Adaptive Ranking Mutation Operator based Differential Evolution for Constrained Optimization

Wenyin Gong, Zhihua Cai, and Dingwen Liang

***Abstract*—Differential evolution (DE) is a powerful evolutionary algorithm (EA) for numerical optimization. Combining with the constraint-handling techniques, recently, DE has been successfully used for the constrained optimization problems (COPs). In this paper, we propose the Adaptive Ranking Mutation OperatoR (ARMOR) for DE when solving the COPs. The ARMOR is expected to make DE converge faster and achieve feasible solutions faster. In ARMOR, the solutions are adaptively ranked according to the situation of the current population. More specifically, the population is classified into three situations, *i.e.*, infeasible situation, semi-feasible situation, and feasible situation. In the infeasible situation, the solutions are ranked only based on their constraint violations; in the semi-feasible situation, they are ranked according to the transformed fitness; while in the feasible situation, the objective function value is used to assign ranks to different solutions. In addition, the selection probability of each solution is calculated differently in different situations. The ARMOR is simple, and it can be easily combined with most of constrained DE (CDE) variants. As illustrations, we integrate our approach into three representative CDE variants to evaluate its performance. 24 benchmark functions presented in CEC 2006 and 18 benchmark functions presented in CEC 2010 are chosen as the test suite. Experimental results verify our expectation that the ARMOR is able to accelerate the original CDE variants in the majority of test cases. Additionally, ARMOR-based CDE is able to provide highly competitive results compared with other state-of-the-art EAs.**

***Index Terms*—Differential evolution, adaptive ranking mutation operator (ARMOR), constrained optimization.**

I. Introduction

MOST OF the real-world optimization problems in science and engineering involve a number of inequality and/or equality constraints, which modify the shape of the search space. These problems can be viewed as constrained optimization problems (COPs). Evolutionary algorithms (EAs) [1] have been successfully used for solving optimization problems. Although the original versions of EAs lack a mechanism to tackle constraints, coupled with constraint-handling techniques, nowadays, EAs get success when solving the COPs [2], [3], [4].

Differential evolution (DE), which was firstly proposed by Storn and Price [5], is one of the most powerful evolutionary algorithms for global numerical optimization. The advantages of DE are its ease of use, simple structure, speed, efficacy, and robustness [6], [7]. Recently, DE has obtained many successful applications in diverse domains [8], [7], [9].

In the DE algorithm, the core operator is the *differential mutation*, and generally, the parents in the mutation are always randomly chosen from the current population. Since the parent vectors in the mutation are selected randomly, it may lead DE to be good at exploring the search space and locating the region of global minimum, but be slow at exploitation of the solutions [10]. In this paper, we modify our previous proposed ranking-based mutation operator [11] and use it for solving the COPs. The Adaptive Ranking Mutation OperatoR (ARMOR) for DE is proposed, where the population is adaptively ranked according to its current situation. In different situations, different criteria are used to sort the population. Additionally, different methods are used to calculate the selection probabilities in different situations. The major advantages of ARMOR are its simplicity and generality, which makes it be easily combined with most of constrained DE (CDE) variants. As illustrations, three representative CDEs are chosen to combine with ARMOR to evaluate its performance. We choose 24 benchmark functions presented in CEC 2006 [12] and 18 benchmark functions presented in CEC 2010 [13] as the test suite. Experimental results verify our expectation that the ARMOR is able to accelerate these CDE variants in the majority of test cases.

The main contributions of this paper are as follows:

- The ARMOR, which is the core contribution of this paper, is proposed in DE for the COPs to balance the exploitation and exploration abilities of the algorithm.
- Our proposed ARMOR is integrated into three representative CDE variants (*i.e.*, ECHT-DE [14], $(\mu + \lambda)$-CDE [15], and DSS-MDE [16]) to verify its performance. Experimental results indicate that the ARMOR can make the CDE variants converge faster and achieve feasible solutions faster.
- Comprehensive experiments are conducted through benchmark functions.

The rest of this paper is organized as follows. Section II briefly introduces the constrained optimization problems and the original DE algorithm. In Section III, we describe some related constraint-handling techniques and representative CDE methods. In the next section, Section IV, our proposed ARMOR is presented in detail. In Section V, the comprehensive experiments are performed using benchmark functions to evaluate the performance of our approach. In the last section,

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61203307 and 61375066, the Natural Science Foundation of Hubei Province under Grant No. 2013CFA004, the Fundamental Research Funds for the Central Universities at China University of Geosciences (Wuhan) under Grant No. CUG130413 and CUG090109, and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20110145120009.

The authors are all with School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China. (Email: wenyingong@yahoo.com; wygong@cug.edu.cn; zhcai@cug.edu.cn).

Section VI draws the conclusions from this work and points out the possible future work.

II. Preliminaries

In this section, we first briefly introduce the constrained optimization problems (COPs) used in this work. Then, the original differential evolution algorithm is briefly described.

A. Constrained Optimization Problems

Without loss of generality, in this work, we consider the constrained minimization problem, which can be formalized 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 ($i \in \{1, \ldots, n\}$) 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, *i.e.*, $x_i \in [l_i, u_i]$.

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 a given 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

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

where $j \in \{q+1, \cdots, m\}$ and δ 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

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

B. Differential Evolution

The DE algorithm is initially proposed for the unconstrained numerical optimization problems [5]. The main procedure of DE is briefly described in the following subsections.

1) Initialization: The DE population consists of μ vectors, initially, it is generated at random. For example, for the j-th variable of the i-th vector $\mathbf{x}_i$ it is initialized as follows:

$$x_{i,j} = l_j + \text{rndreal}(0,1) \cdot (u_j - l_j) \tag{6}$$

where $i = 1, \cdots, \mu$, $j = 1, \cdots, n$, and $\text{rndreal}(0,1)$ is a uniformly distributed random real number in $(0, 1)$.

2) Mutation: After initialization, the mutation operation is applied to generate the mutant vector $\mathbf{v}_i$ for each target vector $\mathbf{x}_i$ in the current population. The classical mutation strategy is "DE/rand/1":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{7}$$

where $F \in (0, 1+)$ is the mutation scaling factor, $r_1, r_2, r_3 \in \{1, \cdots, \mu\}$ are mutually different integers randomly generated, and $r_1 \neq r_2 \neq r_3 \neq i$.

3) Crossover: In order to diversify the current population, following mutation, DE employs the crossover operator to produce the trial vector $\mathbf{u}_i$ between $\mathbf{x}_i$ and $\mathbf{v}_i$. The most commonly used operator is the *binomial* or *uniform* crossover performed on each component as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } (\text{rndreal}(0,1) < Cr \text{ or } j == j_{\text{rand}}) \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{8}$$

where Cr is the crossover rate and j_{rand} is a randomly generated integer within $\{1, n\}$. Note that the notation "$a == b$" indicates a is equal to b.

4) Selection: Finally, to keep the population size constant in the following generations, the selection operation is employed to determine whether the trial or the target vector survives to the next generation. In DE, the *one-to-one tournament selection* for unconstrained optimization problems is used:

$$\mathbf{x}_i = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i, & \text{otherwise} \end{cases} \tag{9}$$

where $f(\mathbf{x})$ is the objective function to be optimized.

III. Related Work

A. Constraint-handling Techniques

Originally, EAs lack a mechanism to deal with the constraints for the COPs. In order to solve the COPs using EAs, the constraint-handling techniques are required. Nowadays, there exists a number of constraint-handling techniques. Coello [3] provided a comprehensive survey of the most popular constraint-handling techniques currently used within EAs and grouped them into five categories: 1) penalty functions; 2) special representations and operators; 3) repair algorithms; 4) separate objective and constraints; and 5) hybrid methods. Most recently, Mezura-Montes and Coello [4] reviewed and analyzed the most relevant types of constraint-handling techniques that have been used with nature-inspired algorithms. In this subsection, we describe two techniques that will be used in this work to implement the fitness transformation.

$$f_{\text{final}}(\mathbf{x}_i) = \begin{cases} f'(\mathbf{x}_i), & \text{for feasible solution} \\ v(\mathbf{x}_i), & \text{if there is no feasible solution} \\ \sqrt{f'(\mathbf{x}_i)^2 + v(\mathbf{x}_i)^2} + [(1-\varphi)\cdot v(\mathbf{x}_i) + r_f \cdot f'(\mathbf{x}_i)], & \text{otherwise} \end{cases} \quad (14)$$

1) Improved Adaptive Trade-off Model: The improved adaptive trade-off model (IATM) was proposed by Wang and Cai [15], which is the improved version of ATM [17]. In IATM, the population contains three situations, *i.e.*, infeasible situation, semi-feasible situation, and feasible situation.

In the infeasible situation, the solutions are ranked based on their constraint violations in ascending order. Then, to get the next population, the first half ($\mu/2$) solutions are selected from the top $\mu/2$ solutions of the ranked population to steer the population to feasibility, while the other $\mu/2$ solutions are randomly chosen from the rest solutions to promote diversity.

In the semi-feasible situation, the population contains both feasible and infeasible solutions. The population is now divided into the feasible group (Z_1) and the infeasible group (Z_2) based on the feasibility of each solution. Then, the objective function value $f(\mathbf{x}_i)$ of the solution $\mathbf{x}_i$ is converted into

$$f'(\mathbf{x}_i) = \begin{cases} f(\mathbf{x}_i), & i \in Z_1 \\ \max\{\varphi \cdot f(\mathbf{x}_{\text{best}}) + (1-\varphi)\cdot f(\mathbf{x}_{\text{worst}}), f(\mathbf{x}_i)\}, & i \in Z_2 \end{cases} \quad (10)$$

where φ 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

$$f_{\text{nor}}(\mathbf{x}_i) = \frac{f'(\mathbf{x}_i) - \min\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j)}{\max\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j) - \min\limits_{j \in Z_1 \cup Z_2} f'(\mathbf{x}_j)} \quad (11)$$

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

$$G_{\text{nor}}(\mathbf{x}_i) = \begin{cases} 0, & i \in Z_1 \\ \frac{G(\mathbf{x}_i) - \min\limits_{j \in Z_2} G(\mathbf{x}_j)}{\max\limits_{j \in Z_2} G(\mathbf{x}_j) - \min\limits_{j \in Z_2} G(\mathbf{x}_j)}, & i \in Z_2 \end{cases} \quad (12)$$

Then, the final fitness function is obtained as follows

$$f_{\text{final}}(\mathbf{x}_i) = f_{\text{nor}}(\mathbf{x}_i) + G_{\text{nor}}(\mathbf{x}_i) \quad (13)$$

Afterward, μ solutions with the smallest f_{final} are selected for the next population.

In the feasible situation, since all solutions are feasible in the population, COPs can be viewed as unconstrained optimization problems. Therefore, μ solutions with the smallest objective function values are chosen for the next population.

2) Adaptive Penalty Formulation: Tessema and Yen proposed an adaptive penalty function for solving the COPs in [18]. In order to exploit infeasible individuals with low objective function value and low constraint violation, in this technique, the number of feasible solutions in the current population is used to determine the penalty value assigned to infeasible solutions.

In [18], for each solution $\mathbf{x}_i$, a final fitness value is evaluated as shown in Equation (14), where φ is the feasible ratio of the current population.

$f'(\mathbf{x}_i)$, which is the normalized fitness value of $\mathbf{x}_i$ ($i = 1, \cdots, \mu$), is formulated as:

$$f'(\mathbf{x}_i) = \frac{f(\mathbf{x}_i) - \min\limits_{j=\overline{1,\mu}} f(\mathbf{x}_j)}{\max\limits_{j=\overline{1,\mu}} f(\mathbf{x}_j) - \min\limits_{j=\overline{1,\mu}} f(\mathbf{x}_j)} \quad (15)$$

$v(\mathbf{x}_i)$ is the constraint violation that is calculated as the sum of the normalized violation of each constraint divided by the total number of constraints:

$$v(\mathbf{x}_i) = \frac{1}{m}\sum_{j=1}^{m} \frac{G_j(\mathbf{x}_i)}{G_{\max,j}} \quad (16)$$

where $G_j(\mathbf{x}_i)$ is calculated by Equation (4), and

$$G_{\max,j} = \max_{k \in \{1,\mu\}} G_j(\mathbf{x}_k).$$

As mentioned-above, we can see that the adaptive penalty formulation method is parameter-less and favors slightly infeasible solutions with a good objective function value.

B. Constrained DEs

Combining with the constraint-handling techniques, the DE algorithm has been successfully used for solving the COPs. We will briefly discuss some representative CDEs as follows.

Storn [19] proposed the constraint adaptation with DE (CADE) for the COPs, which is a multi-member DE. Lampinen presented a Pareto dominance-based constraint-handling method to handle nonlinear constraint functions [20]. Mezura-Montes *et al.* proposed a multi-member diversity-based DE (MDDE) for the COPs in [21], [22]. Similar to CADE, in MDDE each target parent is allowed to generate more than one offspring. In addition, Deb's feasibility rules [23] and a diversity mechanism are adopted to handle the constraints in MDDE. In CEC 2006 competition on constrained real parameter optimization [12], several CDE variants were presented and some of them get the front ranks. For example, εDE [24], proposed by Takahama and Sakai, ranks the first in this competition. In εDE, the ε constrained method is used to handle the constraints. In [25], 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 [26], Huang *et al.* proposed an extended SaDE method for the COPs, where the replacement criterion was modified for tackling constraints. Brest *et al.* presented a self-adaptive DE variant to solve COPs [27], 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 [28], in which 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) [16], where the comparison probability P_f decreases dynamically following the evolution process. Ali and Kajee-Bagdadi presented local exploration-based DE for solving COPs, where a periodic local exploration technique is incorporated into DE [29]. In [30], 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. To provide some insights about the behavior of DE variants for solving COPs, Mezura-Montes *et al.* presented an empirical study on CDE in [31]. 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 COPs [32], in which each constraint-handling technique has its own subpopulation. Elsayed *et al.* presented the ISDE-L method [33], where multiple search operators, constraint handling techniques, and a local search procedure are used. Wang and Cai proposed a $(\mu + \lambda)$-CDE for the COPs [15]. 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. In [34], Elsayed *et al.* proposed the SAMSDE method, in which the population is divided into a number of sub-populations and a self-adaptive learning process is used to adjust the sub-population sizes based on their success [34]. Very recently, Wang and Cai presented the CMODE method [35], in which DE is combined with multiobjective optimization to deal with COPs. Mohamed and Sabry proposed a novel constrained optimization based on modified DE algorithm (COMDE) [36], where a new directed mutation strategy is presented and a modified constraint-handling technique is employed to handle constraints. In [37], 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 [38] is periodically applied. In ISAMODE-CMA, the dynamic penalty constraint-handling technique is used to tackle constraints of a problem. Elsayed *et al.* proposed an adaptive DE variant in [39] for the COPs, where the best combination of the scaling factor F, the crossover rate Cr, and the population size μ are adaptively selected based on an adaptive mechanism. In [40], two self-adaptive DE algorithms are presented for the COPs, where a heuristic mixing of operators is incorporated.

As briefly reviewed above, different CDE variants have been proposed for the COPs in the DE literature. To obtain promising results, different techniques are integrated into DE. For example, εDE [24] combined with the gradient-based mutation; ECHT-DE [32] integrated different constraint-handling techniques into DE; ISAMODE-CMA [37] adopted mixed DE mutations and CMA-ES procedure. Generally, combining DE with other techniques is effective to enhance its performance; however, they are usually more complicated than the original DE algorithm. Based on this consideration, we will present the ARMOR technique in DE for the COPs. Unlike the previous techniques, the proposed ARMOR is simple, and it does not destroy the DE structure and not increase DE overall complexity significantly.

IV. ARMOR: Adaptive Ranking Mutation Operator

In this section, we will introduce our proposed ARMOR for the DE algorithm in detail. The core idea behind our proposed ARMOR is elucidated from four aspects.

A. Motivations

In DE, the core operator is the *differential mutation* operator. Through the mutation operator, the *mutant* vector is generated. Generally, the parents in the mutation operator are chosen randomly from the current population. 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 $r_1, r_2, r_3 \in \{1, \mu\}$ and $r_1 \neq r_2 \neq r_3 \neq i$ [5]. However, since all parents are chosen randomly, it may lead the DE algorithm to be good at exploring the search space and locating the region of global minimum, but be slow at exploitation of the solutions [10]. On the other hand, in the nature, good species always contain more useful information, and hence, they are more likely to be selected to propagate offspring. Therefore, based on the above motivations, in this work, we present the ARMOR technique for DE to utilize the better solutions to guide the search. In this way, the exploitation ability of DE can be enhanced, and hence its convergence speed is able to be accelerated.

B. Adaptive Ranking Technique

In ARMOR, we first need to rank the population. Suppose that the population is sorted from the best to the worst based on a *criterion*, then the ranking of $\mathbf{x}_i$ is assigned as follows:

$$R_i = \mu - i + 1, \quad i = 1, 2, \cdots, \mu \tag{17}$$

According to Equation (17), the best individual in the current population will obtain the highest ranking.

In order to make the ranking-based mutation operator in DE be suitable for the COPs, we modify our previous proposed ranking technique [11], 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 mentioned in [15], the population may experience three situations, *i.e.*, i) infeasible situation, where the population is entirely composed of infeasible solutions; ii) semi-infeasible situation, where the population consists of both feasible and infeasible solutions; and iii) feasible situation, where the population contains feasible solutions only. In different situations, the population is ranked as follows.

1) Ranking in Infeasible Situation: In the infeasible situation, the population contains only infeasible solutions. In this situation, we sort the population according to the constraint violations (*e.g.*, $G(\mathbf{x})$ in Equation (5)) in ascending order. The objective function values are not considered at all. Finally, the population is ranked as

$$G(\mathbf{x}_1) \leq G(\mathbf{x}_2) \leq \cdots \leq G(\mathbf{x}_\mu) \tag{18}$$

2) Ranking in Semi-feasible Situation: As suggested in [15], 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 considerations. Therefore, in order to balance the influence of objective function value and constraint violation, *fitness transformation* techniques (such as the methods presented in [41], [18], [15], etc) could be a good choice. As illustrations, in this work, we adopt the adaptive fitness transformation (AFT) method (Equation (13)) proposed in [15] and adaptive penalty formulation (APF) method (Equation (14)) proposed in [18] 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 (17). Finally, the population is ranked as

$$f_{\text{final}}(\mathbf{x}_1) \leq f_{\text{final}}(\mathbf{x}_2) \leq \cdots \leq f_{\text{final}}(\mathbf{x}_\mu) \tag{19}$$

3) Ranking in Feasible Situation: In this situation, all individuals in the population are feasible, and the COPs can be treated 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, *i.e.*, the population is ranked as

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_\mu) \tag{20}$$

To sum up, in ARMOR 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,
3) objective function values in the feasible situation.

C. Selection Probability Calculation

After obtaining the ranking of each solution $\mathbf{x}_i$, we then calculate its selection probability p_i. Different from the method presented in [11] for unconstrained optimization problems, in this work, the selection probability is calculated according to the situation of the current population for the COPs. In different situations, different methods are used to calculate the selection probability. For example, the probabilities can be calculated as follows:

- In the infeasible situation:

$$p_i = 0.5 \cdot \left[1.0 - \cos\left(\frac{R_i}{\mu} \cdot \pi\right)\right] \tag{21}$$

- In the semi-feasible situation:

$$p_i = 0.5 \cdot \left[1.0 - \cos\left(\frac{R_i}{\mu} \cdot \pi\right)\right] \tag{22}$$

- In the feasible situation:

$$p_i = \frac{\arccos\left(1.0 - 2.0 \cdot \frac{R_i}{\mu}\right)}{\pi} \tag{23}$$

where $i = 1, \cdots, \mu$.

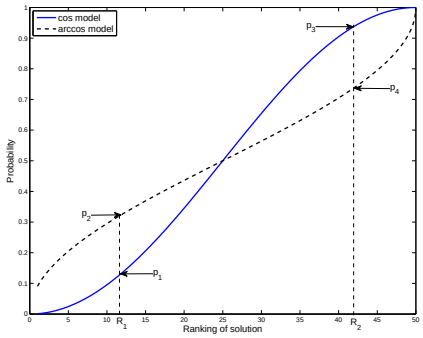


Fig. 1. Relation between the selection probability and the ranking of solution for different models. The population size $\mu = 50$ is used.

1) Explanations: In the infeasible and semi-feasible situations, the cosine model (see Equations (21) and (22)) is used to calculated the selection probability of each solution; while in the feasible situation, the inverse function of Equation (21), *i.e.*, arccosine model, is employed for probability calculation. The relation between the selection probability and the ranking of each individual for the two models is plotted in Fig. 1.

As shown in Fig. 1, let $R_1(R_2)$ be the ranking of solution $\mathbf{x}_1(\mathbf{x}_2)$ and $R_2 > R_1$, then we can get that $p_3 - p_1 > p_4 - p_2$. It means that for the two solutions there are larger probability differences in the cosine model than those in the arccosine model. Therefore, in the cosine model the selection on better solutions will be more dominant than the selection on worse solutions. On the other hand, in the arccosine model, better solutions will less dominate the worse ones due to their smaller probability differences.

Based on the analysis, in the infeasible situation, since all solutions are infeasible, the solutions with small constraint violations should get more chance to be selected to steer the population towards feasibility. In order to get the feasible solutions faster we use the cosine model to calculate the selection probabilities. In this way, better solutions with smaller constraint violations will more dominate the worse ones.

In the semi-feasible situation, some important feasible individuals and important infeasible individuals are assigned higher rankings, since these individuals contain more useful information: On the one hand, 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. Therefore, these individuals should be paid more attention and be more dominant than the worst ones. Based on this consideration, we also use the cosine model to calculate the selection probabilities as shown in Equation (22).

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, the arccosine model is employed to calculate

the selection probabilities. In this manner, better individuals will less dominate the worse ones.

It is worth noting that when the cosine and arccosine models are used to calculate the selection probabilities in different situations, no additional parameters are introduced in the ARMOR. In addition, since the ranking is assigned by Equation (17), the worst solution has a small selection probability (not 0). In this way, the worst solution will not be completely discarded in the selection process.

Algorithm 1 Ranking-based vector selection for "DE/rand/1"

Input: The target vector index i
Output: The selected vector indexes r_1, r_2, r_3
Randomly select $r_1 \in \{1, \mu\}$; {base vector index}
while $\text{rndreal}[0, 1) > p_{r_1}$ **or** $r_1 == i$ **do**
    Randomly select $r_1 \in \{1, \mu\}$;
end while
Randomly select $r_2 \in \{1, \mu\}$; {terminal vector index}
while $\text{rndreal}[0, 1) > p_{r_2}$ **or** $r_2 == r_1$ **or** $r_2 == i$ **do**
    Randomly select $r_2 \in \{1, \mu\}$;
end while
Randomly select $r_3 \in \{1, \mu\}$;
while $r_3 == r_2$ **or** $r_3 == r_1$ **or** $r_3 == i$ **do**
    Randomly select $r_3 \in \{1, \mu\}$;
end while

D. Vector Selection

As presented in [11], after calculating the selection probability of each individual, another issue is that in the mutation operator which vectors should be selected according to the selection probabilities. In this work, the *base* vector and the *terminal* point of the difference vector are selected based on their selection probabilities, while other vectors in the mutation operator are selected randomly as the original DE algorithm. For example, for the "DE/rand/1" mutation the vectors are selected as shown in Algorithm 1. From Algorithm 1 we can see that the vectors with higher rankings (or selection probabilities) are more likely to be chosen as the base vector or the terminal point in the mutation operator. Note that in Algorithm 1 we only illustrate the vector selection for "DE/rand/1", for other mutation operators the vector selection is similar to Algorithm 1.

E. Remarks

Although this work is the extension of our previous work in [11], however, there are significant differences between them: i) The work in [11] is only for unconstrained problems, while this work is for constrained problems. ii) The ranking in [11] is only based on the objective function value, while in this work since the constraints should be considered, the ranking is assigned based on different criteria in different situations. And iii) the calculation of selection probability is also different compared with [11]. In this work, different methods are used to calculate the selection probabilities in different situations.

This work is similar to the work presented in [42], however, the differences between them are significant: i) In [42], the ranking scheme classifies the solutions in the population into different groups, but in this work the ranking technique is used to assign rankings for the solutions based on different criteria. And ii) in [42] different solutions are selected from different considered groups for the crossover operator, while in this work the selection probability is used to control the selection of different solutions for the DE mutation.

Note that the additional complexity of the proposed ARMOR is population sorting and probability calculation. The complexity of population sorting is $O(\mu \cdot \log(\mu))$, and the complexity of probability calculation is $O(\mu)$. As analyzed in [43], most of DE variants have the overall complexity of $O(t_{\max} \cdot \mu \cdot n)$, where $t_{\max}$ is the maximal generations, and n is the dimension of the problem . Therefore, when combining our proposed ARMOR with CDEs, the ARMOR does not significantly increase the overall complexity of CDEs. This might make the ARMOR be easy to integrate into other DE variants for the COPs.

In order to verify the performance of ARMOR, in the following section, we will integrate it into three representative CDEs, *i.e.*, ECHT-DE [14], $(\mu + \lambda)$-CDE [15], and DSS-MDE [16]. These three CDEs are chosen due to their promising performance obtained and *nearly-pure* DE procedure. For example, they do not use the memetic procedure. This can make us focus on the effectiveness of ARMOR for the DE algorithm.

V. Experimental Results and Analysis

In this section, we perform comprehensive experiments to evaluate the performance of ARMOR. Firstly, the ARMOR is combined with ECHT-DE [14], and the method is referred to as ECHT-ARMOR-DE. ECHT-ARMOR-DE is used to optimize the functions presented in CEC 2006 [12] and CEC 2010 [13] to verify the enhanced performance of ARMOR. Then, the ARMOR is integrated into $(\mu + \lambda)$-CDE [15] and DSS-MDE [16] to test the capability of ARMOR to improve other CDEs.

A. Benchmark Functions

In this work, the benchmark functions presented in CEC 2006 [12] and CEC 2010 [13] for the competition on constrained optimization are selected as the test suite. In the CEC 2006 competition, there are 24 COPs. In the CEC 2010 competition, there are 18 scalable COPs, $D = 10$ and $D = 30$ are evaluated in this work. Due to the tight space limitation, we omit the details of these functions, interested readers can refer to them in [12] and [13], respectively.

B. Parameter Settings

For ECHT-ARMOR-DE and ECHT-DE, we use the following parameters, which are set the same as used in [14].

- population size: $\mu = 50$;
- crossover rate: $Cr \in (0.1, 0.9)$ in steps of 0.1;
- scaling factor: $F \in (0.4, 0.9)$ in steps of 0.1;

For the functions in CEC 2006, the maximal number of function evaluations (Max_NFEs) for all benchmark problems are set to be $240,000$ [37]. To compare the results of different algorithms, each function is optimized over 50 independent runs. We use the same set of initial random populations

to evaluate different algorithms, *i.e.*, all of the compared algorithms are started from the same initial population in each out of 50 runs. While for the functions in CEC 2010 at $D = 10$ and $D = 30$, the Max_NFEs are set to be $200,000$ and $600,000$ [13], respectively. For these functions, each function is optimized over 25 runs as recommended in [13].

C. Performance Criteria

In order to compare the results among different algorithms, in this work, we adopt the following performance criteria which have been presented in other literature.

- **NFEs**$_\epsilon$ [12]: 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) [12]: It is equal to the number of success runs over total number of runs. A success run means that within Max_NFEs the algorithm finds a feasible solution $\mathbf{x}$ satisfying $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 1e-4$.
- **NFEs**$_f$: It is used to record the number of function evaluations in each run for finding a feasible solution.
- **Feasible rate** (FR) [13]: It equals to the number of feasible runs over total number of runs.
- **Convergence graphs** [12]: The graphs show the median error performance ($f(\mathbf{x}) - f(\mathbf{x}^*)$) of the total number of runs.
- **Acceleration rate** (AR): Similar to the acceleration rate in [44], this criterion is used to compare the convergence speed between two algorithms. It is defined as follows:

$$AR = \frac{\mathrm{ANFEs}_{\epsilon,\mathrm{A}}/SR_{\mathrm{A}}}{\mathrm{ANFEs}_{\epsilon,\mathrm{B}}/SR_{\mathrm{B}}} \tag{24}$$

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

D. On the Convergence Rate

Firstly, the convergence rate between ECHT-DE and ECHT-ARMOR-DE is compared through the functions in CEC 2006. As mentioned in Section IV-B, in the semi-feasible situation, different fitness transformation techniques can be used to calculate the final transformed fitness $f_{\mathrm{final}}(\mathbf{x}_i)$ for each solution $\mathbf{x}_i$. In this work, the AFT method presented in [15] and the APF method presented in [18] are used as illustrations. Therefore, there are two ECHT-ARMOR-DE variants, *i.e.*, ECHT-ARMOR-DE1 with AFT method and ECHT-ARMOR-DE2 with APF method. For the two ECHT-ARMOR-DE variants, the parameters used are described in Section V-B. The NFEs$_\epsilon$, SR, and AR values of ECHT-DE, ECHT-ARMOR-DE1, and ECHT-ARMOR-DE2 are reported in Table I[2], where the overall best and the second best NFEs$_\epsilon$ values are highlighted in **gray boldface** and **boldface**, respectively. Hereinafter, since for the three algorithms there are no successful runs in functions g20 and g22, the results of these functions are not reported. The convergence curves of the selected functions are plotted in Fig. 2.

From Table I and Fig. 2, it can be observed that

- With respect to the NFEs$_\epsilon$, both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 require less NFEs$_\epsilon$ values than ECHT-DE in 21 out of 22 test functions. Only in function g11, ECHT-DE obtains slightly better NFEs$_\epsilon$ values than those of ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2.
- For SR, all of the three ECHT-DE variants can successfully solve 19 functions in all 50 runs. In function g19, there are no successful runs obtained by ECHT-DE, however, ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 can successfully solve this function in 50 and 49 runs, respectively. In g23, ECHT-DE gets $SR = 0.72$, while both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 obtain $SR = 1.0$. In g02, ECHT-DE obtains the best SR value, followed by ECHT-ARMOR-DE2 and ECHT-ARMOR-DE1. The reason might be that the ARMOR technique enhances the exploitation ability of the algorithm, yet slightly decreases its exploration ability. While for g02 it has large feasible space ($\rho = 100\%$ see [12]), the ARMOR technique may lead to the algorithm not to explore the large feasible space sufficiently.
- In terms of the accelerate rate AR, it is clear that in 19 functions both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 consistently obtain $AR > 1$ compared with ECHT-DE. The averaged AR value between ECHT-ARMOR-DE1 and ECHT-DE is 1.30, which means that ECHT-ARMOR-DE1 performs 30% faster than ECHT-DE in overall. Also, ECHT-ARMOR-DE2 provides 27% faster than ECHT-DE in overall.
- Fig. 2 clearly shows that ECHT-ARMOR-DE1 gets the fastest convergence rate, followed by ECHT-ARMOR-DE2 and ECHT-DE.

To sum up, according to the above analysis, the experimental results verified one of our expectation that the ARMOR technique is capable of accelerating the convergence rate of EHCT-DE in the majority of test functions. It is worth noting that the ARMOR technique can enhance the exploitation ability of CDE variants, however, if the DE mutations are more exploitative, it may lead to premature convergence.

E. Comparison with Other State-of-the-art EAs

In the previous section, we verified that the ARMOR technique is able to accelerate the convergence rate of EHCT-DE. In this section, the quality of the final solutions obtained by ECHT-DE, ECHT-ARMOR-DE1, and ECHT-ARMOR-DE2 is compared. In addition, they also compared with other state-of-the-art EAs for the COPs. These algorithms are AIS-ZYH [45], ISAMODE-CMA [37], SAMODE [46], ECHT-EP2 [32], εDE [24], and ATMES [17]. AIS-ZYH [45] is an artificial immune system based approach for the COPs. SAMODE [46] is a multiple search operators based DE, where different operators are selected adaptively. ISAMODE-CMA [37] is an improved version of SAMODE. In

[1]Indeed, $\mathrm{ANFEs}_{\epsilon,\mathrm{A}}/SR_{\mathrm{A}}$ is the successful performance (SP) of algorithm A as presented in [31]. It can be used to measure the speed and reliability of an algorithm.

[2]Due to the tight space limitation, all of the experimental results, including the figures and tables, are provided in the supplemental file.

ISAMODE-CMA, both mixed mutation operators and CMA-ES based local search are implemented. ECHT-EP2 [32] is evolutionary programming based on ensemble of constraint-handling techniques. εDE [24], which gets the first ranking in the CEC 2006 competition, is a ε constrained DE with gradient-based mutation and feasible elites. ATMES [17] is an adaptive trade-off model based evolution strategy for the COPs. We choose these EAs for comparisons due to their good performance obtained. The Max_NFEs of AIS-ZYH are $350,000$, the Max_NFEs of εDE are $500,000$, while other 8 EAs have the Max_NFEs $= 240,000$.

The mean and standard deviation of the objective function values for each algorithm are shown in Table II. "NA" means not available. Note that the results of AIS-ZYH, ISAMODE-CMA, SAMODE, ECHT-EP2, εDE, and ATMES are directly obtained from their corresponding literature. In addition, for ECHT-DE, ECHT-ARMOR-DE1, ECHT-ARMOR-DE2, AIS-ZYH, ISAMODE-CMA, SAMODE, ECHT-EP2, and εDE, based on the mean values in Table II in the 22 functions, the final rankings obtained by the Friedman test[3] are shown in Fig. 3. In addition, due to the importance of the multiple-problem statistical analysis [48], we present the results of the multiple-problem Wilcoxon signed-rank test in Table III, where "$\bullet$" means that the method in the row improves the method of the column, and "$\circ$" means that the method in the column improves the method of the row. Upper diagonal of level significance at $\alpha = 0.1$, and lower diagonal level of significance at $\alpha = 0.05$.

According to the results show in Table II, it is clearly seen that both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 consistently obtain highly-competitive results in all functions compared with other seven EAs. Both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 can get the optimal solutions in 20 functions in all runs.

With respect to the average rankings of different algorithms by the Friedman test, the p-value is 0.209197, which means that there are no significant differences for the compared algorithms in all functions. Additionally, Fig. 3 shows that εDE gets the first ranking among eight algorithms, while ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 obtain the second and third ranking, respectively. Although εDE is the best one among the compared algorithms, it requires the most Max_NFEs[4]. In addition, in εDE the gradient-based mutation is used to enhance its performance.

Based on the multiple-problem analysis by the Wilcoxon test, Table III reveals that both ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 significantly outperform AIS-ZYH and SAMODE. εDE obtains significantly better results than AIS-ZYH, SAMODE, and ECHT-EP2. SAMODE is the worst one, which has been significantly outperformed by five out of seven methods. Additionally, Table III also shows that there are no significant differences among ECHT-ARMOR-DE1, ECHT-ARMOR-DE2, ECHT-DE, ISAMODE-CMA, and εDE by the Wilcoxon test. The reason is that in the majority of the CEC 2006 functions, the six algorithms can obtain the optimal solutions in all runs as shown in Table II.

In general, the ARMOR technique improves the performance of ECHT-DE in terms of the quality of the final solutions. The ARMOR-based ECHT-DE variants provide highly-competitive results compared with other EAs in the CEC 2006 test suite.

Based on the results in Sections V-D and V-E, we see that ECHT-ARMOR-DE1 gets slightly better results than ECHT-ARMOR-DE2, in the following sections, we mainly focus on the ECHT-ARMOR-DE1 method, and for simplicity, we use ECHT-ARMOR-DE for short.

F. Experiments on CEC 2010 Benchmark Functions

From the experimental results on the CEC 2006 test suite, we verified the enhanced performance of ARMOR in terms of the quality of final solutions and the convergence rate. To better understand the performance of ARMOR, ECHT-ARMOR-DE is evaluated on the functions in CEC 2010 at $D = 10$ and $D = 30$. The results of ECHT-ARMOR-DE are compared with those of ECHT-DE as reported in [14]. In addition, ECHT-ARMOR-DE and ECHT-DE are also compared with other EAs, *i.e.*, AIS-ZYH [45], εDEg[5] [49], and IEMA [50]. Note that the results of ECHT-DE, AIS-ZYH, εDEg, and IEMA are directly retrieved from their corresponding literature.

1) Results at $D = 10$: Table IV reports the results of ECHT-DE and ECHT-ARMOR-DE for the functions in CEC 2010 at $D = 10$, where the better mean values are highlighted in **boldface**. The comparisons among different EAs are reported in Table V in terms of the mean values of final solutions. Based on the mean values shown in Table V, the average rankings of these algorithms[6] by the Friedman test are described in Fig. 4(a), and the results of the multiple-problem analysis by the Wilcoxon test are shown in Table VI.

From Table IV, in terms of the mean results we see that in 11 out of 18 functions ECHT-ARMOR-DE is better than ECHT-DE. In 4 functions both of them can obtain the optimal solutions in all 25 runs. Only in two functions C08 and C09, ECHT-DE gets better results than ECHT-ARMOR-DE. Considering the feasible rate, in 17 functions ECHT-ARMOR-DE obtains $FR = 1.0$, while ECHT-DE gets $FR = 1.0$ in 16 functions.

Comparison with other EAs, Table V shows that ECHT-ARMOR-DE can obtain the first best mean values in 8 functions, and in 3 functions it gets the second best mean results. According to the averaging rankings shown in Fig. 4(a) we can see that ECHT-ARMOR-DE gets the first ranking, followed by εDEg, AIS-ZYH, IEMA, and ECHT-DE. The results

[3]The statistic results of the Friedman test and the Wilcoxon test are calculated by the KEEL software tool [47].

[4]To make a fair comparison between ECHT-ARMOR-DE1 and εDE, the Max_NFEs=$500,000$ is used for ECHT-ARMOR-DE1. In this case, for g02, ECHT-ARMOR-DE1 gets the mean and standard deviation values with -0.8035440 and $2.24E-05$, respectively. For g19, ECHT-ARMOR-DE1 can obtain the optimal solution (32.65559) in all 50 runs. For other 20 functions in Table II, both ECHT-ARMOR-DE1 and εDE get the global optimal solutions in all runs. Thus, the performance of ECHT-ARMOR-DE1 can be highly-competitive to that of εDE in the CEC 2006 test suite when the Max_NFEs=$500,000$.

[5]εDEg is a ε constrained DE with gradient-based mutation, which obtains the first overall ranking in the CEC 2010 competition [51].

[6]Since there are infeasible solutions in C11 and C12 for ECHT-DE, the results of these two functions are not used for statistical analysis.

in Table VI indicate that ECHT-ARMOR-DE is competitive to AIS-ZYH and εDEg. ECHT-ARMOR-DE is significantly better than ECHT-DE and IEMA by the Wilcoxon test at $\alpha = 0.05$. εDEg is significantly better than ECHT-DE and IEMA by the Wilcoxon test at $\alpha = 0.1$ and $\alpha = 0.05$, respectively.

2) Results at $D = 30$: For the CEC 2010 benchmark functions at $D = 30$, the quality of final solutions of ECHT-DE and ECHT-ARMOR-DE are tabulated in Table VII. Table VIII shows the comparisons with other EAs in terms of the mean values. Additionally, the averaging rankings by the Friedman test and the multiple-problem analysis by the Wilcoxon test are reported in Fig. 4(b) and Table IX[7], respectively.

Similar to the results at $D = 10$, ECHT-ARMOR-DE can obtain better mean results than ECHT-DE in the majority of the test functions (11 out of 18). It is worse than ECHT-DE in four functions (C01, C04, C10, and C17).

Table VIII shows that ECHT-ARMOR-DE gets the first best mean values in 4 functions, and the second best results in 2 functions. εDEg respectively obtains the first and second best mean values in 8 and 2 functions. For AIS-ZYH, in 4 functions it gets the first best results, and in 8 functions it obtains the second best results. The average rankings shown in Fig. 4(b) tells that εDEg is the overall best method, followed by AIS-ZYH, ECHT-ARMOR-DE, and ECHT-DE. According the multiple-problem analysis by the Wilcoxon test, the results in Table IX indicate that ECHT-ARMOR-DE, εDEg, and AIS-ZYH significantly outperform ECHT-DE at $\alpha = 0.05$. There are no significant differences among ECHT-ARMOR-DE, εDEg, and AIS-ZYH in terms of the Wilcoxon test.

3) Remarks: By integrating the ARMOR into ECHT-DE, ECHT-ARMOR-DE improves the performance of ECHT-DE in the CEC 2010 benchmark function at $D = 10$ and $D = 30$. Especially, at $D = 10$ ECHT-ARMOR-DE obtains the first average ranking by the Friedman test. The performance of ECHT-DE and ECHT-ARMOR-DE decreases at $D = 30$, however, ECHT-ARMOR-DE still gets better results than ECHT-DE.

It is worth mentioning that the main contribution of this work is the proposed ARMOR, not to propose a competitor. The ARMOR is integrated into ECHT-DE to evaluate the enhanced performance of ARMOR, and it may be useful to improve other CDEs. We will try to verify this expectation in the following section.

G. Discussions

In the previous sections, the performance of ARMOR is combined with ECHT-DE, and ECHT-ARMOR-DE improves ECHT-DE and provides competitive results compared with other EAs in the CEC 2006 and CEC 2010 benchmark functions. In this section, we address four other issues as follows.

[7]Also, C11 and C12 are not used due to the infeasible solutions in these functions. In addition, IEMA is not used for statistical comparison, because in four functions it can not find any feasible solutions in all runs.

1) ARMOR for Other CDEs: As discussed in Section III-B, there are other CDEs for the COPs. Thus, we might be asked that "Is the ARMOR useful to other CDEs?" In order to answer this question, in this section, ARMOR is integrated into two representative CDEs, *i.e.*, (μ+λ)-CDE [15] and DSS-MDE [16]. The two ARMOR-based variants are respectively referred to as (μ+λ)-ARMOR-CDE and DSS-ARMOR-MDE. For ($\mu+\lambda$)-ARMOR-CDE, the parameters are set to the same as used in [15], *e.g.*, $\mu = 70, \lambda = 210, F = 0.8, Cr = 0.9$. For DSS-ARMOR-MDE, the population size is set to be $\mu = 90$, and other parameters are kept the same as used in [16], *i.e.*, $Cr = 0.9, F = \text{rndreal}(0.3, 0.9)$, and $n_o = 5$. We use a larger population size than that of DSS-MDE ($\mu = 50$), because both DSS-MDE and DSS-ARMOR-MDE obtain higher success rates than small population size used in the original DSS-MDE. For both (μ+λ)-ARMOR-CDE and DSS-ARMOR-MDE, Max_NFEs=$500,000$. The two algorithms are performed over 50 independent runs for each function. Note that in ARMOR the AFT method is used to calculate the transformed fitness in the semi-feasible situation. The results of NFEs$_\epsilon$, SR, and AR are tabulated in Table X, and the better results are highlighted in **boldface**. Note that the results of ($\mu + \lambda$)-CDE are obtained from [15] in Table 6. While for DSS-MDE, we performed it in all functions due to the changed population size.

Compared (μ+λ)-ARMOR-CDE with (μ+λ)-CDE, Table X clearly shows that ($\mu + \lambda$)-ARMOR-CDE requires less NFEs$_\epsilon$ than (μ+λ)-CDE in the majority of test cases. In 20 out of 22 functions, ($\mu + \lambda$)-ARMOR-CDE is better than ($\mu + \lambda$)-CDE in terms of the mean NFEs values. Only in two functions g05 and g15, ($\mu + \lambda$)-CDE is slightly better than ($\mu + \lambda$)-ARMOR-CDE. In 20 functions, both ($\mu + \lambda$)-ARMOR-CDE and ($\mu + \lambda$)-CDE can solve them successfully in all runs. In the rest two functions, ($\mu + \lambda$)-ARMOR-CDE obtains higher SR value than ($\mu + \lambda$)-CDE in g21, while it loses seriously in g02. In terms of the AR criterion, the average AR is 1.43, which means that (μ+λ)-ARMOR-CDE converges 43% faster than ($\mu + \lambda$)-CDE in overall.

Comparisons on the performance between DSS-ARMOR-MDE and DSS-MDE, from Table X, we can observe that in all test functions DSS-ARMOR-MDE needs less mean NFEs$_\epsilon$ than DSS-MDE. With respect to SR, DSS-ARMOR-MDE performs better than DSS-MDE in two functions (g21 and g23), while DSS-MDE wins in two functions (g02 and g13). In the rest 18 functions, both DSS-ARMOR-MDE and DSS-MDE get the $SR = 1.0$. The average SR of DSS-ARMOR-MDE is slightly better than that of DSS-MDE. In addition, DSS-ARMOR-MDE performs 104% faster than DSS-MDE in overall, since the average $AR = 2.04$.

Therefore, from the above results and analysis, we can say that our proposed ARMOR is of benefit to ($\mu + \lambda$)-CDE and DSS-MDE. It accelerates their convergence speed, but does not decrease the success rate in the majority of the functions. Hence, the ARMOR can be similarly useful for performance enhancement of other CDEs.

2) On the Robustness: The robustness of the proposed ARMOR is also important to indicate its effectiveness. As mentioned in Section IV-C, there are no additional parameters

introduced in ARMOR; hence, to evaluate its robustness, we will modify the parameters of DE to show how quality of solutions changed. Since in ECHT-DE [32] both Cr and F are adaptively controlled, in this section, we use $(\mu+\lambda)$-CDE [15] as the basis algorithm. In the DE literature for the COPs (such as [24], [16], [15]), $Cr = 0.9$ has obtained very promising results. In addition, as suggested in [31], the most useful F values are $0.6 \leq F \leq 0.9$. Therefore, we fix $Cr = 0.9$; while $F = 0.7$ and $F = 0.9$ are used in $(\mu + \lambda)$-CDE and $(\mu + \lambda)$-ARMOR-CDE to evaluate the robustness. All other parameters are kept the same as used in [15]. The results are reported in Table XI.

According to the results in Table XI, we can see that regardless of different F values $(\mu + \lambda)$-ARMOR-CDE is able to accelerate the convergence rate of $(\mu + \lambda)$-CDE in the majority of the functions. The averaged AR value is 1.33 and 1.31 for $F = 0.7$ and $F = 0.9$, respectively. Thus, we can conclude that the ARMOR is still capable of accelerating $(\mu + \lambda)$-CDE with different F values. And the robustness of ARMOR is not significantly influenced by different F settings in $(\mu + \lambda)$-CDE.

3) Influence of Other Probability Calculation Models: In Section IV-C, the cosine and arccosine models are used to calculate the selection probabilities in different situations. Other models may also be used for probability calculation. To address this issue, the following models are used.

- In the infeasible situation:

$$p_i = \begin{cases} 1.0, & 1 \leq i < \frac{\mu}{2} \\ \frac{R_i}{\frac{\mu}{2}}, & \frac{\mu}{2} \leq i \leq \mu \end{cases} \tag{25}$$

- In the semi-feasible situation:

$$p_i = \left(\frac{R_i}{\mu}\right)^{2.0}, \quad i = 1, \cdots, \mu \tag{26}$$

- In the feasible situation:

$$p_i = \left(\frac{R_i}{\mu}\right)^{0.5}, \quad i = 1, \cdots, \mu \tag{27}$$

The above models are also based on the considerations that: i) In the infeasible situation, the main task is to steer the population into feasible space. ii) In the semi-feasible situation, both important feasible solutions and important infeasible solutions are more dominant than the worse ones. And iii) in the feasible situation, better solutions less dominate worse solutions to promote the diversity.

The ARMOR with these models are also integrated into ECHT-DE [14], two ECHT-ARMOR-DE variants, *i.e.*, ECHT-ARMOR-DE3 with AFT method and ECHT-ARMOR-DE4 with APF method, are evaluated in the CEC 2006 functions. For ECHT-ARMOR-DE3 and ECHT-ARMOR-DE4, the parameter settings are the same as described in Section V-B. The results of NFEs_ϵ, SR, and AR are reported in Table XII. All results are averaged over 50 independent runs.

Like the results in Section V-D, in the majority of functions both ECHT-ARMOR-DE3 and ECHT-ARMOR-DE4 requires less NFEs_ϵ values. Moreover, they can provide higher mean SR values than ECHT-DE. In terms of the AR values, the average AR are 1.26 and 1.21 for ECHT-DE vs ECHT-ARMOR-DE3 and ECHT-DE vs ECHT-ARMOR-DE4, respectively. Therefore, the above models can also accelerate the convergence rate of ECHT-DE in overall.

4) On the NFEs_f Performance: Another issue is that "Can the ARMOR make the algorithm achieve feasible solutions faster?" In order to answer this question, in this section, the NFEs_f, FR, and AR' values of ECHT-DE, ECHT-ARMOR-DE1, and ECHT-ARMOR-DE3[8] are compared in Table XIII. All results are averaged over 50 runs. AR' is similar to AR and it is calculated as

$$AR' = \frac{\text{ANFEs}_{\text{f,A}}/FR_{\text{A}}}{\text{ANFEs}_{\text{f,B}}/FR_{\text{B}}} \tag{28}$$

where $\text{ANFEs}_{\text{f,A}}$ and FR_{A} are respectively the average NFEs_f and FR values of algorithm A. $AR' > 1$ indicates algorithm B gets feasible solution faster than algorithm A. Note that in Table XIII, the results of g02, g04, g12, g19, and g24 are omitted, because in these functions the feasible solutions are obtained in the initial population. The results of g20 are not reported in Table XIII, since the three algorithms can not find feasible solutions in all runs.

From the results shown in Table XIII, we can observe that ECHT-ARMOR-DE1 requires less mean NFEs_f values than ECHT-DE in 12 out of 18 functions, while it is worse than ECHT-DE in 6 functions. In function g22, there are no feasible solutions found by ECHT-DE, however, ECHT-ARMOR-DE1 can provide $FR = 0.88$. $AR' = 1.11$ between ECHT-DE and ECHT-ARMOR-DE1, which means that ECHT-ARMOR-DE1 achieves the feasible solutions 11% faster than ECHT-DE in overall. Compared the results between ECHT-DE and ECHT-ARMOR-DE3, similar results can be obtained. ECHT-ARMOR-DE3 wins in 13 functions with respect to the mean NFEs_f values, and it gets the feasible solutions 7% faster than ECHT-DE in overall.

In general, for the two methods to calculate the selection probabilities in ARMOR, the ARMOR-based ECHT-DE variants can achieve feasible solutions faster than ECHT-DE in the majority of test functions. In addition, the ARMOR technique can make ECHT-DE obtain the feasible solutions in some functions with many active constraints, such as g22.

VI. Conclusions and Future Work

In order to accelerate the convergence rate and achieve feasible solutions faster for the constrained DE methods when solving the COPs, in this paper, we propose an adaptive ranking mutation operator (ARMOR). In ARMOR, based on the situation of the current population, the solutions are adaptively ranked. In addition, in different situations the calculation of selection probability of each solution is different, and different models can be used to calculate the probabilities. The ARMOR is simple, and does not increase the overall complexity of CDEs. It is easy to integrate into most of

[8]In the infeasible situation, the population is ranked only based on the constraint violation of each individual. In this way, ECHT-ARMOR-DE1 and ECHT-ARMOR-DE2 (also ECHT-ARMOR-DE3 and ECHT-ARMOR-DE4) have the same NFEs_f and FR values in the same function. Hence, the results of ECHT-DE are compared with those of ECHT-ARMOR-DE1 and ECHT-ARMOR-DE3.

CDEs, and as illustrations, it is combined with ECHT-DE, $(\mu + \lambda)$-CDE, and DSS-MDE. Experimental results verified our expectations that the proposed ARMOR is able to make the CDEs converge faster and find feasible solution faster. Additionally, ECHT-ARMOR-DE provides fairly-competitive results compared with other state-of-the-art EAs in the CEC 2006 and CEC 2010 benchmark functions.

The ranking-based mutation operators may be useful in 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.

ACKNOWLEDGMENTS

The authors are grateful to Dr. Wang for his constructive suggestions. They also thank Prof. Luo and Prof. Suganthan for kindly providing the source codes of DSS-MDE and ECHT-DE, respectively. The source codes of $(\mu + \lambda)$-CDE are available online by Dr. Wang.

REFERENCES

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, Apr 1997.

[2] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, 1996.

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

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

[5] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[6] R. Storn, "On the usage of differential evolution for function optimization," in *1996 Biennial Conference of the North American Fuzzy Information Processing Society (AFIPS.)*, 1996, pp. 519 – 523.

[7] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61 – 106, 2010.

[8] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer-Verlag, 2005.

[9] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. on Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[10] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. on Evol. Comput.*, vol. 12, no. 1, pp. 107–125, Feb 2008.

[11] W. Gong and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2066 – 2081, 2013.

[12] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2006.

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

[14] R. Mallipeddi and P. Suganthan, "Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems," in *IEEE Congress on Evolutionary Computation (CEC), 2010*, 2010, pp. 1–8.

[15] Y. Wang and Z. Cai, "Constrained evolutionary optimization by means of $(\mu+\lambda)$-differential evolution and improved adaptive trade-off model," *Evolutionary Computation*, vol. 19, no. 2, pp. 249 – 285, 2011.

[16] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043 – 3074, 2008.

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

[18] B. Tessema and G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 3, pp. 565 – 578, may 2009.

[19] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22 – 34, Apr 1999.

[20] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1468 – 1473.

[21] E. Mezura-Montes, J. Velázquez-Reyes, and 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.

[22] E. Mezura-Montes, C. A. Coello Coello, J. Velázquez-Reyes, and L. Muñoz-Dávila, "Multiple trial vectors in differential evolution for engineering design," *Engineering Optimization*, vol. 39, no. 5, pp. 567–589, 2007.

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

[24] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 1 – 8.

[25] E. Mezura-Montes, J. Velazquez-Reyes, and C. Coello Coello, "Modified differential evolution for constrained optimization," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 25 –32.

[26] V. Huang, A. Qin, and P. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 17 – 24.

[27] J. Brest, V. Zumer, and M. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 215 – 222.

[28] F. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 340 – 356, 2007.

[29] M. Ali and Z. Kajee-Bagdadi, "A local exploration-based differential evolution algorithm for constrained global optimization," *Applied Mathematics and Computation*, vol. 208, no. 1, pp. 31 – 48, 2009.

[30] E. Mezura-Montes and A. Palomeque-Ortiz, "Self-adaptive and deterministic parameter control in differential evolution for constrained optimization," in *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence, E. Mezura-Montes, Ed. Springer Berlin Heidelberg, 2009, vol. 198, pp. 95–120.

[31] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: An empirical study," *Information Sciences*, vol. 10, no. 22, pp. 4223–4262, November 2010.

[32] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. on Evol. Comput.*, vol. 14, no. 4, pp. 561 – 579, 2010.

[33] S. M. Elsayed, R. A. Sarker, and D. Essam, "Integrated strategies differential evolution algorithm with a local search for constrained optimization," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 2618–2625.

[34] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "On an evolutionary approach for constrained optimization problem solving," *Appl. Soft Comput.*, vol. 12, no. 10, pp. 3208–3227, 2012.

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

[36] A. W. Mohamed and H. Z. Sabry, "Constrained optimization based on modified differential evolution algorithm," *Information Sciences*, vol. 194, pp. 171 – 208, 2012.

[37] S. Elsayed, R. Sarker, and D. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 89 – 99, 2013.
[38] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1 – 18, 2003.
[39] S. Elsayed, R. Sarker, and T. Ray, "Parameters adaptation in differential evolution," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1 – 8.
[40] S. Elsayed, R. Sarker, and D. Essam, "Self-adaptive differential evolution incorporating a heuristic mixing of operators," *Computational Optimization and Applications*, pp. 1–20, 2012, in press.
[41] R. Farmani and J. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445 – 455, Oct. 2003.
[42] E. Z. Elfeky, R. A. Sarker, and D. Essam, "Analyzing the simple ranking and selection process for constrained evolutionary optimization," *J. Comput. Sci. Technol.*, vol. 23, no. 1, pp. 19–34, 2008.
[43] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*. Berlin: Springer-Verlag, 2009.
[44] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Trans. on Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb 2008.
[45] W. Zhang, G. Yen, and Z. He, "Constrained optimization via artificial immune system," *IEEE Transactions on Cybernetics*, vol. 44, no. 2, pp. 185 – 198, 2014.
[46] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers & Operations Research*, vol. 38, no. 12, pp. 1877 – 1896, 2011.
[47] J. Alcalá-Fdez, L. Sánchez, and S. García, "KEEL: A software tool to assess evolutionary algorithms to data mining problems," 2012. [Online]. Available: http://www.keel.es/
[48] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3 – 18, 2011.
[49] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," in *IEEE Congress on Evolutionary Computation (CEC), 2010*, 2010, pp. 1–9.
[50] H. Singh, T. Ray, and W. Smith, "Performance of infeasibility empowered memetic algorithm for CEC 2010 constrained optimization problems," in *IEEE Congress on Evolutionary Computation (CEC), 2010*, 2010, pp. 1–8.
[51] P. N. Suganthan, "Comparison of results on the 2010 CEC benchmark function set," Nanyang Technological University, Singapore, Tech. Rep., 2010.
[52] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182 –1 97, 2002.

SUPPLEMENT FILE

TABLE I

COMPARISON ON THE NFE$_\epsilon$ VALUES OF ECHT-DE, ECHT-ARMOR-DE1, AND ECHT-ARMOR-DE2 FOR THE CEC 2006 FUNCTIONS. ALL RESULTS ARE AVERAGED OVER 50 RUNS.

Prob	ECHT-DE (1)			ECHT-ARMOR-DE1 (2)			ECHT-ARMOR-DE2 (3)			AR	AR
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	(1) vs (2)	(1) vs (3)
g01	1.384E+05	4.06E+03	1.00	**9.739E+04**	3.14E+03	1.00	**1.064E+05**	3.28E+03	1.00	1.42	1.30
g02	8.205E+04	6.25E+03	**0.42**	**6.689E+04**	1.41E+04	0.30	**7.672E+04**	3.34E+04	**0.40**	0.88	1.02
g03	1.161E+05	1.53E+03	1.00	**1.136E+05**	3.20E+03	1.00	**1.149E+05**	2.64E+03	1.00	1.02	1.01
g04	6.470E+04	2.43E+03	1.00	**4.283E+04**	1.45E+03	1.00	**4.599E+04**	1.75E+03	1.00	1.51	1.41
g05	1.204E+05	1.45E+03	1.00	**1.195E+05**	8.75E+02	1.00	**1.201E+05**	1.23E+03	1.00	1.01	1.00
g06	2.224E+04	1.24E+03	1.00	**1.536E+04**	7.04E+02	1.00	**1.630E+04**	6.67E+02	1.00	1.45	1.36
g07	1.088E+05	4.87E+03	1.00	**6.718E+04**	2.80E+03	1.00	**7.013E+04**	2.97E+03	1.00	1.62	1.55
g08	2.644E+03	4.15E+02	1.00	**2.216E+03**	3.07E+02	1.00	**2.340E+03**	3.77E+02	1.00	1.19	1.13
g09	4.194E+04	1.81E+03	1.00	**2.902E+04**	1.15E+03	1.00	**3.054E+04**	1.14E+03	1.00	1.44	1.37
g10	1.855E+05	7.49E+03	1.00	**1.023E+05**	3.75E+03	1.00	**1.071E+05**	3.47E+03	1.00	1.81	1.73
g11	**5.820E+04**	1.85E+04	1.00	**6.265E+04**	1.55E+04	1.00	6.272E+04	1.42E+04	1.00	0.93	0.93
g12	3.072E+03	7.83E+02	1.00	**2.552E+03**	5.69E+02	1.00	**2.424E+03**	6.06E+02	1.00	1.20	1.27
g13	1.109E+05	4.39E+03	1.00	**1.092E+05**	4.70E+03	1.00	**1.092E+05**	4.70E+03	1.00	1.01	1.01
g14	1.401E+05	6.59E+03	1.00	**1.302E+05**	3.33E+03	1.00	**1.318E+05**	3.26E+03	1.00	1.08	1.06
g15	1.083E+05	5.58E+03	1.00	**1.074E+05**	5.01E+03	1.00	**1.076E+05**	4.96E+03	1.00	1.01	1.01
g16	3.019E+04	1.37E+03	1.00	**2.005E+04**	1.06E+03	1.00	**2.200E+04**	1.04E+03	1.00	1.51	1.37
g17	1.174E+05	1.59E+03	1.00	**1.167E+05**	1.04E+03	1.00	**1.170E+05**	1.12E+03	1.00	1.01	1.00
g18	1.431E+05	2.02E+04	1.00	**8.000E+04**	9.28E+03	1.00	**7.895E+04**	1.09E+04	1.00	1.79	1.81
g19	NA	NA	0.00	**1.830E+05**	1.60E+04	**1.00**	**1.868E+05**	1.73E+04	**0.98**	NA	NA
g21	1.734E+05	6.82E+03	1.00	**1.476E+05**	1.34E+04	1.00	**1.517E+05**	3.10E+03	1.00	1.17	1.14
g23	2.274E+05	5.72E+03	0.72	**1.706E+05**	4.29E+03	**1.00**	**1.751E+05**	4.19E+03	**1.00**	1.85	1.80
g24	8.120E+03	7.81E+02	1.00	**6.032E+03**	4.97E+02	1.00	**6.348E+03**	5.48E+02	1.00	1.35	1.28
avg	–		0.915	–		**0.968**	–		**0.972**	**1.30**	**1.27**

TABLE II

COMPARED THE QUALITY OF FINAL SOLUTIONS OF OUR APPROACH WITH OTHER STATE-OF-THE-ART EAS FOR ALL CEC 2006 FUNCTIONS.

Prob	Criteria	ECHT-ARMOR-DE1	ECHT-ARMOR-DE2	EHCT-DE	AIS-ZYH	ISAMODE-CMA	SAMODE	ECHT-EP2	εDE	ATMES
g01	Mean	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.60E-14
g02	Mean	-0.7900342	-0.7952637	-0.7936387	**-0.8021930**	-0.79244	-0.79873521	-0.799822	**-0.8036191**	-0.790148
	Std	1.23E-02	8.10E-03	1.12E-02	5.19E-10	2.80E-02	8.80E-03	1.26E-02	1.75E-08	1.30E-02
g03	Mean	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1.0005**	**-1**
	Std	0.00E+00	0.00E+00	0.00E+00	1.77E-11	0.00E+00	0.00E+00	0.00E+00	2.96E-31	5.90E-05
g04	Mean	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.5386**	**-30665.539**	**-30665.539**	**-30665.539**
	Std	0.00E+00	0.00E+00	0.00E+00	3.69E-13	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.40E-12
g05	Mean	**5126.497**	**5126.497**	**5126.497**	**5126.498**	**5126.497**	**5126.497**	**5126.497**	**5126.497**	5127.648
	Std	0.00E+00	0.00E+00	0.00E+00	1.70E-02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.80E+00
g06	Mean	**-6961.813875**	**-6961.813875**	**-6961.814**	-6961.81385	**-6961.813875**	**-6961.813875**	**-6961.814**	**-6961.813875**	**-6961.814**
	Std	0.00E+00	0.00E+00	0.00E+00	1.90E-12	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.60E-12
g07	Mean	**24.3062**	**24.3062**	**24.3062**	24.3557	**24.3062**	24.3096	**24.3063**	**24.3062**	24.316
	Std	0.00E+00	0.00E+00	1.14E-10	8.20E-03	0.00E+00	1.59E-03	3.19E-05	2.18E-15	1.10E-02
g08	Mean	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.09582504**	**-0.095825**	**-0.095825**	**-0.095825**
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.61E-08	1.23E-32	2.80E-17
g09	Mean	**680.63**	**680.63**	**680.63**	680.65	**680.63**	**680.63**	**680.63**	**680.63**	**680.639**
	Std	0.00E+00	0.00E+00	0.00E+00	1.20E-08	0.00E+00	1.16E-05	0.00E+00	0.00E+00	1.00E-02
g10	Mean	**7049.24802**	**7049.24802**	**7049.24802**	7049.57032	**7049.24802**	7059.81345	**7049.249**	**7049.24802**	7250.437
	Std	0.00E+00	0.00E+00	4.18E-07	4.50E-04	5.42E-06	7.86E+00	6.60E-04	4.24E-13	1.20E+02
g11	Mean	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.75**
	Std	0.00E+00	0.00E+00	0.00E+00	1.40E-08	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.40E-04
g12	Mean	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E-03
g13	Mean	**0.05394**	**0.05394**	**0.05394**	**0.05394**	**0.05394**	**0.05394**	**0.05394**	**0.05394**	**0.05396**
	Std	0.00E+00	0.00E+00	0.00E+00	7.80E-10	0.00E+00	1.75E-08	1.00E-12	0.00E+00	1.30E-05
g14	Mean	**-47.764888**	**-47.764888**	**-47.764888**	-47.764881	**-47.764888**	-47.68115	-47.7648	**-47.764888**	NA
	Std	0.00E+00	0.00E+00	3.26E-13	1.00E-12	0.00E+00	4.04E-02	2.72E-05	1.39E-15	NA
g15	Mean	**961.71502**	**961.71502**	**961.71502**	**961.71502**	**961.71502**	**961.71502**	**961.71502**	**961.71502**	NA
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.01E-13	0.00E+00	NA
g16	Mean	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	NA
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.12E-10	1.58E-30	NA
g17	Mean	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	**8853.5397**	NA
	Std	0.00E+00	0.00E+00	0.00E+00	1.90E-09	0.00E+00	1.15E-05	2.13E-08	1.21E-27	NA
g18	Mean	**-0.866025**	**-0.866025**	**-0.866024**	**-0.866025**	**-0.866025**	**-0.866024**	**-0.866025**	**-0.866025**	NA
	Std	0.00E+00	0.00E+00	5.15E-06	1.30E-15	0.00E+00	7.04E-07	1.00E-09	2.18E-17	NA
g19	Mean	**32.65560**	**32.65560**	32.65654	**32.65559**	**32.65559**	32.75734	32.66230	**32.65560**	NA
	Std	8.39E-06	1.90E-05	7.76E-04	0.00E+00	6.46E-07	6.15E-02	3.40E-03	1.26E-05	NA
g21	Mean	**193.72451**	**193.72451**	**193.72451**	196.72451	**193.72451**	193.771375	**193.7438**	**193.72451**	NA
	Std	0.00E+00	0.00E+00	0.00E+00	1.10E+00	0.00E+00	1.96E-02	1.65E-02	3.34E-14	NA
g23	Mean	**-400.0551**	**-400.0551**	**-400.0546**	-399.8743	-395.62403	-360.817656	-373.2178	**-400.0551**	NA
	Std	2.68E-09	1.76E-08	2.18E-03	2.00E+00	7.79E+00	1.96E+01	3.37E+01	1.11E-14	NA
g24	Mean	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	NA
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.52E-29	NA

TABLE III

RANKS COMPUTED BY THE WILCOXON TEST FOR STATE-OF-THE-ART EAS ON CEC 2006 BENCHMARK FUNCTIONS. HEREINAFTER, ● = THE METHOD IN THE ROW IMPROVES THE METHOD OF THE COLUMN. ○ = THE METHOD IN THE COLUMN IMPROVES THE METHOD OF THE ROW. UPPER DIAGONAL OF LEVEL SIGNIFICANCE AT $\alpha = 0.1$, LOWER DIAGONAL LEVEL OF SIGNIFICANCE AT $\alpha = 0.05$.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
ECHT-ARMOR-DE1 (1)	-	116.0	145.5	187.0●	127.0	183.5●	170.5	116.0
ECHT-ARMOR-DE2 (2)	137.0	-	167.5	187.0●	148.0	183.5●	171.5	116.0
EHCT-DE (3)	107.5	85.5	-	173.5	128.5	173.5	165.5	85.5
AIS-ZYH (4)	66.0	66.0	79.5○	-	87.0	139.5	108.0	50.0○
ISAMODE-CMA (5)	126.0	105.0	124.5	166.0	-	183.5●	170.5	105.0
SAMODE (6)	69.5	69.5	79.5○	113.5	69.5	-	52.5○	52.5○
ECHT-EP2 (7)	82.5	81.5	87.5	145.0	82.5	200.5●	-	63.5○
εDE (8)	137.0	137.0	167.5	203.0●	148.0	200.5●	189.5●	-

TABLE IV

COMPARISON ON THE PERFORMANCE OF ECHT-DE AND ECHT-ARMOR-DE FOR FUNCTIONS IN THE CEC 2010 TEST SUITE AT $D = 10$.

Prob	ECHT-DE						ECHT-ARMOR-DE					
	Best	Median	Worst	Mean	Std	FR	Best	Median	Worst	Mean	Std	FR
C01	-7.4730E-01	-7.4730E-01	-7.4060E-01	-7.4700E-01	1.4E-03	1.00	-7.4730E-01	-7.4730E-01	-7.4060E-01	-7.4700E-01	1.4E-03	1.00
C02	-2.2777E+00	-2.2777E+00	-2.2612E+00	-2.2744E+00	6.7E-03	1.00	-2.2777E+00	-2.2777E+00	-2.2612E+00	**-2.2770E+00**	3.3E-03	1.00
C03	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0E+00	1.00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0E+00	1.00
C04	-1.0000E-05	-1.0000E-05	-1.0000E-05	-1.0000E-05	0.0E+00	1.00	-1.0000E-05	-1.0000E-05	-1.0000E-05	-1.0000E-05	0.0E+00	1.00
C05	-4.8361E+02	-4.3495E+02	-2.7057E+02	-4.1145E+02	7.6E+01	1.00	-4.8361E+02	-4.8361E+02	-4.8361E+02	**-4.8361E+02**	0.0E+00	1.00
C06	-5.7866E+02	-5.7866E+02	-3.6855E+02	-5.6247E+02	4.5E+01	1.00	-5.7866E+02	-5.7866E+02	-5.7866E+02	**-5.7866E+02**	4.0E-13	1.00
C07	0.0000E+00	0.0000E+00	3.9866E+00	1.3290E-01	7.3E-01	1.00	0.0000E+00	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0E+00	1.00
C08	0.0000E+00	7.0979E+00	2.6115E+01	**6.1566E+00**	6.5E+00	1.00	0.0000E+00	1.0942E+01	1.0942E+01	7.5262E+00	5.0E+00	1.00
C09	0.0000E+00	0.0000E+00	4.4082E+00	**1.4691E-01**	8.0E-01	1.00	0.0000E+00	0.0000E+00	4.4082E+00	1.7633E-01	8.8E-01	1.00
C10	0.0000E+00	0.0000E+00	4.1727E+01	1.7117E+00	7.7E+00	1.00	0.0000E+00	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0E+00	1.00
C11	-1.5000E-03	-1.5000E-03	-8.7300E-02‡	-4.4000E-03	1.6E-02	NA	-1.5227E-03	-1.5227E-03	-8.7342E-02‡	-4.2716E-02	4.4E-02	0.52
C12	-1.9920E-01	-1.9920E-01	-5.5435E+02‡	-1.7187E+02	2.2E+02	NA	-1.9925E-01	-1.9925E-01	-1.9925E-01	**-1.9925E-01**	1.6E-13	**1.00**
C13	-6.8429E+01	-6.3518E+01	-6.1649E+01	-6.5121E+01	2.4E+00	1.00	-6.8429E+01	-6.8429E+01	-6.2276E+01	**-6.7169E+01**	2.1E+00	1.00
C14	0.0000E+00	0.0000E+00	1.7191E+07	7.0242E+05	3.2E+06	1.00	0.0000E+00	0.0000E+00	0.0000E+00	**0.0000E+00**	0.0E+00	1.00
C15	0.0000E+00	1.2216E+10	1.8693E+14	2.3392E+13	5.3E+13	1.00	0.0000E+00	3.6732E+00	4.4974E+00	**2.8246E+00**	1.6E+00	1.00
C16	0.0000E+00	3.0437E-02	1.6351E-01	3.9327E-02	4.3E-02	1.00	0.0000E+00	0.0000E+00	2.3426E-01	**2.8478E-02**	5.0E-02	1.00
C17	0.0000E+00	0.0000E+00	1.0884E+00	1.1152E-01	3.3E-01	1.00	0.0000E+00	6.1630E-33	6.1630E-33	**3.6978E-33**	3.1E-33	1.00
C18	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0E+00	1.00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0E+00	1.00

‡ indicates the solution is infeasible.

TABLE V

COMPARED THE RESULTS OF ECHT-ARMOR-DE WITH OTHER EAS FOR FUNCTIONS IN THE CEC 2010 TEST SUITE AT $D = 10$.

Prob	ECHT-DE	AIS-ZYH	εDEg	IEMA	ECHT-ARMOR-DE
C01	-7.4700E-01 ± 1.40E-03	**-7.4705E-01 ± 1.30E-03**	**-7.4704E-01 ± 1.32E-03**	-7.4319E-01 ± 4.33E-03	-7.4700E-01 ± 1.40E-03
C02	-2.2744E+00 ± 6.70E-03	-2.2748E+00 ± 2.00E-03	-2.2588E+00 ± 2.38E-02	**-2.2777E+00 ± 1.82E-07**	**-2.2770E+00 ± 3.30E-03**
C03	**0.0000E+00 ± 0.00E+00**	3.7472E-09 ± 4.81E-04	**0.0000E+00 ± 0.00E+00**	6.2346E-07 ± 1.40E-06	**0.0000E+00 ± 0.00E+00**
C04	**-1.0000E-05 ± 0.00E+00**	-9.9712E-06 ± 4.28E-03	-9.9185E-06 ± 1.54E-07	-9.3702E-06 ± 8.99E-08	**-1.0000E-05 ± 0.00E+00**
C05	-4.1145E+02 ± 7.63E+01	**-4.7996E+02 ± 6.30E+00**	**-4.8361E+02 ± 3.89E-13**	-3.7916E+02 ± 1.79E+02	**-4.8361E+02 ± 0.00E+00**
C06	-5.6247E+02 ± 4.51E+01	**-5.7995E+02 ± 7.30E-08**	-5.7865E+02 ± 3.62E-03	-5.5147E+02 ± 7.36E+01	**-5.7866E+02** ± 4.00E-13
C07	1.3290E-01 ± 7.28E-01	1.1735E-08 ± 2.70E+00	**0.0000E+00 ± 0.00E+00**	3.2569E-09 ± 3.39E-09	**0.0000E+00 ± 0.00E+00**
C08	6.1566E+00 ± 6.45E+00	**4.0919E+00 ± 1.46E+00**	6.7275E+00 ± 5.56E+00	**4.0702E+00 ± 6.38E+00**	7.5262E+00 ± 5.00E+00
C09	**1.4691E-01 ± 8.05E-01**	2.6980E+01 ± 7.50E+01	**0.0000E+00 ± 0.00E+00**	1.9511E+12 ± 5.40E+12	1.7633E-01 ± 8.82E-01
C10	1.7117E+00 ± 7.66E+00	1.6200E+03 ± 5.00E+02	**0.0000E+00 ± 0.00E+00**	2.5613E+12 ± 3.97E+12	**0.0000E+00 ± 0.00E+00**
C11	-4.4000E-03 ± 1.57E-02‡	-9.1995E-04 ± 8.23E-04	**-1.5227E-03 ± 6.34E-11**	**1.5227E-03 ± 2.73E-08**	-4.2716E-02 ± 4.38E-02‡
C12	-1.7187E+02 ± 2.21E+02‡	**-4.3577E+02 ± 6.02E+01**	**-3.3673E+02 ± 1.78E+02**	-6.4817E-01 ± 2.19E+00	-1.9925E-01 ± 1.61E-13
C13	-6.5121E+01 ± 2.38E+00	-6.7874E+01 ± 3.11-E01	**-6.8429E+01 ± 1.02E-06**	**-6.8018E+01 ± 1.40E+00**	-6.7169E+01 ± 2.15E+00
C14	7.0242E+05 ± 3.19E+06	1.2213E-04 ± 2.90E-08	**0.0000E+00 ± 0.00E+00**	5.6308E+01 ± 1.82E+02	**0.0000E+00 ± 0.00E+00**
C15	2.3392E+13 ± 5.30E+13	**5.1855E-09 ± 1.10E-08**	**1.7990E-01 ± 8.81E-01**	1.5753E+08 ± 6.04E+08	2.8246E+00 ± 1.63E+00
C16	3.9327E-02 ± 4.28E-02	**9.9593E-18 ± 6.27E-15**	3.7021E-01 ± 3.71E-01	3.3030E-02 ± 2.26E-02	**2.8478E-02 ± 4.99E-02**
C17	1.1152E-01 ± 3.32E-01	2.9340E+00 ± 2.29E+00	1.2496E-01 ± 1.93E-01	**3.1509E-03 ± 1.57E-02**	**3.6978E-33 ± 3.08E-33**
C18	**0.0000E+00 ± 0.00E+00**	1.6590E+00 ± 1.27E+00	9.6788E-19 ± 1.81E-18	1.6179E-14 ± 3.82E-14	**0.0000E+00 ± 0.00E+00**

‡ indicates that there are infeasible solutions in this function over 25 independent runs.

TABLE VI

RANKS COMPUTED BY THE WILCOXON TEST FOR STATE-OF-THE-ART EAS ON CEC 2010 BENCHMARK FUNCTIONS AT $D = 10$.

	(1)	(2)	(3)	(4)	(5)
ECHT-DE (1)	–	46.0	37.0○	63.0	22.0○
AIS-ZYH (2)	90.0	–	50.0	98.0	52.0
εDEg (3)	99.0	86.0	–	105.0●	73.0
IEMA (4)	73.0	38.0	31.0	–	24.0○
ECHT-ARMOR-DE (5)	114.0●	84.0	63.0	112.0●	–

TABLE VII
COMPARISON ON THE PERFORMANCE OF ECHT-DE AND ECHT-ARMOR-DE FOR FUNCTIONS IN THE CEC 2010 TEST SUITE AT $D = 30$.

Prob	ECHT-DE						ECHT-ARMOR-DE					
	Best	Median	Worst	Mean	Std	FR	Best	Median	Worst	Mean	Std	FR
C01	-8.2170E-01	-8.0120E-01	-7.5570E-01	**-7.9940E-01**	1.79E-02	1.00	-8.1806E-01	-8.0029E-01	-7.3601E-01	-7.8992E-01	2.51E-02	1.00
C02	-2.2251E+00	-2.0662E+00	-1.3511E+00	-1.9943E+00	2.10E-01	1.00	-2.2607E+00	-2.1900E+00	-1.9746E+00	**-2.1706E+00**	7.36E-02	1.00
C03	3.2433E-21	1.0983E+02	1.8496E+02	9.8920E+01	6.26E+01	1.00	2.5801E-24	2.8673E+01	2.8673E+01	**2.6380E+01**	7.94E+00	1.00
C04	-3.3015E-06	-2.9456E-06	4.6205E-01	**-1.0257E-06**	9.01E-02	1.00	-3.3326E-06	9.9236E-05	1.0886E+00	8.3713E-02	2.89E-01	1.00
C05	-2.1368E+02	-1.6300E+02	4.7719E+02	-1.0642E+02	1.67E+02	1.00	-4.8122E+02	-4.7647E+02	7.6414E+01	**-4.3335E+02**	1.46E+02	1.00
C06	-2.9572E+02	-1.4732E+02	2.6353E+02	-1.3762E+02	9.89E+01	1.00	-5.3010E+02	-5.2465E+02	1.2454E+02	**-4.8931E+02**	1.32E+02	1.00
C07	0.0000E+00	0.0000E+00	3.9866E+00	1.3290E-01	7.28E-01	1.00	0.0000E+00	3.4286E-26	1.1045E-24	**1.0789E-25**	2.20E-25	1.00
C08	0.0000E+00	0.0000E+00	5.8567E+02	3.3585E+01	1.11E+02	1.00	0.0000E+00	8.5541E-26	1.5113E+02	**2.0101E+01**	4.70E+01	1.00
C09	0.0000E+00	0.0000E+00	6.5710E+02	4.2441E+01	1.38E+02	1.00	0.0000E+00	2.2153E-25	1.1527E+02	**4.6110E+00**	2.31E+01	1.00
C10	0.0000E+00	3.1309E+01	4.7510E+02	**5.3381E+01**	8.83E+01	1.00	6.0209E-13	3.1309E+01	5.3332E+02	6.5536E+01	1.07E+02	1.00
C11	-4.0000E-04	-2.0000E-04	2.0400E-02‡	2.6000E-03	6.00E-03	NA	-3.9234E-04	-3.9234E-04	1.8671E-02‡	1.1327E-03	5.28E-03	0.92
C12	-1.9930E-01	-1.9930E-01	-7.4816E+02‡	-2.5129E+01	1.37E+02	NA	-1.9926E-01	-1.9926E-01	7.6343E-01	**-1.6076E-01**	1.93E-01	**1.00**
C13	-6.8429E+01	-6.4619E+01	-6.0939E+01	-6.4583E+01	1.67E+00	1.00	-6.7416E+01	-6.4908E+01	-6.0769E+01	**-6.4646E+01**	1.97E+00	1.00
C14	0.0000E+00	0.0000E+00	3.7101E+06	1.2368E+05	6.77E+05	1.00	1.5809E-27	4.4875E-26	1.1507E+04	**6.6135E+02**	2.47E+03	1.00
C15	1.9922E+09	8.5527E+10	2.3252E+12	1.9409E+11	4.35E+11	1.00	1.1716E-04	2.1603E+01	5.9937E+09	**3.1316E+08**	1.20E+09	1.00
C16	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.00E+00	1.00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.00E+00	1.00
C17	0.0000E+00	1.9273E-01	1.8986E+00	**2.7496E-01**	3.78E-01	1.00	3.3564E-16	4.2103E-01	1.2633E+00	4.0336E-01	3.51E-01	1.00
C18	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.00E+00	1.00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.00E+00	1.00

‡ indicates the solution is infeasible.

TABLE VIII
COMPARED THE RESULTS OF ECHT-ARMOR-DE WITH OTHER EAS FOR FUNCTIONS IN THE CEC 2010 TEST SUITE AT $D = 30$.

Prob	ECHT-DE	AIS-ZYH	εDEg	IEMA	ECHT-ARMOR-DE
C01	-7.9940E-01 ± 1.79E-02	**-8.2011E-01 ± 3.25E-04**	**-8.2087E-01 ± 7.10E-04**	-8.1777E-01 ± 4.79E-03	-7.8992E-01 ± 2.51E-02
C02	-1.9943E+00 ± 2.10E-01	**-2.2125E+00 ± 2.84E-03**	**-2.1745E+00 ± 1.20E-02**	-1.5045E+00 ± 2.14E+00	-2.1706E+00 ± 7.36E-02
C03	9.8920E+01 ± 6.26E+01	6.6758E+01 ± 4.26E+02	**2.8838E+01 ± 8.00E-01**	–	**2.6380E+01 ± 7.94E+00**
C04	**-1.0257E-06 ± 9.01E-02**	**1.9761E-03 ± 1.61E-03**	8.1630E-03 ± 3.06E-03	–	8.3713E-02 ± 2.89E-01
C05	-1.0642E+02 ± 1.67E+02	**-4.3611E+02 ± 2.51E+01**	**-4.4955E+02 ± 2.89E+00**	-2.7093E+02 ± 1.41E+00	-4.3335E+02 ± 1.46E+02
C06	-1.3762E+02 ± 9.89E+01	-4.5426E+02 ± 4.79E+01	**-5.2791E+02 ± 4.74E-01**	-1.3288E+02 ± 5.61E+02	**-4.8931E+02 ± 1.32E+02**
C07	1.3290E-01 ± 7.28E-01	1.0730E+00 ± 1.61E+00	**2.6036E-15 ± 1.23E-15**	8.4861E-10 ± 4.84E-10	**1.0789E-25 ± 2.20E-25**
C08	3.3585E+01 ± 1.11E+02	**1.6531E+00 ± 6.41E-01**	**7.8315E-14 ± 4.85E-14**	1.7703E+01 ± 4.08E+01	2.0101E+01 ± 4.70E+01
C09	4.2441E+01 ± 1.38E+02	**1.5654E+00 ± 1.96E+00**	1.0721E+01 ± 2.82E+01	2.9879E+07 ± 4.50E+07	**4.6110E+00 ± 2.31E+01**
C10	5.3381E+01 ± 8.83E+01	**1.7847E+01 ± 1.88E+01**	3.3262E+01 ± 4.54E-01	1.5834E+07 ± 1.68E+07	6.5536E+01 ± 1.07E+02
C11	2.6000E-03 ± 6.00E-03‡	**-1.5790E-04 ± 4.67E-05**	**-2.8638E-04 ± 2.71E-05**	–	1.1327E-03 ± 5.28E-03
C12	-2.5129E+01 ± 1.37E+02‡	**4.2881E-06 ± 4.52E-04**	3.5623E+02 ± 2.89E+02‡	–	-1.6076E-01 ± 1.93E-01‡
C13	-6.4583E+01 ± 1.67E+00	**-6.6236E+01 ± 2.27E-01**	-6.5353E+01 ± 5.73E+01	**-6.7487E+01 ± 9.83E-01**	-6.4646E+01 ± 1.97E+00
C14	1.2368E+05 ± 6.77E+05	**8.6828E-07 ± 3.14E-07**	**3.0894E-13 ± 5.61E-13**	6.1524E-02 ± 3.07E-01	6.6135E+02 ± 2.47E+03
C15	1.9409E+11 ± 4.35E+11	**3.4128E+01 ± 3.82E+01**	**2.1603E+01 ± 1.10E-04**	2.2949E+08 ± 4.64E+08	3.1316E+08 ± 1.20E+09
C16	**0.0000E+00 ± 0.00E+00**	8.2062E-02 ± 1.12E-01	2.1684E-21 ± 1.06E-20	1.6329E-03 ± 8.16E-03	**0.0000E+00 ± 0.00E+00**
C17	**2.7496E-01 ± 3.78E-01**	3.6051E+00 ± 2.54E+00	6.3265E+00 ± 4.99E+00	**8.8397E-02 ± 1.51E-01**	4.0336E-01 ± 3.51E-01
C18	**0.0000E+00 ± 0.00E+00**	4.0152E+01 ± 1.80E+01	8.7546E+01 ± 1.66E+02	4.7384E-14 ± 6.57E-14	**0.0000E+00 ± 0.00E+00**

‡ indicates that there are infeasible solutions in this function over 25 independent runs.

TABLE IX
RANKS COMPUTED BY THE WILCOXON TEST FOR STATE-OF-THE-ART EAS ON CEC 2010 BENCHMARK FUNCTIONS AT $D = 30$.

	(1)	(2)	(3)	(4)
ECHT-DE (1)	-	27.0○	22.0○	24.5○
AIS-ZYH (2)	109.0●	-	60.0	82.0
εDEg (3)	114.0●	76.0	-	95.0
ECHT-ARMOR-DE (4)	111.5●	54.0	41.0	-

TABLE X

INFLUENCE OF ARMOR TO $(\mu + \lambda)$-CDE AND DSS-MDE IN ALL BENCHMARK FUNCTIONS. THE NFEs$_\epsilon$, SR, AND AR RESULTS ARE REPORTED. ALL RESULTS ARE AVERAGED OVER 50 INDEPENDENT RUNS.

Prob	$(\mu+\lambda)$-CDE			$(\mu+\lambda)$-ARMOR-CDE			AR	DSS-MDE			DSS-ARMOR-MDE			AR
	Mean	Std	SR	Mean	Std	SR		Mean	Std	SR	Mean	Std	SR	
g01	89,000.0	893.7	1.00	**49,109.2**	**2,676.3**	1.00	1.81	196,659.0	5,186.6	1.00	**97,245.0**	**2,480.2**	1.00	2.02
g02	277,379.0	7,489.7	**0.96**	**103,954.8**	**10,080.5**	0.58	1.61	169,979.1	12,339.6	**0.64**	**93,802.5**	**9,282.4**	0.48	1.36
g03	111,025.0	5,474.4	1.00	**91,499.8**	**22,335.8**	1.00	1.21	207,981.0	22,060.7	1.00	**69,120.0**	**6,957.7**	1.00	3.01
g04	30,620.0	296.5	1.00	**19,478.2**	**847.6**	1.00	1.57	73,305.0	2,073.7	1.00	**44,667.0**	**1,562.6**	1.00	1.64
g05	**165,079.0**	**78.3**	1.00	165,235.0	225.5	1.00	1.00	64,782.0	2,378.7	1.00	**29,088.0**	**996.6**	1.00	2.23
g06	11,032.0	158.1	1.00	**8,852.2**	**490.6**	1.00	1.25	25,398.0	1,070.2	1.00	**15,813.0**	**525.5**	1.00	1.61
g07	141,038.0	1,177.3	1.00	**77,602.0**	**3,654.6**	1.00	1.82	171,423.0	6,752.6	1.00	**89,496.0**	**3,123.6**	1.00	1.92
g08	2,010.0	62.1	1.00	**1,636.6**	**272.4**	1.00	1.23	4,563.0	682.8	1.00	**3,141.0**	**438.7**	1.00	1.45
g09	39,953.0	466.7	1.00	**28,831.6**	**1,733.2**	1.00	1.39	59,616.0	2,091.1	1.00	**32,949.0**	**1,404.0**	1.00	1.81
g10	188,725.0	1,945.9	1.00	**94,750.6**	**5,013.0**	1.00	1.99	256,455.0	12,864.6	1.00	**127,998.0**	**5,273.4**	1.00	2.00
g11	79,475.0	3,214.4	1.00	**78,122.8**	**14,321.2**	1.00	1.02	26,118.0	7,315.2	1.00	**8,064.0**	**2,481.5**	1.00	3.24
g12	4,908.0	219.1	1.00	**4,131.4**	**917.4**	1.00	1.19	4,545.0	1,210.4	1.00	**3,825.0**	**976.0**	1.00	1.19
g13	148,237.0	380.9	1.00	**146,805.4**	**2,616.0**	1.00	1.01	77,922.0	16,135.1	**1.00**	**48,310.0**	**40,194.5**	0.90	1.45
g14	176,671.0	697.5	1.00	**170,703.4**	**3,689.9**	1.00	1.03	286,848.0	9,003.9	1.00	**160,938.0**	**7,599.4**	1.00	1.78
g15	**130,622.0**	**2,276.8**	1.00	134,344.0	13,350.4	1.00	0.97	47,376.0	2,576.0	1.00	**18,027.0**	**1,181.9**	1.00	2.63
g16	19,154.0	231.4	1.00	**12,707.8**	**812.4**	1.00	1.51	38,853.0	1,867.6	1.00	**23,931.0**	**1,194.0**	1.00	1.62
g17	183,962.0	535.3	1.00	**170,535.4**	**673.2**	1.00	1.08	103,734.0	10,994.0	1.00	**93,222.0**	**43,174.0**	1.00	1.11
g18	215,068.0	6,770.8	1.00	**73,645.6**	**9,800.9**	1.00	2.92	190,935.0	11,554.7	1.00	**95,319.0**	**7,405.9**	1.00	2.00
g19	268,374.0	3,178.6	1.00	**123,995.2**	**6,145.9**	1.00	2.16	385,794.0	13,514.3	1.00	**208,179.0**	**16,856.6**	1.00	1.85
g21	209,896.0	1,151.1	0.92	**192,138.6**	**4,465.7**	**0.98**	1.16	198,980.6	8,054.9	0.96	**95,002.5**	**13,796.7**	**0.98**	2.14
g23	263,695.0	2,096.2	1.00	**205,609.6**	**4,235.1**	1.00	1.28	464,537.4	29,660.9	0.38	**239,121.0**	**41,365.4**	**1.00**	5.11
g24	5,059.0	84.2	1.00	**3,854.2**	**286.2**	1.00	1.31	11196	1022.3	1.00	**6858.0**	**588.9**	1.00	1.63
avg	–		**0.99**	–		0.98	1.43	–		0.95	–		**0.97**	2.04

TABLE XI

ON THE ROBUSTNESS OF ARMOR TO $(\mu + \lambda)$-CDE WITH DIFFERENT F VALUES IN ALL BENCHMARK FUNCTIONS. THE NFEs$_\epsilon$, SR, AND AR RESULTS ARE REPORTED. ALL RESULTS ARE AVERAGED OVER 50 INDEPENDENT RUNS.

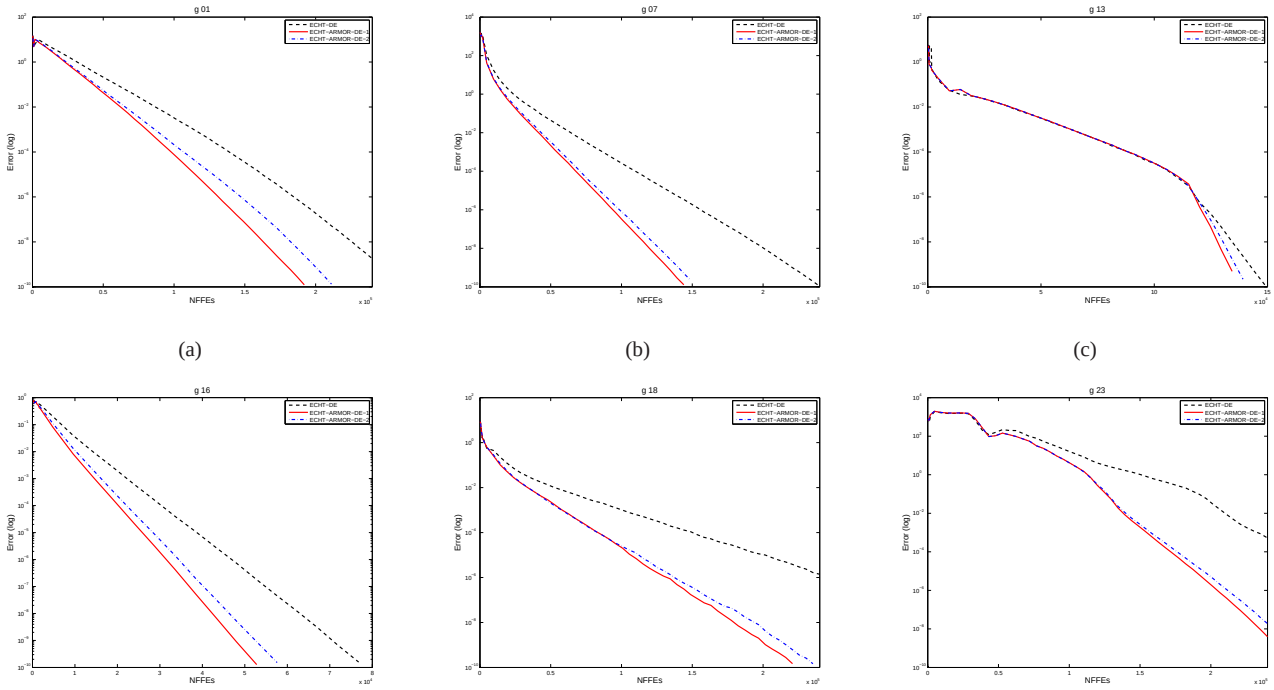
Prob	$F = 0.7, Cr = 0.9$							$F = 0.9, Cr = 0.9$						
	$(\mu+\lambda)$-CDE			$(\mu+\lambda)$-ARMOR-CDE			AR	$(\mu+\lambda)$-CDE			$(\mu+\lambda)$-ARMOR-CDE			AR
	Mean	Std	SR	Mean	Std	SR		Mean	Std	SR	Mean	Std	SR	
g01	57,715.0	2,699.4	1.00	**37,874.2**	**1,717.3**	1.00	1.52	110,101.6	5,804.8	1.00	**65,405.2**	**3,182.2**	1.00	1.68
g02	110,626.5	11,284.8	**0.74**	**59,500.0**	**7,949.9**	0.48	1.21	250,282.3	21,290.2	**0.78**	**135,377.5**	**13,903.2**	0.56	1.33
g03	62,137.6	13,323.8	1.00	**60,365.2**	**14,130.9**	1.00	1.03	135,608.2	39,116.5	1.00	**115,297.0**	**31,331.6**	1.00	1.18
g04	26,076.4	1,264.5	1.00	**18,289.6**	**1,088.8**	1.00	1.43	32,519.2	1,443.3	1.00	**22,775.2**	**1,157.7**	1.00	1.43
g05	**165,592.0**	**334.1**	1.00	165,671.8	1,339.0	1.00	1.00	164,651.2	379.6	1.00	**164,042.2**	**615.8**	1.00	1.00
g06	10,200.4	440.0	1.00	**8,197.0**	**394.0**	1.00	1.24	11,775.4	673.2	1.00	**10,074.4**	**644.5**	1.00	1.17
g07	82,461.4	5,190.4	1.00	**53,809.0**	**3,509.2**	1.00	1.53	197,898.4	10,933.1	1.00	**117,875.8**	**6,665.3**	1.00	1.68
g08	1,792.0	268.3	1.00	**1,540.0**	**224.5**	1.00	1.16	2,136.4	312.8	1.00	**1,687.0**	**345.3**	1.00	1.27
g09	28,197.4	1,384.8	1.00	**20,834.8**	**1,005.5**	1.00	1.35	57,043.0	3,429.8	1.00	**41,515.6**	**2,205.7**	1.00	1.37
g10	NA	NA	0.00	**172,912.4**	**135,380.0**	**0.68**	NA	227,227.0	10,175.6	1.00	**140,186.2**	**8,647.6**	1.00	1.62
g11	74,851.0	15,790.3	1.00	**73,557.4**	**13,074.6**	1.00	1.02	83,137.6	18,846.3	1.00	**82,231.2**	**1,029.7**	1.00	1.01
g12	4,799.2	1,088.5	1.00	**3,921.4**	**908.6**	1.00	1.22	4,286.8	891.8	1.00	**3,505.6**	**752.7**	1.00	1.22
g13	145,852.0	3,453.0	1.00	**143,764.6**	**3,624.4**	1.00	1.01	147,099.4	2,201.3	1.00	**142,139.2**	**4,512.8**	1.00	1.03
g14	169,514.8	4,572.9	1.00	**167,234.2**	**3,096.4**	1.00	1.01	188,129.2	4,194.7	1.00	**180,434.8**	**10,629.5**	1.00	1.04
g15	126,716.8	9,819.8	1.00	**113,432.2**	**11,834.4**	1.00	1.12	141,110.2	7,286.3	1.00	**120,328.6**	**10,598.6**	1.00	1.17
g16	15,072.4	1,025.4	1.00	**11,103.4**	**551.9**	1.00	1.36	21,137.2	1,476.1	1.00	**15,089.2**	**1,142.6**	1.00	1.40
g17	170,753.8	743.7	1.00	**170,417.8**	**615.8**	1.00	1.00	168,884.8	1,336.0	1.00	**168,494.2**	**1,232.9**	1.00	1.00
g18	86,669.8	12,945.1	1.00	**46,446.4**	**5,000.4**	1.00	1.87	185,080.0	23,037.5	1.00	**97,539.4**	**12,953.4**	1.00	1.90
g19	125,427.4	6,384.4	1.00	**80,155.6**	**5,063.5**	1.00	1.56	338,665.6	13,775.0	1.00	**185,966.2**	**9,905.3**	1.00	1.82
g21	239,855.0	107,912.4	0.12	**184,525.0**	**3,367.7**	**0.28**	3.03	216,785.5	6,734.3	0.94	**196,390.0**	**3,847.6**	0.94	1.10
g23	210,065.8	3,884.4	1.00	**201,968.2**	**22,026.7**	1.00	1.04	274,430.8	12,279.3	1.00	**230,011.6**	**6,849.2**	1.00	1.19
g24	4,774.0	422.1	1.00	**3,795.4**	**284.0**	1.00	1.26	5248.6	454.2	1.00	**4446.4**	**394.3**	1.00	1.18
avg	–		0.90	–		**0.93**	1.33	–		**0.99**	–		0.98	1.31

TABLE XII
INFLUENCE OF DIFFERENT MODELS IN ARMOR. THE $NFEs_\epsilon$ VALUES OF ECHT-DE, ECHT-ARMOR-DE3, AND ECHT-ARMOR-DE4 FOR THE CEC 2006 FUNCTIONS ARE REPORTED.

Prob	ECHT-DE (1)			ECHT-ARMOR-DE3 (2)			ECHT-ARMOR-DE4 (3)			AR	AR
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	(1) vs (2)	(1) vs (3)
g01	1.384E+05	4.06E+03	1.00	**1.011E+05**	3.91E+03	1.00	**1.130E+05**	3.50E+03	1.00	1.37	1.22
g02	8.205E+04	6.25E+03	**0.42**	**6.509E+04**	8.91E+03	0.26	**7.411E+04**	2.34E+04	**0.30**	0.78	0.79
g03	1.161E+05	1.53E+03	1.00	**1.137E+05**	2.53E+03	1.00	**1.150E+05**	2.46E+03	1.00	1.02	1.01
g04	6.470E+04	2.43E+03	1.00	**4.450E+04**	1.32E+03	1.00	**4.946E+04**	1.44E+03	1.00	1.45	1.31
g05	1.204E+05	1.45E+03	1.00	**1.196E+05**	6.85E+02	1.00	1.204E+05	1.27E+03	1.00	1.01	1.00
g06	2.224E+04	1.24E+03	1.00	**1.549E+04**	7.53E+02	1.00	**1.722E+04**	7.64E+02	1.00	1.44	1.29
g07	1.088E+05	4.87E+03	1.00	**7.144E+04**	3.18E+03	1.00	**7.645E+04**	3.42E+03	1.00	1.52	1.42
g08	2.644E+03	4.15E+02	1.00	**2.172E+03**	2.80E+02	1.00	**2.328E+03**	4.17E+02	1.00	1.22	1.14
g09	4.194E+04	1.81E+03	1.00	**2.956E+04**	1.09E+03	1.00	**3.268E+04**	1.48E+03	1.00	1.42	1.28
g10	1.855E+05	7.49E+03	1.00	**1.109E+05**	4.18E+03	1.00	**1.203E+05**	5.02E+03	1.00	1.67	1.54
g11	**5.820E+04**	1.85E+04	1.00	6.264E+04	1.11E+04	1.00	**6.178E+04**	1.34E+04	1.00	0.93	0.94
g12	3.072E+03	7.83E+02	1.00	**2.588E+03**	5.63E+02	1.00	**2.316E+03**	5.38E+02	1.00	1.19	1.33
g13	1.109E+05	4.39E+03	1.00	**1.090E+05**	4.76E+03	1.00	**1.090E+05**	4.76E+03	1.00	1.02	1.02
g14	1.401E+05	6.59E+03	1.00	**1.293E+05**	3.06E+03	1.00	**1.317E+05**	3.48E+03	1.00	1.08	1.06
g15	1.083E+05	5.58E+03	1.00	**1.057E+05**	7.51E+03	1.00	**1.074E+05**	7.56E+03	1.00	1.03	1.01
g16	3.019E+04	1.37E+03	1.00	**2.055E+04**	9.11E+02	1.00	**2.329E+04**	1.19E+03	1.00	1.47	1.30
g17	1.174E+05	1.59E+03	1.00	**1.168E+05**	9.70E+02	1.00	**1.171E+05**	1.06E+03	1.00	1.00	1.00
g18	1.431E+05	2.02E+04	1.00	**9.254E+04**	1.26E+04	1.00	**8.807E+04**	1.05E+04	1.00	1.55	1.63
g19	NA	NA	0.00	**1.863E+05**	1.34E+04	**1.00**	**2.016E+05**	1.41E+04	**1.00**	NA	NA
g21	1.734E+05	6.82E+03	1.00	**1.468E+05**	2.57E+03	1.00	**1.564E+05**	3.05E+03	1.00	1.18	1.11
g23	2.274E+05	5.72E+03	0.72	**1.719E+05**	6.00E+03	**1.00**	**1.799E+05**	5.70E+03	**1.00**	1.84	1.76
g24	8.120E+03	7.81E+02	1.00	**6.132E+03**	3.78E+02	1.00	**6.556E+03**	6.01E+02	1.00	1.32	1.24
avg	–		0.915	–		**0.966**	–		**0.968**	**1.26**	1.21

TABLE XIII
COMPARISON ON THE $NFEs_f$ VALUES OF ECHT-DE, ECHT-ARMOR-DE1, AND ECHT-ARMOR-DE3 FOR THE CEC 2006 FUNCTIONS.

Prob	ECHT-DE (1)			ECHT-ARMOR-DE1 (2)			AR'	ECHT-ARMOR-DE3 (3)			AR'
	Mean	Std	FR	Mean	Std	FR	(1) vs (2)	Mean	Std	FR	(1) vs (3)
g01	3.292E+03	6.14E+02	1.00	**2.352E+03**	3.99E+02	1.00	1.40	**2.720E+03**	4.64E+02	1.00	1.21
g03	**4.296E+04**	1.00E+04	1.00	**4.375E+04**	1.01E+04	1.00	0.98	4.426E+04	1.09E+04	1.00	0.97
g05	**1.160E+05**	2.33E+03	1.00	1.165E+05	2.11E+03	1.00	1.00	**1.160E+05**	2.79E+03	1.00	1.00
g06	1.448E+03	3.25E+02	1.00	**1.104E+03**	2.56E+02	1.00	1.31	**1.168E+03**	2.47E+02	1.00	1.24
g07	2.412E+03	5.35E+02	1.00	**1.936E+03**	3.91E+02	1.00	1.25	**2.232E+03**	4.49E+02	1.00	1.08
g08	2.440E+02	1.09E+02	1.00	**2.360E+02**	7.76E+01	1.00	1.03	**2.400E+02**	9.04E+01	1.00	1.02
g09	2.960E+02	1.35E+02	1.00	**2.800E+02**	1.07E+02	1.00	1.06	**2.920E+02**	1.29E+02	1.00	1.01
g10	2.480E+03	5.92E+02	1.00	**1.920E+03**	3.59E+02	1.00	1.29	**2.040E+03**	3.81E+02	1.00	1.22
g11	2.248E+04	1.35E+04	1.00	**2.090E+04**	1.15E+04	1.00	1.08	**2.082E+04**	1.14E+04	1.00	1.08
g13	1.109E+05	4.39E+03	1.00	**1.092E+05**	4.70E+03	1.00	1.01	**1.090E+05**	4.76E+03	1.00	1.02
g14	**1.109E+05**	6.80E+03	1.00	1.136E+05	4.79E+03	1.00	0.98	**1.122E+05**	6.07E+03	1.00	0.99
g15	1.033E+05	6.46E+03	1.00	**1.010E+05**	6.33E+03	1.00	1.02	**9.962E+04**	7.84E+03	1.00	1.04
g16	1.236E+03	4.38E+02	1.00	**1.028E+03**	3.18E+02	1.00	1.20	**1.088E+03**	3.86E+02	1.00	1.14
g17	**1.116E+05**	3.05E+03	1.00	**1.129E+05**	2.44E+03	1.00	0.99	1.132E+05	2.45E+03	1.00	0.99
g18	7.568E+03	6.94E+02	1.00	**5.416E+03**	5.83E+02	1.00	1.40	**6.232E+03**	7.68E+02	1.00	1.21
g21	**1.095E+05**	5.52E+03	1.00	1.131E+05	3.01E+03	1.00	0.97	**1.127E+05**	4.15E+03	1.00	0.97
g22	NA	NA	0.00	**2.183E+05**	1.12E+04	0.88	NA	**2.273E+05**	9.15E+03	0.30	NA
g23	**1.061E+05**	2.69E+03	1.00	1.076E+05	4.25E+03	1.00	0.99	**1.060E+05**	3.38E+03	1.00	1.00
avg	–		0.944	–		**0.993**	**1.11**	–		**0.961**	**1.07**



(a) (b) (c)

(d) (e) (f)

Fig. 2. Convergence graphs of ECHT-DE, ECHT-ARMOR-DE1, and ECHT-ARMOR-DE2 for the selected functions in CEC 2006. (a) g01; (b) g07; (c) g13; (d) g16; (e) g18 (f) g23.

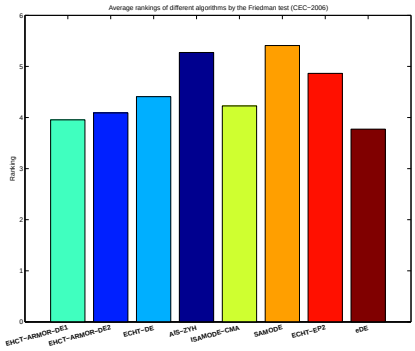
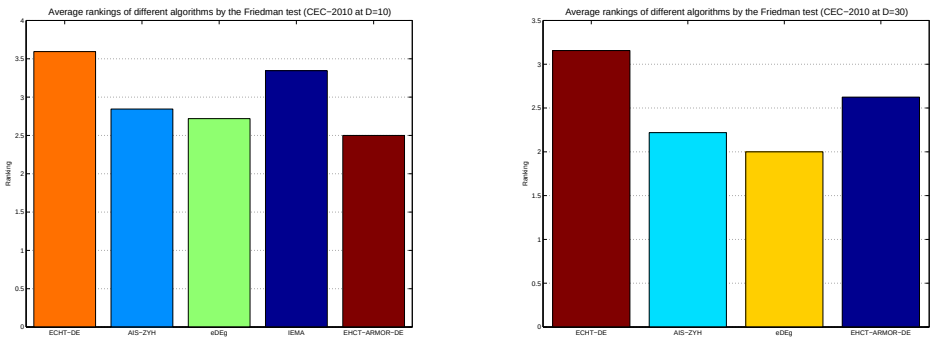


Fig. 3. Average rankings of different algorithms by the Friedman test for the CEC 2006 functions. The lower the ranking, the better the performance obtained by the algorithm. The p-value computed by the Friedman test is 0.209197.



(a) (b)

Fig. 4. Average rankings of different algorithms by the Friedman test for the CEC 2010 functions with respect to the mean quality of final solutions. (a) $D = 10$; The p-value computed by the Friedman test is 0.260226. (b) $D = 30$; The p-value computed by the Friedman test is 0.059022.