# Engineering optimization by means of an improved constrained differential evolution

Wenyin Gong*[,a], Zhihua Cai[a], Dingwen Liang[a]

[a]*School of Computer Science,*
*China University of Geosciences, Wuhan 430074, P.R. China*

**Abstract**

To efficiently optimize the constrained engineering problems, in this paper, an improved constrained differential evolution (DE) method is proposed, where two improvements are presented. Firstly, to make the DE algorithm converge faster, a ranking-based mutation operator that is suitable to the constrained optimization problems is presented. Secondly, an improved dynamic diversity mechanism is proposed to maintain either infeasible or feasible solutions in the population. Combining the two improvements with the DE algorithm, the proposal is referred to as rank-iMDDE, for short. To evaluate the performance of rank-iMDDE, 24 benchmark functions presented in CEC'2006 are selected as the test suite. Moreover, five widely used constrained engineering benchmark problems and four constrained mechanical design problems from the literature are chosen to test the capability of rank-iMDDE for the engineering problems. Experimental results indicate that rank-iMDDE is able to improve the performance of DE in terms of the quality of the final solutions, the convergence rate, and the successful rate. Additionally, it can provide fairly-competitive results compared with other state-of-the-art evolutionary algorithms in both benchmark functions and engineering problems.

*Key words:* Engineering optimization, differential evolution, ranking-based mutation, dynamic diversity mechanism, constrained optimization

## 1. Introduction

In the real-world, most of engineering design problems involve inequality and/or equality constraints. For example, the design of the speed reducer is to minimize its weight under constraints on bending stress of the gear teeth, surfaces stress, transverse deflections of the shafts, and stresses in the shafts [1, 2]. Generally, these problems can be treated as the constrained optimization problems (COPs). In the last few decades, the use of evolutionary algorithms (EAs) for the COPs has obtained considerable attention [3, 4, 5, 6, 7].

Among different EAs, differential evolution (DE) is a simple yet efficient algorithm for the numerical optimization [8]. Recently, coupled with the constraint-handling techniques, DE has been used to solve the COPs [9, 10, 2, 11, 12]. For more details, interested readers can refer to two good surveys of DE in [13] and [14], and the references therein.

Although DE has gotten success in diverse fields, it may suffer slowly at exploitation of the solutions [15] due to the randomly selected parents in the mutation operation. For example, in the classical "DE/rand/1" mutation, three parent vectors $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$, and $\mathbf{x}_{r_3}$ are selected randomly from the current population. The indexes $r_1, r_2$, and $r_3$ satisfy $r_1, r_2, r_3 \in \{1, Np\}$ and $r_1 \neq r_2 \neq r_3 \neq i$, where $Np$ is the population size [8]. In the nature, good species always contain more useful information, and hence, they are more likely to be selected to propagate offspring. Inspired by the phenomenon, in this paper, we proposed an improved constrained DE variant for the constrained engineering optimization problems, where a ranking-based mutation operator is presented to accelerate the convergence rate of DE. In addition, to maintain the diversity of the population, an improved dynamic diversity mechanism is proposed,

*Corresponding author. Tel: +86-27-67883716.
*Email addresses:* wenyingong@yahoo.com; wygong@cug.edu.cn (Wenyin Gong), zhcai@cug.edu.cn (Zhihua Cai)

which can maintain either infeasible or feasible solutions in the population. In addition, the multiple trial vectors generation technique proposed in MDDE [2] is also used. The proposed method can be viewed as an improved variant of MDDE [2], therefore, it is referred to as rank-iMDDE, for short.

To evaluate the performance of rank-iMDDE, 24 benchmark functions presented in CEC'2006 [16] are selected as the test suite. Additionally, five widely used constrained engineering benchmark problems and four constrained mechanical design problems are also chosen from the literature to verify the capability of rank-iMDDE for the engineering applications. Experimental results indicate that rank-iMDDE is able to accelerate the convergence rate of MDDE, and it can also provide better results than MDDE with respect to the solution quality and successful rate. Moreover, compared with other state-of-the-art EAs, rank-iMDDE can obtain highly-competitive results in both benchmark functions and engineering problems.

The main contributions of this work are two-fold. Firstly, an improved constrained DE variant (rank-iMDDE) is proposed for the COPs. In rank-iMDDE, the adaptive ranking-based mutation operator and improved dynamic diversity mechanism are proposed. Secondly, the performance of rank-iMDDE is comprehensively evaluated through benchmark functions and engineering problems.

The rest of this paper is organized as follows. In Section 2, the formulation of the COPs is briefly described. Section 3 introduces the DE algorithm and the constrained DE variants in brief. In Section 4, the proposed rank-iMDDE is presented in detail, followed by the experiments and analysis in Section 5. Finally, in Section 6, the paper is concluded and some possible future work is pointed out.

## 2. Problem formulation

Without loss of generality, in this work, we consider the constrained minimization problem, which can be formalized as a pair $(\mathcal{S}, f)$ , where $\mathcal{S} \subseteq \mathbb{R}^n$ is a bounded set on $\mathbb{R}^n$ and $f : \mathcal{S} \to \mathbb{R}$ is an $n$-dimensional real-valued function. The minimization COP can be formulated as

$$\min \ f(\mathbf{x}), \quad \mathbf{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n \tag{1}$$

subject to

$$\begin{cases} g_j(\mathbf{x}) \leq 0, & j = 1, \cdots, q \\ h_j(\mathbf{x}) = 0, & j = q+1, \cdots, m \end{cases} \tag{2}$$

where $\mathbf{x}$ is the vector of solution, $x_i$ is the $i$-th ($i \in \{1, n\}$) decision variable of $\mathbf{x}$, $q$ is the number of inequality constraints, and $m - q$ is the number of equality constraints (in both cases, constraints could be linear or nonlinear). Generally, for each variable $x_i$ it satisfies a constrained boundary

$$l_i \leq x_i \leq u_i, \ i = 1, \cdots, n$$

The feasible region $\mathcal{F} \subseteq \mathcal{S}$ is defined by the $m$ inequality and/or equality constraints. Any point $\mathbf{x} \in \mathcal{F}$ is called a feasible solution; otherwise, it is an infeasible solution. For an inequality constraint which satisfies $g_j(\mathbf{x}) = 0$ ($j \in \{1, \cdots, q\}$) at any point $\mathbf{x} \in \mathcal{F}$, we will say it is *active* at $\mathbf{x}$. Obviously, all the equality constraints are considered active at all points in feasible region $\mathcal{F}$.

In the evolutionary constrained optimization, the equality constraints are always converted into inequality constraints for the COPs as

$$|h_j(\mathbf{x})| - \delta \leq 0 \tag{3}$$

where $j \in \{q+1, \cdots, m\}$ and $\delta$ is a positive tolerance value. The distance of a solution $\mathbf{x}$ from the $j$-th constraint can be constructed as

$$G_j(\mathbf{x}) = \begin{cases} \max\{0, g_j(\mathbf{x})\}, & 1 \leq j \leq q \\ \max\{0, |h_j(\mathbf{x})| - \delta\}, & q+1 \leq j \leq m \end{cases} \tag{4}$$

Then, the distance of the solution $\mathbf{x}$ from the boundaries of the feasible set, which also reflects the degree of its constraint violation, can be denoted as

$$G(\mathbf{x}) = \sum_{j=1}^{m} G_j(\mathbf{x}) \tag{5}$$

## 3. Differential evolution and constrained DEs

### *3.1. Differential evolution*

The DE algorithm [8] is a simple EA for numerical optimization. It creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness value. This is a rather greedy selection scheme that often outperforms traditional EAs. The pseudo-code of the original DE algorithm is shown in Algorithm 1, where $n$ is the number of decision variables, $F$ is the mutation scaling factor, $Cr$ is the probability of crossover operator. rndint$(1, n)$ is a uniformly distributed random integer number between 1 and $n$. rndreal$_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$; it is generated anew for each value of $j$. As for the terminal conditions, we can either fix the maximum number of function evaluations (NFEs) *Max_NFEs* or the precision of a desired solution *VTR* (value to reach).

**Algorithm 1** The DE algorithm with "DE/rand/1/bin"

```
Generate the initial population;
Evaluate the fitness for each individual;
while the halting criterion is not satisfied do
    for i = 1 to Np do
        Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i;
        j_rand = rndint(1, n);
        for j = 1 to n do
            if rndreal_j(0, 1) < Cr or j == j_rand then
                u_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j});
            else
                u_{i,j} = x_{i,j};
            end if
        end for
    end for
    for i = 1 to Np do
        Evaluate the offspring u_i;
        if f(u_i) is better than or equal to f(x_i) then
            Replace x_i with u_i;
        end if
    end for
end while
```

### *3.2. Constrained DE variants*

Combining with the constraint-handling techniques, the DE algorithm has been successfully used for solving the COPs. In this subsection, we will briefly discuss some representative constrained DE (CDE) variants.

The first attempt to apply DE for the COPs is the constraint adaptation with DE (CADE) proposed by Storn [9]. CADE is a multi-member DE that generates more than one ($n_o > 1$) offspring for each individual with the DE operators, and then only one of the $n_o + 1$ individuals (both the $n_o$ offspring and target individual) will be selected for the next generation. Lampinen presented a Pareto dominance-based constraint-handling method to handle nonlinear constraint functions [17]. Becerra and Coello presented a cultured DE for the COPs [18], where the cultural algorithm is applied to use different knowledge sources to influence the variation operator of DE. Mezura-Montes *et al.* proposed a multi-member diversity-based DE (MDDE) for the COPs in [19, 2]. Similar to CADE, in MDDE each target parent is allowed to generate more than one offspring. In the CEC'2006 competition on the constrained real parameter optimization [16], several CDE variants were proposed and some of them secured front ranks. For example, $\varepsilon$DE [10], proposed by Takahama and Sakai, ranks the first in this competition. In $\varepsilon$DE, the $\varepsilon$ constrained method is used to handle the constraints; in addition, a gradient-based mutation is introduced to find feasible point by using the gradient of constraints at an infeasible point [10]. In [20], Mezura-Montes *et al.* presented a modified DE (MDE) for the COPs. In MDE, a modified mutation operator is presented. Additionally, a dynamic diversity mechanism is added into MDE to maintain infeasible solutions located in promising areas of the search space. In [21], Huang *et al.* proposed an extended SaDE method for the COPs. Compared with the original SaDE method [22], the replacement criterion was modified for tackling constraints. Brest *et al.* presented a self-adaptive DE variant to solve the COPs [23], where three DE mutation operators are used and the parameters of $Cr$ and $F$ are self-adaptively updated. Huang *et al.* proposed a co-evolution mechanism based DE for the COPs [24]. In [24], a co-evolution model is presented and DE is used to perform evolutionary search in spaces of both solutions and penalty factors. Zhang *et al.*

proposed a dynamic stochastic ranking-based multi-member DE (DSS-MDE) [25], where the comparison probability $P_f$ decreases dynamically following the evolution process. Ali and Kajee-Bagdadi presented local exploration-based DE for solving the COPs, where a periodic local exploration technique is incorporated into DE [26]. In [27], Mezura-Montes and Palomeque-Ortiz proposed a modified DE for the COPs, where the parameters related to DE and the constraint-handling mechanism are deterministically and self-adaptively controlled. With the aim of providing some insights about the behavior of DE variants for solving the COPs, Mezura-Montes *et al.* presented an empirical study on CDE in [28]. Since no single constraint-handling technique is able to outperform all others on every problem, Mallipeddi and Suganthan proposed an ensemble of constraint handling techniques (ECHT) to solve the COPs [29], in which each constraint-handling technique has its own subpopulation. Wang and Cai proposed a $(\mu+\lambda)$-CDE for the COPs [11]. 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. Recently, Wang and Cai presented the CMODE method [12], in which DE is combined with multiobjective optimization to deal with the COPs. Mohamed and Sabry proposed a novel constrained optimization based on a modified DE algorithm (COMDE) [30], where a new directed mutation strategy is presented. Additionally, a modified constraint-handling technique based on the feasibility and the sum of constraints violations is employed to handle constraints. In [31], Elsayed *et al.* presented an improved DE algorithm (ISAMODE-CMA) that adopts a mix of different DE mutation operators. Moreover, in order to enhance the local search ability of the algorithm, the CMA-ES [32] is periodically applied. In ISAMODE-CMA, the dynamic penalty constraint-handling technique is used to tackle constraints of a problem.

## 4. The proposal: rank-iMDDE

In this section, the proposed rank-iMDDE method is presented in detail. In rank-iMDDE, there are two major improvements. Firstly, a ranking-based mutation operator is presented to accelerate the convergence rate of DE. Secondly, an improved dynamic diversity mechanism is proposed to maintain the diversity of the population. The core idea behind rank-iMDDE is elucidated as follows.

### *4.1. The ranking-based mutation operator*

#### *4.1.1. Adaptive ranking technique*

In the ranking-based mutation operator, the population needs to be ranked first. Suppose that the population is sorted from the best to the worst based on a *criterion*, then the ranking of an individual $\mathbf{x}_i$ is assigned as follows:

$$R_i = Np - i, \quad i = 1, \cdots, Np \tag{6}$$

According to Equation (6), the best individual in the current population will obtain the highest ranking.

To make the ranking-based mutation operator in DE be suitable to the COPs, we modify our previous proposed ranking technique [33], which is only based on the objective function value for unconstrained optimization problems. In this work, when solving the COPs, the population is adaptively ranked according to the situation of the current population as follows:

- **Ranking in the 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, we sort the population according to the constraint violation (*e.g.*, $G(\mathbf{x})$ in Equation (5)) of each individual in ascending order. The objective function values are not considered at all.

- **Ranking in the semi-feasible situation**: As suggested in [11], in the semi-feasible situation, some important feasible individuals (those with small objective function values) and infeasible individuals (those with small objective function values and slight constraint violations) should be obtained more consideration. Therefore, in order to balance the influence of objective function value and constraint violation, *fitness transformation* techniques could be a good choice. As an illustration, in this work, we adopt the adaptive fitness transformation (AFT) method proposed in [11] to calculate the final transformed fitness value $f_{\text{final}}(\mathbf{x}_i)$ of each individual. Afterwards, the population is sorted according to $f_{\text{final}}(\mathbf{x}_i)$ in ascending order. In this way, the individuals that have lower final transformed fitness values will obtain higher rankings based on Equation (6).

In the AFT method, the population is divided into the feasible group ($Z_1$) and the infeasible group ($Z_2$) based on the feasibility of each solution. Thereafter, the objective function value $f(\mathbf{x}_i)$ of the solution $\mathbf{x}_i$ is converted into

$$f'(\mathbf{x}_i) = \begin{cases} f(\mathbf{x}_i), & i \in Z_1 \\ \max\{\varphi \cdot f(\mathbf{x}_{\text{best}}) + (1-\varphi) \cdot f(\mathbf{x}_{\text{worst}}), f(\mathbf{x}_i)\}, & i \in Z_2 \end{cases} \tag{7}$$

where $\varphi$ is the feasibility ratio of the last population, and $\mathbf{x}_{\text{best}}$ and $\mathbf{x}_{\text{worst}}$ 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

$$f_{\text{nor}}(\mathbf{x}_i) = \frac{f'(\mathbf{x}_i) - \min\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j)}{\max\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j) - \min\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j)} \tag{8}$$

If we use Equation (5) to calculate the constraint violation of each solution, the normalized constraint violation can be evaluated as

$$G_{\text{nor}}(\mathbf{x}_i) = \begin{cases} 0, & i \in Z_1 \\ \frac{G(\mathbf{x}_i) - \min\limits_{j \in Z_2} G(\mathbf{x}_j)}{\max\limits_{j \in Z_2} G(\mathbf{x}_j) - \min\limits_{j \in Z_2} G(\mathbf{x}_j)}, & i \in Z_2 \end{cases} \tag{9}$$

Finally, the final fitness function is obtained as follows

$$f_{\text{final}}(\mathbf{x}_i) = f_{\text{nor}}(\mathbf{x}_i) + G_{\text{nor}}(\mathbf{x}_i) \tag{10}$$

- **Ranking in the feasible situation**: In this situation, all individuals in the population are feasible, and the COPs can be viewed as unconstrained optimization problems. Thus, we only need to rank the population according to the objective function value $f(\mathbf{x}_i)$ of each individual in ascending order.

In summary, in the ranking-based mutation operator the current population is adaptively ranked based on the following three criteria:

1) constraint violations in the infeasible situation,

2) transformed fitness values in the semi-feasible situation, and

3) objective function values in the feasible situation.

It is worth mentioning that although this work is the modification of our previous work in [33], however, there are significant differences compared with our previous work: i) The work in [33] is only for unconstrained problems, whereas this work is for constrained problems. ii) The rankings in [33] are only based on the objective function values, while in this work since the constraints should be considered, the ranking are assigned based on different criteria in different situations. And iii) the calculation of selection probabilities is also different from [33]. In this work, different methods are used to calculate the selection probability in different situations in the following subsection.

*4.1.2. Selection probability calculation*

After obtaining the ranking of each individual, we then calculate the selection probability $p_i$ for each individual $\mathbf{x}_i$. Different from the method presented in [33] for unconstrained optimization problems, in this work, the selection probabilities are calculated according to the situation of the current population for the COPs. In different situations, different methods are used to calculate the selection probabilities as follows.

- **Probability in the infeasible situation**: Since all individuals are infeasible in this situation, the individuals with small constraint violations should get more chance to be selected to steer the population towards feasibility. Therefore, in this situation, we calculate the selection probability for each individual as follows:

$$p_i = \begin{cases} 1.0, & 1 \le i < \frac{Np}{3} \\ \frac{R_i}{\frac{2 \cdot Np}{3}}, & \frac{Np}{3} \le i \le Np \end{cases} \tag{11}$$

Similar to ($\mu$+$\lambda$)-CDE [11], in the infeasible situation, the first $Np/3$ individuals with lower constraint violations in the ranked population will always get the selection probabilities with $p_i = 1.0$ ($i = 1, \cdots, Np/3$) to promote feasibility, whereas for the rest $2 \cdot Np/3$ individuals their selection probabilities are linearly decreased.

- **Probability in the semi-feasible situation**: In this situation, the selection probabilities are evaluated as

$$p_i = \left(\frac{R_i}{Np}\right)^{k_1}, \quad i = 1, \cdots, Np \tag{12}$$

where $k_1$ is a user-defined coefficient. In the semi-feasible situation, some important feasible individuals and important infeasible individuals are assigned higher rankings, and these individuals contain more useful information. The important feasible individuals with small objective functions are able to guide the algorithm to find the global optimum. On the other hand, the important infeasible individuals with slight constraint violations and small objective function values can promote the algorithm to find feasible solutions (especially when the proportion of the feasible region is very small) or to obtain the optimum when it is located exactly on the boundaries of the feasible region. Thus, these individuals should be paid more attention and be more dominant than the worst individuals. Based on these considerations, we recommend $k_1$ is greater than 1.0. The reason is that $k_1 > 1.0$ can assign greater selection pressure for individuals with higher rankings than $k_1 \leq 1.0$ [34]. In this work, $k_1 = 2.0$ is set to be a default value, and its influence will be discussed in Section 5.8.

- **Probability in the feasible situation**: The selection probabilities in this situation are calculated as follows

$$p_i = \left(\frac{R_i}{Np}\right)^{k_2}, \quad i = 1, \cdots, Np \tag{13}$$

where $k_2$ is also a user-defined coefficient. As mentioned-above, in the feasible situation, the COPs can be treated as unconstrained optimization problems. In order to maintain the diversity of the population and avoid trapping into the local optima, we suggest to set $k_2 \leq 1.0$. In this manner, better individuals will less dominate the worse ones. In this work, $k_2 = 0.5$ is set to be a default value, and its influence will also be discussed in Section 5.8.

#### *4.1.3. Vector Selection*

As presented in [33], after calculating the selection probability of each individual in the above subsection, the other issue is that in the mutation operator which vectors should be selected according to the selection probabilities. The vector selection used in this work is the same as the method proposed in [33], *i.e.*, only the base vector and the terminal vector are selected based on their selection probabilities. More details can be found in [33].

### *4.2. The proposed rank-iMDDE method*

In the previous subsection, the ranking-based mutation operator is proposed to enhance the exploitation ability of DE when solving the COPs. In this subsection, it is integrated into a CDE variant (iMDDE), which is an improved version of MDDE [19, 2]. Combining the ranking-based mutation with iMDDE, the rank-iMDDE is presented in details as follows.

#### *4.2.1. Improved dynamic diversity mechanism*

In order to maintain either infeasible or feasible individuals in the population, Mezura-Montes *et al.* [19, 2] proposed the static diversity mechanism, where a static selection ratio ($S_r = 0.45$) is used to control the selection process as follows:

- If rndreal$(0, 1) < S_r$, the selection is performed based only on the objective function values, regardless of feasibility;
- otherwise, Deb's feasibility rules [35] are used to compare different individuals.

**Algorithm 2** The rank-iMDDE algorithm for the COPs

```
Generate and evaluate the initial population;
Set t = 1, Cr = 0.9, S_{r_0} = 0.7, n_o = 5, k_1 = 2.0, and k_2 = 0.5;
while the halting criterion is not satisfied do
    Calculate feasibility ratio of the current population;                                   ⇐
    According to the current situation, sort the population based on different criteria;     ⇐
    Calculate the selection probability for each individual as aforementioned;               ⇐
    Update S_r as shown in Equation (14);                                                    ⇐
    for i = 1 to Np do
        F = rndreal(0.3, 0.9);
        for k = 1 to n_o do
            Select r_1, r_2, r_3 based on the selection probabilities;                       ⇐
            if rndreal(0, 1) < 1.0/n_o then
                Generate the offspring c with "DE/rand/1/exp";                               ⇐
            else
                Generate the offspring c with "DE/rand/1/bin";
            end if
            Deal with the violated variables in c based on the boundary-handling technique as shown in Equation (15).
            Evaluate the offspring c;
            if k > 1 then
                if c is better than u_i based on Deb's feasibility rules then
                    u_i ← c;
                end if
            else
                u_i ← c;
            end if
        end for
    end for
    for i = 1 to Np do
        if rndreal(0, 1) < S_r then
            if f(u_i) ≤ f(x_i) then
                x_i ← u_i;
            end if
        else
            if u_i is better than x_i based on Deb's feasibility rules then
                x_i ← u_i;
            end if
        end if
    end for
    t ← t + 1;
end while
```

Recent studies indicate that the dynamic control of $S_r$ is able to balance the search between feasible and infeasible regions, and hence, it can improve the performance of static diversity mechanism [20, 25, 27] for the COPs. Therefore, inspired by the methods proposed in [20] and [25] we present a dynamic control method for $S_r$ as

$$S_r = \begin{cases} S_{r_0}, & t = 1 \\ S_{r_0}\left(1 - \frac{t}{2t_{\max}/3}\right), & 2 \le t \le \frac{2t_{\max}}{3} \\ 0.025, & \frac{2t_{\max}}{3} < t \le t_{\max} \end{cases} \tag{14}$$

where $t$ is the current generation number, $S_{r_0} = 0.70$ is the initial value of the selection ratio $S_r$, and $t_{\max}$ is the maximal generation number. In this way, when $t \le \frac{2t_{\max}}{3}$, the individuals with small fitness objective values will obtain more chance to be selected into the next population due to the higher $S_r$ value. While in the last third part of the process $S_r$ is set to be 0.025, which means that the feasibility of individuals will be more dominant during this period.

*4.2.2. Boundary-handling method*

After using the DE mutation operator to generate the mutant vector $\mathbf{v}_i$, some components $v_{i,j}$ ($i = 1, \cdots, Np$ and $j = 1, \cdots, n$) may violate the boundary constraint, *i.e.* $v_{i,j} \notin [l_j, u_j]$. In this situation, we should make these components be within their corresponding boundary. As mentioned in [36], the boundary-handling method has significant influence to the performance of DE. In this work, for the iMDDE method we use the *reinitialization* method (see Equation (15)), *i.e.*, when one of the decision variable violates its boundary constraint, it is generated with the uniform distribution within the boundary [36] as follows:

$$v_{i,j} = \text{rndreal}(l_j, u_j), \quad \text{if } v_{i,j} \notin [l_j, u_j] \tag{15}$$

*4.3. Framework of rank-iMDDE*

By integrating the ranking-based mutation into iMDDE, rank-iMDDE for the COPs is presented. The pseudo-code of rank-iMDDE is described in Algorithm 2, where $n_o > 1$ is number of offspring generated by each target individual. The major differences between MDDE [19, 2] and rank-iMDDE are highlighted in "⇐" in Algorithm 2. Specifically, there are three major differences between rank-iMDDE and MDDE.

- Lines 4, 5, 6, and 11 are responsible for choosing $r_1, r_2, r_3$ based on the ranking-based mutation operator.
- In line 7, the dynamic update of $S_r$ is performed at each generation.
- In lines 12 to 16, both "DE/rand/1/exp" and "DE/rand/1/bin" are used to generate the $n_o$ offspring. The reason is that the combination of "DE/rand/1/exp" and "DE/rand/1/bin" for generating the offspring is able to enhance the performance of DE for the COPs [37]. In addition, the probability to use "DE/rand/1/exp" is $1.0/n_o$. This means that in expectation there is only one out of $n_o$ offspring that is generated by "DE/rand/1/exp", while other $n_0 - 1$ offspring are generated by "DE/rand/1/bin".

Note that the last two aspects are the differences between iMDDE and MDDE.

The total complexity of MDDE is $O(t_{\max} \cdot Np \cdot n_o \cdot n)$, where $t_{\max}$ is the maximal generation number. Since $n_o \ll Np$, the total complexity of MDDE is $O(t_{\max} \cdot Np \cdot n)$. Compared with MDDE, rank-iMDDE does not significantly increase the overall complexity. The additional complexity of the proposed rank-iMDDE is population sorting and probability calculation, as shown in Algorithm 2. The complexity of population sorting is $O(Np \cdot \log(Np))$, and the complexity of probability calculation is $O(Np)$. Thus, rank-iMDDE has the total complexity of $O(t_{\max} \cdot Np \cdot (n_o \cdot n + \log(Np) + 1))$. In general, the population size $Np$ is set to be proportional to the problem dimension $n$ in the DE literature [38]. Therefore, the total complexity of both MDDE and rank-iMDDE is $O(t_{\max} \cdot n^2)$, which is the same as the original DE algorithm and many other DE variants.

## 5. Experimental results and analysis

In this section, comprehensive experiments are performed through both benchmark functions and engineering problems to evaluate the performance of rank-iMDDE.

*5.1. Benchmark functions*

In this work, the benchmark functions presented in CEC'2006 [16] for the competition on constrained single objective optimization are selected as the test suite. This test suite contains 24 COPs, which are described in Table 1, where $n$ is the number of decision variables, $\rho = |\mathcal{F}|/|\mathcal{S}|$ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints, and NE is the number of nonlinear equality constraints. $a$ is the number of active constraints at $\mathbf{x}$. More details for these functions can be found in [16].

*5.2. Parameter settings*

For rank-iMDDE (also including other iMDDE variants which will be discussed in Sections 5.6 and 5.7), in all experiments, we use the following parameters unless a change is mentioned.

- population size: $Np = 90$ [19, 2];
- crossover rate: $Cr = 0.90$ [19, 2, 25];
- scaling factor: $F = \text{rndreal}(0.30, 0.90)$ [19, 2, 25];
- number of offspring generate by each target individual: $n_o = 5$ [19, 2, 25];
- tolerance of equality: $\delta = 1e-4$ [16, 19, 2, 25];
- initial value of the selection ratio: $S_{r_0} = 0.70$;

Table 1: Details of 24 benchmark test functions.

| Prob | $n$ | Type of function | $\rho$ | LI | NI | LE | NE | $a$ | $f(\mathbf{x}^*)$ |
|---|---|---|---|---|---|---|---|---|---|
| g01 | 13 | quadratic | 0.01% | 9 | 0 | 0 | 0 | 6 | -15.0000000000 |
| g02 | 20 | nonlinear | 100.00% | 0 | 2 | 0 | 0 | 1 | -0.8036191042 |
| g03 | 10 | polynomial | 0.00% | 0 | 0 | 0 | 1 | 1 | -1.0005001000 |
| g04 | 5 | quadratic | 52.12% | 0 | 6 | 0 | 0 | 2 | -30665.5386717834 |
| g05 | 4 | cubic | 0.00% | 2 | 0 | 0 | 3 | 3 | 5126.4967140071 |
| g06 | 2 | cubic | 0.01% | 0 | 2 | 0 | 0 | 2 | -6961.8138755802 |
| g07 | 10 | quadratic | 0.00% | 3 | 5 | 0 | 0 | 6 | 24.3062090681 |
| g08 | 2 | nonlinear | 0.86% | 0 | 2 | 0 | 0 | 0 | -0.0958250415 |
| g09 | 7 | polynomial | 0.51% | 0 | 4 | 0 | 0 | 2 | 680.6300573745 |
| g10 | 8 | linear | 0.00% | 3 | 3 | 0 | 0 | 6 | 7049.2480205286 |
| g11 | 2 | quadratic | 0.00% | 0 | 0 | 0 | 1 | 1 | 0.7499000000 |
| g12 | 3 | quadratic | 4.77% | 0 | 1 | 0 | 0 | 0 | -1.0000000000 |
| g13 | 5 | nonlinear | 0.00% | 0 | 0 | 0 | 3 | 3 | 0.0539415140 |
| g14 | 10 | nonlinear | 0.00% | 0 | 0 | 3 | 0 | 3 | -47.7648884595 |
| g15 | 3 | quadratic | 0.00% | 0 | 0 | 1 | 1 | 2 | 961.7150222899 |
| g16 | 5 | nonlinear | 0.02% | 4 | 34 | 0 | 0 | 4 | -1.9051552586 |
| g17 | 6 | nonlinear | 0.00% | 0 | 0 | 0 | 4 | 4 | 8853.5396748064 |
| g18 | 9 | quadratic | 0.00% | 0 | 13 | 0 | 0 | 6 | -0.8660254038 |
| g19 | 15 | nonlinear | 33.48% | 0 | 5 | 0 | 0 | 0 | 32.6555929502 |
| g20 | 24 | linear | 0.00% | 0 | 6 | 2 | 12 | 16 | 0.2049794002 |
| g21 | 7 | linear | 0.00% | 0 | 1 | 0 | 5 | 6 | 193.7245100700 |
| g22 | 22 | linear | 0.00% | 0 | 1 | 8 | 11 | 19 | 236.4309755040 |
| g23 | 9 | linear | 0.00% | 0 | 2 | 3 | 1 | 6 | -400.0551000000 |
| g24 | 2 | linear | 79.66% | 0 | 2 | 0 | 0 | 2 | -5.5080132716 |

- two coefficients in the ranking-based mutation: $k_1 = 2.0$ and $k_2 = 0.50$ (their influence will be discussed in Section 5.8).

The maximal number of function evaluations (Max_NFEs) for all benchmark problems are set to be $240,000$ [31]. To compare the results of different algorithms, each function is optimized over 100 independent runs. We use the same set of initial random populations to evaluate different algorithms in a similar way done in [15], *i.e.*, all of the compared algorithms are started from the same initial population in each out of 100 runs.

### *5.3. Performance criteria*

To compare the results among different algorithms, in this work, the following performance criteria are used, which have been presented in other literature.

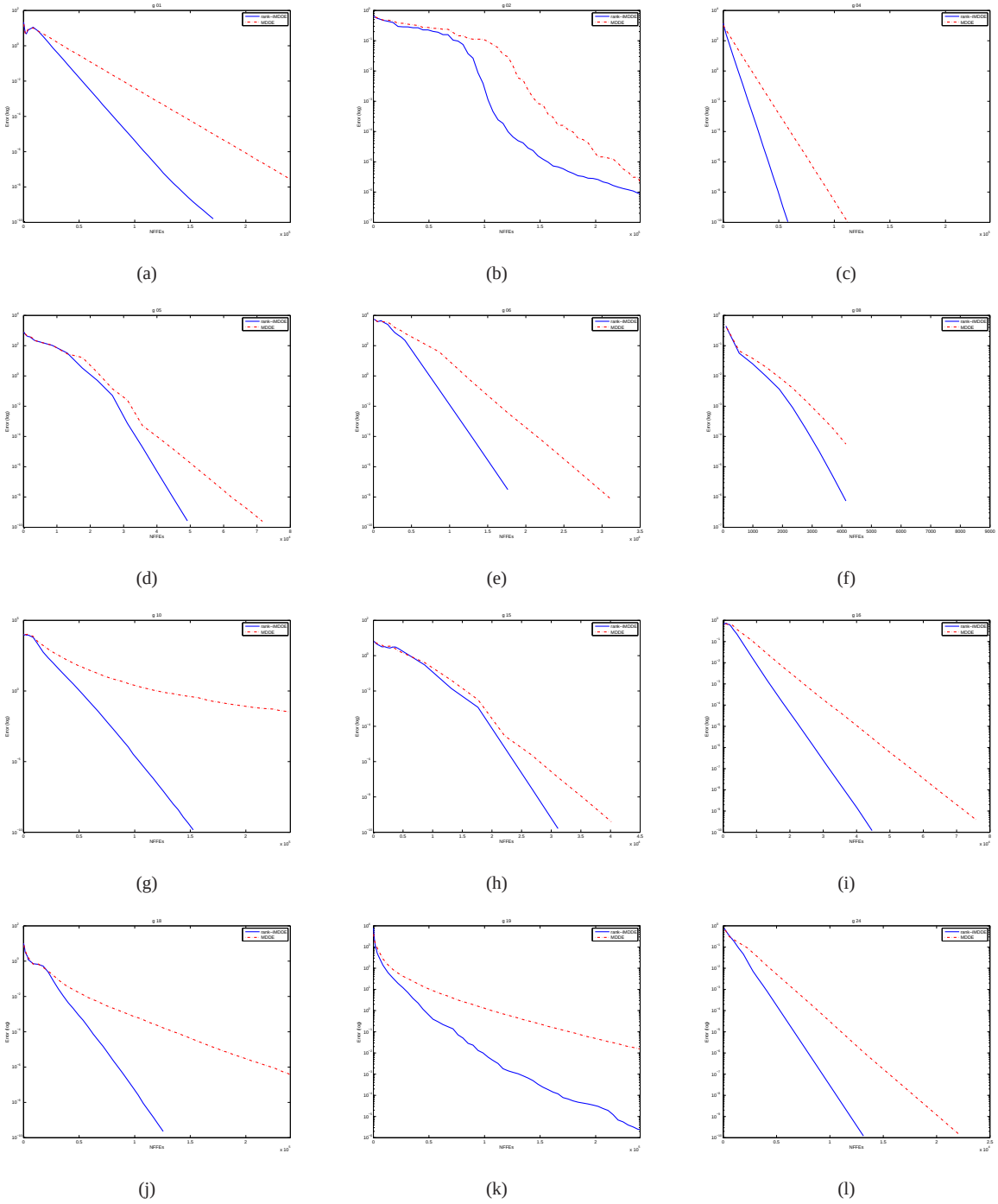
- **NFEs** [16]: It is used to record the number of function evaluations in each run for finding a solution satisfying $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 1e-4$ and $\mathbf{x}$ is feasible, where $\mathbf{x}^*$ is the known-optimal solution of a specific problem.
- **Success rate** ($SR$) [16]: It is equal to the number of success runs over total runs. A success run means that within Max_NFEs the algorithm finds a feasible solution $\mathbf{x}$ satisfying $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 1e-4$.
- **Convergence graphs** [16]: The graphs show the median error performance ($f(\mathbf{x}) - f(\mathbf{x}^*)$) of the total runs.
- **Acceleration rate** ($AR$): Similar to the acceleration rate presented in [39], this criterion is used to compare the convergence speed between two algorithms. It is defined as follows:

$$AR = \frac{\text{ANFEs}_\text{A}/SR_\text{A}}{\text{ANFEs}_\text{B}/SR_\text{B}} \tag{16}$$

  where $\text{ANFEs}_\text{A}$ and $SR_\text{A}$ are respectively the average NFEs and $SR$ values of algorithm A[1]. $AR > 1$ indicates algorithm B converges faster than algorithm A.

In addition, the objective function value $f(\mathbf{x})$ of the final solution in each run is saved, and its best, median, worst, mean, and standard deviation values are also recorded.

[1]Indeed, $\text{ANFEs}_\text{A}/SR_\text{A}$ is the successful performance ($SP$) of algorithm A as presented in [28]. It can be used to measure the speed and reliability of an algorithm.




(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 1: Convergence graphs of rank-iMDDE and MDDE for the selected functions. (a) g01; (b) g02; (c) - (e): g04 - g06; (f) g08; (g) g10; (h): g15; (i): g16; (j) g18; (k) g19; (l) g24.

Table 2: Statistical results obtained by rank-iMDDE and MDDE for all benchmark function, where "NF" means no feasible solution is found.

| Prob | rank-iMDDE | | | | | | MDDE | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Median | Worst | Mean | Std | $SR$ | Best | Median | Worst | Mean | Std | $SR$ |
| g01 | **-15** | **-15** | **-15** | **-15** | 0.00E+0 | 1.00 | -14.99999999 | -14.99999997 | -14.9999999 | -14.99999997 | 1.02E-8 | 1.00 |
| g02 | **-0.80361905** | **-0.80361882** | -0.771749392 | -0.802021189 | 4.57E-3 | 0.86 | -0.803618401 | -0.803614222 | **-0.7926051** | **-0.8025171** | 3.31E-3 | **0.90** |
| g03 | **-1.0005001** | **-1.0005001** | **-1.0005001** | **-1.0005001** | 0.00E+0 | 1.00 | **1.0005001** | **1.0005001** | **1.0005001** | **1.0005001** | 9.66E-14 | 1.00 |
| g04 | **-30665.5387** | **-30665.5387** | **-30665.5387** | **-30665.53867** | 0.00E+0 | 1.00 | **-30665.5387** | **-30665.539** | **-30665.539** | **-30665.539** | 0.00E+0 | 1.00 |
| g05 | **5126.496714** | **5126.496714** | **5126.496714** | **5126.496714** | 0.00E+0 | 1.00 | **5126.496714** | **5126.49671** | **5126.49671** | **5126.49671** | 0.00E+0 | 1.00 |
| g06 | **-6961.81388** | **-6961.81388** | **-6961.81388** | **-6961.813876** | 0.00E+0 | 1.00 | **-6961.81388** | **-6961.8139** | **-6961.8139** | **-6961.8139** | 0.00E+0 | 1.00 |
| g07 | **24.3062091** | **24.3062091** | **24.3062091** | **24.3062091** | 0.00E+0 | 1.00 | **24.3062091** | 24.3062092 | 24.3062100 | 24.3062092 | 1.34E-7 | 1.00 |
| g08 | **-0.09582504** | **-0.09582504** | **-0.09582504** | **-0.095825041** | 0.00E+0 | 1.00 | **-0.09582504** | **-0.095825** | **-0.095825** | **-0.095825** | 0.00E+0 | 1.00 |
| g09 | **680.6300574** | **680.6300574** | **680.6300574** | **680.6300574** | 0.00E+0 | 1.00 | **680.6300574** | **680.630057** | **680.630057** | **680.630057** | 0.00E+0 | 1.00 |
| g10 | **7049.248021** | **7049.248021** | **7049.248021** | **7049.248021** | 0.00E+0 | **1.00** | 7049.248107 | 7049.259608 | 7049.719147 | 7049.280508 | 5.99E-2 | 0.01 |
| g11 | **0.7499** | **0.7499** | **0.7499** | **0.7499** | 0.00E+0 | 1.00 | **0.7499** | **0.7499** | **0.7499** | **0.7499** | 0.00E+0 | 1.00 |
| g12 | **-1** | **-1** | **-1** | **-1** | 0.00E+0 | 1.00 | **-1** | **-1** | **-1** | **-1** | 0.00E+0 | 1.00 |
| g13 | **0.053941514** | **0.053941514** | **0.053941514** | **0.053941514** | 0.00E+0 | **1.00** | **0.053941514** | **0.05394151** | 0.438802615 | 0.065487347 | 6.60E-2 | 0.97 |
| g14 | **-47.7648885** | **-47.7648885** | **-47.7648885** | **-47.76488846** | 0.00E+0 | **1.00** | NF | NF | NF | NF | NF | 0.00 |
| g15 | **961.7150223** | **961.7150223** | **961.7150223** | **961.7150223** | 0.00E+0 | 1.00 | **961.7150223** | **961.715022** | **961.715022** | **961.715022** | 0.00E+0 | 1.00 |
| g16 | **-1.90515526** | **-1.90515526** | **-1.90515526** | **-1.905155259** | 0.00E+0 | 1.00 | **-1.90515526** | **-1.9051553** | **-1.9051553** | **-1.9051553** | 0.00E+0 | 1.00 |
| g17 | **8853.539675** | **8853.539675** | **8853.539675** | **8853.539675** | 0.00E+0 | **1.00** | **8853.539675** | **8853.53967** | 8854.175004 | 8853.550185 | 7.56E-2 | 0.98 |
| g18 | **-0.8660254** | **-0.8660254** | **-0.8660254** | **-0.866025404** | 0.00E+0 | 1.00 | **-0.8660254** | -0.8660251 | -0.8660210 | -0.8660249 | 6.69E-7 | 1.00 |
| g19 | **32.65559313** | **32.65559552** | **32.65640159** | **32.65561099** | 8.30E-5 | **0.97** | 32.66122743 | 32.67016614 | 32.68969114 | 32.67088106 | 5.70E-3 | 0.00 |
| g20 | NF | NF | NF | NF | NF | 0.00 | NF | NF | NF | NF | NF | 0.00 |
| g21 | **193.7245101** | **193.7245101** | **193.7245101** | **193.7245101** | 0.00E+0 | 1.00 | **193.7245101** | 193.7245102 | 193.7245205 | 193.7245106 | 1.53E-6 | 1.00 |
| g22 | NF | NF | NF | NF | NF | 0.00 | NF | NF | NF | NF | NF | 0.00 |
| g23 | **-400.0551** | **-400.0549541** | **-378.0715141** | **-398.1808652** | 4.51E+0 | **0.48** | NF | NF | NF | NF | NF | 0.00 |
| g24 | **-5.50801327** | **-5.50801327** | **-5.50801327** | **-5.508013272** | 0.00E+0 | 1.00 | **-5.50801327** | **-5.5080133** | **-5.5080133** | **-5.5080133** | 0.00E+0 | 1.00 |
| avg | | | – | | | **0.89** | | | – | | | 0.74 |

### *5.4. General Performance of rank-iMDDE*

Because rank-iMDDE is an improved variant of MDDE [2], in this section, the general performance of rank-iMDDE and MDDE is compared through the benchmark functions presented in CEC'2006 [16]. For the two methods, the parameters used are described in Section 5.2. The two approaches are performed over 100 independent runs for each benchmark function with the Max_NFEs=240, 000. The results of the final solutions are shown in Table 2, 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. 1.

From the results shown in Table 2 and Fig. 1, it can be seen that:

- In 19 out of 24 functions, rank-iMDDE consistently obtains the optimal solutions over all 100 runs, whereas MDDE only gets the optimal solutions over all runs in 11 functions.
- Considering the mean values, rank-iMDDE is able to provide better results than MDDE in 10 functions. In 11 functions, both rank-iMDDE and MDDE get the same mean values. Only in function g02, MDDE obtains better mean values than rank-iMDDE.
- With respect to the successful rate, rank-iMDDE gets higher $SR$ values than MDDE in 6 functions. Especially, in 4 functions (g10, g14, g19, and g23), rank-iMDDE extremely improves the successful rate of MDDE. The overall $SR$ value of rank-iMDDE is 0.89, which is higher than that of MDDE (0.74).
- For the convergence rate, Fig. 1 clearly indicates that rank-iMDDE converges faster than MDDE.
- In two functions (g20 and g22), both rank-iMDDE and MDDE can not obtain a feasible solution over all 100 runs. Therefore, in the following experiments, the results of these two functions are not reported.

To summarize, the results confirm our expectation that the proposed rank-iMDDE improves the performance of MDDE in terms of the quality of the final functions, the convergence rate, and the successful rate in the majority of benchmark functions.

### *5.5. Compared with other state-of-the-art EAs*

In the previous subsection, the efficacy of rank-iMDDE is verified through the benchmark functions. In this section, rank-iMDDE is compared with other state-of-the-art EAs for the COPs. These algorithms are ISAMODE-CMA [31], SAMODE [40], ECHT-EP2 [29], ATMES [41], and SMES [42]. SAMODE [40] is a multiple search operators based DE, where different operators are selected adaptively. ISAMODE-CMA [31] is an improved version

Table 3: Compared the quality of final solutions of our approach with other state-of-the-art EAs for all benchmark function.

| Prob | Criteria | rank-iMDDE | ISAMODE-CMA [31] | SAMODE [40] | ECHT-EP2 [29] | ATMES [41] | SMES [42] |
|---|---|---|---|---|---|---|---|
| g01 | Best | **-15** | **-15** | **-15** | **-15** | **-15** | **-15** |
| | Mean | **-15** | **-15** | **-15** | **-15** | **-15** | **-15** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.60E-14 | 0.00E+00 |
| g02 | Best | **-0.80361905** | **-0.8036191** | **-0.803619** | **-0.803619** | -0.803339 | **-0.803601** |
| | Mean | **-0.80202119** | -0.79244 | -0.7987352 | **-0.799822** | -0.790148 | -0.785238 |
| | Std | 4.57E-03 | 2.80E-02 | 8.80E-03 | 1.26E-02 | 1.30E-02 | 1.67E-02 |
| g03 | Best | **-1.0005001** | **-1.0005** | **-1.0005** | **-1.0005** | **-1** | **-1** |
| | Mean | **-1.0005001** | **-1.0005** | **-1.0005** | **-1.0005** | **-1** | **-1** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 5.90E-05 | 2.09E-05 |
| g04 | Best | **-30665.539** | **-30665.539** | **-30665.54** | **-30665.54** | **-30665.54** | **-30665.539** |
| | Mean | **-30665.539** | **-30665.539** | **-30665.54** | **-30665.54** | **-30665.54** | **-30665.539** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 7.40E-12 | 0.00E+00 |
| g05 | Best | **5126.496714** | **5126.497** | **5126.497** | **5126.497** | **5126.498** | 5126.599 |
| | Mean | **5126.496714** | **5126.497** | **5126.497** | **5126.497** | **5127.648** | 5174.492 |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.80E+00 | 5.01E+01 |
| g06 | Best | **-6961.81388** | **-6961.813875** | **-6961.814** | **-6961.814** | **-6961.814** | **-6961.814** |
| | Mean | **-6961.81388** | **-6961.813875** | **-6961.814** | **-6961.814** | **-6961.814** | **-6961.284** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 4.60E-12 | 1.85E+00 |
| g07 | Best | **24.30620907** | **24.3062** | **24.3062** | **24.3062** | **24.306** | 24.327 |
| | Mean | **24.30620907** | **24.3062** | 24.3096 | **24.3063** | 24.316 | 24.475 |
| | Std | 0.00E+00 | 0.00E+00 | 1.59E-03 | 3.19E-05 | 1.10E-02 | 1.32E-01 |
| g08 | Best | **-0.09582504** | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** |
| | Mean | **-0.09582504** | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.61E-08 | 2.80E-17 | 0.00E+00 |
| g09 | Best | **680.6300574** | **680.63** | **680.63** | **680.63** | **680.63** | **680.632** |
| | Mean | **680.6300574** | **680.63** | **680.63** | **680.63** | **680.639** | 680.643 |
| | Std | 0.00E+00 | 0.00E+00 | 1.16E-05 | 0.00E+00 | 1.00E-02 | 1.55E-02 |
| g10 | Best | **7049.248021** | **7049.24802** | **7049.248** | **7049.2483** | 7052.253 | 7051.903 |
| | Mean | **7049.248021** | **7049.24802** | 7059.81345 | **7049.249** | 7250.437 | 7253.047 |
| | Std | 0.00E+00 | 5.42E-06 | 7.86E+00 | 6.60E-04 | 1.20E+02 | 1.36E+02 |
| g11 | Best | **0.7499** | **0.7499** | **0.7499** | **0.7499** | **0.75** | **0.75** |
| | Mean | **0.7499** | **0.7499** | **0.7499** | **0.7499** | **0.75** | **0.75** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 3.40E-04 | 1.52E-04 |
| g12 | Best | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| | Mean | **-1** | **-1** | **-1** | **-1** | **-1** | **-1** |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.00E-03 | 0.00E+00 |
| g13 | Best | **0.053941514** | **0.053942** | **0.053942** | **0.053942** | **0.05395** | 0.053986 |
| | Mean | **0.053941514** | **0.053942** | **0.053942** | **0.053942** | **0.053959** | 0.166385 |
| | Std | 0.00E+00 | 0.00E+00 | 1.75E-08 | 1.00E-12 | 1.30E-05 | 1.77E-01 |
| g14 | Best | **-47.7648885** | **-47.764888** | **-47.76489** | **-47.7649** | NA | NA |
| | Mean | **-47.7648885** | **-47.764888** | -47.68115 | **-47.7648** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 4.04E-02 | 2.72E-05 | NA | NA |
| g15 | Best | **961.7150223** | **961.71502** | **961.71502** | **961.71502** | NA | NA |
| | Mean | **961.7150223** | **961.71502** | **961.71502** | **961.71502** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.01E-13 | NA | NA |
| g16 | Best | **-1.90515526** | **-1.905155** | **-1.905155** | **-1.905155** | NA | NA |
| | Mean | **-1.90515526** | **-1.905155** | **-1.905155** | **-1.905155** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.12E-10 | NA | NA |
| g17 | Best | **8853.539675** | **8853.5397** | **8853.5397** | **8853.5397** | NA | NA |
| | Mean | **8853.539675** | **8853.5397** | **8853.5397** | **8853.5397** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 1.15E-05 | 2.13E-08 | NA | NA |
| g18 | Best | **-0.8660254** | **-0.866025** | **-0.866025** | **-0.866025** | NA | NA |
| | Mean | **-0.8660254** | **-0.866025** | **-0.866024** | **-0.866025** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 7.04E-07 | 1.00E-09 | NA | NA |
| g19 | Best | **32.65559313** | **32.655593** | **32.655593** | **32.6591** | NA | NA |
| | Mean | **32.65561099** | **32.655593** | 32.75734 | 32.6623 | NA | NA |
| | Std | 8.30E-05 | 6.46E-07 | 6.15E-02 | 3.40E-03 | NA | NA |
| g21 | Best | **193.7245101** | **193.72451** | **193.72451** | **193.7246** | NA | NA |
| | Mean | **193.7245101** | **193.72451** | 193.771375 | **193.7438** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 1.96E-02 | 1.65E-02 | NA | NA |
| g23 | Best | **-400.0551** | **-400.0551** | -396.16573 | **-398.9731** | NA | NA |
| | Mean | **-398.180865** | **-395.62403** | -360.81766 | -373.2178 | NA | NA |
| | Std | 4.51E+00 | 7.79E+00 | 1.96E+01 | 3.37E+01 | NA | NA |
| g24 | Best | **-5.50801327** | **-5.508013** | **-5.508013** | **-5.508013** | NA | NA |
| | Mean | **-5.50801327** | **-5.508013** | **-5.508013** | **-5.508013** | NA | NA |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | NA | NA |

Table 4: Average rankings of rank-iMDDE, ISAMODE-CMA, SAMODE, and ECHT-EP2 by the Friedman test for the 22 functions in terms of the mean values.

| Algorithm | Ranking |
|---|---|
| rank-iMDDE | **1.7500** |
| ISAMODE-CMA | *2.4091* |
| SAMODE | 3.1818 |
| ECHT-EP2 | 2.6591 |

Table 5: Influence of ranking-based mutation and dynamic diversity mechanism to the performance of our approach. The NFEs, *SR*, and *AR* results are reported herein. All results are averaged over 100 independent runs.

| Prob | MDDE-b/e (1) | | | iMDDE (2) | | | rank-MDDE-b/e (3) | | | rank-iMDDE (4) | | | *AR* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Std | *SR* | Mean | Std | *SR* | Mean | Std | *SR* | Mean | Std | *SR* | 4 vs 1 | 4 vs 2 | 4 vs 3 |
| g01 | 142,897.5 | 3,426.1 | 1.00 | 153,931.5 | 3,356.3 | 1.00 | **76,072.5** | **2,303.8** | 1.00 | **80,482.5** | **2,471.7** | 1.00 | 1.78 | 1.91 | 0.95 |
| g02 | 178,325.0 | 15,140.2 | **0.90** | 181,472.8 | 16,819.4 | **0.96** | **114,165.0** | **13,167.3** | 0.86 | **118,733.0** | **15,049.9** | 0.86 | 1.44 | 1.37 | 0.96 |
| g03 | 106,393.5 | 7,233.4 | 1.00 | 92,664.0 | 6,681.6 | 1.00 | **63,045.0** | **5,101.8** | 1.00 | **49,572.0** | **3,948.4** | 1.00 | 2.15 | 1.87 | 1.27 |
| g04 | 60,507.0 | 2,014.1 | 1.00 | 61,060.5 | 2,105.3 | 1.00 | **30,415.5** | **913.3** | 1.00 | **31,648.5** | **771.9** | 1.00 | 1.91 | 1.93 | 0.96 |
| g05 | 40,279.5 | 1,570.2 | 1.00 | 55,570.5 | 1,561.9 | 1.00 | **26,986.5** | **1,102.0** | 1.00 | **33,615.0** | **943.3** | 1.00 | 1.20 | 1.65 | 0.80 |
| g06 | 21,348.0 | 953.2 | 1.00 | 21,658.5 | 872.9 | 1.00 | **12,541.5** | **620.3** | 1.00 | **12,942.0** | **475.7** | 1.00 | 1.65 | 1.67 | 0.97 |
| g07 | 137,128.5 | 5,907.7 | 1.00 | 136,926.0 | 5,501.6 | 1.00 | **60,979.5** | **2,679.2** | 1.00 | **62,275.5** | **2,618.2** | 1.00 | 2.20 | 2.20 | 0.98 |
| g08 | 3,906.0 | 521.5 | 1.00 | 4,630.5 | 700.9 | 1.00 | **2,740.5** | **383.2** | 1.00 | **2,961.0** | **377.2** | 1.00 | 1.32 | 1.56 | 0.93 |
| g09 | 46,332.0 | 1,856.1 | 1.00 | 46,638.0 | 1,636.1 | 1.00 | **25,312.5** | **1,162.6** | 1.00 | **24,849.0** | **1,039.2** | 1.00 | 1.86 | 1.88 | 1.02 |
| g10 | 230,040.0 | – | 0.01 | 224,012.5 | 11,685.5 | **0.71** | **89,172.0** | **3,567.4** | **1.00** | **92,718.0** | **3,157.3** | **1.00** | 248.11 | 3.40 | 0.96 |
| g11 | 8,779.5 | 1,714.7 | 1.00 | 12,915.0 | 3,746.7 | 1.00 | **6,822.0** | **2,633.2** | 1.00 | **7,339.5** | **1,304.4** | 1.00 | 1.20 | 1.76 | 0.93 |
| g12 | 4,149.0 | 1,216.9 | 1.00 | 3,631.5 | 890.1 | 1.00 | **3,577.5** | **799.8** | 1.00 | **3,100.5** | **615.8** | 1.00 | 1.34 | 1.17 | 1.15 |
| g13 | 62,598.2 | 13,174.5 | **0.97** | **58,702.5** | **9,078.6** | **1.00** | 95,962.9 | 61,334.1 | 0.59 | **38,988.0** | **5,428.8** | **1.00** | 1.66 | 1.51 | 4.17 |
| g14 | – | – | 0.00 | 206,514.0 | 7,190.1 | **1.00** | **120,514.5** | **12,561.4** | **1.00** | **127,552.5** | **4,964.0** | **1.00** | – | 1.62 | 0.94 |
| g15 | 20,061.0 | 1,261.1 | 1.00 | 30,505.5 | 1,543.1 | 1.00 | **15,043.5** | **1,271.0** | 1.00 | **19,066.5** | **955.2** | 1.00 | 1.05 | 1.60 | 0.79 |
| g16 | 32,004.0 | 1,581.9 | 1.00 | 33,961.5 | 1,411.3 | 1.00 | **16,843.5** | **857.8** | 1.00 | **18,526.5** | **882.7** | 1.00 | 1.73 | 1.83 | 0.91 |
| g17 | 98,107.3 | 28,906.9 | **0.98** | **85,684.5** | **4,165.4** | **1.00** | 150,500.9 | 50,916.9 | 0.69 | **64,539.0** | **19,854.8** | **1.00** | 1.55 | 1.33 | 3.38 |
| g18 | 133,056.0 | 12,509.5 | **1.00** | 137,621.8 | 7,078.9 | **0.99** | **51,299.1** | **3,610.3** | **0.99** | **60,084.0** | **3,951.5** | **1.00** | 2.21 | 2.31 | 0.86 |
| g19 | – | – | 0.00 | – | – | 0.00 | **180,748.4** | **21,419.3** | **0.95** | **181,296.2** | **21,601.8** | **0.97** | – | – | 1.02 |
| g21 | 160,146.0 | 10,142.3 | **1.00** | 186,080.9 | 11,043.8 | **0.99** | **85,005.5** | **23,407.0** | 0.97 | **89,617.5** | **9,787.2** | **1.00** | 1.79 | 2.10 | 0.98 |
| g23 | – | – | 0.00 | – | – | 0.00 | **203,019.5** | **22,499.2** | **0.88** | **205,336.9** | **23,185.5** | **0.48** | – | – | 0.54 |
| g24 | 8,946.0 | 714.8 | 1.00 | 9,009.0 | 693.0 | 1.00 | **5,436.0** | **479.8** | 1.00 | **5,490.0** | **344.4** | 1.00 | 1.63 | 1.64 | 0.99 |
| avg | – | – | 0.81 | – | – | 0.89 | – | – | **0.95** | – | – | **0.97** | 1.65 | 1.82 | 1.20 |

of SAMODE. In ISAMODE-CMA, both mixed mutation operators and CMA-ES based local search are implemented. ECHT-EP2 [29] is evolutionary programming based on ensemble of constraint-handling techniques. ATMES [41] is an adaptive trade-off model based evolution strategy for the COPs. SMES [42] is a simple multi-member evolution strategy to solve the COPs, where a simple diversity mechanism based on allowing infeasible solutions to remain in the population is presented to handle the constraints. We choose these five EAs for comparisons due to their good performance obtained and the same Max_NFEs $(240,000)$ used. In Table 3, the best, mean, and standard deviation of the objective function values of the final solutions for each algorithms are shown. The overall best and the second best results among the six EAs are highlighted in **grey boldface** and **boldface**, respectively. "NA" means not available. Note that the results of ISAMODE-CMA, SAMODE, ECHT-EP2, ATMES, and SMES are directly obtained from their corresponding literature. In addition, for rank-iMDDE, ISAMODE-CMA, SAMODE, and ECHT-EP2, based on the mean values in Table 3, the final rankings obtained by the Friedman test[2] are shown in Table 4.

According to the results shown in Table 3, it can be clearly seen that rank-iMDDE consistently obtains highly-competitive results in all functions compared with other five EAs. In terms of the best results, rank-iMDDE gets the best or similar values among the six EAs in all 22 functions. With respect to the mean results, only in one function g19, rank-iMDDE is slightly worse than ISAMODE-CMA. In the rest 21 functions, rank-iMDDE is able to obtain better or similar results compared with other five EAs.

Furthermore, according to the mean values of rank-iMDDE, ISAMODE-CMA, SAMODE, and ECHT-EP2 shown in Table 3, the $p$-value computed by the Iman-Daveport test is $1.670E-03$, which means that the differences are significant between the compared algorithms in all functions at $\alpha = 0.05$. With respect to the average rankings of all algorithms by the Friedman test, Table 4 shows that our proposed rank-iMDDE gets the first ranking among four algorithms, followed by ISAMODE-CMA, ECHT-EP2, and SAMODE.

[2]The statistic results of the Friedman test and the Iman-Daveport test are calculated by the KEEL software tool [43].

*5.6. Influence of ranking-based mutation operator*

In the above experiments, we have verified that rank-iMDDE is efficient and highly-competitive. In this section, the influence of the ranking-based mutation operator on the performance of rank-iMDDE is evaluated. For this purpose, rank-iMDDE is compared with iMDDE. The only one difference between rank-iMDDE and iMDDE is that in iMDDE three vector indices $r_1, r_2, r_3$ in the mutation are only randomly selected as presented in the original DE algorithm. In addition, we remove the dynamic diversity mechanism as presented in Section 4.2 by using the static diversity mechanism, *i.e.*, $S_r$ is set to be 0.45 during the whole evolution process. The two algorithms are referred to as rank-MDDE-b/e and MDDE-b/e[3], for short. The parameters are set to be the same as presented in Section 5.2 for the four methods. The NFEs, *SR*, and *AR* values are compared as shown in Table 5. All results are averaged over 100 runs. The overall best and the second best results are highlighted in **grey boldface** and **boldface**, respectively.

When comparing the performance between rank-iMDDE and iMDDE, it is clear that rank-iMDDE requires less mean NFEs than iMDDE in all cases. The average *AR* value is 1.82, which means that rank-iMDDE performs 82% faster than iMDDE in overall. Moreover, rank-iMDDE obtains higher average *SR* value than that of iMDDE ($0.97 > 0.89$). Especially, in functions g19 and g23, there are no success runs for iMDDE, while rank-iMDDE gets *SR* with 0.97 and 0.48, respectively, in the two functions.

Comparison between rank-MDDE-b/e and MDDE-b/e, similar results can be observed in Table 5. rank-MDDE-b/e converges faster in 20 out of 22 cases; only in two functions g13 and g17, MDDE-b/e is better than rank-MDDE-b/e. In addition, rank-MDDE-b/e is able to obtain the higher average *SR* value than MDDE-b/e ($0.95 > 0.81$).

However, by carefully looking the results, we can see that in g02 rank-iMDDE and rank-MDDE-b/e provide lower *SR* values than their corresponding iMDDE and MDDE-b/e. The reason might be that the ranking-based mutation operator enhances the exploitation ability of the algorithm, yet slightly decreases its exploration ability. While for g02 it has large feasible space ($\rho = 100\%$ see in Table 1), the ranking-based mutation operator may lead the algorithm not to explore the large feasible space sufficiently.

In general, from the results and analysis we can conclude that the ranking-based mutation operator is of benefit to the performance enhancement of iMDDE and MDDE-b/e for the COPs. It not only accelerates the algorithms, but also makes them more efficient (with respect to the success rate).

*5.7. Effect of dynamic diversity mechanism*

In iMDDE as presented in Section 4.2, the improved dynamic diversity mechanism is proposed to dynamically control the selection ratio $S_r$. In this section, its influence on the performance of rank-iMDDE is studied. Also, the four algorithms mentioned in Section 5.6 are compared, *i.e.*, rank-iMDDE vs rank-MDDE-b/e, and iMDDE vs MDDE-b/e. The results are tabulated in Table 5. From the results, we can observe that

- The dynamic diversity mechanism is able to make the algorithm get higher success rate than the static one, for example, *SR* of iMDDE is 0.89, which is greater than that of MDDE-b/e (0.81). Also, rank-iMDDE gets higher *SR* value than rank-MDDE-b/e ($0.97 > 0.95$).

- However, the dynamic diversity mechanism leads to greater NFEs values in most of cases. For example, in 17 out of 22 functions rank-MDDE-b/e needs less NFEs values than rank-iMDDE.

- Compared the *AR* value between rank-iMDDE and rank-MDDE-b/e, the average *AR* value is 1.20, which indicates that rank-iMDDE gets 20% faster than rank-MDDE-b/e in overall.

In general, the dynamic diversity mechanism is able to provide higher success rate and make the algorithm more robust than the static one, although it requires greater NFEs to reach $f(\mathbf{x}) - f(\mathbf{x}^*) < 1e-4$.

[3]MDDE-b/e means that the combination of "DE/rand/1/bin" and "DE/rand/1/exp" is used. It is used to differ from the original MDDE as proposed in [19, 2]. It is worth pointing out that we compared the performance of MDDE-b/e with that of MDDE through benchmark functions, and in most of the test cases, MDDE-b/e provides better results than MDDE. However, for the sake of brevity, we did not report these results. Interested readers can contact the first author for more details.

Table 6: Parameter study on $k_1$ and $k_2$ of the ranking-based mutation operator on the performance of rank-iMDDE in all benchmark functions. The mean NFEs, and $SR$ results are reported herein. All results are averaged over 100 independent runs.

| Prob | iMDDE | | $k_1 = 0.5, k_2 = 2.0$ | | $k_1 = k_2 = 0.5$ | | $k_1 = k_2 = 1.0$ | | $k_1 = 5.0, k_2 = 0.2$ | | $k_1 = k_2 = 2.0$ | | $k_1 = 2.0, k_2 = 0.5$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | $SR$ | Mean | $SR$ | Mean | $SR$ | Mean | $SR$ | Mean | $SR$ | Mean | $SR$ | Mean | $SR$ |
| g01 | 142,897.5 | 1.00 | 112,711.5 | 1.00 | 114,853.5 | 1.00 | 98,163.0 | 1.00 | **61,929.0** | 1.00 | **80,365.5** | 1.00 | 80,482.5 | 1.00 |
| g02 | 178,325.0 | **0.90** | **87,907.2** | 0.67 | 128,007.9 | 0.84 | 101,660.6 | 0.80 | 131,737.8 | **0.89** | **77,711.4** | 0.63 | 118,733.0 | 0.86 |
| g03 | 106,393.5 | 1.00 | 67,531.5 | 1.00 | 67,536.0 | 1.00 | 57,690.0 | 1.00 | **41,395.5** | 1.00 | **49,572.0** | 1.00 | **49,572.0** | 1.00 |
| g04 | 60,507.0 | 1.00 | 44,973.0 | 1.00 | 44,959.5 | 1.00 | 38,106.0 | 1.00 | **24,246.0** | 1.00 | **31,648.5** | 1.00 | **31,648.5** | 1.00 |
| g05 | 40,279.5 | 1.00 | 35,563.5 | 1.00 | 35,563.5 | 1.00 | 34,708.5 | 1.00 | **32,521.5** | 1.00 | **33,615.0** | 1.00 | **33,615.0** | 1.00 |
| g06 | 21,348.0 | 1.00 | 16,173.0 | 1.00 | 16,587.0 | 1.00 | 14,607.0 | 1.00 | **10,966.5** | 1.00 | 12,946.5 | 1.00 | **12,942.0** | 1.00 |
| g07 | 137,128.5 | 1.00 | 96,129.0 | 1.00 | 96,232.5 | 1.00 | 79,542.0 | 1.00 | **45,954.0** | 1.00 | **62,275.5** | 1.00 | **62,275.5** | 1.00 |
| g08 | 3,906.0 | 1.00 | 3,721.5 | 1.00 | 3,721.5 | 1.00 | 3,321.0 | 1.00 | **2,488.5** | 1.00 | **2,961.0** | 1.00 | **2,961.0** | 1.00 |
| g09 | 46,332.0 | 1.00 | 33,309.0 | 1.00 | 35,203.5 | 1.00 | 29,920.5 | 1.00 | **18,693.0** | 1.00 | **24,561.0** | 1.00 | 24,849.0 | 1.00 |
| g10 | 230,040.0 | 0.01 | 147,987.0 | **1.00** | 149,256.0 | **1.00** | 121,486.5 | **1.00** | **67,721.2** | **0.96** | **92,718.0** | **1.00** | **92,718.0** | **1.00** |
| g11 | 8,779.5 | 1.00 | 8,217.0 | 1.00 | 8,217.0 | 1.00 | 7,834.5 | 1.00 | **6,705.0** | 1.00 | **7,339.5** | 1.00 | **7,339.5** | 1.00 |
| g12 | 4,149.0 | 1.00 | 3,645.0 | 1.00 | 3,645.0 | 1.00 | 3,105.0 | 1.00 | **2,990.0** | 1.00 | **3,100.5** | 1.00 | **3,100.5** | 1.00 |
| g13 | 62,598.2 | **0.97** | **38,988.0** | **1.00** | **38,988.0** | **1.00** | 39,037.5 | **1.00** | **38,997.0** | **1.00** | **38,988.0** | **1.00** | **38,988.0** | **1.00** |
| g14 | – | 0.00 | 143,428.5 | 1.00 | 145,746.0 | 1.00 | 136,300.5 | 1.00 | **119,592.0** | 1.00 | **127,237.5** | 1.00 | 127,552.5 | 1.00 |
| g15 | 20,061.0 | 1.00 | 19,719.0 | 1.00 | 19,719.0 | 1.00 | 19,273.5 | 1.00 | **18,765.0** | 1.00 | **19,066.5** | 1.00 | **19,066.5** | 1.00 |
| g16 | 32,004.0 | 1.00 | 25,348.5 | 1.00 | 25,308.0 | 1.00 | 21,784.5 | 1.00 | **14,481.0** | 1.00 | **18,526.5** | 1.00 | **18,526.5** | 1.00 |
| g17 | 98,107.3 | **0.98** | 70,182.0 | **1.00** | 70,128.0 | **1.00** | 66,172.5 | **1.00** | 69,048.0 | **1.00** | **63,652.5** | **1.00** | **64,539.0** | **1.00** |
| g18 | 133,056.0 | **1.00** | 92,992.5 | **1.00** | 92,974.5 | **1.00** | 76,135.5 | **1.00** | **44,330.6** | **0.96** | **60,084.0** | **1.00** | **60,084.0** | **1.00** |
| g19 | – | 0.00 | 215,980.1 | 0.86 | 233,370.0 | 0.30 | 204,662.7 | **0.99** | 193,030.0 | 0.45 | **175,266.0** | **1.00** | **181,296.2** | 0.97 |
| g21 | 160,146.0 | 1.00 | 109,350.0 | 1.00 | 109,642.5 | 1.00 | 99,225.0 | 1.00 | **80,280.0** | 1.00 | 89,622.0 | 1.00 | **89,617.5** | 1.00 |
| g23 | – | 0.00 | 222,896.3 | 0.24 | 225,294.5 | 0.11 | 217,002.5 | 0.36 | **193,477.5** | **0.52** | **202,763.1** | **0.52** | 205,336.9 | **0.48** |
| g24 | 8,946.0 | 1.00 | 6,745.5 | 1.00 | 6,993.0 | 1.00 | 6,223.5 | 1.00 | **4,563.0** | 1.00 | **5,463.0** | 1.00 | 5,490.0 | 1.00 |
| avg | – | 0.81 | – | 0.94 | – | 0.92 | – | **0.96** | – | 0.94 | – | **0.96** | – | **0.97** |

### *5.8. Parameter study*

In the proposed ranking-based mutation operator, there are two user-defined coefficients ($k_1$ and $k_2$) in the probability calculation models. In Section 4.1, we suggested that $k_1$ is set to be greater than 1.0, while $k_2$ is less than or equal to 1.0. In this section, the influence of $k_1$ and $k_2$ on the performance of rank-iMDDE is evaluated. We set different $k_1$ and $k_2$ values as: (a) $k_1 = 0.5, k_2 = 2.0$, (b) $k_1 = k_2 = 0.5$, (c) $k_1 = k_2 = 1.0$, (d) $k_1 = 5.0, k_2 = 0.2$, and (e) $k_1 = k_2 = 2.0$. All other parameters for rank-iMDDE are kept unchanged as described in Section 5.2. Each algorithm is performed over 100 independent runs for each function. The results of mean NFEs and $SR$ are given in Table 6, in which the results of iMDDE and rank-iMDDE with default parameter settings (*i.e.*, $k_1 = 2.0, k_2 = 0.5$) are also reported. The overall best, second best, and worst results are highlighted in **grey boldface**, **boldface**, and `teletype`, respectively.

According to the results reported in Table 6, it can be observed that

- Generally, the six rank-iMDDE variants with different $k_1$ and $k_2$ settings are able to converge faster than iMDDE. Additionally, they are also capable of providing higher average $SR$ values than that of iMDDE. It means that the enhanced performance by the ranking-based mutation operator is not influenced by different $k_1$ and $k_2$ settings.
- With respect to the NFEs values, we can observe that rank-iMDDE variants with the same $k_1$ value and different $k_2$ values (*e.g.*, $k_1 = 0.5, k_2 = 2.0$ vs $k_1 = k_2 = 0.5$ and $k_1 = 2.0, k_2 = 0.5$ vs $k_1 = k_2 = 2.0$) provide similar NFEs in most of test cases. The reason is that the population is mainly in the semi-feasible situation during the whole evolution process.
- rank-iMDDE with $k_1 = 5.0, k_2 = 0.2$ requires the smallest NFEs in the majority of the functions, because of the higher selection pressure on the better solutions in the semi-feasible situation. However, it leads to premature convergence in some complexity functions, such as g10 and g19.
- If $k_1$ is less than 1.0, for example $k_1 = 0.5$, rank-iMDDE converges slower than $k_1 > 1.0$, due to the lower selection pressure on the better solutions.

In summary, the ranking-based mutation operator is able to enhance the performance of iMDDE in terms of both convergence rate and success rate regardless of different $k_1$ and $k_2$ settings. Since the population is mainly in the semi-feasible situation, the ranking in this situation plays the leading role. Thus, $k_1$ has more important influence on the performance than $k_2$. According to the results, we suggest that $k_1 \in [2.0, 5.0]$ and $k_2 \in [0.5, 1.0+)$ are good choices. The default setting with $k_1 = 2.0$ and $k_2 = 0.5$ is a reasonable, but not optimal, setting in the ranking-based mutation operator.

Table 7: Comparison on the best results between rank-iMDDE and BIANCA on the non-convex test cases.

| | case 1 | | case 2 | | case 3 | |
|---|---|---|---|---|---|---|
| | BIANCA [44] | rank-iMDDE | BIANCA [44] | rank-iMDDE | BIANCA [44] | rank-iMDDE |
| $x_1$ | 10.71119 | 10.71252 | 11.27620 | 11.29429 | 10.45320 | 10.47395 |
| $x_2$ | 2.96646 | 2.96338 | 2.46284 | 2.47128 | 2.71465 | 2.73013 |
| $g_1$ | -4.43000E-03 | **-1.12850E-11** | – | – | -3.86538E-03 | **-4.38089E-12** |
| $g_2$ | – | – | -1.09853E-04 | **-1.76889E-12** | -3.30729E-03 | **-1.16759E-12** |
| $f$ | -8.09933 | **-8.11646** | -9.49783 | **-9.50127** | -7.37696 | **-7.48573** |

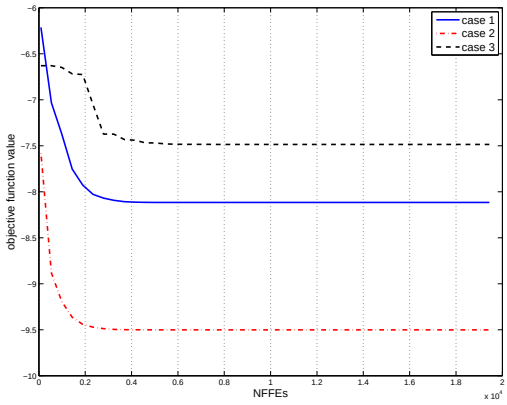



Figure 2: Convergence graphs of rank-iMDDE for the non-convex test cases.

### *5.9. rank-iMDDE for non-convex COPs*

In this section, the effectiveness of rank-iMDDE is further evaluated on the non-convex COPs. With this purpose, three test problems presented in [44] are selected. These problems have the same highly non-convex objective function:

$$f(x_1, x_2) = -\exp\left(ka\sqrt{x_1^2 + x_2^2}\right)\sin(ax_1)\cos(2bx_2) \tag{17}$$

whereas they have different constraints as follows:

- Case 1:

$$g_1(x_1, x_2) = \exp\left(cx_1^2\right) - 1 - x_2 \leq 0 \tag{18}$$

- Case 2:

$$g_2(x_1, x_2) = \alpha\left[\left(\frac{\beta}{4\pi}x_1 + \gamma\right)\sin\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right) - \delta\right] \leq 0 \tag{19}$$

- Case 3:

$$\begin{aligned} g_1(x_1, x_2) &= \exp\left(cx_1^2\right) - 1 - x_2 \leq 0 \\ g_2(x_1, x_2) &= \alpha\left[\left(\tfrac{\beta}{4\pi}x_1 + \gamma\right)\sin\left(\tfrac{\pi}{2}x_1\right)\sin\left(\tfrac{\pi}{2}x_2\right) - \delta\right] \leq 0 \end{aligned} \tag{20}$$

where $x_1 \in [0, 4\pi]$, $x_2 \in [0, 2\pi]$, $a = 1, b = 0.6, k = 0.2, c = 0.012, \alpha = 0.1, \beta = 0.4, \gamma = 0.8$, and $\delta = 0.7$. As described in [44], the constraint $g_1(x_1, x_2)$ is active. The constraint $g_2(x_1, x_2)$ shown in Equation (19) is highly non-convex, which leads the COPs to be defined over a disjoint search space characterized by "barely" infeasible regions [44].

To make a fair comparison with the results reported in [44], the Max_NFEs are set to be $20,000$ for rank-iMDDE as used in [44]. All other parameters are kept unchanged as described in Section 5.2. For each test case, rank-iMDDE is performed over 100 independent runs. The results are given in Table 7, where the better objective function

Table 8: Comparison on the results of welded beam design (case 1).

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| SCM [1] | 2.3854347 | 6.3996785 | 3.0025883 | 9.60E-01 | 33,095 |
| DSS-MDE [25] | **2.38095658** | **2.38095658** | **2.38095658** | 3.19E-10 | 24,000 |
| AATM [45] | 2.3823262 | 2.3915924 | 2.3869762 | 2.2E-03 | 30,000 |
| DELC [46] | **2.38095658** | **2.38095658** | **2.38095658** | 2.60E-12 | 20,000 |
| rank-iMDDE | **2.38095658** | **2.38095658** | **2.38095658** | **7.18E-14** | **19,830** |

and constraint function values are highlighted in **boldface**. In addition, the convergence graphs of rank-iMDDE are plotted in Fig. 2.

From Table 7, it is clear that rank-iMDDE consistently gets the better results compared with BIANCA in all cases. The constraint function values are very close to zero in the three cases for the best solutions obtained by rank-iMDDE. In addition, according to the convergence curves shown in Fig. 2, we see that rank-iMDDE is able to obtain the nearly optimal solutions very quickly: In cases 1 and 2, it requires around $2,500$ NFEs; in case 3, it needs about $4,000$ NFEs.

From the above analysis, we can see that rank-iMDDE is also effective for the non-convex COPs, in which the objective function is highly non-convex and the constraints are active and/or non-convex as presented in the above test cases.

### *5.10. rank-iMDDE for constrained engineering benchmark problems*

Experimental results on benchmark functions verified that rank-iMDDE is very efficient and it provides highly-competitive results compared with other state-of-the-art EAs when solving the COPs. In this section, the potential of rank-iMDDE is also tested through 5 widely used constrained engineering benchmark problems. The five problems are: i) welded beam design (case 1 [1] and case 2 [24]), ii) tension/compression spring design [1], iii) speed reducer design [1], iv) three-bar truss design [1], and v) pressure vessel design [24]. 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, rank-iMDDE adopts different population size and Max_NFEs, while other parameters are kept the same as shown in Section 5.2. The population size and Max_NFEs of rank-iMDDE for different problems are

- For welded beam design (case 1 and 2): $\mu = 30$ and Max_NFEs = $19,830$;
- For spring design: $\mu = 65$ and Max_NFEs = $19,565$;
- For reducer design and three-bar truss design: $\mu = 20$ and Max_NFEs = $19,920$;
- For pressure vessel design: $\mu = 65$ and Max_NFEs = $23,465$.

For each problem, rank-iMDDE is performed over 100 independent runs. The best result obtained by rank-iMDDE in each problem is shown in Table 14.

#### *5.10.1. Welded Beam Design (Case 1)*

rank-iMDDE is compared with four EAs in this problem. The four EAs are: 1) society and civilization algorithm (SCM) [1], 2) dynamic stochastic ranking based DE (DSS-MDE) [25], 3) accelerated adaptive trade-off model based EA (AATM) [45], and 4) DE with level comparison (DELC) [46]. The results of these algorithms are shown in Table 8. A result in **boldface** means a better (or best) solution obtained. From the results in Table 8, we can observe that rank-iMDDE obtains the optimal solution in this problem in all 100 runs. Moreover, with the smallest Max_NFEs, rank-iMDDE is able to provide the smallest standard deviation value compared with other EAs.

#### *5.10.2. Welded Beam Design (Case 2)*

This problem is another version of welded beam design (case 1). In this problem, rank-iMDDE is compared with: 1) co-evolutionary DE (Co-DE) [24], 2) multiple trial vector based DE (MDDE) [2], 3) DELC [46], 4) COMDE [30], 5) CVI-PSO [47], 6) BIANCA [44], and 7) MVDE [48]. The results are tabulated in Table 9. Similar to welded beam design (case 1), rank-iMDDE successfully solve this problem in all runs, and it gets the smallest standard deviation value with Max_NFEs=19, 830 among all compared EAs. Even with Max_NFEs=15, 000, rank-iMDDE also gets highly-competitive results compared with other EAs.

Table 9: Comparison on the results of welded beam design (case 2).

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| Co-DE [24] | 1.733461 | 1.824105 | 1.768158 | 2.22E-02 | 204,800 |
| MDDE [2] | **1.725** | **1.725** | **1.725** | 1.00E-15 | 24,000 |
| DELC [46] | **1.724852** | **1.724852** | **1.724852** | 4.10E-13 | 20,000 |
| COMDE [30] | **1.724852309** | **1.724852309** | **1.724852309** | 1.60E-12 | 20,000 |
| CVI-PSO [47] | **1.724852** | 1.727665 | 1.725124 | 6.12E-04 | 25,000 |
| BIANCA [44] | **1.724852** | 1.793233 | 1.752201 | 2.30E-02 | 80,000 |
| MVDE [48] | 1.7248527 | 1.7249215 | 1.7248621 | 7.88E-06 | **15,000** |
| rank-iMDDE | **1.724852309** | **1.724852309** | **1.724852309** | **9.06E-16** | 19,830 |
| | **1.724852309** | **1.724852309** | **1.724852309** | 7.71E-11 | **15,000** |

Table 10: Comparison on the results of tension/compression spring design.

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| SCM [1] | 0.012669249 | 0.016717272 | 0.012922669 | 5.90E-04 | 25,167 |
| Co-DE [24] | 0.0126702 | 0.01279 | 0.012703 | 2.70E-05 | 204,800 |
| MDDE [2] | **0.012665** | 0.012674 | 0.012666 | 2.00E-06 | 24,000 |
| DSS-MDE [25] | **0.012665233** | 0.012738262 | 0.012669366 | 1.25E-05 | 24,000 |
| AATM [45] | 0.012668262 | 0.012861375 | 0.012708075 | 4.50E-05 | 25,000 |
| DELC [46] | **0.012665233** | **0.012665575** | 0.012665267 | **1.30E-07** | 20,000 |
| COMDE [30] | **0.012665232** | 0.012676809 | 0.012667168 | 3.09E-06 | 24,000 |
| CVI-PSO [47] | 0.0126655 | 0.0128426 | 0.012731 | 5.58E-05 | 25,000 |
| BIANCA [44] | 0.012671 | 0.012913 | 0.012681 | 5.12E-05 | 80,000 |
| MVDE [48] | 0.012665272 | 0.012719055 | 0.012667324 | 2.45E-06 | **10,000** |
| rank-iMDDE | **0.012665233** | 0.01266765 | **0.012665264** | 2.45E-07 | 19,565 |
| | **0.012665233** | 0.01266743 | 0.012665297 | 8.48E-07 | **10,000** |

*5.10.3. Tension/Compression Spring Design*

This problem is to minimize the weight under four constraints. rank-iMDDE is compared with ten different EAs in this problem. These methods are SCM [1], Co-DE [24], MDDE [2], DSS-MDE [25], AATM [45], DELC [46], COMDE [30], CVI-PSO [47], BIANCA [44], and MVDE [48]. According to the results given in Table 10, we can see that there are five algorithms (*i.e.*, MDDE, DSS-MDE, DELC, COMDE, and rank-iMDDE) that can obtain the optimal solution in this problem. Our proposed rank-iMDDE gets the best mean value in all algorithms. While DELC provides the best results with respect to the worst and standard deviation values; rank-iMDDE is slightly worse than DELC in these two metrics.

*5.10.4. Speed Reducer Design*

The weight is minimized in the speed reducer design. In this problem, rank-iMDDE is compared with SCM [1], MDDE [2], DSS-MDE [25], AATM [45], DELC [46], COMDE [30], and MVDE [48], and the results are shown in Table 11. It is clear that rank-iMDDE is capable of getting the optimal solution in all runs. In addition, it can obtain the smallest standard deviation value yet with the smallest Max_NFEs among all compared algorithms.

*5.10.5. Three-Bar Truss Design*

The three-bar truss design is to minimize the volume subject to stress constraints. In Table 12, the results of rank-iMDDE are compared with those of SCM [1], DSS-MDE [25], AATM [45], DELC [46], COMDE [30], and MVDE [48]. Clearly, rank-iMDDE obtains the global optimal solution in all runs with the smallest Max_NFEs. It also provides the smallest standard deviation value compared with other EAs.

*5.10.6. Pressure Vessel Design*

The results of Co-DE [24], MDDE [2], DELC [46], COMDE [30], CVI-PSO [47], BIANCA [44], MVDE [48], and rank-iMDDE are given in Table 13. From the results, we can see that there are four algorithms (MDDE, DELC, COMDE, and rank-iMDDE) that are able to solve this problem in all runs. In terms of the standard deviation, MDDE is the best one, followed by rank-iMDDE, DELC, COMDE, and Co-DE.

To sum up, within the above-mentioned engineering benchmark problems, rank-iMDDE is able to successfully solve welded beam (case 1 and 2), speed reducer, three-bar truss, and pressure vessel over all 100 independent runs

Table 11: Comparison on the results of speed reducer design.

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| SCM [1] | 2994.744241 | 3009.964736 | 3001.758264 | 4.00E+00 | 54,456 |
| MDDE [2] | 2996.357 | 2996.39 | 2996.367 | 8.20E-03 | 24,000 |
| DSS-MDE [25] | **2994.471066** | **2994.471066** | **2994.471066** | 3.58E-12 | 30,000 |
| AATM [45] | 2994.516778 | 2994.659797 | 2994.585417 | 3.30E-02 | 40,000 |
| DELC [46] | **2994.471066** | **2994.471066** | **2994.471066** | 1.90E-12 | 30,000 |
| COMDE [30] | **2994.471066** | **2994.471066** | **2994.471066** | 1.54E-12 | 21,000 |
| MVDE [48] | **2994.471066** | 2994.471069 | **2994.471066** | 2.82E-07 | 30,000 |
| rank-iMDDE | **2994.471066** | **2994.471066** | **2994.471066** | **7.93E-13** | **19,920** |

Table 12: Comparison on the results of three-bar truss design.

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| SCM [1] | 263.8958466 | 263.96975 | 263.9033 | 1.30E-02 | 17,610 |
| DSS-MDE [25] | **263.8958434** | 263.8958498 | 263.8958436 | 9.72E-07 | 15,000 |
| AATM [45] | 263.8958435 | 263.90041 | 263.8966 | 1.10E-03 | 17,000 |
| DELC [46] | **263.8958434** | **263.8958434** | **263.8958434** | 4.30E-14 | 10,000 |
| COMDE [30] | **263.8958433** | **263.8958433** | **263.8958433** | 5.34E-13 | 7,000 |
| MVDE [48] | **263.8958434** | 263.8958548 | **263.8958434** | 2.58E-07 | 7,000 |
| rank-iMDDE | **263.8958434** | **263.8958434** | **263.8958434** | **0.00E+00** | **4,920** |

Table 13: Comparison on the results of pressure vessel design.

| Algorithm | Best | Worst | Mean | Std | Max_NFEs |
|---|---|---|---|---|---|
| Co-DE [24] | 6059.734 | 6371.0455 | 6085.2303 | 4.30E+01 | 204,800 |
| MDDE [2] | **6059.702** | **6059.702** | **6059.702** | **1.00E-12** | 24,000 |
| DELC [46] | **6059.7143** | **6059.7143** | **6059.7143** | 2.10E-11 | 30,000 |
| COMDE [30] | **6059.714335** | **6059.714335** | **6059.714335** | 3.62E-10 | 30,000 |
| CVI-PSO [47] | **6059.7143** | 6820.4101 | 6292.1231 | 2.88E+02 | 25,000 |
| BIANCA [44] | **6059.9384** | 6447.3251 | 6182.0022 | 1.22E+02 | 80,000 |
| MVDE [48] | 6059.714387 | 6090.533528 | 6059.997236 | 2.91E+00 | **15,000** |
| rank-iMDDE | **6059.714335** | **6059.714335** | **6059.714335** | 1.95E-12 | 23,465 |
| | **6059.714335** | **6059.714335** | **6059.714335** | 7.57E-07 | **15,000** |

Table 14: The best results obtained by rank-iMDDE for each constrained engineering design problem used in this work.

| Prob | Best solution $\mathbf{x}$ | Best result $f(\mathbf{x})$ |
|---|---|---|
| Welded beam design (case 1) | 0.24436898, 6.21751972, 8.29147139, 0.24436898 | 2.38095658 |
| Welded beam design (case 2) | 0.20572964, 3.47048867, 9.03662391, 0.20572964 | 1.724852309 |
| Tension/compression spring design | 0.35671718, 0.05168904, 11.28899860 | 0.012665233 |
| Speed reducer design | 3.5, 0.7, 17, 7.3, 7.71531991, 3.35021467, 5.28665446 | 2994.471066 |
| Three-bar truss design | 0.78867513, 0.40824829 | 263.8958434 |
| Pressure vessel design | 13, 7, 42.09844560, 176.63659584 | 6059.714335 |

yet with the smallest Max_NFEs compared with other EAs. For the spring design, rank-iMDDE also obtains the near-optimal solution ($f(\mathbf{x}) - f(\mathbf{x}^*) \leq 1e-8$) in 95 out of 100 runs. Thus, we can conclude that our proposed rank-iMDDE is also able to deal with constrained engineering benchmark problems.

### *5.11. rank-iMDDE for constrained mechanical design problems from literature*

According to the results on the constrained engineering benchmark problems shown in Section 5.10, it can be observed that rank-iMDDE provides very promising results when comparing with other state-of-the-art EAs. In this section, four further constrained mechanical design problems are considered, including the step-cone pulley, hydrodynamic thrust bearing, rolling element bearing, and Belleville spring. These problems are selected from [49], which have different natures of design variables, objective functions, and constraints. The objective functions of all problems are minimized herein. For their detailed descriptions, interested readers can refer to the literature in [49]. The results of rank-iMDDE is compared with those of MDDE, TLBO, and ABC. The results of TLBO and ABC are obtained from [49]. For rank-iMDDE and MDDE, the population size and Max_NFEs of rank-iMDDE for different problems are

- For the step-cone pulley problem: $\mu = 30$ and Max_NFEs = $15,000$;

Table 15: Comparison on the results of different algorithms on the constrained mechanical design problems from the literature.

| Prob | Criterion | ABC [49] | TLBO [49] | MDDE | rank-iMDDE |
|---|---|---|---|---|---|
| Step-cone pulley | Best | 16.634655 | 16.634510 | 14.488038 | **14.487968** |
| | Mean | 36.099500 | 24.011358 | 16.725652 | **15.472300** |
| | Worst | 145.470500 | 74.022951 | 18.169821 | **17.833931** |
| Hydrostatic thrust bearing | Best | **1625.442760** | 1625.443000 | 1638.403234 | 1625.460142 |
| | Mean | 1861.554000 | 1797.707980 | 1759.103885 | **1724.727935** |
| | Worst | 2144.836000 | 2096.801270 | 2553.358476 | **1894.734127** |
| Rolling element bearing | Best | **-81859.741600** | -81859.740000 | -81858.836832 | -81859.732421 |
| | Mean | -81496.000000 | -81438.987000 | -81848.703534 | **-81859.010377** |
| | Worst | -78897.810000 | -80807.855100 | -81701.180671 | **-81838.757577** |
| Belleville spring | Best | **1.979675** | **1.979675** | **1.979675** | **1.979675** |
| | Mean | 1.995475 | 1.979688 | 1.982256 | **1.979675** |
| | Worst | 2.104297 | 1.979757 | 2.104326 | **1.979683** |

Table 16: The best results obtained by rank-iMDDE for the constrained mechanical design problems.

| Prob | Best solution $\mathbf{x}$ | Best result $f(\mathbf{x})$ |
|---|---|---|
| Step-cone pulley | 99.999998, 34.581577, 47.581631, 63.437726, 76.067314 | 14.487968 |
| Hydrostatic thrust bearing | 5.955817, 5.389051, 5.358711, 2.269693 | 1625.460142 |
| Rolling element bearing | 125.719053, 21.425590, 11.375940, 0.515000, 0.515000, 0.434445, 0.636030, 0.300000, 0.079193, 0.674157 | −81859.732421 |
| Belleville spring | 12.010000, 10.030473, 0.204143, 0.200000 | 1.979675 |

- For the hydrostatic thrust bearing problem: $\mu = 80$ and Max_NFEs = $25,000$;
- For the rolling element bearing problem: $\mu = 25$ and Max_NFEs = $10,000$;
- For the Belleville spring problem: $\mu = 25$ and Max_NFEs = $15,000$.

Other parameters are kept unchanged as shown in Section 5.2. Note that the Max_NFEs are set the same as used in [49]. For each problem, rank-iMDDE and MDDE are performed over 100 independent runs.

The results are reported in Table 15, where the best overall results are highlighted in **boldface**. The best solution of each mechanical problem obtained by rank-iMDDE is tabulated in Table 16. From Table 15, it can be seen that

- For the step-cone pulley problem, rank-iMDDE consistently obtains the best results compared with other three algorithms in terms of the best, mean, and worst solutions.
- For the hydrostatic thrust bearing problem, ABC can get the best result in terms of the best solution, followed by TLBO, rank-iMDDE, and MDDE. The best solution of rank-iMDDE is very close to that of ABC. With respect to the mean and worst solutions, rank-iMDDE still ranks the first among the four compared algorithms.
- Similar to the results shown in the hydrostatic thrust bearing problem, for the rolling element bearing problem, rank-iMDDE gets the best results with respect to the mean and worst solutions. For the best solution, rank-iMDDE obtains slightly worse result when comparing with that of ABC and TLBO.
- For the Belleville spring problem, all of the four algorithms can obtain the same best solution. However, in terms of the mean and worst solutions, rank-iMDDE gets the best results.

To sum up, the proposed rank-iMDDE is still capable of providing the overall best and most robust results for the constrained mechanical problems compared with other three algorithms.

### *5.12. Discussions*

Combining with the constraint-handling techniques, the DE algorithm has been used for the COPs recently. In this work, the ranking-based mutation operator and an improved dynamic diversity mechanism are proposed to enhance the performance of DE when solving the COPs. From the comprehensive experiments through both benchmark functions and engineering problems, we can conclude that

- The proposed ranking-based mutation operator is efficient to enhance the performance of CDEs in terms of the convergence speed and the success rate.

- In the ranking-based mutation operator, there are two coefficients $k_1$ and $k_2$ introduced to calculate the selection probabilities of the solutions. Regardless of different parameter settings of $k_1$ and $k_2$, rank-iMDDE is capable of improving the performance of iMDDE consistently. The default setting with $k_1 = 2.0$ and $k_2 = 0.5$ is a reasonable choice, but may be not optimal. In our future work, we will try to study the adaptive setting of $k_1$ and $k_2$ in the ranking-based mutation operator.
- When comparing with other state-of-the-art EAs for the COPs, rank-iMDDE is highly-competitive in terms of the final solutions, convergence speed, and success rate.

## 6. Conclusions and future work

To accelerate the convergence rate and maintain the population diversity of the DE algorithm when solving the COPs, in the paper, an improved constrained DE (rank-iMDDE) is proposed, where the ranking-based mutation and improved dynamic diversity mechanism are presented. The performance of rank-iMDDE is evaluated by 24 benchmark functions presented in CEC'2006 and five widely used engineering benchmarks and four constrained mechanical design problems. Experimental results verify the superiority of rank-iMDDE when comparing with other EAs.

In rank-iMDDE, the combination of "DE/rand/1/bin" and "DE/rand/1/exp" is adopted. Recent studies indicate that the adaptive ensemble of different mutation strategies is able to enhance the performance of DE [22, 50, 51, 31]. Therefore, one possible future work is to combine rank-based CDEs with strategy adaptation techniques to further improve the performance of CDEs. In addition, the ranking-based mutation operators may also be useful to the multiobjective optimization. For example, the non-dominated sorting method [52] can be possibly used to rank solutions in the multiobjective optimization. In our future, we will try to verify this expectation. In this work, the selected engineering design problems are not computationally expensive, whereas in the real world there exist many complex engineering problems that are computationally expensive, such as the discovery of low-energy pure water isomers [53], potential energy minimization [7], etc. Combining rank-iMDDE with local search methods and surrogate models like [54, 55], another future direction is developing the memetic rank-iMDDE algorithm for the complex computationally expensive problems.

The source code of the proposed rank-iMDDE can be obtained from the first author upon request.

## Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under Grant Nos. 61203307, 61075063, and 61375066, the Fundamental Research Funds for the Central Universities at China University of Geosciences (Wuhan) under Grant Nos. CUG130413 and CUG090109, and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20110145120009.

## References

[1] T. Ray, K. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior, IEEE Transactions on Evolutionary Computation 7 (4) (2003) 386 – 396.
[2] E. Mezura-Montes, C. A. Coello Coello, J. Velázquez-Reyes, L. Muñoz-Dávila, Multiple trial vectors in differential evolution for engineering design, Engineering Optimization 39 (5) (2007) 567–589.
[3] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evol. Comput. 4 (1) (1996) 1–32.
[4] C. A. C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering 191 (11-12) (2002) 1245 – 1287.
[5] E. Mezura-Montes, C. A. C. Coello, Constraint-handling in nature-inspired numerical optimization: Past, present and future, Swarm and Evolutionary Computation 1 (4) (2011) 173 – 194.
[6] M. Daneshyari, G. Yen, Constrained multiple-swarm particle swarm optimization within a cultural framework, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 42 (2) (2012) 475 – 490.
[7] S. Handoko, C. K. Kwoh, Y.-S. Ong, Feasibility structure modeling: An effective chaperone for constrained memetic algorithms, IEEE Transactions on Evolutionary Computation 14 (5) (2010) 740–758.
[8] R. Storn, K. Price, Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces, J. of Global Optim. 11 (4) (1997) 341–359.

[9] R. Storn, System design by constraint adaptation and differential evolution, IEEE Transactions on Evolutionary Computation 3 (1) (1999) 22 – 34.
[10] T. Takahama, S. Sakai, Constrained optimization by the $\varepsilon$ constrained differential evolution with gradient-based mutation and feasible elites, in: IEEE Congress on Evolutionary Computation, 2006, pp. 1 – 8.
[11] Y. Wang, Z. Cai, Constrained evolutionary optimization by means of $(\mu + \lambda)$-differential evolution and improved adaptive trade-off model, Evolutionary Computation 19 (2) (2011) 249 – 285.
[12] Y. Wang, Z. Cai, Combining multiobjective optimization with differential evolution to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 16 (1) (2012) 117–134.
[13] F. Neri, V. Tirronen, Recent advances in differential evolution: A survey and experimental analysis, Artificial Intelligence Review 33 (1-2) (2010) 61 – 106.
[14] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. on Evol. Comput. 15 (1) (2011) 4–31.
[15] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. on Evol. Comput. 12 (1) (2008) 107–125.
[16] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, Tech. rep., Nanyang Technological University, Singapore (2006).
[17] J. Lampinen, A constraint handling approach for the differential evolution algorithm, in: IEEE Congress on Evolutionary Computation, Vol. 2, 2002, pp. 1468 – 1473.
[18] R. L. Becerra, C. A. C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering 195 (33 - 36) (2006) 4303 – 4322.
[19] E. Mezura-Montes, J. Velázquez-Reyes, C. A. Coello Coello, Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization, in: Proceedings of the Conference on Genetic and Evolutionary Computation, 2005, pp. 225–232.
[20] E. Mezura-Montes, J. Velazquez-Reyes, C. Coello Coello, Modified differential evolution for constrained optimization, in: IEEE Congress on Evolutionary Computation, 2006, pp. 25 –32.
[21] V. Huang, A. Qin, P. Suganthan, Self-adaptive differential evolution algorithm for constrained real-parameter optimization, in: IEEE Congress on Evolutionary Computation, 2006, pp. 17 – 24.
[22] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. on Evol. Comput. 13 (2) (2009) 398–417.
[23] J. Brest, V. Zumer, M. Maucec, Self-adaptive differential evolution algorithm in constrained real-parameter optimization, in: IEEE Congress on Evolutionary Computation, 2006, pp. 215 – 222.
[24] F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics and Computation 186 (1) (2007) 340 – 356.
[25] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178 (15) (2008) 3043 – 3074.
[26] M. Ali, Z. Kajee-Bagdadi, A local exploration-based differential evolution algorithm for constrained global optimization, Applied Mathematics and Computation 208 (1) (2009) 31 – 48.
[27] E. Mezura-Montes, A. Palomeque-Ortiz, Self-adaptive and deterministic parameter control in differential evolution for constrained optimization, in: E. Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Vol. 198 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2009, pp. 95–120.
[28] E. Mezura-Montes, M. E. Miranda-Varela, R. del Carmen Gómez-Ramón, Differential evolution in constrained numerical optimization: An empirical study, Information Sciences 10 (22) (2010) 4223–4262.
[29] R. Mallipeddi, P. N. Suganthan, Ensemble of constraint handling techniques, IEEE Trans. on Evol. Comput. 14 (4) (2010) 561 – 579.
[30] A. W. Mohamed, H. Z. Sabry, Constrained optimization based on modified differential evolution algorithm, Information Sciences 194 (2012) 171 – 208.
[31] S. Elsayed, R. Sarker, D. Essam, An improved self-adaptive differential evolution algorithm for optimization problems, IEEE Transactions on Industrial Informatics 9 (1) (2013) 89 – 99.
[32] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary Computation 11 (1) (2003) 1 – 18.
[33] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Transactions on Cybernetics (2013) 1 – 16In press.
[34] T. Bäck, Selective pressure in evolutionary algorithms: a characterization of selection mechanisms, in: Proceedings of the First IEEE Conference on Evolutionary Computation, 1994, pp. 57 – 62.
[35] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2-4) (2000) 311 – 338.
[36] J. Arabas, A. Szczepankiewicz, T. Wroniak, Experimental comparison of methods to handle boundary constraints in differential evolution, in: Parallel Problem Solving from Nature - PPSN XI, Vol. 6239 of LNCS, 2010, pp. 411 – 420.
[37] T. Takahama, S. Sakai, Efficient constrained optimization by the $\varepsilon$ constrained adaptive differential evolution, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, 2010, pp. 1 – 8.
[38] J. Zhang, A. C. Sanderson, Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization, Springer-Verlag, Berlin, 2009.
[39] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition-based differential evolution, IEEE Trans. on Evol. Comput. 12 (1) (2008) 64–79.
[40] S. M. Elsayed, R. A. Sarker, D. L. Essam, Multi-operator based evolutionary algorithms for solving constrained optimization problems, Computers & Operations Research 38 (12) (2011) 1877 – 1896.
[41] Y. Wang, Z. Cai, Y. Zhou, W. Zeng, An adaptive tradeoff model for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 80 – 92.
[42] E. Mezura-Montes, C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 1 – 17.

[43] J. Alcalá-Fdez, L. Sánchez, S. García, KEEL: A software tool to assess evolutionary algorithms to data mining problems (2012).
URL http://www.keel.es/
[44] M. Montemurro, A. Vincenti, P. Vannucci, The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms, Computer Methods in Applied Mechanics and Engineering 256 (0) (2013) 70 – 87.
[45] Y. Wang, Z. Cai, Y. Zhou, Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization, International Journal for Numerical Methods in Engineering 77 (11) (2009) 1501–1534.
[46] L. Wang, L.-p. Li, An effective differential evolution with level comparison for constrained engineering design, Structural and Multidisciplinary Optimization 41 (2010) 947–963.
[47] I. Mazhoud, K. Hadj-Hamou, J. Bigeon, P. Joyeux, Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism, Engineering Applications of Artificial Intelligence 26 (4) (2013) 1263 – 1273.
[48] V. V. de Melo, G. L. Carosio, Investigating multi-view differential evolution for solving constrained engineering design problems, Expert Systems with Applications 40 (9) (2013) 3370 – 3377.
[49] R. Rao, V. Savsani, D. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (3) (2011) 303 – 315.
[50] W. Gong, Z. Cai, C. X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics: Part B – Cybernetics 41 (2) (2011) 397–413.
[51] W. Gong, A. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: An empirical study, Information Sciences 181 (24) (2011) 5364 – 5386.
[52] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182 –1 97.
[53] H. Soh, Y.-S. Ong, Q. C. Nguyen, Q. H. Nguyen, M. Habibullah, T. Hung, J.-L. Kuo, Discovering unique, low-energy pure water isomers: Memetic exploration, optimization, and landscape analysis, IEEE Transactions on Evolutionary Computation 14 (3) (2010) 419–437.
[54] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Transactions on Evolutionary Computation 14 (3) (2010) 329–355.
[55] M. N. Le, Y. S. Ong, S. Menzel, Y. Jin, B. Sendhoff, Evolution by adapting surrogates, Evolutionary Computation 21 (2) (2013) 313–340.