With respect to CEC'2006, these algorithms are SAMODE (Elsayed et al. 2011), ATMES (Omran and Salman 2009), TLBO (Rao et al. 2011), IVPSO (Sun et al. 2011) and ABC (Karaboga and Akay 2011). Secondly, the GOBL-CDE framework is applied to other CDE algorithms including $(\mu + \lambda)$-CDE, ECHT-DE and MDDE, named as GOBL-$(\mu + \lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE, respectively.

Tables 9 and 10 show the mean of the objective function values of the final solutions for each algorithm in CEC'2006. The overall best results among the algorithms are highlighted in boldface. "NA" means not available.

In Table 9, we compare the results of GOBL-rank-iMDDE and GOBL-εDEag with those of SAMODE, ATMES, TLBO, IVPSO, and ABC in all benchmark functions. It is obvious that ours are better than those of SAMODE, ATMES, TLBO, IVPSO, and ABC.

According to the results shown in Table 10, it can be clearly seen that GOBL-$(\mu + \lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE consistently obtain the best known optimal solutions in almost all functions, and compared with $(\mu + \lambda)$-CDE, ECHT-DE and MDDE, respectively, GOBL-$(\mu + \lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE get highly competitive results in all functions. Especially GOBL-MDDE gets better values than MDDE in 13 out of 24 functions, and gets the same value in other functions.

Tables 11 and 12 show the results of these six algorithms in 18 the benchmark functions in CEC'2010 for 10 and 30 dimensions. With regard to 10D function instances, GOBL-$(\mu + \lambda)$-CDE and GOBL-ECHT-DE get better values than $(\mu+\lambda)$-CDE and ECHT-DE in 15 out of 18 functions, respectively; GOBL-MDDE get better values than MDDE in 13 out of 18 functions. With regard to 30D function instances, GOBL-$(\mu+\lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE get better values than $(\mu + \lambda)$-CDE, ECHT-DE and MDDE in 15 out of 18 functions, respectively.

The Friedman test is used to obtain the ranking of different algorithms for all problems (Derrac et al. 2011; Alcal-Fdez et al. 2008). The Friedman test was carried out and the

Table 6 Number of FEs to achieve the success condition by GOBL-εDEag and εDEag for all benchmark functions of CEC'2006

Prob	GOBL-εDEag					εDEag				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
g01	**58731**	**45654**	**47372**	**48377**	1.60631E+04	61712	57122	59345	59308	1.15629E+03
g02	**162349**	**124061**	**128606**	**128356**	2.72805E+03	175206	126152	146911	149825	1.57193E+04
g03	97330	**53570**	**83894**	**80280**	1.25808E+04	**91328**	86748	89473	89407	1.07732E+03
g04	**17673**	**12406**	**15217**	**14465**	1.17967E+04	28206	24800	26098	26216	9.09203E+02
g05	117108	**74608**	**81994**	**84406**	1.05731E+04	**98589**	96812	97379	97431	4.02579E+02
g06	**8047**	**6404**	7339	7441	2.15203E+03	8382	6499	7316	**7381**	4.60634E+02
g07	13450	81606	**66237**	82203	1.11869E+04	78963	69506	74476	**74303**	2.40937E+03
g08	**1033**	557	1189	**1078**	7.64281E+02	1334	**327**	**1182**	1139	1.94050E+02
g09	25208	**17609**	23951	**23037**	2.37792E+04	**24790**	19530	**23172**	23121	1.15244E+03
g10	**121617**	105609	**105533**	**104096**	1.57305E+04	122387	**93743**	105799	105234	6.77696E+03
g11	29732	26386	27053	27307	9.63663E+02	**29510**	**5407**	**16821**	**16420**	6.57072E+03
g12	**4507**	4203	**4111**	**4103**	7.55211E+01	5540	**1645**	4155	4124	8.12554E+02
g13	**61774**	8826	**27865**	**26302**	4.56851E+03	68608	**8287**	33594	34738	1.59581E+04
g14	**107992**	**61385**	**88461**	**84028**	1.38338E+04	121656	106816	112526	113439	4.19509E+03
g15	100691	**10133**	**57152**	**48528**	3.28189E+04	**90593**	57729	87185	84216	7.14719E+03
g16	16676	14254	15570	15093	5.39090E+03	**13923**	**12347**	**13001**	**12986**	4.00629E+02
g17	131531	105701	121610	120680	6.56778E+03	**100144**	**97274**	**99021**	**98861**	7.91636E+02
g18	**64225**	51679	**54753**	**56483**	9.63150E+03	72112	**51035**	59232	59153	4.32170E+03
g19	**208745**	**171184**	**191584**	**189692**	1.07619E+04	451685	319636	354060	356350	2.82156E+04
g20	5000000	5000000	5000000	5000000	0.00000E+00	5000000	5000000	5000000	5000000	0.00000E+00
g21	312194	94735	255465	236451	6.57500E+04	**216905**	**126194**	**133224**	**135143**	1.69588E+04
g22	5000000	5000000	5000000	5000000	0.00000E+00	5000000	5000000	5000000	5000000	0.00000E+00
g23	**150287**	**119438**	**137925**	**137241**	8.05700E+03	281071	158742	193593	200765	2.76710E+04
g24	**2983**	**2394**	**2596**	**2614**	1.87796E+03	3474	2661	2928	2952	1.84842E+02

results are shown in Table 13. With respect to the average rankings of different algorithms by the Friedman test, it is clearly shown that GOBL-rank-iMDDE, GOBL-εDEag, GOBL-$(\mu+\lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE consistently obtain better rankings in all algorithms for the functions at CEC'2006 and CEC'2010.

From Table 13, we find that GOBL-εDEag and εDEag are ranked 1 and 2 in CEC'2010, respectively, and what εDEag was the winner algorithm in the CEC'2010 competition is verified again. It is obvious that GOBL-rank-iMDDE, GOBL-εDEag, GOBL-$(\mu+\lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE are better than rank-iMDDE, εDEag, $(\mu+\lambda)$-CDE, ECHT-DE and MDDE, respectively, in CEC'2006 and CEC'2010.

To this end, the Wilcoxon Signed Rank Test (Corder and Foreman 2009) is used to perform statistically significant testing in CEC'2006 and CEC'2010. Table 14 shows that GOBL-rank-iMDDE, GOBL-εDEag, GOBL-$(\mu+\lambda)$-CDE, GOBL-ECHT-DE and GOBL-MDDE are better than rank-iMDDE, εDEag, $(\mu+\lambda)$-CDE, ECHT-DE and MDDE for test functions in CEC'2006 and CEC'2010, respectively. This means that the GOBL-CDE framework significantly improves the performance of the CDE algorithms for test functions in CEC'2006 and CEC'2010, when the GOBL-CDE framework was applied to these CDE algorithms. Note that with regard to 10D functions in CEC'2010, GOBL-rank-iMDDE gets almost same R+ and R− against rank-iMDDE; however, GOBL-rank-iMDDE gets more bigger R+ than R− against rank-iMDDE in 30D functions. There are similar performance in GOBL-εDEag and εDEag. This also shows that the higher dimensions are functions in CEC'2010, the better performance can GOBL-CDE obtain.

The non-parametric analysis over the obtained results using the Wilcoxon signed ranks test showed that the proposed framework, despite its simplicity, had a remarkable performance over a wide and various set of test problems.

4.5 Application to real-world problem

The GOBL framework keeps the simple structure of the CDE algorithm, and can be easily applied to other advanced CDE variants to further improve the performance. This provides

Table 7 Function values achieved by GOBL-εDEag and εDEag for all benchmark functions of CEC'2010 for 10D

Prob	GOBL-εDEag					εDEag				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
c01	**−7.40872E−01**	**−7.47310E−01**	**−7.47310E−01**	**−7.47172E−01**	1.59660E−02	−7.40557E−01	**−7.47310E−01**	**−7.47310E−01**	−7.47040E−01	1.32334E−03
c02	**−2.21034E+00**	−2.27763E+00	−2.24615E+00	**−2.25943E+00**	1.59544E−02	−2.17450E+00	**−2.27770E+00**	**−2.26950E+00**	−2.25887E+00	2.38978E−02
c03	1.17827E−09	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	2.35468E−10	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00
c04	3.53367E−03	7.64351E−05	1.85588E−03	1.79998E−03	1.08878E−03	**−9.28230E−06**	**−9.99235E−06**	**−9.97728E−06**	**−9.91845E−06**	1.54673E−07
c05	**−4.83611E+02**	**−4.83611E+02**	**−4.83611E+02**	**−4.83611E+02**	1.99000E−06	**−4.83611E+02**	**−4.83611E+02**	**−4.83611E+02**	**−4.83611E+02**	3.89035E−13
c06	**−5.78646E+02**	**−5.78662E+02**	**−5.78659E+02**	**−5.78658E+02**	3.50763E−03	−5.78645E+02	−5.78658E+02	−5.78653E+02	−5.78653E+02	3.62717E−03
c07	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00
c08	**1.09416E+01**	**0.00000E+00**	**1.05729E+01**	**5.81957E+00**	5.43779E+00	1.53754E+01	**0.00000E+00**	1.09415E+01	6.72753E+00	5.56065E+00
c09	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00
c10	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00
c11	3.29331E−02	**−1.58048E−01**	−1.38616E−03	**−1.17359E−02**	3.98546E−02	**−1.52271E−03**	−1.52271E−03	**−1.52271E−03**	−1.52271E−03	6.34104E−11
c12	3.55825E+02	**−8.87038E+02**	−1.99187E−01	−2.41140E+02	4.01202E+02	**−1.98913E−01**	−5.70090E+02	**−4.23133E+02**	**−3.36735E+02**	1.78217E+02
c13	−3.87749E+01	**−6.84294E+01**	−6.84253E+01	−6.05749E+01	1.08041E+01	**−6.84294E+01**	**−6.84294E+01**	**−6.84294E+01**	**−6.84294E+01**	1.02596E−06
c14	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**	0.00000E+00
c15	9.26652E+02	**0.00000E+00**	**0.00000E+00**	3.76058E+01	1.85224E+02	**4.49745E+00**	**0.00000E+00**	**0.00000E+00**	**1.79898E−01**	8.81316E−01
c16	**9.18108E−01**	3.22221E−02	3.00736E−01	**3.33388E−01**	2.57111E−01	1.01827E+00	**0.00000E+00**	**2.81984E−01**	3.70205E−01	3.71048E−01
c17	**1.59697E−01**	1.22829E−13	**3.67356E−03**	**1.57144E−02**	3.36304E−02	7.30177E−01	**1.46318E−17**	5.65333E−03	1.24956E−01	1.93720E−01
c18	1.15438E−12	**3.68101E−25**	5.31710E−15	7.74782E−14	2.38268E−13	**9.22703E−18**	3.73144E−20	**4.09791E−19**	**9.67877E−19**	1.81123E−18

Table 8 Function values achieved by GOBL-εDEag and εDEag for all benchmark functions of CEC'2010 for 30D

Prob	GOBL-εDEag					εDEag				
	Worst	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std
c01	**−8.20133E−01**	**−8.21826E−01**	**−8.20633E−01**	**−8.21747E−01**	7.51483E−02	−8.19547E−01	−8.21826E−01	−8.20617E−01	−8.20869E−01	7.10389E−04
c02	**−2.15686E+00**	**−2.27637E+00**	**−2.23119E+00**	**−2.21763E+00**	3.50889E−02	−2.11710E+00	−2.16925E+00	−2.15215E+00	−2.15142E+00	1.19758E−02
c03	1.87414E+02	**1.17738E−10**	4.44964E+01	7.41040E+01	6.31123E+01	**3.27801E+01**	2.86735E+01	**2.86735E+01**	**2.88379E+01**	8.04716E−01
c04	3.32811E−01	**1.60274E−03**	**4.49136E−03**	**6.29269E−02**	9.20558E−02	**1.77789E−02**	4.69811E−03	6.94761E−03	8.16297E−03	3.06779E−03
c05	−4.39791E+02	**−4.83587E+02**	**−4.76399E+02**	**−4.68057E+02**	1.61930E+01	**−4.42159E+02**	−4.53131E+02	−4.50040E+02	−4.49546E+02	2.89911E+00
c06	**−5.28630E+02**	**−5.30389E+02**	**−5.29714E+02**	**−5.29564E+02**	5.59536E−01	−5.26454E+02	−5.28575E+02	−5.28041E+02	−5.27907E+02	4.74838E−01
c07	3.98662E−12	5.23641E−12	9.36915E−12	2.04314E−12	8.17634E−01	**5.48192E−15**	**1.14711E−15**	**2.11443E−15**	**2.60363E−15**	1.23343E−15
c08	2.23673E−13	3.34111E−14	8.29069E−14	9.39926E−14	4.46639E+01	**2.57811E−13**	**2.51869E−14**	**6.51151E−14**	**7.83146E−14**	4.85518E−14
c09	4.89838E+04	**1.10490E−16**	**2.19354E−09**	**4.17919E+00**	1.35006E+02	**1.05276E+02**	2.77067E−16	1.12461E−08	1.07214E+01	2.82192E+01
c10	4.97215E+03	**8.77224E−13**	**3.26973E+01**	**3.12511E+01**	1.34965E+03	**3.46324E+01**	3.25200E+01	3.32890E+01	3.32618E+01	4.54558E−01
c11	5.18653E−02	**−3.46173E−02**	4.58815E−02	3.82999E−02	1.91885E−02	**−2.23634E−04**	−3.26846E−04	**−2.84330E−04**	**−2.86388E−04**	2.70761E−05
c12	6.66783E+02	**−9.03684E+02**	**−5.40820E+02**	**−3.95674E+02**	4.46644E+02	**5.46172E+02**	−1.99145E−01	5.33713E+02	3.56233E+02	2.88925E+02
c13	−5.01175E+01	**−6.69674E+01**	−6.09042E+01	−6.02141E+01	4.47717E+00	**−6.42969E+01**	−6.64247E+01	**−6.53151E+01**	**−6.53531E+01**	5.73301E−01
c14	2.67580E+01	2.05353E−09	6.84221E−02	1.39545E+00	5.35703E+00	**2.92351E−12**	**5.01586E−14**	**1.35931E−13**	**3.08941E−13**	5.60841E−13
c15	5.55960E+02	**1.71148E−08**	2.16065E+01	5.96790E+01	1.19860E+02	**2.16040E+01**	2.16035E+01	**2.16038E+01**	**2.16038E+01**	1.10483E−04
c16	**6.18577E−22**	**0.00000E+00**	**0.00000E+00**	**6.40177E−22**	1.57551E−01	5.42101E−20	**0.00000E+00**	**0.00000E+00**	2.16840E−21	1.06230E−20
c17	**1.13541E+01**	**3.82317E−05**	**1.03139E−01**	**8.66869E−01**	2.39049E+00	1.88906E+01	2.16572E−01	5.31595E+00	6.32649E+00	4.98669E+00
c18	**4.77556E−01**	**6.03995E−05**	**6.36551E−03**	**6.93354E−02**	1.32173E−01	7.37536E+02	1.22605E+00	2.67950E+01	8.75457E+01	1.66475E+02

Table 9 Comparison of the quality of final solutions of our framework with other state-of-the-art EAs for all benchmark functions of CEC'2006

Prob	GOBL-rank-iMDDE	GOBL-εDEag	SAMODE	ATMES	IVPSO	ABC	TLBO
g01	**−15**	**−15**	**−15**	**−15**	**−15**	**−15**	**−15**
g02	**−0.803619085**	−0.803618318	−0.7987352	−0.790148	−0.769889	−0.792412	NA
g03	**−1.0005001**	**−1.00050004**	**−1.0005**	−1	−1.005010	−1.000	−1
g04	**−30,665.53867**	**−30,665.53867**	**−30,665.539**	**−30,665.539**	**−30,665.538672**	**−30,665.539**	NA
g05	**5126.496714**	**5126.496714**	**5126.497**	5127.648	**5126.492646**	5185.714	NA
g06	**−6961.813876**	**−6961.813876**	**−6961.814**	**−6961.814**	−6961. 813876	−6961.813	NA
g07	**24.30620907**	**24.3062091**	24.3096	24.316	24.429669	24.473	NA
g08	**−0.095825041**	**−0.095825041**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	NA
g09	**680.6300574**	**680.6300574**	**680.63**	680.639	680.630077	680.640	680.633
g10	**7049.248021**	**7049.248021**	7059.81345	7250.437	7053.619895	7224.407	7083.6732
g11	**0.7499**	**0.7499**	**0.7499**	**0.75**	**0.749000**	0.750	NA
g12	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**
g13	**0.053941514**	**0.053941514**	**0.053942**	0.053959	1.795008	0.968	NA
g14	**−47.76488846**	**−47.76488846**	−47.68115	NA	NA	NA	NA
g15	**961.7150223**	**961.7150223**	**961.71502**	NA	NA	NA	NA
g16	**−1.905155259**	**−1.905155259**	**−1.905155**	NA	NA	NA	NA
g17	**8853.533875**	**8853.539675**	**8853.5397**	NA	NA	NA	NA
g18	**−0.866025404**	**−0.866025358**	**−0.866025**	NA	NA	NA	NA
g19	**32.65559295**	**32.65559295**	32.6591	NA	NA	NA	NA
g20	0.204974675	**0.202435629**	NA	NA	NA	NA	NA
g21	**193.7245101**	**193.7245101**	193.7438	NA	NA	NA	NA
g22	10,170.08901	**3565.694744**	NA	NA	NA	NA	NA
g23	**−400.0551**	**−400.0551**	−373.2178	NA	NA	NA	NA
g24	**−5.508013272**	**−5.508013272**	**−5.508013**	NA	NA	NA	NA

a promising approach for solving the real-world engineering design problems because most advanced CDE variants have obtained achievement in these problems. In order to study the performance of solving the real-world engineering design problems, the potential of GOBL-CDE is also tested on 5 widely used constrained engineering benchmark problems.

(1) Welded beam design [Case 1 (De Melo and Carosio 2013) and Case 2 (Huang et al. 2007)]

In this problem, a welded beam must be designed for minimum cost subjected to some constraints named: shear stress (τ), bending stress in the beam (θ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints. This problem presents four design variables: $h(x_1)$, $l(x_2)$, $t(x_3)$, and $b(x_4)$, where $0.125 \leq x_1 \leq 10.0, 0.1 \leq x_2 \leq 10.0, 0.1 \leq x_3 \leq 10.0$ and $0.1 \leq x_4 \leq 10.0$ in case 1, and where $0.1 \leq x_1 \leq 2.0, 0.1 \leq x_2 \leq 10.0, 0.1 \leq x_3 \leq 10.0$ and $0.1 \leq x_4 \leq 2.0$ in case 2.

(2) Tension/compression spring design (De Melo and Carosio 2013)

A tension/compression spring must be designed for minimum weight subject to the following constraints: shear stress, minimum deflection, surge frequency, limits on outside diameter and on design variables. The design variables are: the wire diameter ($x_1 = D$), the mean coil diameter ($x_2 = d$), and the number of active coils ($x_3 = N$). The design optimization problem presents three continuous variables and four nonlinear inequality constraints with the following ranges: $0.25 \leq x_1 \leq 1.3$; $0.05 \leq x_2 \leq 2.0$, and $2 \leq x_3 \leq 15$.

(3) Speed reducer design (De Melo and Carosio 2013)

The weight of a speed reducer must be minimized subject to 11 constraints regarding the following characteristics: bending stress of the gear teeth, surface stress, transverse defections of the shafts and stresses in the shaft. The problem has seven variables to be minimized ($x_1, \ldots, x_7$): the face width, module of the teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings, and the diameter of the first and second shafts, with the ranges $2.6 \leq x_1 \leq 3.6, 0.7 \leq x_2 \leq 0.8, 17 \leq x_3 \leq 28, 7.3 \leq x_4 \leq 8.3, 7.3 \leq x_5 \leq 8.3, 2.9 \leq x_6 \leq 3.9$, and $5 \leq x_7 \leq 5.5$. This is a mixed integer programming problem, where all variables are continuous, except x_3 that is integer.

(4) Three bar truss design (De Melo and Carosio 2013)

This problem considers a 3-bar planar truss structure. The volume of a loaded 3-bar truss must be minimized subject

Table 10 Comparison of the quality of final solutions of our framework with other CDE algorithms for all benchmark functions of CEC'2006

Prob	GOBL-($\mu + \lambda$)-CDE	($\mu + \lambda$)-CDE	GOBL-ECHT-DE	ECHT-DE	GOBL-MDDE	MDDE
g01	**−15**	**−15**	**−14.99999963**	−14.997526	**−15**	−14.99999997
g02	**−0.803619101**	**−0.803619101**	**−0.803618624**	−0.786177234	**−0.803619077**	−0.8025171
g03	**−1.0005001**	**−1.0005001**	**−1.0005001**	**−1.0005001**	**−1.0005001**	**1.0005001**
g04	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	**−30,665.53867**	**−30,665.539**
g05	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	**5126.496714**	**5126.49671**
g06	**−6961.813876**	**−6961.813876**	**−6961.813876**	**−6961.813876**	**−6961.813876**	**−6961.8139**
g07	**24.30620907**	**24.30620907**	**24.30620907**	24.30621474	**24.30620907**	24.3062092
g08	**−0.095825041**	**−0.095825041**	**−0.095825041**	**−0.095825041**	**−0.095825041**	**−0.095825**
g09	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	**680.6300574**	**680.630057**
g10	**7049.248021**	**7049.248021**	**7049.248021**	7049.248344	**7049.248021**	7049.280508
g11	**0.7499**	**0.7499**	**0.749946719**	**0.7499**	**0.7499**	**0.7499**
g12	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**
g13	**0.053941514**	**0.053941514**	**0.053941514**	0.111281064	**0.053941514**	0.065487347
g14	**−47.76488846**	**−47.76488846**	**−47.76488701**	−47.76487859	**−42.72788828**	NA
g15	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	**961.7150223**	**961.715022**
g16	**−1.905155259**	**−1.905155259**	**−1.905155259**	**−1.905155259**	**−1.905155259**	**−1.9051553**
g17	**8853.533875**	**8853.533875**	**8853.539675**	**8853.539675**	**8853.539675**	8853.550185
g18	**−0.866025404**	**−0.866025404**	**−0.866025404**	−0.865945344	**−0.866025404**	−0.8660249
g19	**32.65559295**	**32.65559295**	**32.65573101**	32.96557911	**32.65773795**	32.67088106
g20	**0.204974681**	0.204974688	**1.697904699**	5.706497966	**0.112712015**	NA
g21	**193.7245101**	**193.7245101**	**193.7245101**	193.7245162	**193.7245103**	193.7245106
g22	**10,542.29732**	10,899.118	**1104.914714**	9863.090491	**1414.689222**	NA
g23	**−400.0551**	**−400.0551**	**−400.0551**	−360.5007407	**−49.15365373**	NA
g24	**−5.508013272**	**−5.508013272**	**−5.508013272**	**−5.508013272**	**−5.508013272**	**−5.5080133**

Table 11 Comparison of the quality of final solutions of our framework with other CDE algorithms for 10D benchmark functions of CEC'2010

Prob	GOBL-($\mu + \lambda$)-CDE	($\mu + \lambda$)-CDE	GOBL-ECHT-DE	ECHT-DE	GOBL-MDDE	MDDE
c01	−7.46230E−01	**−8.21884E−01**	−7.45272E−01	**−7.47300E−01**	**−7.46291E−01**	−7.45191E−01
c02	**−2.19418E+00**	3.12004E+00	**−1.41345E+00**	−3.57600E−01	**−1.86008E+00**	−1.83236E+00
c03	1.05900E+14	**8.17444E+13**	**0.00000E+00**	4.05510E−27	**5.85786E+00**	**5.85786E+00**
c04	**−1.00000E−05**	4.07705E+00	9.47977E−01	**−9.72770E−06**	**−1.00000E−05**	−8.13813E−06
c05	**−3.17867E+02**	5.18509E+02	**2.75811E+02**	3.61854E+02	1.50223E+02	**1.32582E+02**
c06	**−3.10967E+02**	4.63479E+02	**−3.61030E+02**	2.44167E+02	**−5.77733E+02**	−5.76662E+02
c07	**0.00000E+00**	4.92486E+00	**0.00000E+00**	5.13810E−28	**1.66095E−25**	1.59463E−01
c08	**8.94019E+00**	2.39545E+01	**6.23070E+00**	7.08580E+00	**5.01326E+00**	5.71844E+00
c09	**1.99217E+12**	3.69464E+13	**0.00000E+00**	9.85000E−28	5.65018E+01	**4.75644E+01**
c10	**1.90314E+12**	3.72879E+13	**8.90511E+00**	4.17259E+01	**4.05538E+01**	4.15492E+01
c11	**−1.52271E−03**	2.67863E−03	**−1.80101E−01**	−1.50000E−03	**−7.01778E−02**	−6.67450E−02
c12	**−1.99246E−01**	6.34296E−01	**−5.79718E+02**	−2.83678E+01	**−2.21494E+02**	−1.81951E+02
c13	**−6.43764E+01**	−6.32549E+01	−6.74981E+01	**−6.75697E+01**	−6.81759E+01	**−6.83153E+01**
c14	**0.00000E+00**	2.14258E+11	**1.65331E−17**	6.55160E+03	1.36759E−12	**1.77075E−13**
c15	**4.83127E+12**	8.75919E+12	**5.36931E+12**	1.80150E+13	2.07350E+00	**2.00003E+00**
c16	7.49247E−02	**3.47851E−02**	**6.27925E−01**	1.03920E+00	**0.00000E+00**	1.52984E−03
c17	**0.00000E+00**	1.39569E+03	**0.00000E+00**	5.29826E+01	**2.30668E−03**	4.60835E−03
c18	**0.00000E+00**	1.16819E+04	**7.01651E−09**	7.38200E+00	**3.02842E−09**	2.13656E−04

Table 12 Comparison of the quality of final solutions of our framework with other CDE algorithms for 30D benchmark functions of CEC'2010

Prob	GOBL-($\mu+\lambda$)-CDE	($\mu+\lambda$)-CDE	GOBL-ECHT-DE	ECHT-DE	GOBL-MDDE	MDDE
c01	−8.08107E−01	**−8.17161E−01**	−7.9043E−01	**−8.1441E−01**	**−8.13083E−01**	−8.12791E−01
c02	**−1.26120E+00**	2.14449E+00	**3.6421E+00**	4.5511E+00	**1.81070E+00**	1.83599E+00
c03	**1.42012E+13**	2.62549E+14	**2.1192E+08**	5.1552E+11	**8.41883E+07**	3.18197E+08
c04	**2.60533E+00**	7.14062E+00	4.1227E+00	**9.9666E−02**	**2.00014E−03**	1.64784E−03
c05	**3.25308E+02**	5.14703E+02	5.1753E+02	**4.8434E+02**	4.08596E+02	**3.72475E+02**
c06	**−9.45110E+01**	4.81840E+02	**5.2512E+02**	5.2665E+02	**4.20213E+02**	4.39119E+02
c07	**1.59466E−01**	5.39014E+00	**6.3424E−08**	2.8303E−03	**1.59465E−02**	7.97325E−02
c08	**2.55532E+01**	2.76514E+01	**4.0858E+00**	4.7799E+00	1.54692E+01	**6.61962E+00**
c09	**2.48417E+13**	3.64132E+13	**1.6718E+13**	2.8925E+13	**1.24408E+13**	1.56884E+13
c10	**1.93631E+13**	3.76773E+13	**2.3021E+13**	3.0964E+13	**1.13570E+13**	1.22725E+13
c11	3.42135E−03	**1.92124E−03**	**−8.3099E−03**	2.2626E−03	**−5.90090E+64**	−5.90090E+64
c12	**−8.29654E−02**	4.18799E−01	**−1.4915E+02**	−1.0523E+01	**−3.74874E+01**	−1.24557E+01
c13	−6.14210E+01	**−6.35987E+01**	**−6.4874E+01**	−6.2403E+01	**−6.82815E+01**	−6.81661E+01
c14	**1.97696E+05**	3.25035E+11	**3.9821E+00**	3.8755E+11	1.47747E+10	**1.45958E+02**
c15	**1.06823E+13**	1.06988E+13	**1.0428E+14**	1.2610E+14	**1.93873E+13**	2.00803E+13
c16	**1.85573E−02**	3.62956E−02	**1.1245E+00**	1.1492E+00	**2.81001E−02**	3.43764E−02
c17	**7.34378E−01**	1.41642E+03	**1.1635E+03**	1.4008E+03	**6.34223E+02**	7.58007E+02
c18	**5.03325E−03**	1.63365E+04	**9.1335E+03**	2.4402E+04	**1.35717E+04**	1.42887E+04

Table 13 Average ranking of all the algorithms by the Friedman test for CEC'2006 and CEC'2010

Algorithms	Ranking		
	CEC'2006	CEC'2010	
		10D	30D
GOBL-rank-iMDDE	5.0625 (3)	5.3056 (6)	4.6944 (3)
Rank-iMDDE	5.375 (6)	6.0278 (7)	5.3056 (5)
GOBL-εDEag	5.2917 (5)	3.9381 (1)	2.8889 (1)
εDEag	5.4167 (7)	4.0944 (2)	2.9167 (2)
GOBL-($\mu+\lambda$)-CDE	4.5417 (1)	6.2778 (8)	5.8889 (7)
($\mu+\lambda$)-CDE	4.625 (2)	8.9722 (10)	8.4167 (10)
GOBL-ECHT-DE	5.4167 (7)	4.8889 (5)	6.6389 (8)
ECHT-DE	7.4375 (9)	7.0278 (9)	7.8056 (9)
GOBL-MDDE	5.2292 (4)	4.1389 (3)	5.0556 (4)
MDDE	6.6042 (10)	4.4722 (4)	5.3889 (6)

to stress (σ) constraints on each of the truss members. The design optimization problem presents two continuous variables ($x_1 = A_1$ and $x_2 = A_2$) and three nonlinear inequality constraints with the following ranges: $06 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$.

(5) Pressure vessel design (Huang et al. 2007)

A cylindrical pressure vessel with two hemispherical heads must be designed for minimum fabrication cost. Four variables are identified: thickness of the pressure vessel ($x_1 = T_s$, $1 \leq x_1 \leq 99$); thickness of the head ($x_2 = T_h$, $1 \leq x_2 \leq 99$); inner radius of the vessel ($x_3 = R$, $10 \leq x_3 \leq 200$), and length of the vessel without heads ($x_4 = L$, $10 \leq x_4 \leq 200$). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates. As some EAs use floating point, the values for x_1 and x_2 are truncated to integers. R and L are treated as continuous variables.

Table 14 The Wilcoxon sign rank test results for our framework against other CDE algorithms

Algorithms	CEC'2006		CEC'2010			
			10D		30D	
	R+	R−	R+	R−	R+	R−
GOBL-rank-iMDDE vs. rank-iMDDE	**195**	105	**78**	75	**119**	34
GOBL-εDEag vs. εDEag	**173**	127	**86**	85	**105**	66
GOBL-rank-iMDDE vs. rank-iMDDE	**173.5**	126.5	**150**	21	**144**	9
GOBL-ECHT-DE vs. ECHT-DE	**234.5**	41.5	**153**	18	**149**	22
GOBL-MDDE vs. MDDE	**232**	68	**102**	51	**132.5**	38.5

Table 15 Comparison on the results of constrained engineering benchmark problems

Algorithm	p01(case 1)		p01(case 2)		p02		p03		p04		p05	
	Mean	NFEs	Mean	NFEs	Mean	NFEs	Mean	NFEs	Mean	NFEs	Mean	NFEs
CVI-PSO	2.386976	30,000 (9)	1.725124	25,000 (9)	0.012731	25,000 (9)	**2994.471066**	24,000 (6)	**263.8958434**	7500 (8)	6820.4101	25,000 (6)
MDDE	**2.380956581**	24,580 (8)	1.725	24,000 (8)	0.012666	24,000 (7)	2996.367	24,000 (6)	**263.8958434**	5530 (5)	**6059.702**	24,000 (5)
DELC	**2.38095658**	20,000 (5)	1.724852	20,000 (6)	0.012665267	20,000 (6)	**2994.471066**	30,000 (8)	**263.8958434**	10,000 (9)	**6059.7143**	30,000 (8)
MVDE	**2.38095658**	20,000 (5)	1.7248621	15,000 (3)	0.012667324	10,000 (1)	**2994.471066**	30,000 (8)	**263.8958434**	7000 (6)	6059.997236	15,000 (1)
COMDE	**2.38095658**	24,000 (7)	**1.724852309**	20,000 (6)	0.012667168	24,000 (7)	**2994.471066**	21,000 (5)	**263.8958433**	7000 (6)	**6059.714335**	30,000 (8)
Rank-iMDDE	**2.38095658**	19,830 (4)	**1.724852309**	15,000 (3)	0.012665264	19,565 (4)	**2994.471066**	19,920 (4)	**263.8958434**	4920 (3)	**6059.714335**	23,450 (5)
εDEag	**2.38095658**	14,820 (3)	**1.724852309**	13,718 (2)	**0.012665233**	20,000 (5)	**2994.471066**	15,676 (2)	**263.8958434**	5000 (4)	**6059.714335**	22,980 (4)
GOBL-rank-iMDDE	**2.38095658**	11,130 (2)	**1.724852309**	15,000 (3)	**0.012665233**	18,930 (3)	**2994.471066**	17,880 (3)	**263.8958434**	3330 (1)	**6059.714335**	22,430 (3)
GOBL-εDEag	**2.38095658**	10,634 (1)	**1.724852309**	13,400 (1)	**0.012665233**	18,806 (2)	**2994.471066**	12,314 (1)	**263.8958434**	4340 (2)	**6059.714335**	19,823 (2)

For the sake of space limitation, the formulations of these problems are omitted here. Interested readers can find them in their corresponding literature. Due to different features of different problems, GOBL-CDE adopts different population size and Max_NFEs, while other parameters are kept the same as shown in Sect. 4.1. The population size of GOBL-CDE for different problems is 30, 50, 20, 20, and 50; and Max_NFEs were limited to 20,000, 20,000, 20,000, 5000, and 25,000, respectively.

GOBL-CDE including GOBL-rank-iMDDE and GOBL-εDEag is compared with seven EAs. The seven EAs are: (1) MVDE (De Melo and Carosio 2013), (2) multiple trial vector based DE (MDDE) (Mezura-Montes et al. 2007), (3) DELC (Wang and Li 2010), (4) COMDE (Mohamed and Sabry 2012), (5) CVI-PSO (Mazhoud et al. 2013), (6) rank-iMDDE (Gong et al. 2014), and (7) εDEag (Takahama and Sakai 2009).

The results of these algorithms are shown in Table 15. A value in boldface means that a better (or the best) solution was obtained. Note that the results of CVI-PSO, MDDE, DELC, MVDE, COMDE are directly obtained from their corresponding literature. From the results in Table 15, we can observe that GOBL-rank-iMDDE and GOBL-εDEag obtain the optimal solution in this problem in all 50 runs. Moreover, GOBL-εDEag is able to provide the smallest NFEs in problem p01, p02 and p03 compared with other EAs. GOBL-rank-iMDDE also exhibits smaller NFEs than other EAs.

5 Conclusion

The DE is one simple and effective EA for global optimization; it employs constraint handling techniques to solve the COPs. The GOBL is generalized opposition-based learning, which by comparing the fitness of an individual to its transformation and retaining the fitter one in the population accelerates the search process. The objective of this paper is to introduce new framework of the CDE which employs the GOBL to accelerate the CDE without making premature convergence. The proposed framework named as GOBL-CDE can be applied to the most state-of-the-art EAs, and in GOBL-CDE, all algorithms similarly use the transformed population initialization to achieve fitter initial individuals and their difference is in applying transformed generation jumping. At last, the proposed framework is applied to the rank-iMDDE and εDEag algorithms, and experimental results on benchmark functions show that the GOBL-CDE is an effective approach to enhance the performance of most of the DE variants studied.

Acknowledgments The research of the authors was supported by the National Nature Science Foundation of China (No. 61103037, 61170193, 61370185), Nature Science Foundation of Guangdong Province (No. S2013010011858, 2013010013432), Guangdong Higher School Scientific Innovation Project (No. 2013KJCX0174, 2013KJCX 0178).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Research involving human participants and/or animals This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Ahandani MA, Alavi-Rad H (2012) Opposition-based learning in the shuffled differential evolution algorithm. Soft Comput 16:1303–1337

Alcal-Fdez J, Snchez L, Garcła S (2008) KEEL: a software tool to assess evolutionary algorithms to data mining problems. Soft Comput 13(3):307–318

Alpaydin E (2004) Introduction to machine learning. MIT Press, Cambridge

Al-Qunaieer FS, Tizhoosh HR, Rahnamayan S (2010) Opposition based computing—a survey. Int Jt Conf Neural Netw 2010:1–7

Balamurugan R, Subramanian S (2009) Emission-constrained dynamic economic dispatch using opposition-based self-adaptive differential evolution algorithm. Int Energy J 10:267–277

Bošković B, Brest J, Zamuda A, Greiner S, Žumer V (2011) History mechanism supported differential evolution for chess evaluation function tuning. Soft Comput 15:667–682

Brest J (2009) Constrained real-parameter optimization with ε-self-adaptive differential evolution constraint-handling. Constraint-handling in evolutionary optimization, vol 198. Springer, Berlin, pp 73–93

Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191(11–12):1245–1287

Corder G, Foreman D (2009) Nonparametric statistics for non-statisticians: a step-by-step approach. Wiley, Hoboken

De Melo VV, Carosio GL (2013) Investigating multi-view differential evolution for solving constrained engineering design problems. Expert Syst Appl 40(9):3370–3377

Derrac J, Garcła S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18

Elsayed SM, Sarker RA, Essam DL (2011) Multi-operator based evolutionary algorithms for solving constrained optimization problems. Comput Oper Res 38(12):1877–1896

Gao W, Yen GG, Liu S (2015) A dual-population differential evolution with coevolution for constrained optimization. IEEE Trans Cybern 45(5):1094–1107

Goldberg DE, Samtani M (1986) Engineering optimization via genetic algorithm. In: Proceedings of 9th conference on electronic computation. University of Alabama, pp 471–482

Gong W, Cai Z, Liang D (2014) Engineering optimization by means of an improved constrained differential evolution. Comput Methods Appl Mech Eng 268:884–904

Gong W, Cai Z, Liang D (2015) Adaptive ranking mutation operator based differential evolution for constrained optimization. IEEE Trans Cybern 45(4):716–727

Guo SM, Yang CC, Chang HY et al (2015) Constraint-activated differential evolution for constrained min-max optimization problems: theory and methodology. Expert Syst Appl 42(3):1626–1636

Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1):1–18

Huang F, Wang L, He Q (2007) An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 186(1):340–356

Jia G, Wang Y, Cai Z et al (2013) An improved $(\mu+\lambda)$-constrained differential evolution for constrained optimization. Inf Sci 222:302–322

Karaboga D, Akay B (2011) A modified Artificial Bee Colony (ABC) algorithm for constrained optimization Problems. Appl Soft Comput 11:3021–3031

Liang JJ, Runarsson TP, Mezura-Montes E et al (2006) Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real parameter optimization, Technical Report. Nanyang Technological University, Singapore

Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640

Mallipeddi R, Suganthan PN (2010) Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization, Technical Report. Nanyang Technological University, Singapore

Mallipeddi R, Suganthan PN (2010) Ensemble of constraint handling techniques. IEEE Trans Evol Comput 14(4):561–579

Mazhoud I, Hadj-Hamou K, Bigeon J et al (2013) Particle swarm optimization for solving engineering problems: a new constraint-handling mechanism. Eng Appl Artif Intell 26(4):1263–1273

Mezura-Montes E, Coello CAC, Vel'azquez-Reyes J et al (2007) Multiple trial vectors in differential evolution for engineering design. Eng Optim 39(5):567–589

Mezura-Montes E, Vel'azquez-Reyes J, Coello CAC (2005) Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization. In: Proceedings of the conference on genetic and evolutionary computation, pp 225–232

Michalewicz Z (1995) A survey of constraint handling techniques in evolutionary computation methods. In: Proceedings of the 4th annual conference on evolutionary programming. The MIT Press, Cambridge, pp 135–155

Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. Inf Sci 194:171–208

Omran MGH, Salman A (2009) Constrained optimization using CODEQ. Chaos Solitons Fractals 42(2):662–668

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Secaucus

Rahnamayan S, Tizhoosh HR, Salama MMA (2006) Opposition versus randomness in soft computing techniques. Appl Soft Comput 8:906–918

Rahnamayan S, Tizhoosh HR, Salama MMA (2007) Quasi oppositional differential evolution. In: IEEE congress on evolutionary computation, CEC 2007, pp 2229–2236

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. In: IEEE transactions on evolutionary computation, pp 1264–1279

Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Design 43:303–315

Storn R (1999) System design by constraint adaptation and differential evolution. IEEE Trans Evol Comput 3(1):22–34

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

Subudhi B, Jena D (2009) Nonlinear system identification using opposition based learning differential evolution and neural network techniques. IEEE J Intell Cybern Syst 5:1–13

Sun CL, Zeng JH, Pan JY (2011) An improved vector particle swarm optimization for constrained optimization problems. Inf Sci 181:1153–1163

Takahama T, Sakai S (2006) 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), pp 1–8

Takahama T, Sakai S (2009) Solving difficult constrained optimization problems by the ε-constrained differential evolution with gradient-based mutation. Constraint-handling in evolutionary optimization, vol 198. Springer, Berlin, pp 51–72

Tasgetiren MF, Suganthan PN, Ozcan S et al (2015) A differential evolution algorithm with a variable neighborhood search for constrained function optimization. Adaptation and hybridization in computational intelligence, pp 171–184

Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation, vol 1, Vienna, pp 695–701

Tizhoosh HR (2005) Reinforcement learning based on actions and opposite actions. In: International conference on artificial intelligence and machine learning, Cairo, pp 94–98

Tizhoosh HR (2006) Opposition-based reinforcement learning. J Adv Comput Intell Intell Inform 10(3):578–585

Ventresca M, Tizhoosh HR (2006) Improving the convergence of backpropagation by opposite transfer functions. In: International joint conference on neural networks, pp 4777–4784

Wang H (2012) Opposition-based barebones particle swarm for constrained nonlinear optimization problems. Mathematical Problems in Engineering, pp 1–12

Wang Y, Cai Z, Zhou Y et al (2009) Constrained optimization evolutionary algorithms. J Softw 20(1):11–29

Wang Y, Cai Z (2011) Constrained evolutionary optimization by means of $(\mu+\lambda)$-differential evolution and improved adaptive trade-off model. Evol Comput 19(2):249–285

Wang Y, Cai Z (2012) A dynamic hybrid framework for constrained evolutionary optimization. IEEE Trans Syst Man Cybern Part B Cybern 42(1):203–217

Wang Y, Cai Z (2012) Combining multiobjective optimization with differential evolution to solve constrained optimization problems. IEEE Trans Evol Comput 16(1):117–134

Wang L, Li L-P (2010) An effective differential evolution with level comparison for constrained engineering design. Struct Multidiscip Optim 41:947–963

Wang Y, Wang BC, Li HX et al (2015) Incorporating objective function information into the feasibility rule for constrained evolutionary optimization. IEEE Trans Cybern. doi:10.1109/TCYB.2015.2493239

Wang H, Wu Z, Liu Y et al (2009) Space transformation search: a new evolutionary technique. In: Proceedings of world summit on genetic and evolutionary computation, pp 537–544

Wang H, Wu Z, Rahnamayan S et al (2009) A scalability test for accelerated DE using generalized opposition-based learning. In: Ninth international conference on intelligent systems design and applications, pp 1090–1095

Xu QZ, Wang L, He BM et al (2011) Opposition-based differential evolution using the current optimum for function optimization. J Appl Sci 29(3):308–315

Xu QZ, Wang L, Wang N et al (2014) A review of opposition-based learning from 2005 to 2012. Eng Appl Artif Intell 29:1–12